ACL 2018

The 56th Annual Meeting of the Association for Computational Linguistics

Proceedings of the Conference, Vol. 1 (Long Papers)

July 15 - 20, 2018
Melbourne, Australia

Diamond Sponsors:

Google

ByteDance 字节跳动

SAMSUNG Research

facebook

Platinum Sponsors:

amazon

Bai du 百度

京东

RECRUIT Institute of Technology

Tencent 腾讯

Gold Sponsors:

IBM Research AI

Microsoft

NAVER | LINE

CVTE 视源股份

digital health crc

Silver Sponsors:

Bronze Sponsors:

Supporters:

Student Volunteers Sponsors:

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-948087-32-2 (Volume 1)

Message from the General Chair

It is an honor to write the initial words of this proceedings as General Chair of the 56th Annual Meeting of the Association for Computational Linguistics! This is only the second time that an ACL conference has been held in Australia — the first time was for the joint COLING/ACL conference in June of 2006 in Sydney, and I was one of its Program Chairs. For ACL 2018 we have tried to maintain the welcoming and intimate spirit and the relaxed and genial character of the much smaller ACL conferences of the past in spite of the ever-growing number of researchers in the field and participants in our conferences.

It is my pleasure here to express gratitude to all those without whom this conference would not exist. My biggest thanks go to the Program Chairs Iryna Gurevych and Yusuke Miyao, as well as to Local Chairs Tim Baldwin, Trevor Cohn and Karin Verspoor. They have done a tremendous job to manage the submission and review process, and the local arrangement details, respectively.

I also want to thank all of the other chairs for their very hard work: Workshops Chairs Brendan O'Connor and Eva Maria Vecchi; Tutorials Chairs Yoav Artzi and Jacob Eisenstein; Demo Chairs Fei Liu and Thamar Solorio; Student Research Workshop Organizers Vered Shwartz, Jeniya Tabassum and Rob Voigt; Faculty Advisors to the Student Research Workshop Marie-Catherine de Marneffe, Wanxiang Che and Malvina Nissim; Publications Chairs Shay Cohen, Kevin Gimpel and Wei Lu; Exhibits Coordinator Karin Vespoor; Student Volunteer Coordinator Karin Vespoor; Conference Handbook Chairs Jey Han Lau and Trevor Cohn; Publicity Chair Sarvnaz Karimi; Local Sponsorship Chair Cecile Paris; Webmaster Andrew MacKinlay; and Priscilla Rasmussen, giver of advice and wisdom to all of us as ACL Business Manager.

I also warmly thank the ACL Executive Committee for its guidance and advice on many important issues and concerns as they arose.

I am also extremely grateful to all the sponsors for their great support to the conference.

Many thanks to the area chairs, the reviewers, the invited speakers, the authors of the various papers, posters and presentations.

And, finally, many many thanks to all the participants who will put the final touches on making ACL 2018 an exciting, stimulating and inspiring event!

Claire Cardie
ACL 2018 General Chair
July 2018

Message from the Program Committee Co-Chairs

Welcome to the 56th Annual Meeting of the Association for Computational Linguistics 2018 – or ACL 2018 for short.

In September 2017, Program Committee Co-Chairs (PCs) posted the call for nominations of Area Chairs (AC), Reviewers and Invited Speakers. We received 752 responses in total. Overall, out of 388 valid nominations for area chairs, 299 unique persons were suggested; 110 persons were self-nominations. About 70% of the 56 selected area chairs (later expanded to 61 area chairs due to the high number of submissions) were nominated by the community. For the reviewers, we collected 936 valid nominations. At the PhD level, 139 persons were self-nominations and 129 were nominated by others. At the Postdoc/Ass.Prof. level, 160 were self-nominated, 112 nominated by others. At the Prof. level, 221 persons were self-nominated, 175 nominated by others.

We received 138 unique nominations for invited speakers, from which two invited speakers of the conference were selected:

- Carolyn Penstein Rosé, Language Technologies Institute at Carnegie Mellon University, USA
- Anton van den Hengel, Australian Centre for Visual Technologies at University of Adelaide, Australia

Our community is steadily growing: in total, 1621 submissions were received right after the submission deadline: 1045 long, 576 short papers. 13 erroneous submissions were deleted or withdrawn in the preliminary checks by PCs. 25 papers were rejected without review (16 long, 9 short); the reasons are the violation of the ACL 2018 style and dual submission guidelines. 32 papers were withdrawn before the review period started; the main reason was that the papers have been accepted as the short papers at NAACL HLT 2018. In total, 1551 papers went into the reviewing phase: 1021 long, 530 short papers. 1610 reviewers (1473 primary and 137 secondary reviewers) were involved in the reviewing process; each reviewer has reviewed about 3 papers on average. 3 long and 4 short papers were withdrawn during the reviewing period, and finally 1018 long and 526 short papers were considered during the acceptance decision phase.

The assignment of papers to areas and reviewers has been done in multiple rounds. First round: Initial assignments of papers to areas were determined automatically with the help of the authors' input, while PCs went through all submissions and moved papers to other areas, considering COI and the topical fit. PCs assigned one AC as a meta-reviewer to each paper using Toronto Paper Matching System (TPMS) scores. Second round: ACs looked into the papers in their area, and adjusted meta-reviewer assignments. ACs sent a report to PCs if they found any problems. Third round: PCs made the final decision, considering the workload balance, possible COIs and the topical fit. Fourth round: ACs decided which reviewers would review each paper, based on AC's knowledge about the reviewers, TPMS scores, reviewers' bids, and COI.

We have introduced several innovations to the reviewing process. One of them is an argument-based review form. The reviewers were asked to provide arguments for and against the paper. This has been tremendously helpful for ACs and PCs to analyze the reviews and come up with final recommendations. The authors were asked to respond to the con arguments during the rebuttal. In coordination with the NAACL HLT 2018 PCs, we plan to do some analytics on anonymized reviews and rebuttal statements, with the consent of the reviewers and authors. Our purpose is to improve the quality of the review process. The data will be compiled into a unique corpus for NLP, and will be made available to the research community after appropriate anonymization checks, at the earliest in 2 years after ACL 2018.

We hope to provide data on *how to review* to younger researchers, and to improve the transparency of the reviewing process in general.

The ACL 2018 conference is super-competitive: We accepted 256 out of 1018 submitted long papers and 125 out of 526 short papers, with an overall acceptance rate of 24.7%. The details of the review process are available at the conference homepage. Criteria of acceptance were mainly:

- strengths/weaknesses raised by reviewers and their significance;
- the result of discussions and author responses;
- contribution to CL as the science of language: whether the paper advances (or contributes to) our understanding of language in any way;
- diversity: we do not want to fill ACL with similar papers like achieving 1% improvement on a well-known task.

We also considered the balance of paper types, topics and contributions and re-considered the acceptance when reviewers reported any problem in preliminary checks (*Appropriateness* to *Handling of Human Participants*).

Continuing the tradition, ACL 2018 will feature 20 papers which were accepted for publication in the Transactions of the Association for Computational Linguistics (TACL). The TACL papers were split into 10 oral presentations and 10 poster presentations.

There are many people to thank for who have worked diligently to make ACL 2018 possible. All names are listed in the Program Committee section of the Front Matter.

Since the conference size continues to grow and the organizational complexity increases, we have introduced the role of Program Committee Co-Chair Assistants. In total, 5 senior researchers have supported the PCs during most intensive work phases to handle the communication in a timely manner, draft various documents and effectively prepare decisions.

Thanks to our area chairs for their hard work on recruiting reviewers, managing reviews, leading discussions, and making recommendations.

This program certainly would not be possible without the help of the 1610 reviewers. In particular, 192 reviewers from this list were recognized by the area chairs as outstanding reviewers who have turned in exceptionally well-written and constructive reviews and who have actively engaged themselves in the post-rebuttal discussions.

We are also deeply indebted to the best paper selection committee which consists of 22 members. They had to additionally review 6-8 papers according to the best paper criteria on short notice. Their time and effort in recommending the best paper awards is much appreciated.

We also would like to thank many colleagues for generously sharing their experience in organizing prior ACL conferences and for their advice. We are grateful for the guidance and the support of the ACL presidents Joakim Nivre and Marti Hearst, and the ACL board. We also would like to thank the publication co-chairs Shay Cohen, Kevin Gimpel and Wei Lu (Advisory) and the handbook chair Jey Han Lau for putting together the proceedings and the conference handbook; and Rich Gerber from Softconf for always being responsive to our requests. We would like to thank the ACL Business Manager Priscilla Rasmussen for helping us to sort important things out. Finally, this conference could not have happened without the efforts of the general chair, Claire Cardie. We thank her for the leadership and advice, especially when matters got complicated.

We hope you will enjoy ACL 2018 and contribute to the future success of our community!

ACL 2018 Program Committee Co-Chairs
Iryna Gurevych, TU Darmstadt, Germany
Yusuke Miyao, National Institute of Informatics, Japan

The process for selecting best papers and honourable mentions

The Program Committee Co-Chairs (PCs) have defined a multi-step process. Area Chairs (ACs) were asked to select a number of top papers in their areas satisfying as many as possible of the following criteria:

- high quality
- nominated for the award by at least one primary reviewer
- bringing disruptive ground-breaking innovation as compared to the current mainstream

ACs re-read their finalists and discussed among themselves the merits of the nominee's work with the help of the primary reviews. ACs then submitted the papers to the PCs along with their selection decisions. PCs balanced ACs' nominations for diversity and representativeness among areas and the review consistency. They prepared the papers in Softconf for best-paper reviewing and selection. There were 52 best paper candidates.

In parallel, PCs formed the best paper selection committee (BPC) from 22 experts in the field with a mix of expertise and backgrounds and at a good seniority level. In case of COIs, the BPC member was excluded from the further evaluation process. BPC members reviewed 6-8 papers each and provided a short review with respect to the best paper criteria.

Based on BPC recommendations, there were about 20 papers left in the pool. PCs then re-read those papers and discussed their particular merits. Finally, 6 long papers and 2 short papers were selected as honourable mentions. For the best papers, 3 long papers and 2 short papers were selected for presentation in the closing conference session.

The selected honourable mentions and best papers emphasize the diversity of the ACL in terms of research questions, methods, and interdisciplinarity.

Best Long Papers

- *Finding syntax in human encephalography with beam search.* John Hale, Chris Dyer, Adhiguna Kuncoro and Jonathan Brennan.
- *Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information.* Sudha Rao and Hal Daumé III.
- *Let's do it "again": A First Computational Approach to Detecting Adverbial Presupposition Triggers.* Andre Cianflone, Yulan Feng, Jad Kabbara and Jackie Chi Kit Cheung.

Best Short Papers

- *Know What You Don't Know: Unanswerable Questions for SQuAD.* Pranav Rajpurkar, Robin Jia and Percy Liang.
- *'Lighter' Can Still Be Dark: Modeling Comparative Color Descriptions.* Olivia Winn and Smaranda Muresan.

Invited Talk: Deep Neural Networks, and what they're not very good at

Anton van den Hengel
Professor, School of Computer Science, University of Adelaide

Abstract: Deep Neural Networks have had an incredible impact in a variety of areas within machine learning, including computer vision and natural language processing. Deep Neural Networks use implicit representations that are very high-dimensional, however, and are thus particularly well suited to problems that can be solved by associative recall of previous solutions. They are ill-suited to problems that require human-interpretable representations, explicit manipulation of symbols, or reasoning. The dependency of Deep Neural Networks on large volumes of training data, also means that they are typically only applicable when the problem itself, and the nature of the test data, are predictable long in advance.

The application of Deep Neural Networks to Visual Question Answering has achieved results that would have been thought impossible only a few years ago. It has also thrown a spotlight on the shortcomings of current Deep Nets in solving problems that require explicit reasoning, the use of a knowledge base, or the ability to learn on the fly. In this talk I will illustrate some of the steps being taken to address these problems, and a new learning-to-learn approach that we hope will combine the power of Deep Learning with the significant benefits of explicit-reasoning-based methods.

Bio: Anton van den Hengel is a Professor in the School of Computer Science at the University of Adelaide, the Director of the Australian Institute for Machine Learning, and a Chief Investigator of the Australian Centre for Robotic Vision. Prof. van den Hengel has been a CI on over $60m in external research funding from sources including Google, Canon, BHP Billiton and the ARC, and has won a number of awards, including the Pearcey Foundation Entrepreneur Award, the SA Science Excellence Award for Research Collaboration, and the CVPR Best Paper prize in 2010. He has authored over 300 publications, had 8 patents commercialised, formed 2 start-ups, and has recently had a medical technology achieve first-in-class FDA approval. Current research interests include Deep Learning, vison and language problems, interactive image-based modelling, large-scale video surveillance, and learning from large image databases.

Invited Talk: Who is the Bridge Between the What and the How

Carolyn Penstein Rosé
Professor, School of Computer Science, Carnegie Mellon University

Abstract: This talk reports on over a decade of research where theoretical foundations motivate computational models that produce real world impact in online spaces. Both the earliest philosophers of language and the most recent researchers in computational approaches to social media analysis have acknowledged the distinction between the what of language, namely its propositional content, and the how of language, or its form, style, or framing. What bridges between these realms are social processes that motivate the linguistic choices that result in specific realizations of propositional content situated within social interactions, designed to achieve social goals. These insights allow researchers to make sense of the connection between discussion processes and outcomes from those discussions. These findings motivate on the one hand design of computational approaches to real time monitoring of discussion processes and on the other hand the design of interventions that support interactions in online spaces with the goal of increasing desired outcomes, including learning, health, and wellbeing.

As an example, in this talk we probe into a specific quality of discussion referred to as Transactivity. Transactivity is the extent to which a contribution articulates the reasoning of the speaker, that of an interlocutor, and the relation between them. In different contexts, and within very distinct theoretical frameworks, this construct has been associated with solidarity, influence, expertise transfer, and learning. Within the construct of Transactivity, the cognitive and social underpinnings are inextricably linked such that modeling the who enables prediction of the connection between the what and the how.

Bio: Dr. Carolyn Rosé is a Professor of Language Technologies and Human-Computer Interaction in the School of Computer Science at Carnegie Mellon University. Her research program is focused on better understanding the social and pragmatic nature of conversation, and using this understanding to build computational systems that can improve the efficacy of conversation between people, and between people and computers. In order to pursue these goals, she invokes approaches from computational discourse analysis and text mining, conversational agents, and computer supported collaborative learning. Her research group's highly interdisciplinary work, published in 200 peer reviewed publications, is represented in the top venues in 5 fields: namely, Language Technologies, Learning Sciences, Cognitive Science, Educational Technology, and Human-Computer Interaction, with awards in 3 of these fields. She serves as Past President and Inaugural Fellow of the International Society of the Learning Sciences, Chair of the International Alliance to Advance Learning in the Digital Era, and Executive Editor of the International Journal of Computer-Supported Collaborative Learning.

Organising Committee

General Chair:

Claire Cardie, Cornell University

Program Chairs:

Iryna Gurevych, TU Darmstadt
Yusuke Miyao, National Institute of Informatics

Workshop Chairs:

Brendan O'Connor, University of Massachusetts Amherst
Eva Maria Vecchi, University of Cambridge

Tutorial Chairs:

Yoav Artzi, Cornell University
Jacob Eisenstein, Georgia Institute of Technology

Demo Chairs:

Fei Liu, University of Central Florida
Thamar Solorio, University of Houston

Publications Chairs:

Shay Cohen, University of Edinburgh
Kevin Gimpel, Toyota Technological Institute at Chicago
Wei Lu, Singapore University of Technology and Design (Advisory)

Exhibits Coordinator:

Karin Verspoor, University of Melbourne

Conference Handbook Chairs:

Jey Han Lau, IBM Research
Trevor Cohn, University of Melbourne

Publicity Chair:

Sarvnaz Karimi, CSIRO

Local Sponsorship Chair:

Cecile Paris, CSIRO

Local Chairs:

Tim Baldwin, University of Melbourne
Karin Verspoor, University of Melbourne
Trevor Cohn, University of Melbourne

Program Committee

Program Committee Co-Chairs

Iryna Gurevych, TU Darmstadt, Germany
Yusuke Miyao, National Institute of Informatics, Japan

Program Committee Co-Chair Assistants

Yang Gao, TU Darmstadt, Germany
Ivan Habernal, TU Darmstadt, Germany
Sang Phan, National Institute of Informatics, Japan
Steffen Eger, TU Darmstadt, Germany
Christian Meyer, TU Darmstadt, Germany

Area Chairs

Senior Area Chairs are indicated in boldface.

Dialogue and Interactive Systems

Asli Celikyilmaz
Verena Rieser
Milica Gasic
Jason Williams

Discourse and Pragmatics

Manfred Stede
Ani Nenkova

Document Analysis

Hang Li
Yiqun Liu
Eugene Agichtein

Generation

Ioannis Konstas
Claire Gardent

Information Extraction and Text Mining

Feiyu Xu
Kevin Cohen
Zhiyuan Liu
Ralph Grishman
Yi Yang
Nazli Goharian

Linguistic Theories, Cognitive Modeling and Psycholinguistics

Shuly Wintner
Tim O’Donnell

Machine Learning

Andre Martins
Ariadna Quattoni
Jun Suzuki

Machine Translation

Yang Liu (Tsinghua University)
Matt Post
Lucia Specia
Dekai Wu

Multidisciplinary and Area Chair COI

Yoav Goldberg
Anders Søgaard
Mirella Lapata

Multilinguality

Bernardo Magnini
Tristan Miller

Phonology, Morphology and Word Segmentation

Graham Neubig
Hai Zhao

Question Answering

Lluís Màrquez
Teruko Mitamura
Zornitsa Kozareva
Richard Socher

Resources and Evaluation

Gerard de Melo
Sara Tonelli
Karën Fort

Sentence-level Semantics

Luke Zettlemoyer
Ellie Pavlick
Jacob Uszkoreit

Sentiment Analysis and Argument Mining

Smaranda Muresan
Benno Stein
Yulan He

Social Media

David Jurgens

Jing Jiang

Summarization

Kathleen McKeown
Xiaodan Zhu

Tagging, Chunking, Syntax and Parsing

Liang Huang
Weiwei Sun
Željko Agić
Yue Zhang

Textual Inference and Other Areas of Semantics

Michael Roth
Fabio Massimo Zanzotto

Vision, Robotics, Multimodal, Grounding and Speech

Yoav Artzi
Shinji Watanabe
Timothy Hospedales

Word-level Semantics

Ekaterina Shutova
Roberto Navigli

Best Paper Selection Committee

Timothy Baldwin
Pushpak Bhattacharyya
Phil Blunsom
Johan Bos
Jordan Boyd-Graber
Trevor Cohn
Vera Demberg
Kevin Duh
Katrin Erk
Mark Johnson
Yang Liu (Tsinghua University)
Yuji Matsumoto
Jong Park
Ellie Pavlick
Simone Paolo Ponzetto
Sebastian Riedel
Carolyn Penstein Rosé
Noah A. Smith
Anders Søgaard
Ivan Titov
Benjamin Van Durme
Ming Zhou

Primary Reviewers

Outstanding reviewers are indicated in boldface.

Mourad Abbas, Ahmed Abdelali, Asad Abdi, **Muhammad Abdul-Mageed**, **Omri Abend**, Gilles Adda, Heike Adel, Stergos Afantenos, Apoorv Agarwal, Arvind Agarwal, Rodrigo Agerri, Roee Aharoni, Murat Akbacak, Mohammad Akbari, Alan Akbik, Ahmet Aker, Elif Aktolga, Mohamed Al-Badrashiny, Firoj Alam, Chris Alberti, Hanan Aldarmaki, **Enrique Alfonseca**, David Alfter, Afra Alishahi, Laura Alonso Alemany, Ashjan Alsulaimani, David Alvarez-Melis, Carlos Alzate, **Maxime Amblard**, Daniel Andrade, **Jacob Andreas**, Nicholas Andrews, Ion Androutsopoulos, Anietie Andy, Galia Angelova, Jean-Yves Antoine, **Marianna Apidianaki**, **Emilia Apostolova**, Jun Araki, Kenji Araki, Yuki Arase, Philip Arthur, Masayuki Asahara, Ramón Astudillo, John Atkinson, Giuseppe Attardi, Isabelle Augenstein, Eleftherios Avramidis, Amittai Axelrod, Wilker Aziz, AmirAli Bagher Zadeh, JinYeong Bak, Collin Baker, Simon Baker, Omid Bakhshandeh, Alexandra Balahur, Mithun Balakrishna, Vevake Balaraman, Niranjan Balasubramanian, Timothy Baldwin, Kalika Bali, Miguel Ballesteros, David Bamman, Rafael E. Banchs, Srinivas Bangalore, Mohit Bansal, Roy Bar-Haim, Libby Barak, Ken Barker, Marco Baroni, Alberto Barrón-Cedeño, Sabine Bartsch, Pierpaolo Basile, Valerio Basile, Roberto Basili, Daniel Bauer, Timo Baumann, **Rachel Bawden**, Frederic Bechet, Daniel Beck, Srikanta Bedathur, **Beata Beigman Klebanov**, Núria Bel, Yonatan Belinkov, Meriem Beloucif, Iz Beltagy, Anja Belz, Yassine Benajiba, **Jonathan Berant**, Raffaella Bernardi, Yevgeni Berzak, Laurent Besacier, Steven Bethard, Rahul Bhagat, Archna Bhatia, Sumit Bhatia, Pushpak Bhattacharyya, Ergun Biçici, Ann Bies, Lidong Bing, **Joachim Bingel**, **Or Biran**, Alexandra Birch, Arianna Bisazza, Yonatan Bisk, Johannes Bjerva, Anders Björkelund, Philippe Blache, Frédéric Blain, Eduardo Blanco, Michael Bloodgood, Bernd Bohnet, Ondřej Bojar, Danushka Bollegala, Daniele Bonadiman, Claire Bonial, Francesca Bonin, Kalina Bontcheva, Georgeta Bordea, **Benjamin Börschinger**, Johan Bos, Elizabeth Boschee, **Antoine Bosselut**, Fethi Bougares, Pierrette Bouillon, **Gerlof Bouma**, **Samuel Bowman**, Cem Bozsahin, David Bracewell, James Bradbury, S.R.K. Branavan, António Branco, Chloé Braud, Ted Briscoe, **Chris Brockett**, Julian Brooke, Thomas Brovelli (Meyer), Christian Buck, Paweł Budzianowski, Michael Bugert, Paul Buitelaar, Harry Bunt, Franck Burlot, Hendrik Buschmeier, Miriam Butt, Jan Buys, José G. C. de Souza, Deng Cai, Iacer Calixto, Chris Callison-Burch, Jose Camacho-Collados, **Erik Cambria**, Leonardo Campillos Llanos, B Rosario Campomanes-Alvarez, **Burcu Can**, Yuan Cao, Yunbo Cao, Giuseppe Carenini, Marine Carpuat, Xavier Carreras, John Carroll, Paula Carvalho, Vitor Carvalho, Francisco Casacuberta, Iñigo Casanueva, Helena Caseli, **Tommaso Caselli**, Taylor Cassidy, Giuseppe Castellucci, Daniel Cer, Christophe Cerisara, Özlem Çetinoğlu, Mauro Cettolo, Joyce Chai, Yllias Chali, **Nathanael Chambers**, Muthu Kumar Chandrasekaran, Akshay Chandrashekaran, **Angel Chang**, Baobao Chang, Chia-Hui Chang, Kai-Wei Chang, Yin-Wen Chang, Snigdha Chaturvedi, Wanxiang Che, **Ciprian Chelba**, Bin Chen, Boxing Chen, Chen Chen, Danqi Chen, Hsin-Hsi Chen, Qian Chen, Qingcai Chen, Wenliang Chen, Xinchi Chen, Yubo Chen, **Yun-Nung Chen**, Zhiyuan Chen, Fei Cheng, **Hao Cheng**, Jianpeng Cheng, Pu-Jen Cheng, Colin Cherry, Jackie Chi Kit Cheung, David Chiang, Christian Chiarcos, Jen-Tzung Chien, Hai Leong Chieu, Dokook Choe, Eunsol Choi, Yejin Choi, Kostadin Cholakov, Grzegorz Chrupała, Chenhui Chu, Tagyoung Chung, Kenneth Church, Mark Cieliebak, Philipp Cimiano, Volkan Cirik, Alexander Clark, Stephen Clark, Guillaume Cleuziou, Ann Clifton, **Maximin Coavoux**, **Anne Cocos**, **Arman Cohan**, Kevin Cohen, Shay B. Cohen, Michael Collins, John Conroy, Matthieu Constant, Danish Contractor, Paul Cook, Marta R. Costa-jussà, Ryan Cotterell, **Alain Couillault**, Benoit Crabbé, Danilo Croce, **Paul Crook**, James Cross, Montse Cuadros, Heriberto Cuayahuitl, Silviu-Petru Cucerzan, Lei Cui, Xiaodong Cui, Aron Culotta, Shane Culpepper, Anna Currey, Luis Fernando D'Haro, Giovanni Da San Martino, Walter Daelemans, Xin-Yu Dai, Beatrice Daille, Fahim Dalvi, Cristian Danescu-Niculescu-Mizil, Falavigna Daniele, Kareem Darwish, Abhishek Das, Amitava Das, Dipanjan Das, Tirthankar Dasgupta, Johannes Daxenberger, Gaël de Chalendar, Munmun De Choudhury, Adrià de Gispert, Daniël de Kok, Eric De La Clergerie, **Marie-Catherine de Marneffe**, Renato de Mori, Thierry Declerck, Luciano Del Corro, Louise Deléger, Felice Dell'Orletta, **Claudio Delli Bovi**, Rodolfo Delmonte, Vera Demberg, Dina Demner-Fushman, Lingjia Deng, Pascal Denis, Michael Denkowski, Tejaswini Deoskar, Valeria dePaiva, Nina Dethlefs, **Chris Develder**, Barry Devereux, Bhuwan Dhingra, Luigi Di Caro, Giuseppe Di Fabbrizio, Emanuele Di Rosa, Mona Diab, Gaël Dias, Jana Diesner, Shuoyang Ding, Georgiana Dinu, Stefanie Dipper, Nemanja Djuric, Dmitriy Dligach, Simon Dobnik, Jesse Dodge, A. Seza Doğruöz, Tobias Domhan, Daxiang Dong, **Li Dong**, Shichao Dong, Doug Downey, Mark Dras, Markus Dreyer, Patrick Drouin, Lan Du, Xinya Du, Nan Duan, Xiangyu Duan, Pablo Duboue, Shiran Dudy, **Kevin Duh**, Stefan Daniel Dumitrescu, **Greg Durrett**, **Ondřej Dušek**, Chris Dyer, Hiroshi Echizen'ya, **Judith Eckle-Kohler**, Markus Egg, Patrick Ehlen, Maud Ehrmann, Vladimir Eidelman, Andreas Eisele, **Jacob Eisenstein**, Layla El Asri, Shady Elbassuoni, Mohamed Eldesouki, Michael Elhadad, Desmond Elliott, **Guy Emerson**, Erkut Erdem, **Mihail Eric**, Tomaž Erjavec, **Katrin Erk**, Arash Eshghi, Ramy Eskander, Cristina España-Bonet, Luis Espinosa Anke, Miquel Esplà-Gomis, Kilian Evang, **Ingrid Falk**, Hao Fang, Hui Fang, Licheng Fang, Meng Fang, Benamara Farah, Stefano Faralli, Richárd Farkas, Maryam Fazel-Zarandi, Afsaneh Fazly, Marcello Federico, Christian Federmann, Anna Feldman, Yansong Feng, Olivier Ferret, Alejandro Figueroa, Elena Filatova, Simone Filice, Denis Filimonov, Katja Filippova, Andrew Finch, Mark Finlayson, Orhan Firat, Mark Fishel, Nicholas FitzGerald, Jeffrey Flanigan, **Margaret Fleck**, Lucie Flekova, Radu Florian, Antske Fokkens, Eric Fosler-Lussier, **George Foster**, **James Foulds**, Marc Franco-Salvador, Thomas Francois, Stella Frank, Dayne Freitag, Markus Freitag, André Freitas, **Lea Frermann**, **Daniel Fried**, Annemarie Friedrich, Hagen Fuerstenau, Akinori Fujino, Fumiyo Fukumoto, Sadaoki Furui, Robert Gaizauskas, Olivier Galibert, Irina Galinskaya, Matthias Gallé, Michel Galley, Kuzman Ganchev, Octavian-Eugen Ganea, Bin Gao, Jianfeng Gao, **Wei Gao**, Claire Gardent, **Dan Garrette**, **Albert Gatt**, Eric Gaussier, Tao Ge, **Lieke Gelderloos**, Spandana Gella, Kallirroi Georgila, Pablo Gervás, Marjan Ghazvininejad, Debanjan Ghosh, Sucheta Ghosh, Daniela Gifu, Daniel Gildea, C Lee Giles, Kevin Gimpel, Roxana Girju, **Dimitra Gkatzia**, **Goran Glavaš**, Dan Goldwasser, **Carlos Gómez-Rodríguez**, Hugo Gonçalo Oliveira, Jibing Gong, Yeyun Gong, Graciela Gonzalez-Hernandez, Jesús González-Rubio, Kyle Gorman, Matthew R. Gormley, **Cyril Goutte**, Kartik Goyal, Natalia Grabar, Joao Graca, Jorge Gracia, Yvette Graham, Roger Granada, **Christophe Gravier**, **Gregory Grefenstette**, Cyril Grouin, Marco Guerini, Anupam Guha, Lin Gui, Bruno Guillaume, Curry Guinn, **Kristina Gulordava**, Weiwei Guo, Yuhong Guo, **Arpit Gupta**, Joakim Gustafson, **E. Dario Gutierrez**, Francisco Guzmán, Jeremy Gwinnup, Maryam Habibi, Ben Hachey, Christian Hadiwinoto, Gholamreza Haffari, Udo Hahn, Hannaneh Hajishirzi, Dilek Hakkani-Tur, Keith Hall, **William L. Hamilton**, Michael Hammond, Thierry Hamon, Ting Han, Xianpei Han, Sanda Harabagiu, Christian Hardmeier, Orin Hargraves, Sadid A. Hasan, Kazuma Hashimoto, Eva Hasler, Ahmed Hassan Awadallah, Hany Hassan, Helen Hastie, Katsuhiko Hayashi, He He, Hua He, **Luheng He**, Shizhu He, Xiangnan He, Kenneth Heafield, James Henderson, Matthew Henderson, **Lisa Anne Hendricks**, **Aurélie Herbelot**, Jonathan Herzig, **Jack Hessel**, **John Hewitt**, Felix Hieber, Ryuichiro Higashinaka, Tsutomu Hirao, **Julia Hirschberg**, **Graeme Hirst**, Julian Hitschler, Lydia-Mai Ho-Dac, Vu Cong Duy Hoang, Johannes Hoffart, Chris Hokamp, Chester Holtz, **Ari Holtzman**, Yu Hong, Chiori Hori, Takaaki Hori, Ehsan Hosseini-Asl, Yufang Hou, Julian Hough, Dirk Hovy, Eduard Hovy, **David M. Howcroft**, Renfen Hu, Yuheng Hu, Zhiting Hu, Fei Huang, Hen-Hsen Huang, Lifu Huang, Minlie Huang, Po-Sen Huang, Ruihong Huang, Shujian Huang, Xuanjing Huang, Matthias Huck, Mans Hulden, **Tim Hunter**, Jena D. Hwang, Seung-won Hwang, Ignacio Iacobacci, Nancy Ide, Marco Idiart, Adrian Iftene, Ryu Iida, Naoya Inoue, Kentaro Inui, **Srinivasan Iyer**, Mohit Iyyer, Tommi Jaakkola, **Cassandra L. Jacobs**, **Aaron Jaech**, Kokil Jaidka, **Laura Jehl**, **Yacine Jernite**, Rahul Jha, Donghong Ji, Yangfeng Ji, Robin Jia, Sittichai Jiampojamarn, Hui Jiang, Antonio Jimeno Yepes, Hongyan Jing, **Charles Jochim**, Richard Johansson, Kristiina Jokinen, Gareth Jones, **Aditya Joshi**, Shafiq Joty, Marcin Junczys-Dowmunt, Ákos Kádár, Sylvain Kahane, **Hetunandan Kamisetty**, **Herman Kamper**, Jaap Kamps, Min-Yen Kan, Hiroshi Kanayama, **Dongyeop Kang**, Katharina Kann, Diptesh Kanojia, Dimitri Kartsaklis, Makoto P. Kato, David Kauchak, Daisuke Kawahara, Tatsuya Kawahara, Anna Kazantseva, **Hideto Kazawa**, Chris Kedzie, **Andrew Kehler**, Simon Keizer, Ruth Kempson, Casey Kennington, Fabio Kepler, Nitish Shirish Keskar, Mitesh M. Khapra, Huda Khayrallah, **Douwe Kiela**, Yuta Kikuchi, **Halil Kilicoglu**, Jin-Dong Kim, Sun Kim, **Yoon Kim**, Young-Bum Kim, **Brian Kingsbury**, Eliyahu Kiperwasser, Svetlana Kiritchenko, Chunyu Kit, Dietrich Klakow, Alexandre Klementiev, Manfred Klenner, Sigrid Klerke, **Ro-**

man Klinger, Julien Kloetzer, Kevin Knight, Sosuke Kobayashi, Thomas Kober, **Ekaterina Kochmar**, Philipp Koehn, **Alexander Koller**, Kazunori Komatani, **Rik Koncel-Kedziorski**, **Grzegorz Kondrak**, Lingpeng Kong, **Maximilian Köper**, Stefan Kopp, Selcuk Kopru, Parisa Kordjamshidi, Valia Kordoni, Anna Korhonen, Yannis Korkontzelos, Leila Kosseim, Katsunori Kotani, Lili Kotlerman, Emiel Krahmer, Martin Krallinger, Julia Kreutzer, **Jayant Krishnamurthy**, Kriste Krstovski, Canasai Kruengkrai, **Ivana Kruijff-Korbayova**, **Germán Kruszewski**, Lun-Wei Ku, Marco Kuhlmann, Roland Kuhn, Ashish Kulkarni, Gaurav Kumar, **Shankar Kumar**, Anil Kumar Singh, **Jonathan K. Kummerfeld**, Adhiguna Kuncoro, Souvik Kundu, Gakuto Kurata, Sadao Kurohashi, Nate Kushman, Andrey Kutuzov, Polina Kuznetsova, Majid Laali, Gorka Labaka, Anirban Laha, Mathias Lambert, Vasileios Lampos, **Gerasimos Lampouras**, Ian Lane, Ni Lao, Mirella Lapata, Shalom Lappin, Romain Laroche, Kornel Laskowski, Alberto Lavelli, Alon Lavie, Phong Le, **Joseph Le Roux**, Robert Leaman, Gianluca Lebani, **Chia-ying Lee**, Hung-yi Lee, John Lee, Kenton Lee, Lin-shan Lee, Moontae Lee, Sungjin Lee, Yoong Keok Lee, **Young-Suk Lee**, **Els Lefever**, Fabrice Lefevre, Wenqiang Lei, Jochen L. Leidner, Gaël Lejeune, Alessandro Lenci, Piroska Lendvai, Yves Lepage, Johannes Leveling, Tomer Levinboim, Omer Levy, **Roger Levy**, Mike Lewis, Baoli Li, Cheng-Te Li, Chenliang Li, Chunyuan Li, Fangtao Li, Haizhou Li, Jing Li, Juanzi Li, Junhui Li, Junyi Jessy Li, Piji Li, Qing Li, **Shaohua Li**, Sheng Li, Shoushan Li, Sujian Li, Wenjie Li, Xiangsheng Li, Xiaoli Li, **Xiujun Li**, Yanran Li, Zhenghua Li, Zhixing Li, Zichao Li, **Maria Liakata**, Chen Liang, **Timm Lichte**, Anne-Laure Ligozat, Nut Limsopatham, **Chenghua Lin**, Chin-Yew Lin, Jimmy Lin, Shou-de Lin, Xi Victoria Lin, Xiao Ling, **Tal Linzen**, Zachary Lipton, Pierre Lison, Diane Litman, Marina Litvak, Bing Liu, Changsong Liu, Fei Liu, Jing Liu, Kang Liu, Lemao Liu, Qian Liu, Qiaoling Liu, Qun Liu, Shujie Liu, Ting Liu, Xiaobing Liu, Yang Liu (University of Texas at Dallas), Yang Liu (University of Edinburgh), Yang Liu (Tsinghua University), Yijia Liu, Zhiyuan Liu, Elena Lloret, **Sharid Loáiciga**, José Lopes, Lucelene Lopes, Marcos Lopes, Oier Lopez de Lacalle, Aurelio Lopez-Lopez, Annie Louis, Samuel Louvan, Bin Lu, Qin Lu, **Wei Lu**, Xiaofei Lu, Yanbin Lu, Zhiyong Lu, Yi Luan, **Andy Luecking**, **Stephanie M. Lukin**, Cheng Luo, Jiaming Luo, Xiaoqiang Luo, Zhunchen Luo, Franco M. Luque, Anh Tuan Luu, Teresa Lynn, Ji Ma, Jianqiang Ma, Mingbo Ma, Xuezhe Ma, Wolfgang Macherey, Pranava Swaroop Madhyastha, Nitin Madnani, Pierre Magistry, Simone Magnolini, **Wolfgang Maier**, **Jean Maillard**, Prodromos Malakasiotis, Alfredo Maldonado, Andreas Maletti, Igor Malioutov, Radhika Mamidi, Suresh Manandhar, Gideon Mann, Christopher D. Manning, Diego Marcheggiani, Daniel Marcu, David Mareček, **Matthew Marge**, Alex Marin, Erwin Marsi, Scott Martin, Héctor Martínez Alonso, Bruno Martins, David Martins de Matos, Yuval Marton, **Yann Mathet**, Prashant Mathur, Shigeki Matsubara, Yuji Matsumoto, Yoichi Matsuyama, Takuya Matsuzaki, Austin Matthews, Jonathan May, Karen Mazidi, David McAllester, Diana McCarthy, **R. Thomas Mccoy**, **John Philip McCrae**, Stephen McGregor, Tara McIntosh, Louise McNally, Yashar Mehdad, Sameep Mehta, **Hongyuan Mei**, Oren Melamud, Nurit Melnik, Arul Menezes, Fandong Meng, Slim Mesfar, **Florian Metze**, Haitao Mi, Yishu Miao, Antonio Valerio Miceli Barone, Claudiu Mihăilă, Todor Mihaylov, Elena Mikhalkova, Timothy Miller, Tristan Miller, Bonan Min, **Anne-Lyse Minard**, Michael Minock, Toben Mintz, Shachar Mirkin, Seyed Abolghasem Mirroshandel, Paramita Mirza, **Abhijit Mishra**, Dipendra Misra, Prasenjit Mitra, Vikramjit Mitra, Arpit Mittal, Makoto Miwa, Junta Mizuno, Daichi Mochihashi, Ashutosh Modi, Marie-Francine Moens, Samaneh Moghaddam, Abdelrahman Mohamed, **Saif Mohammad**, Behrang Mohit, Michael Mohler, Mitra Mohtarami, Karo Moilanen, Luis Gerardo Mojica de la Vega, Helena Moniz, Johanna Monti, **Nafise Sadat Moosavi**, Roser Morante, Erwan Moreau, Shinsuke Mori, Hajime Morita, Andrea Moro, David R. Mortensen, Alessandro Moschitti, Nasrin Mostafazadeh, Lili Mou, Diego Moussallem, Nikola Mrkšić, Arjun Mukherjee, Tanmoy Mukherjee, Philippe Muller, Yugo Murawaki, Gabriel Murray, Kenton Murray, Elena Musi, Sung-Hyon Myaeng, Maria Nadejde, Masaaki Nagata, Ajay Nagesh, Tetsuji Nakagawa, Yukiko Nakano, Ndapa Nakashole, Preslav Nakov, Courtney Napoles, Jason Naradowsky, Karthik Narasimhan, **Shashi Narayan**, Shrikanth Narayanan, Masha Naslidnyk, Alexis Nasr, Vivi Nastase, Borja Navarro-Colorado, Adeline Nazarenko, Arvind Neelakantan, Matteo Negri, Aida Nematzadeh, Guenter Neumann, Aurélie Névéol, Denis Newman-Griffis, **Dominick Ng**, Hwee Tou Ng, Jun-Ping Ng, Vincent Ng, Dong Nguyen, Thien Huu Nguyen, Truc-Vien T. Nguyen, Viet-An Nguyen, Garrett Nicolai, Massimo Nicosia, Vlad

Niculae, **Jian-Yun Nie**, Jan Niehues, Ivelina Nikolova, Kristina Nilsson Björkenstam, **Sergiu Nisioi**, **Joakim Nivre**, Cicero Nogueira dos Santos, Hiroshi Noji, Joel Nothman, Damien Nouvel, **Diarmuid Ó Séaghdha**, Stephan Oepen, Kemal Oflazer, Alice Oh, Jong-Hoon Oh, Kiyonori Ohtake, Hidekazu Oiwa, Naoaki Okazaki, Hiroshi G. Okuno, Shereen Oraby, Constantin Orasan, Vicente Ordonez, Petya Osenova, Simon Ostermann, Robert Östling, Myle Ott, Jessica Ouyang, Katja Ovchinnikova, Arzucan Özgür, Muntsa Padró, **Alexis Palmer**, Martha Palmer, Alessio Palmero Aprosio, Joao Palotti, Boyuan Pan, Sinno Jialin Pan, Xiaoman Pan, Alexander Panchenko, Alexandros Papangelis, Denis Paperno, Ankur Parikh, Cecile Paris, **Seong-Bae Park**, Yannick Parmentier, Patrick Paroubek, **Carla Parra Escartín**, Tommaso Pasini, Peyman Passban, **Panupong Pasupat**, Siddharth Patwardhan, Michael J. Paul, Adam Pauls, Romain Paulus, Adam Pease, Pavel Pecina, Bolette Pedersen, Anselmo Peñas, Baolin Peng, Hao Peng, Haoruo Peng, Nanyun Peng, Xiaochang Peng, Laura Perez-Beltrachini, **Isaac Persing**, Casper Petersen, Fabio Petroni, Slav Petrov, Miriam R L Petruck, Hieu Pham, Nghia The Pham, Karl Pichotta, Roberto Pieraccini, Olivier Pietquin, **Mohammad Taher Pilehvar**, Juan Pino, Yuval Pinter, Emily Pitler, Paul Piwek, Barbara Plank, Massimo Poesio, Thierry Poibeau, Heather Pon-Barry, Edoardo Maria Ponti, Simone Paolo Ponzetto, Andrei Popescu-Belis, Maja Popović, Fred Popowich, François Portet, **Christopher Potts**, Vinodkumar Prabhakaran, Daniel Preoţiuc-Pietro, Laurent Prévot, Emily Prud'hommeaux, Matthew Purver, James Pustejovsky, Valentina Pyatkin, **Ashequl Qadir**, Vahed Qazvinian, Peng Qi, Longhua Qian, Lianhui Qin, Long Qiu, Minghui Qiu, Xipeng Qiu, Lizhen Qu, Uwe Quasthoff, Chris Quirk, **Alessandro Raganato**, Altaf Rahman, Jonathan Raiman, Taraka Rama, Vikram Ramanarayanan, Maya Ramanath, Rohan Ramanath, Carlos Ramisch, Jinfeng Rao, Ari Rappoport, Mohammad Sadegh Rasooli, Pushpendre Rastogi, Soumya Ray, Manny Rayner, Simon Razniewski, Livy Real, Siva Reddy, Chris Reed, **Ines Rehbein**, Georg Rehm, Marek Rei, Roi Reichart, Julia Reinspach, Ehud Reiter, **Steffen Remus**, Xiang Ren, Zhaochun Ren, Christian Retore, Corentin Ribeyre, Kyle Richardson, Matthew Richardson, Mark Riedl, Martin Riedl, Jason Riesa, Stefan Riezler, German Rigau, Ellen Riloff, **Laura Rimell**, Fabio Rinaldi, Alan Ritter, Kirk Roberts, Molly Roberts, Ina Roesiger, Marcus Rohrbach, Oleg Rokhlenko, Laurent Romary, Salvatore Romeo, Carolyn Penstein Rosé, Andrew Rosenberg, **Sara Rosenthal**, Sophie Rosset, **Paolo Rosso**, Benjamin Roth, **Michael Roth**, Sascha Rothe, Johann Roturier, Jacobo Rouces, Subhro Roy, Alla Rozovskaya, Raphael Rubino, Sebastian Ruder, **Rachel Rudinger**, Maja Rudolph, Pablo Ruiz, Anna Rumshisky, Josef Ruppenhofer, Vasile Rus, Irene Russo, Delia Rusu, Attapol Rutherford, Mrinmaya Sachan, Kugatsu Sadamitsu, Markus Saers, Kenji Sagae, Benoît Sagot, Saurav Sahay, Cem Sahin, Magnus Sahlgren, Patrick Saint-Dizier, Hassan Sajjad, Keisuke Sakaguchi, Sakriani Sakti, Mohammad Salameh, Elizabeth Salesky, Avneesh Saluja, Rajhans Samdani, Mark Sammons, Germán Sanchis-Trilles, **Marina Santini**, Ruhi Sarikaya, Kamal Sarkar, **Agata Savary**, Asad Sayeed, Carolina Scarton, **Tatjana Scheffler**, **Christian Scheible**, Niko Schenk, Yves Scherrer, Frank Schilder, David Schlangen, Natalie Schluter, Allen Schmaltz, Helmut Schmid, Alexandra Schofield, William Schuler, Björn Schuller, Sabine Schulte im Walde, Claudia Schulz, Sebastian Schuster, Didier Schwab, H. Andrew Schwartz, Lane Schwartz, Roy Schwartz, Donia Scott, Djamé Seddah, Joao Sedoc, Abigail See, Frederique Segond, Satoshi Sekine, Ethan Selfridge, Mark Seligman, Jean Senellart, Rico Sennrich, Minjoon Seo, Christophe Servan, Burr Settles, Armin Seyeditabari, Lei Sha, Izhak Shafran, Kashif Shah, Pararth Shah, Samira Shaikh, Igor Shalyminov, Lifeng Shang, Yan Shao, Ehsan Shareghi, Rebecca Sharp, Lanbo She, Shiqi Shen, Shuming Shi, Hiroyuki Shindo, Koichi Shinoda, Eyal Shnarch, **Vered Shwartz**, Advaith Siddharthan, Avi Sil, Carina Silberer, Max Silberztein, Sameer Singh, Kairit Sirts, Gabriel Skantze, **Noam Slonim**, Kevin Small, **Noah A. Smith**, Jan Šnajder, Richard Socher, Artem Sokolov, **Luca Soldaini**, **Swapna Somasundaran**, Wei Song, Yangqiu Song, Radu Soricut, Aitor Soroa, **Daniil Sorokin**, René Speck, Jennifer Spenader, **Manuela Speranza**, Matthias Sperber, Caroline Sporleder, **Vivek Srikumar**, Somayajulu Sripada, Shashank Srivastava, **Christian Stab**, Sanja Štajner, **Gabriel Stanovsky**, Mark Steedman, Pontus Stenetorp, **Mitchell Stern**, Mark Stevenson, Brandon Stewart, Sebastian Stober, Matthew Stone, Svetlana Stoyanchev, Veselin Stoyanov, Karl Stratos, **Kristina Striegnitz**, **Jannik Strötgen**, **Emma Strubell**, Tomek Strzalkowski, Sara Stymne, Hui Su, Jinsong Su, Keh-Yih Su, Yu Su, David Suendermann-Oeft, Alane Suhr, Ang Sun, Huan Sun, Le Sun, Lin Sun, Weiwei Sun, Xu Sun, Hanna Suominen, Mihai Surdeanu, Simon Suster, Hisami Suzuki, **Swabha**

Swayamdipta, **John Sylak-Glassman**, Stan Szpakowicz, Idan Szpektor, Oscar Täckström, Prasad Tadepalli, Kaveh Taghipour, Nina Tahmasebi, Sho Takase, David Talbot, Aleš Tamchyna, **Akihiro Tamura**, Chenhao Tan, Liling Tan, Niket Tandon, Duyu Tang, Jiliang Tang, Jintao Tang, **Chris Tanner**, Fangbo Tao, Marta Tatu, Christoph Teichmann, Serra Sinem Tekiroglu, Irina Temnikova, Joel Tetreault, Simone Teufel, Kapil Thadani, Stefan Thater, Jesse Thomason, Sam Thomson, Fei Tian, Ran Tian, Jörg Tiedemann, Ivan Titov, Takenobu Tokunaga, Katrin Tomanek, Gaurav Singh Tomar, Marc Tomlinson, Antonio Toral, Kentaro Torisawa, Isabel Trancoso, David Traum, Rocco Tripodi, Alexander Trott, Chen-Tse Tsai, Ming-Feng Tsai, Reut Tsarfaty, Yuta Tsuboi, Oren Tsur, Yoshimasa Tsuruoka, Yulia Tsvetkov, Kewei Tu, Zhaopeng Tu, Dan Tufis, Don Tuggener, Gokhan Tur, Marco Turchi, Ferhan Ture, Francis Tyers, Kateryna Tymoshenko, Kiyotaka Uchimoto, Stefan Ultes, **Shyam Upadhyay**, Olga Uryupina, Dmitry Ustalov, Masao Utiyama, Alessandro Valitutti, Tim Van de Cruys, **Benjamin Van Durme**, Paul Van Eecke, **Menno van Zaanen**, Vincent Vandeghinste, Clara Vania, Tony Veale, Mihaela Vela, Paola Velardi, Sumithra Velupillai, Subhashini Venugopalan, Patrick Verga, Marc Verhagen, Rakesh Verma, Karin Verspoor, **Tim Vieira**, David Vilar, **David Vilares**, **Serena Villata**, Aline Villavicencio, Sami Virpioja, **Andreas Vlachos**, Svitlana Volkova, Elena Volodina, Ngoc Thang Vu, **Thuy Vu**, **Ivan Vulić**, V.G.Vinod Vydiswaran, Ekaterina Vylomova, **Henning Wachsmuth**, Joachim Wagner, **Byron Wallace**, Matthew Walter, Xiaojun Wan, Baoxun Wang, Chenguang Wang, Di Wang, Hai Wang, Hongning Wang, Houfeng Wang, Hua Wang, Jin Wang, Josiah Wang, Linlin Wang, Lu Wang, Mingxuan Wang, Quan Wang, Shaojun Wang, Shuohang Wang, Suge Wang, Tong Wang, Wei Wang, William Yang Wang, Xiaolong Wang, Yu-Chun Wang, Zhichun Wang, Zhongqing Wang, **Zhuoran Wang**, Leo Wanner, Nigel Ward, Zeerak Waseem, Patrick Watrin, Bonnie Webber, Ingmar Weber, Julie Weeds, Furu Wei, Zhongyu Wei, Gerhard Weikum, Marion Weller-Di Marco, **Tsung-Hsien Wen**, Michael White, Antoine Widlöcher, **Michael Wiegand**, **John Wieting**, Derry Tanti Wijaya, Adina Williams, Philip Williams, Sam Wiseman, Marcin Woliński, Derek F. Wong, Kam-Fai Wong, Tak-Lam Wong, Bowen Wu, Chun-Kai Wu, Wei Wu, Yu Wu, Zhaohui Wu, Joern Wuebker, Yunqing Xia, Min Xiao, Tong Xiao, Xinyan Xiao, Yanghua Xiao, **Boyi Xie**, Pengtao Xie, Shasha Xie, Caiming Xiong, Chenyan Xiong, Deyi Xiong, Hua Xu, Ruifeng Xu, Wei Xu, Wenduan Xu, Nianwen Xue, Semih Yagcioglu, Yadollah Yaghoobzadeh, Mohamed Yahya, Takehiro Yamamoto, Rui Yan, Zhao Yan, Bishan Yang, Grace Hui Yang, **Jie Yang**, Liu Yang, Qian Yang, Weiwei Yang, Yaqin Yang, Zhilin Yang, Roman Yangarber, Helen Yannakoudakis, Andrew Yates, Mark Yatskar, Lana Yeganova, Meliha Yetisgen, Pengcheng Yin, Wenpeng Yin, Dani Yogatama, Naoki Yoshinaga, Koichiro Yoshino, Steve Young, Bei Yu, Dian Yu, Jianfei Yu, Kai Yu, Liang-Chih Yu, Mo Yu, Zhengtao Yu, Zhou Yu, **François Yvon**, Wlodek Zadrozny, Virginie Zampa, **Amir Zeldes**, Daojian Zeng, Xiaodong Zeng, Kalliopi Zervanou, Torsten Zesch, Deniz Zeyrek, Biao Zhang, Chao Zhang, Congle Zhang, Dongdong Zhang, Hu Zhang, Jiajun Zhang, Justine Zhang, Li Zhang, Meishan Zhang, Min Zhang, Qi Zhang, Sicong Zhang, Wei-Nan Zhang, **Xingxing Zhang**, Yongfeng Zhang, Bing Zhao, Dongyan Zhao, Hai Zhao, Jun Zhao, Sendong Zhao, Tiancheng Zhao, **Tianyu Zhao**, Tiejun Zhao, Wayne Xin Zhao, Zhou Zhao, Guoqing Zheng, Vincent W. Zheng, Victor Zhong, Deyu Zhou, Guodong Zhou, Xiaobing Zhou, Yingbo Zhou, Muhua Zhu, Xiaodan Zhu, Fuzhen Zhuang, Heike Zinsmeister, Ayah Zirikly, Chengqing Zong, Guido Zuccon, Willem Zuidema, Pierre Zweigenbaum.

Secondary Reviewers

Peter A. Rankel, Ranit Aharonov, Domagoj Alagić, Anusha Balakrishnan, Somnath Banerjee, Joost Bastings, Giannis Bekoulis, Gayatri Bhat, Yonatan Bilu, Filip Boltužić, Svetla Boytcheva, Dominique Brunato, Gaëtan Caillaut, Iacer Calixto, Thiago Castro Ferreira, Dhivya Chinnappa, Arman Cohan, Joana Correia, Sujatha Das Gollapalli, Tobias Daudert, Hugues de Mazancourt, Maksym Del, Thomas Demeester, Ying Ding, Zhang Dong, Seth Ebner, Roxanne El Baff, Akiko Eriguchi, Nawshad Farruque, Quentin Feltgen, Elisa Ferracane, Qiaozi Gao, Pepa Gencheva, Bilal Ghanem, Frederic Godin, Nitish Gupta, Shachi H Kumar, Mareike Hartmann, Zhengqiu He, Irazú Hernández, Geert Heyman, Parag Jain, Sarthak Jain, Jyun-Yu Jiang, Xisen Jin, Vidur Joshi, Meizhi Ju, Joo-Kyung Kim, Jooyeon Kim, Elena Kochkina, Bhushan Kotnis, Florian Kreyssig, Sachin Kumar, Ronja Laarmann-Quante, Patrick Lange, Jihwan Lee, Shasha Li, Xinyi Li, Chen Liang, Yuping Lin, Haochen Liu, Qiuzhi Liu, Tong Liu,

Table of Contents

Conference Program

Monday, July 16, 2018

9:00–10:00 ***Welcome Session & Presidential Address***

10:00–10:30 ***Coffee Break***

Session 1A: Word Semantics 1

10:30–10:55 *Probabilistic FastText for Multi-Sense Word Embeddings*
Ben Athiwaratkun, Andrew Wilson and Anima Anandkumar

10:55–11:20 *A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors*
Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart and Sanjeev Arora

11:20–11:45 *Unsupervised Learning of Distributional Relation Vectors*
Shoaib Jameel, Zied Bouraoui and Steven Schockaert

11:45–12:10 *Explicit Retrofitting of Distributional Word Vectors*
Goran Glavaš and Ivan Vulić

Session 1B: Machine Translation 1

10:30–10:55 *Unsupervised Neural Machine Translation with Weight Sharing*
Zhen Yang, Wei Chen, Feng Wang and Bo Xu

10:55–11:20 *Triangular Architecture for Rare Language Translation*
Shuo Ren, Wenhu Chen, Shujie Liu, Mu Li, Ming Zhou and Shuai Ma

11:20–11:45 *Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates*
Taku Kudo

11:45–12:10 *The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation*
Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu and Macduff Hughes

Monday, July 16, 2018 (continued)

Session 1E: Resource, Annotation

10:30–10:55	*Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20,000 English Words* Saif Mohammad
10:55–11:20	*Comprehensive Supersense Disambiguation of English Prepositions and Possessives* Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan and Omri Abend
11:20–11:45	*A Corpus with Multi-Level Annotations of Patients, Interventions and Outcomes to Support Language Processing for Medical Literature* Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain Marshall, Ani Nenkova and Byron Wallace
11:45–12:10	*Efficient Online Scalar Annotation with Bounded Support* Keisuke Sakaguchi and Benjamin Van Durme

Session 1F: Argument Mining

10:30–10:55	*Neural Argument Generation Augmented with Externally Retrieved Evidence* Xinyu Hua and Lu Wang
10:55–11:20	*A Stylometric Inquiry into Hyperpartisan and Fake News* Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff and Benno Stein
11:20–11:45	*Retrieval of the Best Counterargument without Prior Topic Knowledge* Henning Wachsmuth, Shahbaz Syed and Benno Stein

12:10–12:30 ***Short Break***

12:30–14:00 ***Poster Session 1A: Machine Learning***

LinkNBed: Multi-Graph Representation Learning with Entity Linkage
Rakshit Trivedi, Bunyamin Sisman, Xin Luna Dong, Christos Faloutsos, Jun Ma and Hongyuan Zha

Monday, July 16, 2018 (continued)

Monday, July 16, 2018 (continued)

Session 2E: Vision

14:00–14:25	*No Metrics Are Perfect: Adversarial Reward Learning for Visual Storytelling* Xin Wang, Wenhu Chen, Yuan-Fang Wang and William Yang Wang
14:25–14:50	*Bridging Languages through Images with Deep Partial Canonical Correlation Analysis* Guy Rotman, Ivan Vulić and Roi Reichart
14:50–15:15	*Illustrative Language Understanding: Large-Scale Visual Grounding with Image Search* Jamie Kiros, William Chan and Geoffrey Hinton
15:15–15:40	*What Action Causes This? Towards Naive Physical Action-Effect Prediction* Qiaozi Gao, Shaohua Yang, Joyce Chai and Lucy Vanderwende

Session 2F: Sentiment

14:00–14:25	*Transformation Networks for Target-Oriented Sentiment Classification* Xin Li, Lidong Bing, Wai Lam and Bei Shi
14:25–14:50	*Target-Sensitive Memory Networks for Aspect Sentiment Classification* Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou and Yi Chang
14:50–15:15	*Identifying Transferable Information Across Domains for Cross-domain Sentiment Classification* Raksha Sharma, Pushpak Bhattacharyya, Sandipan Dandapat and Himanshu Sharad Bhatt
15:15–15:40	*Unpaired Sentiment-to-Sentiment Translation: A Cycled Reinforcement Learning Approach* Jingjing Xu, Xu SUN, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang and Wenjie Li

15:40–16:10 ***Coffee Break***

Tuesday, July 17, 2018 (continued)

Session 4E: Evaluation

Tuesday, July 17, 2018 (continued)

Tuesday, July 17, 2018 (continued)

Wednesday, July 18, 2018 (continued)

Probabilistic FastText for Multi-Sense Word Embeddings

Ben Athiwaratkun[*]
Cornell University
pa338@cornell.edu

Andrew Gordon Wilson
Cornell University
andrew@cornell.edu

Anima Anandkumar
AWS & Caltech
anima@amazon.com

Abstract

We introduce *Probabilistic FastText*, a new model for word embeddings that can capture multiple word senses, sub-word structure, and uncertainty information. In particular, we represent each word with a Gaussian mixture density, where the mean of a mixture component is given by the sum of n-grams. This representation allows the model to share statistical strength across sub-word structures (e.g. Latin roots), producing accurate representations of rare, misspelt, or even unseen words. Moreover, each component of the mixture can capture a different word sense. Probabilistic FastText outperforms both FASTTEXT, which has no probabilistic model, and dictionary-level probabilistic embeddings, which do not incorporate subword structures, on several word-similarity benchmarks, including English RareWord and foreign language datasets. We also achieve state-of-art performance on benchmarks that measure ability to discern different meanings. Thus, the proposed model is the first to achieve multi-sense representations while having enriched semantics on rare words.

1 Introduction

Word embeddings are foundational to natural language processing. In order to model language, we need word representations to contain as much semantic information as possible. Most research has focused on vector word embeddings, such as WORD2VEC (Mikolov et al., 2013a), where words with similar meanings are mapped to nearby points in a vector space. Following the seminal work of Mikolov et al. (2013a), there have been numerous works looking to learn efficient word embeddings.

One shortcoming with the above approaches to word embedding that are based on a predefined dictionary (termed as dictionary-based embeddings) is their inability to learn representations of rare words. To overcome this limitation, character-level word embeddings have been proposed. FASTTEXT (Bojanowski et al., 2016) is the state-of-the-art character-level approach to embeddings. In FASTTEXT, each word is modeled by a sum of vectors, with each vector representing an n-gram. The benefit of this approach is that the training process can then share *strength* across words composed of common roots. For example, with individual representations for “circum” and “navigation”, we can construct an informative representation for “circumnavigation”, which would otherwise appear too infrequently to learn a dictionary-level embedding. In addition to effectively modelling rare words, character-level embeddings can also represent slang or misspelled words, such as “dogz”, and can share strength across different languages that share roots, e.g. Romance languages share latent roots.

A different promising direction involves representing words with probability distributions, instead of point vectors. For example, Vilnis and McCallum (2014) represents words with Gaussian distributions, which can capture uncertainty information. Athiwaratkun and Wilson (2017) generalizes this approach to multimodal probability distributions, which can naturally represent words with different meanings. For example, the distribution for “rock” could have mass near the word “jazz” and “pop”, but also “stone” and “basalt”. Athiwaratkun and Wilson (2018) further developed this approach to learn hierarchical word representations: for example, the word “music” can

[*] Work done partly during internship at Amazon.

be learned to have a broad distribution, which encapsulates the distributions for “jazz” and “rock”.

In this paper, we propose *Probabilistic FastText* (PFT), which provides probabilistic character-level representations of words. The resulting word embeddings are highly expressive, yet straightforward and interpretable, with simple, efficient, and intuitive training procedures. PFT can model rare words, uncertainty information, hierarchical representations, and multiple word senses. In particular, we represent each word with a Gaussian or a Gaussian mixture density, which we name PFT-G and PFT-GM respectively. Each component of the mixture can represent different word senses, and the mean vectors of each component decompose into vectors of n-grams, to capture character-level information. We also derive an efficient energy-based max-margin training procedure for PFT.

We perform comparison with FASTTEXT as well as existing density word embeddings W2G (Gaussian) and W2GM (Gaussian mixture). Our models extract high-quality semantics based on multiple word-similarity benchmarks, including the rare word dataset. We obtain an average weighted improvement of 3.7% over FASTTEXT (Bojanowski et al., 2016) and 3.1% over the dictionary-level density-based models. We also observe meaningful nearest neighbors, particularly in the multimodal density case, where each mode captures a distinct meaning. Our models are also directly portable to foreign languages without any hyperparameter modification, where we observe strong performance, outperforming FASTTEXT on many foreign word similarity datasets. Our multimodal word representation can also disentangle meanings, and is able to separate different senses in foreign polysemies. In particular, our models attain state-of-the-art performance on SCWS, a benchmark to measure the ability to separate different word meanings, achieving 1.0% improvement over a recent density embedding model W2GM (Athiwaratkun and Wilson, 2017).

To the best of our knowledge, we are the first to develop multi-sense embeddings with high semantic quality for rare words. Our code and embeddings are publicly available. [1]

2 Related Work

Early word embeddings which capture semantic information include Bengio et al. (2003), Collobert and Weston (2008), and Mikolov et al. (2011). Later, Mikolov et al. (2013a) developed the popular WORD2VEC method, which proposes a log-linear model and negative sampling approach that efficiently extracts rich semantics from text. Another popular approach GLOVE learns word embeddings by factorizing co-occurrence matrices (Pennington et al., 2014).

Recently there has been a surge of interest in making dictionary-based word embeddings more flexible. This flexibility has valuable applications in many end-tasks such as language modeling (Kim et al., 2016), named entity recognition (Kuru et al., 2016), and machine translation (Zhao and Zhang, 2016; Lee et al., 2017), where unseen words are frequent and proper handling of these words can greatly improve the performance. These works focus on modeling subword information in neural networks for tasks such as language modeling.

Besides vector embeddings, there is recent work on multi-prototype embeddings where each word is represented by multiple vectors. The learning approach involves using a cluster centroid of context vectors (Huang et al., 2012), or adapting the skip-gram model to learn multiple latent representations (Tian et al., 2014). Neelakantan et al. (2014) furthers adapts skip-gram with a non-parametric approach to learn the embeddings with an arbitrary number of senses per word. Chen et al. (2014) incorporates an external dataset WORDNET to learn sense vectors. We compare these models with our multimodal embeddings in Section 4.

3 Probabilistic FastText

We introduce *Probabilistic FastText*, which combines a probabilistic word representation with the ability to capture subword structure. We describe the probabilistic subword representation in Section 3.1. We then describe the similarity measure and the loss function used to train the embeddings in Sections 3.2 and 3.3. We conclude by briefly presenting a simplified version of the energy function for isotropic Gaussian representations (Section 3.4), and the negative sampling scheme we use in training (Section 3.5).

3.1 Probabilistic Subword Representation

We represent each word with a Gaussian mixture with K Gaussian components. That is, a word

[1] https://github.com/benathi/multisense-prob-fasttext

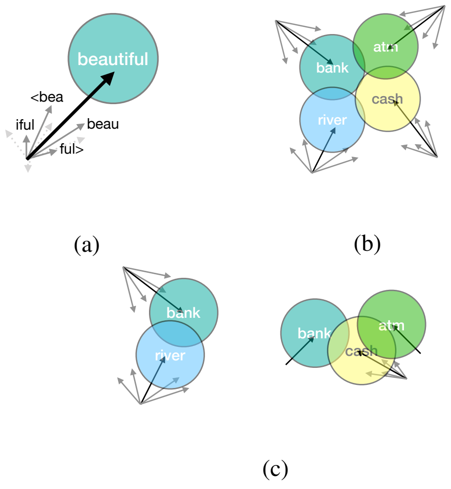


Figure 1: (1a) a Gaussian component and its subword structure. The bold arrow represents the final mean vector, estimated from averaging the grey n-gram vectors. (1b) PFT-G model: Each Gaussian component's mean vector is a subword vector. (1c) PFT-GM model: For each Gaussian mixture distribution, one component's mean vector is estimated by a subword structure whereas other components are dictionary-based vectors.

w is associated with a density function $f(x) = \sum_{i=1}^{K} p_{w,i} \mathcal{N}(x; \vec{\mu}_{w,i}, \Sigma_{w,i})$ where $\{\mu_{w,i}\}_{k=1}^{K}$ are the mean vectors and $\{\Sigma_{w,i}\}$ are the covariance matrices, and $\{p_{w,i}\}_{k=1}^{K}$ are the component probabilities which sum to 1.

The mean vectors of Gaussian components hold much of the semantic information in density embeddings. While these models are successful based on word similarity and entailment benchmarks (Vilnis and McCallum, 2014; Athiwaratkun and Wilson, 2017), the mean vectors are often dictionary-level, which can lead to poor semantic estimates for rare words, or the inability to handle words outside the training corpus. We propose using subword structures to estimate the mean vectors. We outline the formulation below.

For word w, we estimate the mean vector μ_w with the average over n-gram vectors and its dictionary-level vector. That is,

$$\mu_w = \frac{1}{|NG_w| + 1} \left(v_w + \sum_{g \in NG_w} z_g \right) \quad (1)$$

where z_g is a vector associated with an n-gram g, v_w is the dictionary representation of word w, and NG_w is a set of n-grams of word w. Examples of 3,4-grams for a word "beautiful", including the beginning-of-word character '$\langle$' and end-of-word character '$\rangle$', are:

- 3-grams: $\langle$be, bea, eau, aut, uti, tif, ful, ul$\rangle$
- 4-grams: $\langle$bea, beau .., iful ,ful$\rangle$

This structure is similar to that of FASTTEXT (Bojanowski et al., 2016); however, we note that FASTTEXT uses single-prototype deterministic embeddings as well as a training approach that maximizes the negative log-likelihood, whereas we use a multi-prototype probabilistic embedding and for training we maximize the similarity between the words' probability densities, as described in Sections 3.2 and 3.3

Figure 1a depicts the subword structure for the mean vector. Figure 1b and 1c depict our models, Gaussian probabilistic FASTTEXT (PFT-G) and Gaussian mixture probabilistic FASTTEXT (PFT-GM). In the Gaussian case, we represent each mean vector with a subword estimation. For the Gaussian mixture case, we represent one Gaussian component's mean vector with the subword structure whereas other components' mean vectors are dictionary-based. This model choice to use dictionary-based mean vectors for other components is to reduce to constraint imposed by the subword structure and promote independence for meaning discovery.

3.2 Similarity Measure between Words

Traditionally, if words are represented by vectors, a common similarity metric is a dot product. In the case where words are represented by distribution functions, we use the generalized dot product in Hilbert space $\langle \cdot, \cdot \rangle_{L_2}$, which is called the expected likelihood kernel (Jebara et al., 2004). We define the energy $E(f, g)$ between two words f and g to be $E(f, g) = \log \langle f, g \rangle_{L_2} = \log \int f(x) g(x) \, dx$. With Gaussian mixtures $f(x) = \sum_{i=1}^{K} p_i \mathcal{N}(x; \vec{\mu}_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^{K} q_i \mathcal{N}(x; \vec{\mu}_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^{K} p_i = 1$, and $\sum_{i=1}^{K} q_i = 1$, the energy has a closed form:

$$E(f, g) = \log \sum_{j=1}^{K} \sum_{i=1}^{K} p_i q_j e^{\xi_{i,j}} \quad (2)$$

where $\xi_{j,j}$ is the partial energy which corresponds to the similarity between component i of the first word f and component j of the second word g.[2]

$$\begin{aligned}\xi_{i,j} &\equiv \log \mathcal{N}(0; \vec{\mu}_{f,i} - \vec{\mu}_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}) \\ &= -\frac{1}{2}\log\det(\Sigma_{f,i} + \Sigma_{g,j}) - \frac{D}{2}\log(2\pi) \\ -\frac{1}{2}&(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top(\Sigma_{f,i} + \Sigma_{g,j})^{-1}(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})\end{aligned} \tag{3}$$

Figure 2 demonstrates the partial energies among the Gaussian components of two words.

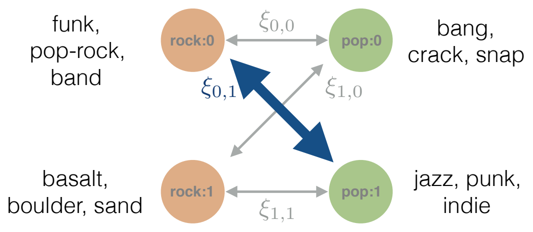


Figure 2: The interactions among Gaussian components of word `rock` and word `pop`. The partial energy is the highest for the pair `rock:0` (the zeroth component of rock) and `pop:1` (the first component of pop), reflecting the similarity in meanings.

3.3 Loss Function

The model parameters that we seek to learn are v_w for each word w and z_g for each n-gram g. We train the model by pushing the energy of a true context pair w and c to be higher than the negative context pair w and n by a margin m. We use Adagrad (Duchi et al., 2011) to minimize the following loss to achieve this outcome:

$$L(f, g) = \max\left[0, m - E(f, g) + E(f, n)\right]. \tag{4}$$

We describe how to sample words as well as its positive and negative contexts in Section 3.5.

This loss function together with the Gaussian mixture model with $K > 1$ has the ability to extract multiple senses of words. That is, for a word with multiple meanings, we can observe each mode to represent a distinct meaning. For instance, one density mode of "star" is close to the densities of "celebrity" and "hollywood" whereas another mode of "star" is near the densities of "constellation" and "galaxy".

[2]The orderings of indices of the components for each word are arbitrary.

3.4 Energy Simplification

In theory, it can be beneficial to have covariance matrices as learnable parameters. In practice, Athiwaratkun and Wilson (2017) observe that spherical covariances often perform on par with diagonal covariances with much less computational resources. Using spherical covariances for each component, we can further simplify the energy function as follows:

$$\xi_{i,j} = -\frac{\alpha}{2} \cdot ||\mu_{f,i} - \mu_{g,j}||^2, \tag{5}$$

where the hyperparameter α is the scale of the inverse covariance term in Equation 3. We note that Equation 5 is equivalent to Equation 3 up to an additive constant given that the covariance matrices are spherical and the same for all components.

3.5 Word Sampling

To generate a context word c of a given word w, we pick a nearby word within a context window of a fixed length ℓ. We also use a word sampling technique similar to Mikolov et al. (2013b). This subsampling procedure selects words for training with lower probabilities if they appear frequently. This technique has an effect of reducing the importance of words such as 'the', 'a', 'to' which can be predominant in a text corpus but are not as meaningful as other less frequent words such as 'city', 'capital', 'animal', etc. In particular, word w has probability $P(w) = 1 - \sqrt{t/f(w)}$ where $f(w)$ is the frequency of word w in the corpus and t is the frequency threshold.

A negative context word is selected using a distribution $P_n(w) \propto U(w)^{3/4}$ where $U(w)$ is a unigram probability of word w. The exponent $3/4$ also diminishes the importance of frequent words and shifts the training focus to other less frequent words.

4 Experiments

We have proposed a probabilistic FASTTEXT model which combines the flexibility of subword structure with the density embedding approach. In this section, we show that our probabilistic representation with subword mean vectors with the simplified energy function outperforms many word similarity baselines and provides disentangled meanings for polysemies.

First, we describe the training details in Section 4.1. We provide qualitative evaluation in Section

4.2, showing meaningful nearest neighbors for the Gaussian embeddings, as well as the ability to capture multiple meanings by Gaussian mixtures. Our quantitative evaluation in Section 4.3 demonstrates strong performance against the baseline models FASTTEXT (Bojanowski et al., 2016) and the dictionary-level Gaussian (W2G) (Vilnis and McCallum, 2014) and Gaussian mixture embeddings (Athiwaratkun and Wilson, 2017) (W2GM). We train our models on foreign language corpuses and show competitive results on foreign word similarity benchmarks in Section 4.4. Finally, we explain the importance of the n-gram structures for semantic sharing in Section 4.5.

4.1 Training Details

We train our models on both English and foreign language datasets. For English, we use the concatenation of UKWAC and WACKYPEDIA (Baroni et al., 2009) which consists of 3.376 billion words. We filter out word types that occur fewer than 5 times which results in a vocabulary size of 2,677,466.

For foreign languages, we demonstrate the training of our model on French, German, and Italian text corpuses. We note that our model should be applicable for other languages as well. We use FRWAC (French), DEWAC (German), ITWAC (Italian) datasets (Baroni et al., 2009) for text corpuses, consisting of 1.634, 1.716 and 1.955 billion words respectively. We use the same threshold, filtering out words that occur less than 5 times in each corpus. We have dictionary sizes of 1.3, 2.7, and 1.4 million words for FRWAC, DEWAC, and ITWAC.

We adjust the hyperparameters on the English corpus and use them for foreign languages. Note that the adjustable parameters for our models are the loss margin m in Equation 4 and the scale α in Equation 5. We search for the optimal hyperparameters in a grid $m \in \{0.01, 0.1, 1, 10, 100\}$ and $\alpha \in \{\frac{1}{5\times10^{-3}}, \frac{1}{10^{-3}}, \frac{1}{2\times10^{-4}}, \frac{1}{1\times10^{-4}}\}$ on our English corpus. The hyperpameter α affects the scale of the loss function; therefore, we adjust the learning rate appropriately for each α. In particular, the learning rates used are $\gamma = \{10^{-4}, 10^{-5}, 10^{-6}\}$ for the respective α values.

Other fixed hyperparameters include the number of Gaussian components $K = 2$, the context window length $\ell = 10$ and the subsampling threshold $t = 10^{-5}$. Similar to the setup in FASTTEXT, we use n-grams where $n = 3, 4, 5, 6$ to estimate the mean vectors.

4.2 Qualitative Evaluation - Nearest neighbors

We show that our embeddings learn the word semantics well by demonstrating meaningful nearest neighbors. Table 1 shows examples of polysemous words such as `rock`, `star`, and `cell`.

Table 1 shows the nearest neighbors of polysemous words. We note that subword embeddings prefer words with overlapping characters as nearest neighbors. For instance, "rock-y", "rockn", and "rock" are both close to the word "rock". For the purpose of demonstration, we only show words with meaningful variations and omit words with small character-based variations previously mentioned. However, all words shown are in the top-100 nearest words.

We observe the separation in meanings for the multi-component case; for instance, one component of the word "bank" corresponds to a financial bank whereas the other component corresponds to a river bank. The single-component case also has interesting behavior. We observe that the subword embeddings of polysemous words can represent both meanings. For instance, both "lava-rock" and "rock-pop" are among the closest words to "rock".

4.3 Word Similarity Evaluation

We evaluate our embeddings on several standard word similarity datasets, namely, SL-999 (Hill et al., 2014), WS-353 (Finkelstein et al., 2002), MEN-3k (Bruni et al., 2014), MC-30 (Miller and Charles, 1991), RG-65 (Rubenstein and Goodenough, 1965), YP-130 (Yang and Powers, 2006), MTurk(-287,-771) (Radinsky et al., 2011; Halawi et al., 2012), and RW-2k (Luong et al., 2013). Each dataset contains a list of word pairs with a human score of how related or similar the two words are. We use the notation DATASET-NUM to denote the number of word pairs NUM in each evaluation set. We note that the dataset RW focuses more on infrequent words and SimLex-999 focuses on the similarity of words rather than relatedness. We also compare PFT-GM with other multi-prototype embeddings in the literature using SCWS (Huang et al., 2012), a word similarity dataset that is aimed to measure the ability of embeddings to discern multiple meanings.

We calculate the Spearman correlation (Spearman, 1904) between the labels and our scores gen-

Word	Co.	Nearest Neighbors
rock	0	rock:0, rocks:0, rocky:0, mudrock:0, rockscape:0, boulders:0 , coutcrops:0,
rock	1	rock:1, punk:0, punk-rock:0, indie:0, pop-rock:0, pop-punk:0, indie-rock:0, band:1
bank	0	bank:0, banks:0, banker:0, bankers:0, bankcard:0, Citibank:0, debits:0
bank	1	bank:1, banks:1, river:0, riverbank:0, embanking:0, banks:0, confluence:1
star	0	stars:0, stellar:0, nebula:0, starspot:0, stars.:0, stellas:0, constellation:1
star	1	star:1, stars:1, star-star:0, 5-stars:0, movie-star:0, mega-star:0, super-star:0
cell	0	cell:0, cellular:0, acellular:0, lymphocytes:0, T-cells:0, cytes:0, leukocytes:0
cell	1	cell:1, cells:1, cellular:0, cellular-phone:0, cellphone:0, transcellular:0
left	0	left:0, right:1, left-hand:0, right-left:0, left-right-left:0, right-hand:0, leftwards:0
left	1	left:1, leaving:0, leavings:0, remained:0, leave:1, enmained:0, leaving-age:0, sadly-departed:0

Word	Nearest Neighbors
rock	rock, rock-y, rockn, rock-, rock-funk, rock/, lava-rock, nu-rock, rock-pop, rock/ice, coral-rock
bank	bank-, bank/, bank-account, bank., banky, bank-to-bank, banking, Bank, bank/cash, banks.**
star	movie-stars, star-planet, G-star, star-dust, big-star, starsailor, 31-star, star-lit, Star, starsign, pop-stars
cell	cellular, tumour-cell, in-cell, cell/tumour, 11-cell, T-cell, sperm-cell, 2-cells, Cell-to-cell
left	left, left/joined, leaving, left,right, right, left)and, leftsided, lefted, leftside

Table 1: Nearest neighbors of PFT-GM (top) and PFT-G (bottom). The notation `w:i` denotes the i^{th} mixture component of the word `w`.

D	50				300				
	W2G	W2GM	PFT-G	PFT-GM	FASTTEXT	W2G	W2GM	PFT-G	PFT-GM
SL-999	29.35	29.31	27.34	**34.13**	38.03	38.84	**39.62**	35.85	39.60
WS-353	71.53	**73.47**	67.17	71.10	73.88	78.25	**79.38**	73.75	76.11
MEN-3K	72.58	73.55	70.61	**73.90**	76.37	78.40	78.76	77.78	**79.65**
MC-30	76.48	79.08	73.54	**79.75**	81.20	82.42	**84.58**	81.90	80.93
RG-65	73.30	74.51	70.43	**78.19**	79.98	80.34	**80.95**	77.57	79.81
YP-130	41.96	**45.07**	37.10	40.91	53.33	46.40	47.12	48.52	**54.93**
MT-287	64.79	66.60	63.96	**67.65**	67.93	67.74	**69.65**	66.41	69.44
MT-771	60.86	60.82	60.40	**63.86**	66.89	70.10	**70.36**	67.18	69.68
RW-2K	28.78	28.62	44.05	**42.78**	48.09	35.49	42.73	**50.37**	49.36
AVG.	42.32	42.76	44.35	**46.47**	49.28	47.71	49.54	49.86	**51.10**

Table 2: Spearman's Correlation $\rho \times 100$ on Word Similarity Datasets.

erated by the embeddings. The Spearman correlation is a rank-based correlation measure that assesses how well the scores describe the true labels. The scores we use are cosine-similarity scores between the mean vectors. In the case of Gaussian mixtures, we use the pairwise maximum score:

$$s(f,g) = \max_{i\in 1,\dots,K} \max_{j\in 1,\dots,K} \frac{\mu_{f,i}\cdot\mu_{g,j}}{||\mu_{f,i}||\cdot||\mu_{g,j}||}. \quad (6)$$

The pair (i,j) that achieves the maximum cosine similarity corresponds to the Gaussian component pair that is the closest in meanings. Therefore, this similarity score yields the most related senses of a given word pair. This score reduces to a cosine similarity in the Gaussian case $(K = 1)$.

4.3.1 Comparison Against Dictionary-Level Density Embeddings and FASTTEXT

We compare our models against the dictionary-level Gaussian and Gaussian mixture embeddings in Table 2, with 50-dimensional and 300-dimensional mean vectors. The 50-dimensional results for W2G and W2GM are obtained directly from Athiwaratkun and Wilson (2017). For comparison, we use the public code[3] to train the 300-dimensional W2G and W2GM models and the publicly available FASTTEXT model[4].

We calculate Spearman's correlations for each of the word similarity datasets. These datasets vary greatly in the number of word pairs; therefore, we mark each dataset with its size for visibil-

[3] https://github.com/benathi/word2gm

[4] https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.en.zip

ity. For a fair and objective comparison, we calculate a weighted average of the correlation scores for each model.

Our PFT-GM achieves the highest average score among all competing models, outperforming both FASTTEXT and the dictionary-level embeddings W2G and W2GM. Our unimodal model PFT-G also outperforms the dictionary-level counterpart W2G and FASTTEXT. We note that the model W2GM appears quite strong according to Table 2, beating PFT-GM on many word similarity datasets. However, the datasets that W2GM performs better than PFT-GM often have small sizes such as MC-30 or RG-65, where the Spearman's correlations are more subject to noise. Overall, PFT-GM outperforms W2GM by 3.1% and 8.7% in 300 and 50 dimensional models. In addition, PFT-G and PFT-GM also outperform FASTTEXT by 1.2% and 3.7% respectively.

4.3.2 Comparison Against Multi-Prototype Models

In Table 3, we compare 50 and 300 dimensional PFT-GM models against the multi-prototype embeddings described in Section 2 and the existing multimodal density embeddings W2GM. We use the word similarity dataset SCWS (Huang et al., 2012) which contains words with potentially many meanings, and is a benchmark for distinguishing senses. We use the maximum similarity score (Equation 6), denoted as MAXSIM. AVESIM denotes the average of the similarity scores, rather than the maximum.

We outperform the dictionary-based density embeddings W2GM in both 50 and 300 dimensions, demonstrating the benefits of subword information. Our model achieves state-of-the-art results, similar to that of Neelakantan et al. (2014).

Model	Dim	$\rho \times 100$
HUANG AVGSIM	50	62.8
TIAN MAXSIM	50	63.6
W2GM MAXSIM	50	62.7
NEELAKANTAN AVGSIM	50	64.2
PFT-GM MAXSIM	50	63.7
CHEN-M AVGSIM	200	66.2
W2GM MAXSIM	200	65.5
NEELAKANTAN AVGSIM	300	**67.2**
W2GM MAXSIM	300	66.5
PFT-GM MAXSIM	300	**67.2**

Table 3: Spearman's Correlation $\rho \times 100$ on word similarity dataset SCWS.

4.4 Evaluation on Foreign Language Embeddings

We evaluate the foreign-language embeddings on word similarity datasets in respective languages. We use Italian WORDSIM353 and Italian SIMLEX-999 (Leviant and Reichart, 2015) for Italian models, GUR350 and GUR65 (Gurevych, 2005) for German models, and French WORDSIM353 (Finkelstein et al., 2002) for French models. For datasets GUR350 and GUR65, we use the results reported in the FASTTEXT publication (Bojanowski et al., 2016). For other datasets, we train FASTTEXT models for comparison using the public code[5] on our text corpuses. We also train dictionary-level models W2G, and W2GM for comparison.

Table 4 shows the Spearman's correlation results of our models. We outperform FASTTEXT on many word similarity benchmarks. Our results are also significantly better than the dictionary-based models, W2G and W2GM. We hypothesize that W2G and W2GM can perform better than the current reported results given proper pre-processing of words due to special characters such as accents.

We investigate the nearest neighbors of polysemies in foreign languages and also observe clear sense separation. For example, *piano* in Italian can mean "floor" or "slow". These two meanings are reflected in the nearest neighbors where one component is close to *piano-piano*, *pianod* which mean "slowly" whereas the other component is close to *piani* (floors), *istrutturazione* (renovation) or *infrastruttre* (infrastructure). Table 5 shows additional results, demonstrating that the disentangled semantics can be observed in multiple languages.

4.5 Qualitative Evaluation - Subword Decomposition

One of the motivations for using subword information is the ability to handle out-of-vocabulary words. Another benefit is the ability to help improve the semantics of rare words via subword sharing. Due to an observation that text corpuses follow Zipf's power law (Zipf, 1949), words at the tail of the occurrence distribution appears much

[5]https://github.com/facebookresearch/fastText.git

Lang.	Evaluation	FASTTEXT	w2g	w2gm	pft-g	pft-gm
FR	WS353	38.2	16.73	20.09	41.0	**41.3**
DE	GUR350	70	65.01	69.26	77.6	**78.2**
	GUR65	81	74.94	76.89	81.8	**85.2**
IT	WS353	57.1	56.02	61.09	60.2	**62.5**
	SL-999	29.3	29.44	**34.91**	29.3	33.7

Table 4: Word similarity evaluation on foreign languages.

Word	Meaning	Nearest Neighbors
(IT) *secondo*	2nd	Secondo (2nd), terzo (3rd) , quinto (5th), primo (first), quarto (4th), ultimo (last)
(IT) *secondo*	according to	conformit (compliance), attenendosi (following), cui (which), conformemente (accordance with)
(IT) *porta*	lead, bring	portano (lead), conduce (leads), portano, porter, portando (bring), costringe (forces)
(IT) *porta*	door	porte (doors), finestrella (window), finestra (window), portone (doorway), serratura (door lock)
(FR) *voile*	veil	voiles (veil), voiler (veil), voilent (veil), voilement, foulard (scarf), voils (veils), voilant (veiling)
(FR) *voile*	sail	catamaran (catamaran), driveur (driver), nautiques (water), Voile (sail), driveurs (drivers)
(FR) *temps*	weather	brouillard (fog), orageuses (stormy), nuageux (cloudy)
(FR) *temps*	time	mi-temps (half-time), partiel (partial), Temps (time), annualis (annualized), horaires (schedule)
(FR) *voler*	steal	envoler (fly), voleuse (thief), cambrioler (burgle), voleur (thief), violer (violate), picoler (tipple)
(FR) *voler*	fly	airs (air), vol (flight), volent (fly), envoler (flying), atterrir (land)

Table 5: Nearest neighbors of polysemies based on our foreign language PFT-GM models.

less frequently. Training these words to have a good semantic representation is challenging if done at the word level alone. However, an n-gram such as 'abnorm' is trained during both occurrences of "abnormal" and "abnormality" in the corpus, hence further augments both words's semantics.

Figure 3 shows the contribution of n-grams to the final representation. We filter out to show only the n-grams with the top-5 and bottom-5 similarity scores. We observe that the final representations of both words align with n-grams "abno", "bnor", "abnorm", "anbnor", "<abn". In fact, both "abnormal" and "abnormality" share the same top-5 n-grams. Due to the fact that many rare words such as "autobiographer", "circumnavigations", or "hypersensitivity" are composed from many common sub-words, the n-gram structure can help improve the representation quality.

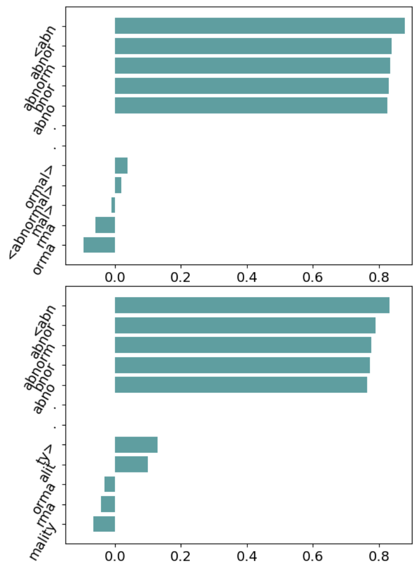


Figure 3: Contribution of each n-gram vector to the final representation for word "abnormal" (top) and "abnormality" (bottom). The x-axis is the cosine similarity between each n-gram vector $z_g^{(w)}$ and the final vector μ_w.

5 Numbers of Components

It is possible to train our approach with $K > 2$ mixture components; however, Athiwaratkun and Wilson (2017) observe that dictionary-level Gaussian mixtures with $K = 3$ do not overall improve word similarity results, even though these mixtures can discover 3 distinct senses for certain words. Indeed, while $K > 2$ in principle allows for greater flexibility than $K = 2$, most words can be very flexibly modelled with a mixture of two Gaussians, leading to $K = 2$ representing a good balance between flexibility and Occam's razor.

Even for words with single meanings, our PFT model with $K = 2$ often learns richer representations than a $K = 1$ model. For example, the two mixture components can learn to cluster together to form a more heavy tailed unimodal distribution which captures a word with one dominant meaning but with close relationships to a wide range of other words.

In addition, we observe that our model with K components can capture more than K meanings. For instance, in $K = 1$ model, the word pairs ("cell", "jail") and ("cell", "biology") and ("cell", "phone") will all have positive similarity scores based on $K = 1$ model. In general, if a word has multiple meanings, these meanings are usually compressed into the linear substructure of the embeddings (Arora et al., 2016). However, the pairs of non-dominant words often have lower similarity scores, which might not accurately reflect their true similarities.

6 Conclusion and Future Work

We have proposed models for probabilistic word representations equipped with flexible sub-word structures, suitable for rare and out-of-vocabulary words. The proposed probabilistic formulation incorporates uncertainty information and naturally allows one to uncover multiple meanings with multimodal density representations. Our models offer better semantic quality, outperforming competing models on word similarity benchmarks. Moreover, our multimodal density models can provide interpretable and disentangled representations, and are the first multi-prototype embeddings that can handle rare words.

Future work includes an investigation into the trade-off between learning full covariance matrices for each word distribution, computational complexity, and performance. This direction can potentially have a great impact on tasks where the variance information is crucial, such as for hierarchical modeling with probability distributions (Athiwaratkun and Wilson, 2018).

Other future work involves co-training PFT on many languages. Currently, existing work on multi-lingual embeddings align the word semantics on pre-trained vectors (Smith et al., 2017), which can be suboptimal due to polysemies. We envision that the multi-prototype nature can help disambiguate words with multiple meanings and facilitate semantic alignment.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear algebraic structure of word senses, with applications to polysemy. *CoRR* abs/1601.03764. http://arxiv.org/abs/1601.03764.

Ben Athiwaratkun and Andrew Gordon Wilson. 2017. Multimodal word distributions. In *ACL*. https://arxiv.org/abs/1704.08424.

Ben Athiwaratkun and Andrew Gordon Wilson. 2018. On modeling hierarchical data via probabilistic order embeddings. *ICLR* .

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation* 43(3):209–226. https://doi.org/10.1007/s10579-009-9081-4.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155. http://www.jmlr.org/papers/v3/bengio03a.html.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.* 49(1):1–47. http://dl.acm.org/citation.cfm?id=2655713.2655714.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1025–1035. http://aclweb.org/anthology/D/D14/D14-1110.pdf.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. pages 160–167. http://doi.acm.org/10.1145/1390156.1390177.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159. http://dl.acm.org/citation.cfm?id=2021068.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.* 20(1):116–131. http://doi.acm.org/10.1145/503104.503110.

Iryna Gurevych. 2005. Using the structure of a conceptual network in computing semantic relatedness. In *Natural Language Processing - IJCNLP 2005, Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005, Proceedings*. pages 767–778.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. pages 1406–1414. http://doi.acm.org/10.1145/2339530.2339751.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR* abs/1408.3456. http://arxiv.org/abs/1408.3456.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*. pages 873–882. http://www.aclweb.org/anthology/P12-1092.

Tony Jebara, Risi Kondor, and Andrew Howard. 2004. Probability product kernels. *Journal of Machine Learning Research* 5:819–844.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. pages 2741–2749.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 911–921. http://aclweb.org/anthology/C/C16/C16-1087.pdf.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL* 5:365–378. https://transacl.org/ojs/index.php/tacl/article/view/1051.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR* abs/1508.00106. http://arxiv.org/abs/1508.00106.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. Sofia, Bulgaria.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. pages 5528–5531. https://doi.org/10.1109/ICASSP.2011.5947611.

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes* 6(1):1–28. https://doi.org/10.1080/01690969108406936.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient nonparametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1059–1069. http://aclweb.org/anthology/D/D14/D14-1113.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*. WWW '11, pages 337–346. http://doi.acm.org/10.1145/1963405.1963455.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM* 8(10):627–633. http://doi.acm.org/10.1145/365628.365657.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR* abs/1702.03859. http://arxiv.org/abs/1702.03859.

C. Spearman. 1904. The proof and measurement of association between two things. *American Journal of Psychology* 15:88–103.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*. pages 151–160. http://aclweb.org/anthology/C/C14/C14-1016.pdf.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR* abs/1412.6623. http://arxiv.org/abs/1412.6623.

Dongqiang Yang and David M. W. Powers. 2006. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*.

Shenjian Zhao and Zhihua Zhang. 2016. An efficient character-level neural machine translation. *CoRR* abs/1608.04738. http://arxiv.org/abs/1608.04738.

G.K. Zipf. 1949. *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press. https://books.google.com/books?id=1tx9AAAAIAAJ.

A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors

Mikhail Khodak*, Nikunj Saunshi*
Princeton University
{mkhodak,nsaunshi}@princeton.edu

Yingyu Liang
University of Wisconsin-Madison
yliang@cs.wisc.edu

Tengyu Ma
Facebook AI Research
tengyuma@stanford.edu

Brandon Stewart, Sanjeev Arora
Princeton University
{bms4,arora}@princeton.edu

Abstract

Motivations like domain adaptation, transfer learning, and feature learning have fueled interest in inducing embeddings for rare or unseen words, n-grams, synsets, and other textual features. This paper introduces *à la carte* embedding, a simple and general alternative to the usual word2vec-based approaches for building such representations that is based upon recent theoretical results for GloVe-like embeddings. Our method relies mainly on a linear transformation that is efficiently learnable using pretrained word vectors and linear regression. This transform is applicable "on the fly" in the future when a new text feature or rare word is encountered, even if only a single usage example is available. We introduce a new dataset showing how the *à la carte* method requires fewer examples of words in context to learn high-quality embeddings and we obtain state-of-the-art results on a nonce task and some unsupervised document classification tasks.

1 Introduction

Distributional word embeddings, which represent the "meaning" of a word via a low-dimensional vector, have been widely applied by many natural language processing (NLP) pipelines and algorithms (Goldberg, 2016). Following the success of recent neural (Mikolov et al., 2013) and matrix-factorization (Pennington et al., 2014) methods, researchers have sought to extend the approach to other text features, from subword elements to n-grams to sentences (Bojanowski et al., 2016; Poliak et al., 2017; Kiros et al., 2015). However, the performance of both word embeddings and their extensions is known to degrade in small corpus settings (Adams et al., 2017) or when embedding sparse, low-frequency features (Lazaridou et al., 2017). Attempts to address these issues often involve task-specific approaches (Rothe and Schütze, 2015; Iacobacci et al., 2015; Pagliardini et al., 2018) or extensively tuning existing architectures such as skip-gram (Poliak et al., 2017; Herbelot and Baroni, 2017).

For computational efficiency it is desirable that methods be able to induce embeddings for only those features (e.g. bigrams or synsets) needed by the downstream task, rather than having to pay a computational *prix fixe* to learn embeddings for all features occurring frequently-enough in a corpus. We propose an alternative, novel solution via *à la carte* embedding, a method which bootstraps existing high-quality word vectors to learn a feature representation in the same semantic space via a linear transformation of the average word embeddings in the feature's available contexts. This can be seen as a shallow extension of the distributional hypothesis (Harris, 1954), "a feature is characterized by the words in its context," rather than the computationally more-expensive "a feature is characterized by the features in its context" that has been used implicitly by past work (Rothe and Schütze, 2015; Logeswaran and Lee, 2018).

Despite its elementary formulation, we demonstrate that the *à la carte* method can learn faithful word embeddings from single examples and feature vectors improving performance on important downstream tasks. Furthermore, the approach is resource-efficient, needing only pretrained embeddings of common words and the text corpus used to train them, and easy to implement and compute via vector addition and linear regression. After motivating and specifying the method, we illustrate these benefits through several applications:

- **Embeddings of rare words:** we introduce a dataset[1] for few-shot learning of word vectors and achieve state-of-the-art results on the task of representing unseen words using only the definition (Herbelot and Baroni, 2017).
- **Synset embeddings:** we show how the method can be applied to learn more fine-grained lexico-semantic representations and give evidence of its usefulness for standard word-sense disambiguation tasks (Navigli et al., 2013; Moro and Navigli, 2015).
- **n-gram embeddings:** we build seven million n-gram embeddings from large text corpora and use them to construct document embeddings that are competitive with unsupervised deep learning approaches when evaluated on linear text classification.

Our experimental results[2] clearly demonstrate the advantages of *à la carte* embedding. For word embeddings, the approach is an easy way to get a good vector for a new word from its definition or a few examples in context. For feature embeddings, the method can embed anything that does not need labeling (such as a bigram) or occurs in an annotated corpus (such as a word-sense). Our document embeddings, constructed directly using *à la carte* n-gram vectors, compete well with recent deep neural representations; this provides further evidence that simple methods can outperform modern deep learning on many NLP benchmarks (Arora et al., 2017; Mu and Viswanath, 2018; Arora et al., 2018a,b; Pagliardini et al., 2018).

2 Related Work

Many methods have been proposed for extending word embeddings to semantic feature vectors, with the aim of using them as interpretable and structure-aware building blocks of NLP pipelines (Kiros et al., 2015; Yamada et al., 2016). Many exploit the structure and resources available for specific feature types, such as methods for sense, synsets, and lexemes (Rothe and Schütze, 2015; Iacobacci et al., 2015) that make heavy use of the graph structure of the Princeton WordNet (PWN) and similar resources (Fellbaum, 1998). By contrast, our work is more general, with incorporation of structure left as an open problem. Embeddings of n-grams are of special interest because they do not need annotation or expert knowledge and can often be effective on downstream tasks. Their computation has been studied both explicitly (Yin and Schutze, 2014; Poliak et al., 2017) and as an implicit part of models for document embeddings (Hill et al., 2016; Pagliardini et al., 2018), which we use for comparison. Supervised and multi-task learning of text embeddings has also been attempted (Wang et al., 2017; Wu et al., 2017).

A main motivation of our work is to learn good embeddings, of both words and features, from only one or a few examples. Efforts in this area can in many cases be split into contextual approaches (Lazaridou et al., 2017; Herbelot and Baroni, 2017) and morphological methods (Luong et al., 2013; Bojanowski et al., 2016; Pado et al., 2016). The current paper provides a more effective formulation for context-based embeddings, which are often simpler to implement, can improve with more context information, and do not require morphological annotation. Subword approaches, on the other hand, are often more compositional and flexible, and we leave the extension of our method to handle subword information to future work. Our work is also related to some methods in domain adaptation and multi-lingual correlation, such as that of Bollegala et al. (2014).

Mathematically, this work builds upon the linear algebraic understanding of modern word embeddings developed by Arora et al. (2018b) via an extension to the latent-variable embedding model of Arora et al. (2016). Although there have been several other applications of this model for natural language representation (Arora et al., 2017; Mu and Viswanath, 2018), ours is the first to provide a general approach for learning semantic features using corpus context.

3 Method Specification

We begin by assuming a large text corpus $\mathcal{C}_\mathcal{V}$ consisting of contexts c of words w in a vocabulary $\mathcal{V}$, with the contexts themselves being sequences of words in $\mathcal{V}$ (e.g. a fixed-size window around the word or feature). We further assume that we have trained word embeddings $\mathbf{v}_w \in \mathbb{R}^d$ on this collo-

[1]Dataset: `nlp.cs.princeton.edu/CRW`
[2]Code: `www.github.com/NLPrinceton/ALaCarte`

cation information using a standard algorithm (e.g. word2vec / GloVe). Our goal is to construct a good embedding $\mathbf{v}_f \in \mathbb{R}^d$ of a text feature f given a set $\mathcal{C}_f$ of contexts it occurs in. Both f and its contexts are assumed to arise via the same process that generates the large corpus $\mathcal{C}_\mathcal{V}$. In many settings below, the number $|\mathcal{C}_f|$ of contexts available for a feature f of interest is much smaller than the number $|\mathcal{C}_w|$ of contexts that the typical word $w \in \mathcal{V}$ occurs in. This could be because the feature is rare (e.g. unseen words, n-grams) or due to limited human annotation (e.g. word senses, named entities).

3.1 A Linear Approach

A naive first approach to construct feature embeddings using context is *additive*, i.e. taking the average over all contexts of a feature f of the average word vector in each context:

$$\mathbf{v}_f^{\text{additive}} = \frac{1}{|\mathcal{C}_f|} \sum_{c \in \mathcal{C}_f} \frac{1}{|c|} \sum_{w \in c} \mathbf{v}_w \tag{1}$$

This formulation reflects the training of commonly used embeddings, which employs additive composition to represent the context (Mikolov et al., 2013; Pennington et al., 2014). It has proved successful in the bag-of-embeddings approach to sentence representation (Wieting et al., 2016; Arora et al., 2017), which can compete with LSTM representations, and has also been given theoretical justification as the *maximum a posteriori* (MAP) context vector under a generative model related to popular embedding objectives (Arora et al., 2016). Lazaridou et al. (2017) use this approach to learn embeddings of unknown word amalgamations, or *chimeras*, given a few context examples.

The additive approach has some limitations because the set of all word vectors is seen to share a few common directions. Simple addition amplifies the component in these directions, at the expense of less common directions that presumably carry more "signal." Stop-word removal can help to ameliorate this (Lazaridou et al., 2017; Herbelot and Baroni, 2017), but does not deal with the fact that content-words also have significant components in the same direction as these deleted words. Another mathematical framework to address this lacuna is to remove the top one or top few principal components, either from the word embeddings themselves (Mu and Viswanath, 2018) or from their summations (Arora et al., 2017). However, this approach is liable to either not remove enough noise or cause too much information loss without careful tuning (c.f. Figure 1).

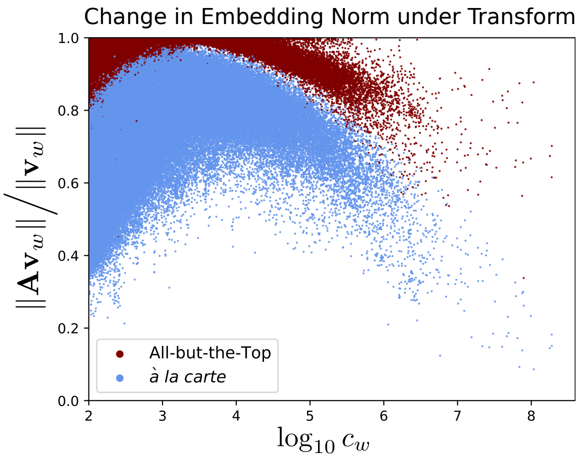


Figure 1: Plot of the ratio of embedding norms after transformation as a function of word count. While All-but-the-Top tends to affect only very frequent words, *à la carte* learns to remove components even from less common words.

We now note that removing the component along the top few principal directions is tantamount to multiplying the additive composition by a fixed (but data-dependent) matrix. Thus a natural extension is to use an arbitrary linear transformation which will be *learned* from the data, and hence guaranteed to do at least as well as any of the above ideas. Specifically, we find the transform that can best recover *existing* word vectors $\mathbf{v}_w$ —which are presumed to be of high quality— from their additive context embeddings $\mathbf{v}_w^{\text{additive}}$. This can be posed as the following linear regression problem

$$\mathbf{v}_w \approx \mathbf{A}\mathbf{v}_w^{\text{additive}} = \mathbf{A}\left(\frac{1}{|\mathcal{C}_w|} \sum_{c \in \mathcal{C}_w} \sum_{w' \in c} \mathbf{v}_{w'}\right) \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{d \times d}$ is learned and we assume for simplicity that $\frac{1}{|c|}$ is constant (e.g. if c has a fixed window size) and is thus subsumed by the transform. After learning the matrix, we can embed any text feature in the same semantic space as the word embeddings via the following expression:

$$\mathbf{v}_f = \mathbf{A}\mathbf{v}_f^{\text{additive}} = \mathbf{A}\left(\frac{1}{|\mathcal{C}_f|} \sum_{c \in \mathcal{C}_f} \sum_{w \in c} \mathbf{v}_w\right) \tag{3}$$

Note that $\mathbf{A}$ is fixed for a given corpus and set of pretrained word embeddings and so does not need to be re-computed to embed different features or feature types.

Algorithm 1: The basic *à la carte* feature embedding induction method. All contexts c consist of sequences of words drawn from the vocabulary $\mathcal{V}$.

```
Data: vocabulary $\mathcal{V}$, corpus $\mathcal{C}_\mathcal{V}$, vectors $\mathbf{v}_w \in \mathbb{R}^d \ \forall\, w \in \mathcal{V}$, feature $f$, corpus $\mathcal{C}_f$ of contexts of $f$
Result: feature embedding $\mathbf{v}_f \in \mathbb{R}^d$
1 for $w \in \mathcal{V}$ do
2     let $\mathcal{C}_w \subset \mathcal{C}_\mathcal{V}$ be the subcorpus of contexts of $w$
3     $\mathbf{u}_w \leftarrow \frac{1}{|\mathcal{C}_w|} \sum_{c \in \mathcal{C}_w} \sum_{w' \in c} \mathbf{v}_{w'}$    // compute each word's context embedding $\mathbf{u}_w$
4 $\mathbf{A} \leftarrow \arg\min_{\mathbf{A} \in \mathbb{R}^{d \times d}} \sum_{w \in \mathcal{V}} \|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2$  // compute context-to-feature transform $\mathbf{A}$
5 $\mathbf{u}_f \leftarrow \frac{1}{|\mathcal{C}_f|} \sum_{c \in \mathcal{C}_f} \sum_{w \in c} \mathbf{v}_w$    // compute feature's context embedding $\mathbf{u}_f$
6 $\mathbf{v}_f \leftarrow \mathbf{A}\mathbf{u}_f$    // transform feature's context embedding
```

Theoretical Justification: As shown by Arora et al. (2018b, Theorem 1), the approximation (2) holds exactly in expectation for some matrix $\mathbf{A}$ when contexts $c \in \mathcal{C}$ are generated by sampling a context vector $\mathbf{v}_c \in \mathbb{R}^d$ from a zero-mean Gaussian with fixed covariance and drawing $|c|$ words using $\mathbb{P}(w|\mathbf{v}_c) \propto \exp\langle \mathbf{v}_c, \mathbf{v}_w \rangle$. The correctness (again in expectation) of (3) under this model is a direct extension. Arora et al. (2018b) use large text corpora to verify their model assumptions, providing theoretical justification for our approach. We observe that the best linear transform $\mathbf{A}$ can recover vectors with mean cosine similarity as high as 0.9 or more with the embeddings used to learn it, thus also justifying the method empirically.

3.2 Practical Details

The basic *à la carte* method, as motivated in Section 3.1 and specified in Algorithm 1, is straightforward and parameter-free (the dimension d is assumed to have been chosen beforehand, along with the other parameters of the original word embeddings). In practice we may wish to modify the regression step in an attempt to learn a better transformation matrix $\mathbf{A}$. However, the standard first approach of using ℓ_2-regularized (Ridge) regression instead of simple linear regression gives little benefit, even when we have more parameters than word embeddings (i.e. when $d^2 > |\mathcal{V}|$).

A more useful modification is to weight each point by some non-decreasing function α of each word's corpus count c_w, i.e. to solve

$$\mathbf{A} = \arg\min_{\mathbf{A} \in \mathbb{R}^{d \times d}} \sum_{w \in \mathcal{V}} \alpha(c_w) \|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2 \qquad (4)$$

where $\mathbf{u}_w$ is the additive context embedding. This reflects the fact that more frequent words likely have better pretrained embeddings. In settings where $|\mathcal{V}|$ is large we find that a hard threshold ($\alpha(c) = \mathbf{1}_{c \geq \tau}$ for some $\tau \geq 1$) is often useful. When we do not have many embeddings we can still give more importance to words with better embeddings via a function such as $\alpha(c) = \log c$, which we use in Section 5.1.

4 One-Shot and Few-Shot Learning of Word Embeddings

While we can use our method to embed any type of text feature, its simplicity and effectiveness is rooted in word-level semantics: the approach assumes pre-existing high quality word embeddings and only considers collocations of features with words rather than with other features. Thus to verify that our approach is reasonable we first check how it performs on word representation tasks, specifically those where word embeddings need to be learned from very few examples. In this section we first investigate how representation quality varies with number of occurrences, as measured by performance on a similarity task that we introduce. We then apply the *à la carte* method to two tasks measuring the ability to learn new or synthetic words from context, achieving strong results on the nonce task of Herbelot and Baroni (2017).

4.1 Similarity Correlation vs. Sample Size

Performance on pairwise word similarity tasks is a standard way to evaluate word embeddings, with success measured via the Spearman correlation between a human score and the cosine similarity between word vectors. An overview of widely used datasets is given by Faruqui and Dyer (2014). However, none of these datasets can be used directly to measure the effect of word frequency on embedding quality, which would help us understand the data requirements of our approach. We address this issue by introducing the *Contextual Rare Words* (CRW) dataset, a subset of 562 pairs from the Rare Word (RW) dataset (Luong et al., 2013) supplemented by 255 sentences (contexts) for each rare word sampled from the Westbury Wikipedia Corpus (WWC) (Shaoul and Westbury, 2010). In addition we provide a subset of the WWC from which all sentences containing these rare words have been removed. The task is to use embeddings trained on this subcorpus to induce rare word embeddings from the sampled contexts.

More specifically, the CRW dataset is constructed using all pairs from the RW dataset where the rarer word occurs between 512 and 10000 times in WWC; this yields a set of 455 distinct rare words. The lower bound ensures that we have a sufficient number of rare word contexts, while the upper bound ensures that a significant fraction of the sentences from the original WWC remain in the subcorpus we provide. In CRW, the first word in every pair is the more frequent word and occurs in the subcorpus, while the second word occurs in the 255 sampled contexts but not in the subcorpus. We provide word2vec embeddings trained on all words occurring at least 100 times in the WWC subcorpus; these vectors include those assigned to the first (non-rare) words in the evaluation pairs.

Evaluation: For every rare word the method under consideration is given eight disjoint subsets containing $1, 2, 4, \dots, 128$ example contexts. The method induces an embedding of the rare word for each subset, letting us track how the quality of rare word vectors changes with more examples. We report the Spearman ρ (as described above) at each sample size, averaged over 100 trials obtained by shuffling each rare word's 255 contexts.

The results in Figure 2 show that our *à la carte* method significantly outperforms the additive baseline (1) and its variants, including stopword removal, SIF-weighting (Arora et al., 2017), and top principal component removal (Mu and Viswanath, 2018). We find that combining SIF-weighting and top component removal also beats these baselines, but still does worse than our method. These experiments consolidate our intuitions from Section 3 that removing common components and frequent words is important and that learning a data-dependent transformation is an effective way to do this. However, if we train

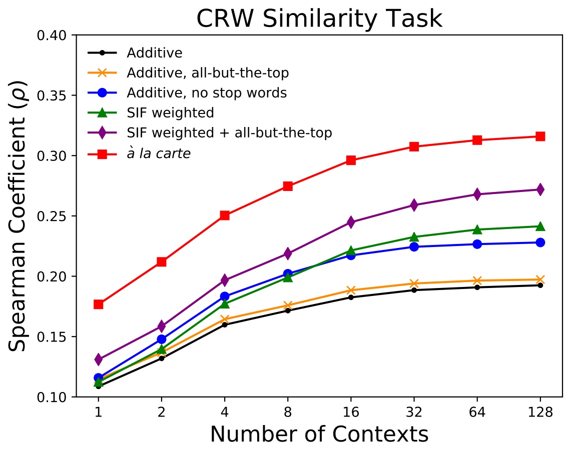


Figure 2: Spearman correlation between cosine similarity and human scores for pairs of words in the CRW dataset given an increasing number of contexts per rare word. Our *à la carte* method outperforms all previous approaches, even when restricted to only eight example contexts.

word2vec embeddings from scratch on the subcorpus together with the sampled contexts we achieve a Spearman correlation of 0.45; this gap between word2vec and our method shows that there remains room for even better approaches for few-shot learning of word embeddings.

4.2 Learning Embeddings of New Concepts: Nonces and Chimeras

We now evaluate our work directly on the tasks posed by Herbelot and Baroni (2017), who developed simple datasets and methods to "simulate the process by which a competent speaker encounters a new word in known contexts." The general goal will be to construct embeddings of new concepts in the same semantic space as a known embedding vocabulary using contextual information consisting of definitions or example sentences.

Nonces: We first discuss the definitional nonce dataset made by the authors themselves, which has a test-set consisting of 300 single-word concepts and their definitions. The task of learning each concept's embedding is simulated by removing or randomly re-initializing its vector and requiring the system to use the remaining embeddings and the definition to make a new vector that is close to the original. Because the embeddings were constructed using data that includes these concepts, an implicit assumption is made that including or excluding one word does not greatly affect the se-

	Nonce (Herbelot and Baroni, 2017)		Chimera (Lazaridou et al., 2017)		
Method	Mean Recip. Rank	Med. Rank	2 Sent.	4 Sent.	6 Sent.
word2vec	0.00007	111012	0.1459	0.2457	0.2498
additive	0.00945	3381	0.3627	0.3701	0.3595
additive, no stop words	0.03686	861	0.3376	0.3624	**0.4080**
nonce2vec	0.04907	623	0.3320	0.3668	0.3890
à la carte	**0.07058**	**165.5**	**0.3634**	**0.3844**	0.3941

Table 1: Comparison with baselines and nonce2vec (Herbelot and Baroni, 2017) on few-shot embedding tasks. Performance on the chimeras task is measured using the Spearman correlation with human ratings. Note that the additive baseline requires removing stop-words in order to improve with more data.

mantic space; this assumption is necessary in order to have a good target vector for the system to be evaluated against.

Using 259,376 word2vec embeddings trained on Wikipedia as the base vectors, Herbelot and Baroni (2017) heavily modify the skip-gram algorithm to successfully learn on one definition, creating the *nonce2vec* system. The original skip-gram algorithm and $\mathbf{v}_w^{\text{additive}}$ are used as baselines, with performance measured as the mean reciprocal rank and median rank of the concept's original vector among the nearest neighbors of the output.

To compare directly to their approach, we use their word2vec embeddings along with contexts from the Wikipedia corpus to construct context vectors $\mathbf{u}_w$ for all words w apart from the 300 nonces. We then learn the *à la carte* transform $\mathbf{A}$, weighting the data points in the regression (4) using a hard threshold of at least 1000 occurrences in Wikipedia. An embedding for each nonce can then be constructed by multiplying $\mathbf{A}$ by the sum over all word embeddings in the nonce's definition. As can be seen in Table 1, this approach significantly improves over both baselines and nonce2vec; the median rank of 165.5 of the original embedding among the nearest neighbors of the nonce vector is very low considering the vocabulary size is more than 250,000, and is also significantly lower than that of all previous methods.

Chimeras: The second dataset Herbelot and Baroni (2017) consider is that of Lazaridou et al. (2017), who construct unseen concepts by combining two related words into a fake nonce word (the "chimera") and provide two, four, or six example sentences for this nonce drawn from sentences containing one of the two component words. The desired nonce embeddings is then evaluated via the correlation of its cosine similarity with the embeddings of several other words, with ratings provided by human judges.

We use the same approach as in the nonce task, except that the chimera embedding is the result of summing over multiple sentences. From Table 1 we see that, while our method is consistently better than both the additive baseline and nonce2vec, removing stop-words from the additive baseline leads to stronger performance for more sentences. Since the *à la carte* algorithm explicitly trains the transform to match the true word embedding rather than human similarity measures, it is perhaps not surprising that our approach is much more dominant on the definitional nonce task.

5 Building Feature Embeddings using Large Corpora

Having witnessed its success at representing unseen words, we now apply the *à la carte* method to two types of feature embeddings: synset embeddings and n-gram embeddings. Using these two examples we demonstrate the flexibility and adaptability of our approach when handling different corpora, base word embeddings, and downstream applications.

5.1 Supervised Synset Embeddings for Word-Sense Disambiguation

Embeddings of synsets, or sets of cognitive synonyms, and related entities such as senses and lexemes have been widely studied, often due to the desire to account for polysemy (Rothe and Schütze, 2015; Iacobacci et al., 2015). Such representations can be evaluated in several ways, including via their use for word-sense disambiguation (WSD), the task of determining a word's sense from context. While current state-of-the-art methods often use powerful recurrent models (Raganato et al., 2017), we will instead use a sim-

	SemEval-2013 Task 12	SemEval-2015 Task 13				
Method	nouns	adj.	nouns	adv.	verbs	comb.
à la carte (SemCor)	60.0	72.2	67.7	85.2	60.6	68.1
à la carte (glosses)	51.8	75.3	62.5	79.0	55.8	64.2
à la carte (combined)	**60.5**	74.1	**70.3**	86.4	59.4	**69.6**
MFS (SemCor)	58.8	**79.5**	60.0	**87.6**	**66.7**	66.8
Raganato et al. (2017)	**66.9**					**72.4**

Table 2: Application of *à la carte* synset embeddings to two standard WSD tasks. As all systems always return exactly one answer, performance is measured in terms of accuracy. Results due to Raganato et al. (2017), who use a bi-LSTM for this task, are given as the recent state-of-the-art result.

ple similarity-based approach that heavily depends on the synset embedding itself and thus serves as a more useful indicator of representation quality. A major target for our simple systems is to beat the most-frequent sense (MFS) method, which returns for each word the sense that occurs most frequently in a corpus such as SemCor. This baseline is "notoriously hard-to-beat," routinely besting many systems in SemEval WSD competitions (Navigli et al., 2013).

Synset Embeddings: We use SemCor (Langone et al., 2004), a subset of the Brown Corpus (BC) (Francis and Kucera, 1979) annotated using PWN synsets. However, because the corpus is quite small we use GloVe trained on Wikipedia instead of on BC itself. The transform $\mathbf{A}$ is learned using context embeddings $\mathbf{u}_w$ computed with windows of size ten around occurrences of w in BC and weighting each word by the log of its count during the regression stage (4). Then we set the context embedding $\mathbf{u}_s$ of each synset s to be the average sum of word embeddings representation over all sentences in SemCor containing s. Finally, we apply the *à la carte* transform to get the synset embedding $\mathbf{v}_s = \mathbf{A}\mathbf{u}_s$.

Sense Disambiguation: To determine the sense of a word w given its context c, we convert c into a vector using the *à la carte* transform $\mathbf{A}$ on the sum of its word embeddings and return the synset s of w whose embedding $\mathbf{v}_s$ is most similar to this vector. We try two different synset embeddings: those induced from SemCor as above and those obtained by embedding a synset using its *gloss*, or PWN-provided definition, in the same way as a nonce in Section 4.2. We also consider a *combined* approach in which we fall back on the gloss vector if the synset does not appear in SemCor and thus has no induced embedding.

As shown in Table 2, synset embeddings induced from SemCor alone beat MFS overall, largely due to good noun results. The method improves further when combined with the gloss approach. While we do not match the state-of-the-art, our success in besting a difficult baseline using very little fine-tuning and exploiting none of the underlying graph structure suggests that the *à la carte* method can learn useful synset embeddings, even from relatively small data.

5.2 N-Gram Embeddings for Classification

As some of the simplest and most useful linguistic features, n-grams have long been a focus of embedding studies. Compositional approaches, such as sums and products of unigram vectors, are often used and work well on some evaluations, but are often order-insensitive or very high-dimensional (Mitchell and Lapata, 2010). Recent work by Poliak et al. (2017) works around this while staying compositional; however, as we will see their approach does not seem to capture a bigram's meaning much better than the sum of its word vectors. n-grams embeddings have also gained interest for low-dimensional document representation schemes (Hill et al., 2016; Pagliardini et al., 2018; Arora et al., 2018a), largely due to the success of their sparse high-dimensional Bag-of-n-Grams (BonG) counterparts (Wang and Manning, 2012). This setting of document embeddings derived from n-gram features will be used for quantitative evaluation in this section.

We build n-gram embeddings using two corpora: 300-dimensional Wikipedia embeddings, which we evaluate qualitatively, and 1600-dimensional embeddings on the Amazon Product Corpus (McAuley et al., 2015), which we use for document classification. For both we use as source embeddings GloVe vectors trained on the respec-

Method	beef up	cutting edge	harry potter	tight lipped
$\mathbf{v}_{w_1}+\mathbf{v}_{w_2}$	meat, out	cut, edges	deathly, azkaban	loose, fitting
$\mathbf{v}^{\text{additive}}_{(w_1,w_2)}$	but, however	which, both	which, but	but, however
ECO	meats, meat	weft, edges	robards, keach	scaly, bristly
Sent2Vec	add, reallocate	science, multidisciplinary	naruto, pokemon	wintel, codebase
à la carte	need, improve	innovative, technology	deathly, hallows	worried, very

Table 3: Closest word embeddings (measured via cosine similarity) to the embeddings of four idiomatic or entity-associated bigrams. From these examples we see that purely compositional methods may struggle to construct context-aware bigram embeddings, even when the features are present in the corpus. On the other hand, adding up corpus contexts (1) is dominated by stop-word information. Sent2Vec is successful on half the examples, reflecting its focus on good sentence, not bigram, embeddings.

tive corpora over words occurring at least a hundred times. Context embeddings are constructed using a window of size ten and a hard threshold at 1000 occurrences is used as the word-weighting function in the regression (4). Unlike Poliak et al. (2017), who can construct arbitrary embeddings but need to train at least two sets of vectors of dimension at least $2d$ to do so, and Yin and Schutze (2014), who determine which n-grams to represent via corpus counts, our *à la carte* approach allows us to train exactly those embeddings that we need for downstream tasks. This, combined with our method's efficiency, allows us to construct more than two million bigram embeddings and more than five million trigram embeddings, constrained only by their presence in the large source corpus.

Qualitative Evaluation: We first compare bigram embedding methods by picking some idiomatic and entity-related bigrams and examining the closest word vectors to their representations. These word-pairs are picked because we expect sophisticated feature embedding methods to encode a better vector than the sum of the two embeddings, which we use as a baseline. From Table 3 we see that embeddings based on corpora rather than composition are better able to embed these bigrams to be close to concepts that are semantically similar. On the other hand, as discussed in Section 3 and evident from these results, the additive context approach is liable to emphasize stop-word directions due to their high frequency.

Document Embedding: Our main application and quantitative evaluation of n-gram vectors is to use them to construct document embeddings. Given a length L document $D=\{w_1,\dots,w_L\}$, we define its embedding $\mathbf{v}_D$ as a weighted concatenation over sums of our induced n-gram embeddings, i.e.

$$\mathbf{v}_D^T = \left(\sum_{t=1}^{L} \mathbf{v}_{w_t}^T \quad \cdots \quad \frac{1}{n} \sum_{t=1}^{L-n+1} \mathbf{v}_{(w_t,\dots,w_{t+n-1})}^T \right)$$

where $\mathbf{v}_{(w_t,\dots,w_{t+n-1})}$ is the embedding of the n-gram $(w_t,\dots,w_{t+n-1})$. Following Arora et al. (2018a), we weight each n-gram component by $\frac{1}{n}$ to reflect the fact that higher-order n-grams have lower quality embeddings because they occur less often in the source corpus. While we concatenate across unigram, bigram, and trigram embeddings to construct our text representations, separate experiments show that simply adding up the vectors of all features also yields a smaller but still substantial improvement over the unigram performance. The higher embedding dimension due to concatenation is in line with previous methods and can also be theoretically supported as yielding a less lossy compression of the n-gram information (Arora et al., 2018a).

In Table 4 we display the result of running cross-validated, ℓ_2-regularized logistic regression on documents from MR movie reviews (Pang and Lee, 2005), CR customer reviews (Hu and Liu, 2004), SUBJ subjectivity dataset (Pang and Lee, 2004), MPQA opinion polarity subtask (Wiebe et al., 2005), TREC question classification (Li and Roth, 2002), SST sentiment classification (binary and fine-grained) (Socher et al., 2013), and IMDB movie reviews (Maas et al., 2011). The first four are evaluated using tenfold cross-validation, while the others have train-test splits.

Despite the simplicity of our embeddings (a concatenation over sums of *à la carte* n-gram vectors), we find that our results are very competitive with many recent unsupervised methods, achieving the best word-level results on two of the tested

Representation	n	d^*	MR	CR	SUBJ	MPQA	TREC	SST (±1)	SST	IMDB
	1	V_1	77.1	77.0	91.0	85.1	86.8	80.7	36.8	88.3
BonG	2	$V_1 + V_2$	77.8	78.1	91.8	85.8	90.0	80.9	39.0	90.0
	3	$V_1 + V_2 + V_3$	77.8	78.3	91.4	85.6	89.8	80.1	42.3	89.8
	1	1600	79.8	81.3	92.6	87.4	85.6	84.1	46.7	89.0
à la carte	2	3200	81.3	83.7	93.5	87.6	89.0	85.8	**47.8**	**90.3**
	3	4800	**81.8**	**84.3**	93.8	87.6	89.0	**86.7**	**48.1**	**90.9**
Sent2Vec[1]	1-2	700	76.3	79.1	91.2	87.2	85.8	80.2	31.0	85.5
DisC[2]	2-3	3200-4800	80.1	81.5	92.6	87.9	90.0	85.5	46.7	89.6
skip-thoughts[3]		4800	80.3	83.8	**94.2**	**88.9**	**93.0**	85.1	45.8	
SDAE[4]		2400	74.6	78.0	90.8	86.9	78.4			
CNN-LSTM[5]		4800	77.8	82.0	93.6	**89.4**	**92.6**			
MC-QT[6]		4800	**82.4**	**86.0**	**94.8**	**90.2**	**92.4**	**87.6**		
byte mLSTM[7]		4096	**86.8**	**90.6**	**94.7**	88.8	90.4	**91.7**	**54.6**	**92.2**

[*] Vocabulary sizes (i.e. BonG dimensions) vary by task; usually 10K-100K.
[1,3,7] (Pagliardini et al., 2018; Kiros et al., 2015; Radford et al., 2017) Evaluation conducted using latest pretrained models. Note that the latest available skip-thoughts implementation returns an error on the IMDB task.
[2,4,5,6] (Arora et al., 2018a; Hill et al., 2016; Gan et al., 2017; Logeswaran and Lee, 2018) Best results from publication.

Table 4: Performance of document embeddings built using *à la carte* n-gram vectors and recent unsupervised word-level approaches on classification tasks, with the character LSTM of (Radford et al., 2017) shown for comparison. Top three results are **bolded** and the best word-level performance is underlined.

datasets. The fact that we do especially well on the sentiment tasks indicates strong exploitation of the Amazon review corpus, which was also used by DisC, CNN-LSTM, and byte mLSTM. At the same time, the fact that our results are comparable to neural approaches indicates that local word-order may contain much of the information needed to do well on these tasks. On the other hand, separate experiments do not show a substantial improvement from our approach over unigram methods such as SIF (Arora et al., 2017) on sentence similarity tasks such as STS (Cer et al., 2017). This could reflect either noise in the n-gram embeddings themselves or the comparative lower importance of local word-order for textual similarity compared to classification.

6 Conclusion

We have introduced *à la carte* embedding, a simple method for representing semantic features using unsupervised context information. A natural and principled integration of recent ideas for composing word vectors, the approach achieves strong performance on several tasks and promises to be useful in many linguistic settings and to yield many further research directions. Of particular interest is the replacement of simple window contexts by other structures, such as dependency parses, that could yield results in domains such as question answering or semantic role labeling. Extensions of the mathematical formulation, such as the use of word weighting when building context vectors as in Arora et al. (2018b) or of spectral information along the lines of Mu and Viswanath (2018), are also worthy of further study.

More practically, the Contextual Rare Words (CRW) dataset we provide will support research on few-shot learning of word embeddings. Both in this area and for n-grams there is great scope for combining our approach with compositional approaches (Bojanowski et al., 2016; Poliak et al., 2017) that can handle settings such as zero-shot learning. More work is needed to understand the usefulness of our method for representing (potentially cross-lingual) entities such as synsets, whose embeddings have found use in enhancing WordNet and related knowledge bases (Camacho-Collados et al., 2016; Khodak et al., 2017). Finally, there remain many language features, such as named entities and morphological forms, whose representation by our method remains unexplored.

Acknowledgments

We thank Karthik Narasimhan and our three anonymous reviewers for helpful suggestions. The work in this paper was in part supported by SRC JUMP, Mozilla Research, NSF grants CCF-1302518 and CCF-1527371, Simons Investigator Award, Simons Collaboration Grant, and ONR-N00014-16-1-2329.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proc. EACL*.

Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018a. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and lstms. In *Proc. ICLR*.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *TACL*.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018b. Linear algebraic structure of word senses, with applications to polysemy. *TACL*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proc. ICLR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. ArXiv.

Danushka Bollegala, David Weir, and John Carroll. 2014. Learning to predict distributions of words across domains. In *Proc. ACL*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *AI*.

Daniel Cer, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. In *Proc. SemEval*.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proc. ACL: System Demonstrations*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

W. Nelson Francis and Henry Kucera. 1979. *Brown Corpus Manual*. Brown University.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proc. EMNLP*.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *JAIR*.

Zellig Harris. 1954. Distributional structure. *Word*, 10:146–162.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: Acquiring new word vectors from tiny data. In *Proc. EMNLP*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proc. NAACL*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. KDD*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proc. ACL-IJCNLP*.

Mikhail Khodak, Andrej Risteski, Christiane Fellbaum, and Sanjeev Arora. 2017. Automated wordnet construction using word embeddings. In *Proc. Workshop on Sense, Concept and Entity Representations and their Applications*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Adv. NIPS*.

Helen Langone, Benjamin R. Haskell, and George A. Miller. 2004. Annotating wordnet. In *Proc. Workshop on Frontiers in Corpus Annotation*.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive Science*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proc. COLING*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *Proc. ICLR*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proc. CoNLL*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proc. ACL-HLT*.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proc. KDD*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Adv. NIPS*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*.

Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proc. SemEval*.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective post-processing for word representations. In *Proc. ICLR*.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Proc. SemEval*.

Sebastian Pado, Aurelie Herbelot, Max Kisselew, and Jan Snajder. 2016. Predictability of distributional semantics in derivational word formation. In *Proc. COLING*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proc. NAACL*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP*.

Adam Poliak, Pushpendre Rastogia, M. Patrick Martin, and Benjamin Van Durme. 2017. Efficient, compositional, order-sensitive n-gram embeddings. In *Proc. EACL*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. ArXiv.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *Proc. EMNLP*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proc. ACL-IJCNLP*.

Cyrus Shaoul and Chris Westbury. 2010. The westbury lab wikipedia corpus.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

Dingquan Wang, Nanyun Peng, and Kevin Duh. 2017. A multi-task learning approach to adapting bilingual word embeddings for cross-lingual named entity recognition. In *Proc. IJCNLP*.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proc. ACL*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Proc. LREC*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proc. ICLR*.

Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017. Starspace: Embed all the things! ArXiv.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proc. CoNLL*.

Wenpeng Yin and Hinrich Schutze. 2014. An exploration of embeddings for generalized phrases. In *Proc. ACL 2014 Student Research Workshop*.

Unsupervised Learning of Distributional Relation Vectors

Shoaib Jameel
School of Computing
Medway Campus
University of Kent, UK
M.S.Jameel@kent.ac.uk

Zied Bouraoui
CRIL CNRS and
Artois University
France
bouraoui@cril.fr

Steven Schockaert
School of Computer Science
and Informatics
Cardiff University, UK
schockaerts1@cardiff.ac.uk

Abstract

Word embedding models such as GloVe rely on co-occurrence statistics to learn vector representations of word meaning. While we may similarly expect that co-occurrence statistics can be used to capture rich information about the relationships between different words, existing approaches for modeling such relationships are based on manipulating pre-trained word vectors. In this paper, we introduce a novel method which directly learns relation vectors from co-occurrence statistics. To this end, we first introduce a variant of GloVe, in which there is an explicit connection between word vectors and PMI weighted co-occurrence vectors. We then show how relation vectors can be naturally embedded into the resulting vector space.

1 Introduction

Word embeddings are vector space representations of word meaning (Mikolov et al., 2013b; Pennington et al., 2014). A remarkable property of these models is that they capture various lexical relationships, beyond mere similarity. For example, (Mikolov et al., 2013b) found that analogy questions of the form "a is to b what c is to ?" can often be answered by finding the word d that maximizes $\cos(w_b - w_a + w_c, w_d)$, where we write w_x for the vector representation of a word x.

Intuitively, the word vector w_a represents a in terms of its most salient features. For example, w_{paris} implicitly encodes that Paris is located in France and that it is a capital city, which is intuitively why the 'capital of' relation can be modeled in terms of a vector difference. Other relationships, however, such as the fact that Macron succeeded Hollande as president of France, are unlikely to be captured by word embeddings. Relation extraction methods can discover such information by analyzing sentences that contain both of the words or entities involved (Mintz et al., 2009; Riedel et al., 2010; dos Santos et al., 2015), but they typically need a large number of training examples to be effective.

A third alternative, which we consider in this paper, is to characterize the relatedness between two words s and t by learning a relation vector r_{st} in an unsupervised way from corpus statistics. Among others, such vectors can be used to find word pairs that are similar to a given word pair (i.e. finding analogies), or to find the most prototypical examples among a given set of relation instances. They can also be used as an alternative to the aforementioned relation extraction methods, by subsequently training a classifier that uses the relation vectors as input, which might be particularly effective in cases where only limited amounts of training data are available (with the case of analogy finding from a single instance being an extreme example).

The most common unsupervised approach for learning relation vectors consists of averaging the embeddings of the words that occur in between s and t, in sentences that contain both (Weston et al., 2013; Fan et al., 2015; Hashimoto et al., 2015). While this strategy is often surprisingly effective (Hill et al., 2016), it is sub-optimal for two reasons. First, many of the words co-occurring with s and t will be semantically related to s or to t, but will not actually be descriptive for the relationship between s and t; e.g. the vector describing the relation between *Paris* and *France* should not be affected by words such as *eiffel* (which only relates to Paris). Second, it gives too much weight to stop-words, which cannot be addressed in a straightforward way as some stop-words are actually crucial for modeling relationships (e.g. prepositions such as 'in' or 'of' or Hearst patterns (Indurkhya and Damerau, 2010)).

In this paper, we propose a method for learning relation vectors directly from co-occurrence statistics. We first introduce a variant of GloVe, in which word vectors can be directly interpreted as smoothed PMI-weighted bag-of-words representations. We then represent relationships between words as weighted bag-of-words representations, using generalizations of PMI to three arguments, and learn vectors that correspond to smoothed versions of these representations.

As far as the possible applications of our methodology is concerned, we imagine that relation vectors can be used in various ways to enrich the input to neural network models. As a simple example, in a question answering system, we could "annotate" mentions of entities with relation vectors encoding their relationship to the different words from the question. As another example, we could consider a recommendation system which takes advantage of vectors expressing the relationship between items that have been bought (or viewed) by a customer and other items from the catalogue. Finally, relation vectors should also be useful for knowledge completion, especially in cases where few training examples per relation type are given (meaning that neural network models could not be used) and where relations cannot be predicted from the already available knowledge (meaning that knowledge graph embedding methods could not be used, or are at least not sufficient).

2 Related Work

The problem of characterizing the relationship between two words has been studied in various settings. From a learning point of view, the most straightforward setting is where we are given labeled training sentences, with each label explicitly indicating what relationship is expressed in the sentence. This fully supervised setting has been the focus of several evaluation campaigns, including as part of ACE (Doddington et al., 2004) and at SemEval 2010 (Hendrickx et al., 2010). A key problem with this setting, however, is that labeled training data is hard to obtain. A popular alternative is to use known instances of the relations of interest as a form of distant supervision (Mintz et al., 2009; Riedel et al., 2010). Some authors have also considered unsupervised relation extraction methods (Shinyama and Sekine, 2006; Banko et al., 2007), in which case the aim is essentially to find clusters of patterns that express similar relationships, although these relationships may not correspond to the ones that are needed for the considered application. Finally, several systems have also used bootstrapping strategies (Brin, 1998; Agichtein and Gravano, 2000; Carlson et al., 2010), where a small set of instances are used to find extraction patterns, which are used to find more instances, which can in turn be used to find better extraction patterns, etc.

Traditionally, relation extraction systems have relied on a variety of linguistic features, such as lexical patterns, part-of-speech tags and dependency parsers. More recently, several neural network architectures have been proposed for the relation extraction problem. These architectures rely on word embeddings to represent the words in the input sentence, and manipulate these word vectors to construct a relation vector. Some approaches simply represent the sentence (or the phrase connecting the entities whose relationship we want to determine) as a sequence of words, and use e.g. convolutional networks to aggregate the vectors of the words in this sequence (Zeng et al., 2014; dos Santos et al., 2015). Another possibility, explored in (Socher et al., 2012), is to use parse trees to capture the structure of the sentence, and to use recursive neural networks (RNNs) to aggregate the word vectors in a way which respects this structure. A similar approach is taken in (Xu et al., 2015), where LSTMs are applied to the shortest path between the two target words in a dependency parser. A straightforward baseline method is to simply take the average of the word vectors (Mitchell and Lapata, 2010). While conceptually much simpler, variants of this approach have obtained state-of-the-art performance for relation classification (Hashimoto et al., 2015) and a variety of tasks that require sentences to be represented as a vector (Hill et al., 2016).

Given the effectiveness of word vector averaging, in (Kenter et al., 2016) a model was proposed that explicitly tries to learn word vectors that generalize well when being averaged. Similarly, the model proposed in (Hashimoto et al., 2015) aims to produce word vectors that perform well for the specific task of relation classification. The ParagraphVector method from (Le and Mikolov, 2014) is related to the aformentioned approaches, but it explicitly learns a vector representation for each paragraph along with the word embeddings. However, this method is computationally expensive, and often fails to outperform simpler approaches (Hill et al., 2016).

To the best of our knowledge, existing methods for learning relation vectors are all based on manipulating pre-trained word vectors. In contrast, we will directly learn relation vectors from corpus statistics, which will have the important advantage that we can focus on words that describe the interaction between the two words s and t, i.e. words that commonly occur in sentences that contain both s and t, but are comparatively rare in sentences that only contain s or only contain t.

Finally, note that our work is fundamentally different from Knowledge Graph Embedding (KGE) (Wang et al., 2014b), (Wang et al., 2014a), (Bordes et al., 2011) in at least two ways: (i) KGE models start from a structured knowledge graph whereas we only take a text corpus as input, and (ii) KGE models represent relations as geometric objects in the "entity embedding" itself (e.g. as translations, linear maps, combinations of projections and translations, etc), whereas we represent words and relations in different vector spaces.

3 Word Vectors as PMI Encodings

Our approach to relation embedding is based on a variant of the GloVe word embedding model (Pennington et al., 2014). In this section, we first briefly recall the GloVe model itself, after which we discuss our proposed variant. A key advantage of this variant is that it allows us to directly interpret word vectors in terms of the Pointwise Mutual Information (PMI), which will be central to the way in which we learn relation vectors.

3.1 Background

The GloVe model (Pennington et al., 2014) learns a vector w_i for each word i in the vocabulary, based on a matrix of co-occurrence counts, encoding how often two words appear within a given window. Let us write x_{ij} for the number of times word j appears in the context of word i in some text corpus. More precisely, assume that there are m sentences in the corpus, and let $\mathcal{P}_i^l \subseteq \{1, ..., n_l\}$ be the set of positions from the l^{th} sentence where the word i can be found (with n_l the length of the sentence). Then x_{ij} is defined as follows:

$$\sum_{l=1}^{m} \sum_{p \in \mathcal{P}_i^l} \sum_{q \in \mathcal{P}_j^l} \mathit{weight}(p, q)$$

where $\mathit{weight}(p,q) = \frac{1}{|p-q|}$ if $0 < |p-q| \leq W$, and $\mathit{weight}(p,q) = 0$ otherwise, where the window size W is usually set to 5 or 10.

The GloVe model learns for each word i two vectors w_i and $\tilde{w}_i$ by optimizing the following objective:

$$\sum_i \sum_{j: x_{ij} \neq 0} f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b_j} - \log x_{ij})^2$$

where f is a weighting function, aimed at reducing the impact of rare terms, and b_i and $\tilde{b_j}$ are bias terms. The GloVe model is closely related to the notion of pointwise mutual information (PMI), which is defined for two words i and j as $\mathit{PMI}(i,j) = \log\left(\frac{P(i,j)}{P(i)P(j)}\right)$, where $P(i,j)$ is the probability of seeing the words i and j if we randomly pick a word position from the corpus and a second word position within distance W from the first position. The PMI between i and j is usually estimated as follows:

$$\mathit{PMI}_X(i,j) = \log\left(\frac{x_{ij} x_{**}}{x_{i*} x_{*j}}\right)$$

where $x_{i*} = \sum_j x_{ij}$, $x_{*j} = \sum_i x_{ij}$ and $x_{**} = \sum_i \sum_j x_{ij}$. In particular, it is straightforward to see that after the reparameterization given by $b_i \mapsto b_i + \log x_{i*} - \log x_{**}$ and $b_j \mapsto b_j + \log x_{*j}$, the GloVe model is equivalent to

$$\sum_i \sum_{\substack{j \\ x_{ij} \neq 0}} f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b_j} - \mathit{PMI}_X(i,j))^2 \tag{1}$$

3.2 A Variant of GloVe

In this paper, we will use the following variant of the formulation in (1):

$$\sum_i \sum_{j \in J_i} \frac{1}{\sigma_j^2}(w_i \cdot \tilde{w}_j + \tilde{b_j} - \mathit{PMI}_S(i,j))^2 \tag{2}$$

Despite its similarity, this formulation differs from the GloVe model in a number of important ways. First, we use smoothed frequency counts instead of the observed frequency counts x_{ij}. In particular, the PMI between words i and j is given as:

$$\mathit{PMI}_S(i,j) = \log\left(\frac{P(i,j)}{P(i)P(j)}\right)$$

where the probabilities are estimated as follows:

$$P(i) = \frac{x_{i*} + \alpha}{x_{**} + n\alpha} \qquad P(j) = \frac{x_{*j} + \alpha}{x_{**} + n\alpha}$$
$$P(i,j) = \frac{x_{ij} + \alpha}{x_{**} + n^2\alpha}$$

where $\alpha \geq 0$ is a parameter controlling the amount of smoothing and n is the size of the vocabulary. This ensures that the estimation of $\mathit{PMI}(i,j)$ is well-defined even in cases where $x_{ij} = 0$, meaning that we no longer have to restrict the inner summation to those j for which $x_{ij} > 0$. For efficiency reasons, in practice, we only consider a small subset of all context words j for which $x_{ij} = 0$, which is similar in spirit to the use of negative sampling in Skip-gram (Mikolov et al., 2013b). In particular, the set J_i contains each j such that $x_{ij} > 0$ as well as M uniformly[1] sampled context words j for which $x_{ij} = 0$, where we choose $M = 2 \cdot |\{j : x_{ij} > 0\}|$.

Second, following (Jameel and Schockaert, 2016), the weighting function $f(x_{ij})$ has been replaced by $\frac{1}{\sigma_j^2}$, where σ_j^2 is the residual variance of the regression problem for context word j, estimated follows:

$$\sigma_j^2 = \frac{1}{|J_j^{-1}|} \sum_{i \in J_j^{-1}} (w_i \cdot \tilde{w_j} + \tilde{b_j} - \mathit{PMI}_S(i,j))^2$$

with $J_j^{-1} = \{i : j \in J_i\}$. Since we need the word vectors to estimate this residual variance, we re-estimate σ_j^2 after every five iterations of the SGD optimization. For the first 5 iterations, where no estimation for σ_j^2 is available, we use the GloVe weighting function.

The use of smoothed frequency counts and residual variance based weighting make the word embedding model more robust for rare words. For instance, if w only co-occurs with a handful of other terms, it is important to prioritize the most informative context words, which is exactly what the use of the residual variance achieves, i.e. σ_j^2 is small for informative terms and large for stop words; see (Jameel and Schockaert, 2016). This will be important for modeling relations, as the relation vectors will often have to be estimated from very sparse co-occurrence counts.

[1]While the negative sampling method used in Skip-gram favors more frequent words, initial experiments suggested that deviating from a uniform distribution almost had no impact in our setting.

Finally, the bias term b_i has been omitted from the model in (2). We have empirically found that omitting this bias term does not affect the performance of the model, while it allows us to have a more direct connection between the vector w_i and the corresponding PMI scores.

3.3 Word Vectors and PMI

Let us define PMI_W as follows:

$$\mathit{PMI}_W(i,j) = w_i{\cdot}\tilde{w_j} + \tilde{b_j}$$

Clearly, when the word vectors are trained according to (2), it holds that $\mathit{PMI}_W(i,j) \approx \mathit{PMI}_S(i,j)$. In other words, we can think of the word vector w_i as a low-dimensional encoding of the vector $(\mathit{PMI}_S(i,1), ..., \mathit{PMI}_S(i,n))$, with n the number of words in the vocabulary. This view allows us to assign a natural interpretation to some word vector operations. In particular, the vector difference $w_i - w_k$ is commonly used as a model for the relationship between words i and k. For a given context word j, we have

$$(w_i - w_k) \cdot \tilde{w_j} = \mathit{PMI}_W(i,j) - \mathit{PMI}_W(k,j)$$

The latter is an estimation of $\log\left(\frac{P(i,j)}{P(i)P(j)}\right) - \log\left(\frac{P(k,j)}{P(k)P(j)}\right) = \log\left(\frac{P(j|i)}{P(j|k)}\right)$. In other words, the vector translation $w_i - w_k$ encodes for each context word j the (log) ratio of the probability of seeing j in the context of i and in the context of k, which is in line with the original motivation underlying the GloVe model (Pennington et al., 2014). In the following section, we will propose a number of alternative vector representations for the relationship between two words, based on generalizations of PMI to three arguments.

4 Learning Global Relation Vectors

We now turn to the problem of learning a vector r_{ik} that encodes how the source word i and target word k are related. The main underlying idea is that r_{ik} will capture which context words j are most closely associated with the word pair (i,k). Whereas the GloVe model is based on statistics about (*main word*, *context word*) pairs, here we will need statistics on (*source word*, *context word*, *target word*) triples. First, we discuss how co-occurrence statistics among three words can be expressed using generalizations of PMI to three arguments. Then we explain how this can be used to learn relation vectors in natural way.

4.1 Co-occurrence Statistics for Triples

Let $\mathcal{P}_i^l \subseteq \{1, ..., n_l\}$ again be the set of positions from the l^{th} sentence corresponding to word i. We define:

$$y_{ijk} = \sum_{l=1}^{m} \sum_{p \in \mathcal{P}_i^l} \sum_{q \in \mathcal{P}_j^l} \sum_{r \in \mathcal{P}_k^l} \mathit{weight}(p, q, r)$$

where $\mathit{weight}(p, q, r) = \max(\frac{1}{q-p}, \frac{1}{r-q})$ if $p < q < r$ and $r - p \leq W$, and $\mathit{weight}(p, q, r) = 0$ otherwise. In other words, y_{ijk} reflects the (weighted) number of times word j appears between words i and k in a sentence in which i and k occur sufficiently close to each other, in that order. Note that by taking word order into account in this way, we will be able to model asymmetric relationships.

To model how strongly a context word j is associated with the word pair (i, k), we will consider the following two well-known generalizations of PMI to three arguments (Van de Cruys, 2011):

$$\mathit{SI}^1(i, j, k) = \log\left(\frac{P(i, j)P(i, k)P(j, k)}{P(i)P(j)P(k)P(i, j, k)}\right)$$
$$\mathit{SI}^2(i, j, k) = \log\left(\frac{P(i, j, k)}{P(i)P(j)P(k)}\right)$$

where $P(i, j, k)$ is the probability of seeing the word triple (i, j, k) when randomly choosing a sentence and three (ordered) word positions in that sentence within a window size of W. In addition we will also consider two ways in which PMI can be used more directly:

$$\mathit{SI}^3(i, j, k) = \log\left(\frac{P(i, j, k)}{P(i, k)P(j)}\right)$$
$$\mathit{SI}^4(i, j, k) = \log\left(\frac{P(i, k|j)}{P(i|j)P(k|j)}\right)$$

Note that $\mathit{SI}^3(i, j, k)$ corresponds to the PMI between (i, k) and j, whereas $\mathit{SI}^4(i, j, k)$ is the PMI between i and k conditioned on the fact that j occurs. The measures SI^3 and SI^4 are closely related to SI^1 and SI^2 respectively[2]. In particular, the following identities are easy to show:

$$\mathit{PMI}(i, j) + \mathit{PMI}(j, k) - \mathit{SI}^1(i, j, k) = \mathit{SI}^3(i, j, k)$$
$$\mathit{SI}^2(i, j, k) - \mathit{PMI}(i, j) - \mathit{PMI}(j, k) = \mathit{SI}^4(i, j, k)$$

[2]Note that probabilities of the form $P(i, j)$ or $P(i)$ here refer to marginal probabilities over ordered triples. In contrast, the PMI scores from the word embedding model are based on probabilities over unordered word pairs, as is common for word embeddings.

Using smoothed versions of the counts y_{ijk}, we can use the following probability estimates for $\mathit{SI}^1(i, j, k)$–$\mathit{SI}^4(i, j, k)$:

$$P(i, j, k) = \frac{y_{ijk} + \alpha}{y_{***} + n^3\alpha} \qquad P(i, j) = \frac{y_{ij*} + \alpha}{y_{***} + n^2\alpha}$$
$$P(i, k) = \frac{y_{i*k} + \alpha}{y_{***} + n^2\alpha} \qquad P(j, k) = \frac{y_{*jk} + \alpha}{y_{***} + n^2\alpha}$$
$$P(i) = \frac{y_{i**} + \alpha}{y_{***} + n\alpha} \qquad P(j) = \frac{y_{*j*} + \alpha}{y_{***} + n\alpha}$$
$$P(k) = \frac{y_{**k} + \alpha}{y_{***} + n\alpha}$$

where $y_{ij*} = \sum_k y_{ijk}$, and similar for the other counts. For efficiency reasons, the counts of the form y_{ij*}, y_{i*k} and y_{*jk} are pre-computed for all word pairs, which can be done efficiently due to the sparsity of co-occurrence counts (i.e. these counts will be 0 for most pairs of words), similarly to how to the counts x_{ij} are computed in GloVe. From these counts, we can also efficiently pre-compute the counts y_{i**}, y_{*j*}, y_{**k} and y_{***}. On the other hand, the counts y_{ijk} cannot be pre-computed, since the total number of triples for which $y_{ijk} \neq 0$ is prohibitively high in a typical corpus. However, using an inverted index, we can efficiently retrieve the sentences that contain the words i and k, and since this number of sentences is typically small, we can efficiently obtain the counts y_{ijk} corresponding to a given pair (i, k) whenever they are needed.

4.2 Relation Vectors

Our aim is to learn a vector r_{ik} that models the relationship between i and k. Computing such a vector for each pair of words (which co-occur at least once) is not feasible, given the number of triples (i, j, k) that would need to be considered. Instead, we first learn a word embedding, by optimizing (2). Then, fixing the context vectors $\tilde{w}_j$ and bias terms b_j, we learn a vector representation for a given pair (i, k) of interest by solving the following objective:

$$\sum_{j \in J_{i,k}} (r_{ik} \cdot \tilde{w}_j + \tilde{b_j} - \mathit{SI}(i, j, k))^2 \qquad (3)$$

where SI refers to one of $\mathit{SI}_S^1, \mathit{SI}_S^2, \mathit{SI}_S^3, \mathit{SI}_S^4$. Note that (3) is essentially the counterpart of (1), where we have replaced the role of the PMI measure by SI. In this way, we can exploit the representations of the context words from the word embedding model for learning relation vectors. Note that the factor $\frac{1}{\sigma_j^2}$ has been omitted. This is because words j that are normally relatively uninformative (e.g. stop words), for which σ_j^2 would be high, can actually be very important for characterizing the relationship between i and k. For instance, the phrase "X such as Y" clearly suggests a hyponomy relationship between X and Y, but both 'such' and 'as' would be associated with a high residual variance σ_j^2. The set $J_{i,k}$ contains every j for which $y_{ijk} > 0$ as well as a random sample of m words for which $y_{ijk} = 0$, where $m = 2 \cdot |\{j : y_{ijk} > 0|$. Note that because $\tilde{w_j}$ is now fixed, (3) is a linear least squares regression problem, which can be solved exactly and efficiently.

The vector r_{ik} is based on words that appear between i and k. In the same way, we can learn a vector s_{ik} based on the words that appear before i and a vector t_{ik} based on the words that appear after k, in sentences where i occurs before k. Furthermore, we also learn vectors r_{ki}, s_{ki} and t_{ki} from the sentences where k occurs before i. As the final representation R_{ik} of the relationship between i and k, we concatenate the vectors $r_{ik}, r_{ki}, s_{ik}, s_{ki}, t_{ik}, t_{ki}$ as well as the word vectors w_i and w_k. We write R^l_{ik} to denote the vector that results from using measure *SI*^{l} ($l \in \{1, 2, 3, 4\}$).

5 Experimental Results

In our experiments, we have used the Wikipedia dump from November 2nd, 2015, which consists of 1,335,766,618 tokens. We have removed punctuations and HTML/XML tags, and we have lowercased all tokens. Words with fewer than 10 occurrences have been removed from the corpus. To detect sentence boundaries, we have used the Apache sentence segmentation tool. In all our experiments, we have set the number of dimensions to 300, which was found to be a good choice in previous work, e.g. (Pennington et al., 2014). We use a context window size W of 10 words. The number of iterations for SGD was set to 50. For our model, we have tuned the smoothing parameter α based on held-out tuning data, considering values from $\{0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001\}$. We have noticed that in most of the cases the value of α was automatically selected as 0.00001. To efficiently compute the triples, we have used the Zettair[3] retrieval engine.

As our main baselines, we use three popular unsupervised methods for constructing relation vectors. First, *Diff* uses the vector difference $w_k - w_i$, following the common strategy of modeling relations as vector differences, as e.g. in (Vylomova et al., 2016). Second, *Conc* uses the concatenation of w_i and w_k. This model is more general than *Diff* but it uses twice as many dimensions, which may make it harder to learn a good classifier from few examples. The use of concatenations is popular e.g. in the context of hypernym detection (Baroni et al., 2012). Finally, *Avg* averages the vector representations of the words occurring in sentences that *Diff*, contain i and k. In particular, let r^{avg}_{ik} be obtained by averaging the word vectors of the context words appearing between i and k for each sentence containing i and k (in that order), and then averaging the vectors obtained from each of these sentences. Let s^{avg}_{ik} and t^{avg}_{ik} be similarly obtained from the words occurring before i and the words occurring after k respectively. The considered relation vector is then defined as the concatenation of r^{avg}_{ik}, r^{avg}_{ki}, s^{avg}_{ik}, s^{avg}_{ki}, t^{avg}_{ik}, t^{avg}_{ki}, w_i and w_k. The *Avg* will allow us to directly compare how much we can improve relation vectors by deviating from the common strategy of averaging word vectors.

Table 1: Results for the relation induction task.

Google Analogy							
	Diff	Conc	Avg	R^1_{ik}	R^2_{ik}	R^3_{ik}	R^4_{ik}
Acc	90.0	89.0	89.9	90.0	**92.3**	90.9	90.4
Pre	81.6	78.7	80.8	79.9	87.1	83.2	81.1
Rec	82.6	83.9	83.9	86.0	84.8	84.8	85.5
F1	82.1	81.2	82.3	82.8	**85.9**	84.0	83.3
DiffVec							
	Diff	Conc	Avg	R^1_{ik}	R^2_{ik}	R^3_{ik}	R^4_{ik}
Acc	29.5	28.9	29.7	29.7	**31.3**	30.4	30.1
Pre	19.6	18.7	20.4	21.5	22.9	21.9	22.3
Rec	23.8	22.9	23.7	24.5	25.7	25.3	22.9
F1	21.5	20.6	21.9	22.4	**24.2**	23.5	22.6

5.1 Relation Induction

In the relation induction task, we are given word pairs $(s_1, t_1), ..., (s_k, t_k)$ that are related in some way, and the task is to decide for a number of test examples (s, t) whether they also have this relationship. Among others, this task was considered in (Vylomova et al., 2016), and a ranking version of this task was studied in (Drozd et al., 2016). As test sets we use the Google Analogy Test Set (Mikolov et al., 2013a), which contains instances of 14 different types of relations, and the DiffVec dataset, which was introduced in (Vylomova et al., 2016). This dataset contains instances of 36 dif-

[3]http://www.seg.rmit.edu.au/zettair/

Table 2: Results for the relation induction task using alternative word embedding models.

	GloVe				SkipGram				CBOW			
	Google		DiffVec		Google		DiffVec		Google		DiffVec	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Diff	90.0	81.9	21.2	13.9	89.8	81.9	21.7	14.5	89.9	82.1	17.4	9.7
Conc	88.9	80.4	20.2	11.9	89.2	81.6	20.5	12.0	89.1	81.1	16.4	7.7
Avg	89.8	82.1	21.4	13.9	90.2	82.4	21.8	14.4	89.8	82.2	17.5	10.0
R^1_{ik}	89.7	81.7	20.9	12.5	89.4	81.2	21.1	12.3	89.8	81.9	17.2	9.2
R^2_{ik}	90.0	82.8	21.2	13.4	89.1	81.3	21.1	12.9	90.2	82.4	17.7	10.0
R^3_{ik}	90.0	82.3	20.0	11.2	89.5	81.1	20.5	12.3	89.5	81.1	17.2	9.6
R^4_{ik}	90.0	82.5	20.0	11.4	88.9	80.8	20.6	12.1	90.5	82.2	17.1	8.4

ferent types of relations[4]. Note that both datasets contain a mix of semantic and syntactic relations.

In our evaluation, we have used 10-fold cross-validation (or leave-one-out for relations with fewer than 10 instances). In the experiments, we consider for each relation in the test set a separate binary classification task, which was found to be considerably more challenging than a multi-class classification setting in (Vylomova et al., 2016). To generate negative examples in the training data (resp. test data), we have used three strategies, following (Vylomova et al., 2016). First, for a given positive example (s,t) of the considered relation, we add (t,s) as a negative example. Second, for each positive example (s,t), we generate two negative examples (s,t_1) and (s,t_2) by randomly selecting two tail words t_1,t_2 from the other training (resp. test) examples of the same relation. Finally, for each positive example, we also generate a negative example by randomly selecting two words from the vocabulary. For each relation, we then train a linear SVM classifier. To set the parameters of the SVM, we initially use 25% of the training data for tuning, and then retrain the SVM with the optimal parameters on the full training data.

The results are summarized in Table 1 in terms of accuracy and (macro-averaged) precision, recall and F1 score. As can be observed, our model outperforms the baselines on both datasets, with the R^2_{ik} variant outperforming the others.

To analyze the benefit of our proposed word embedding variant, Table 2 shows the results that were obtained when we use standard word embedding models. In particular, we show results for the standard GloVe model, SkipGram and the Continuous Bag of Words (CBOW) model. As can be observed, our variant leads to better results than the original GloVe model, even for the baselines.

[4]Note that in contrast to (Vylomova et al., 2016) we use all 36 relations from this dataset, including those with very few instances.

Table 3: Relation induction without position weighting (left) and without the relation vectors s_{ik} and t_{ik} (right).

	Google		DiffVec	
	Acc	F1	Acc	F1
R^1_{ik}	89.7	82.4	30.2	22.2
R^2_{ik}	91.0	83.4	30.8	24.1
R^3_{ik}	90.4	83.2	30.1	22.3
R^4_{ik}	90.2	82.9	29.1	21.2

	Google		DiffVec	
	Acc	F1	Acc	F1
R^1_{ik}	90.0	82.5	29.9	22.3
R^2_{ik}	92.3	85.8	31.2	24.2
R^3_{ik}	90.5	83.2	30.2	23.0
R^4_{ik}	90.3	83.1	29.8	22.3

The difference is particularly noticeable for DiffVec. The difference is also larger for our relation vectors than for the baselines, which is expected as our method is based on the assumption that context word vectors can be interpreted in terms of PMI scores, which is only true for our variant.

Similar as in the GloVe model, the context words in our model are weighted based on their distance to the nearest target word. Table 3 shows the results of our model without this weighting, for the relation induction task. Comparing these results with those in Table 1 shows that the weighting scheme indeed leads to a small improvement (except for the accuracy of R^1_{ik} for DiffVec). Similarly, in Table 3, we show what happens if the relation vectors s_{ik}, s_{ki}, t_{ik} and t_{ki} are omitted. In other words, for the results in Table 3, we only use context words that appear between the two target words. Again, the results are worse than those in Table 1 (with the accuracy of R^1_{ik} for DiffVec again being an exception), although the differences are very small in this case. While including the vectors s_{ik}, s_{ki}, t_{ik}, t_{ki} should be helpful, it also significantly increases the dimensionality of the vectors R^l_{ik}. Given that the number of instances per relation is typically quite small for this

Table 4: Results for measuring degrees of prototypicality (Spearman $\rho \times 100$).

Diff	Conc	Avg	R^1_{ik}	R^2_{ik}	R^3_{ik}	R^4_{ik}
17.3	16.7	21.1	22.7	**23.9**	21.8	22.2

task, this can also make it harder to learn a suitable classifier.

5.2 Measuring Degrees of Prototypicality

Instances of relations can often have different degrees of prototypicality. For example, for the relation "X characteristically makes the sound Y", the pair (*dog*,*bark*) should be considered more prototypical than the pair (*floor*,*squeak*), even though both pairs might be considered to be instances of the relation (Jurgens et al., 2012). A suitable relation vector should allow us to rank word pairs according to how prototypical they are as instances of that relation. We evaluate this ability using a dataset that was produced in the aftermath of SemEval 2012 Task 2. In particular, we have used the "Phase2AnswerScaled" data from the platinum rankings dataset, which is available from the SemEval 2012 Task 2 website[5]. In this dataset, 79 ranked list of word pairs are provided, each of which corresponds to a particular relation. For each relation, we first split the associated ranking into 60% training, 20% tuning, and 20% testing (i.e. we randomly select 60% of the word pairs and use their ranking as training data, and similar for tuning and test data). We then train a linear SVM regression model on the ranked word pairs. Note that this task slightly differs from the task that was considered at SemEval 2012, to allow us to use an SVM based model for consistency with the rest of the paper.

We report results using Spearman's ρ in Table 4. Our model again outperforms the baselines, with R^2_{ik} again being the best variant. Interestingly, in this case, the Avg baseline is considerably stronger than Diff and Conc. Intuitively, we might indeed expect that this ranking problem requires a more fine-grained representation than the relation induction setting. Note that the Diff representations were found to achieve near state-of-the-art performance on a closely related task in (Zhila et al., 2013). The only model that was found to perform (slightly) better was a hybrid model, combining Diff representations with linguistic patterns (inspired by (Rink and Harabagiu, 2012)) and lexical databases, among others.

[5]https://sites.google.com/site/semeval2012task2/download

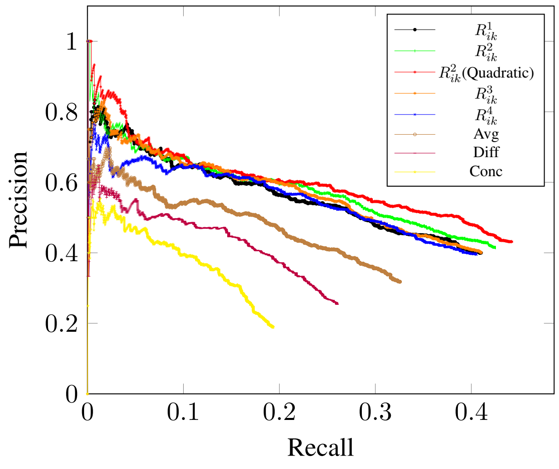


Figure 1: Results for the relation extraction from the NYT corpus: comparison with the main baselines.

5.3 Relation Extraction

Finally, we consider the problem of relation extraction from a text corpus. Specifically, we consider the task proposed in (Riedel et al., 2010), which is to extract (subject,predicate,object) triples from the New York Times (NYT) corpus. Rather than having labelled sentences as training data, the task requires using the existing triples from Freebase as a form of distant supervision, i.e. for some pairs of entities we know some of the relations that hold between them, but not which sentences assert these relationships (if any). To be consistent with published results for this task, we have used a word embedding that was trained from the NYT corpus[6], rather than Wikipedia (using the same preprocessing and set-up). We have used the training and test data that was shared publicly for this task[7], which consist of sentences from articles published in 2005-2006 and in 2007, respectively. Each of these sentences contains two entities, which are already linked to Freebase. We learn relation vectors from the sentences in the training and test sets, and learn a linear SVM classifier based on the Freebase triples that are available in the training set. Initially, we split the training data into 75% training and 25% tuning to find the optimal parameters of the linear SVM model. We tuned the parameters for each test fold sepa-

[6]https://catalog.ldc.upenn.edu/LDC2008T19
[7]http://iesl.cs.umass.edu/riedel/ecml/

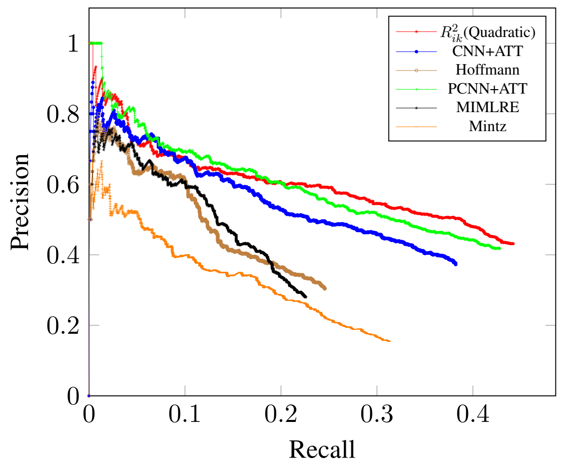


Figure 2: Results for the relation extraction from the NYT corpus: comparison with state-of-the-art neural network models.

rately. For each test fold, we used 25% of the 9 training folds as tuning data. After the optimal parameters have been determined, we retrain the model on the full training data, and apply it on the test fold. We used this approach (rather than e.g. fixing a train/tune/test split) because the total number of examples for some of the relations is very small. After tuning, we re-train the SVM models on the full training data. As the number of training examples is larger for this task, we also consider SVMs with a quadratic kernel.

Following earlier work on this task, we report our results on the test set as a precision-recall graph in Figure 1. This shows that the best performance is again achieved by R^2_{ik}, especially for larger recall values. Furthermore, using a quadratic kernel (only shown for R^2_{ik}) outperforms the linear SVM models. Note that the differences between the baselines are more pronounced in this task, with Avg being clearly better than Diff, which is in turn better than Conc. For this relation extraction task, a large number of methods have already been proposed in the literature, with variants of convolutional neural network models with attention mechanisms achieving state-of-the-art performance[8]. A comparison with these models[9] is shown in Figure 2. The performance of R^2_{ik} is comparable with the state-of-the-art PCNN+ATT model (Lin et al., 2016), outperforming it for larger recall values. This is remarkable, as our model is conceptually much simpler, and has not been specifically tuned for this task. For instance, it could easily be improved by incorporating the attention mechanism from the PCNN+ATT model to focus the relation vectors on the considered task. Similarly, we could consider a supervised variant of (3), in which a learned relation-specific weight is added to each term.

[8]Note that such models would not be suitable for the evaluation tasks in Sections 5.1 and 5.2, due to the very limited number of training examples.

[9]Results for the neural network models have been obtained from `https://github.com/thunlp/TensorFlow-NRE/tree/master/data`.

6 Conclusions

We have proposed an unsupervised method which uses co-occurrences statistics to represent the relationship between a given pair of words as a vector. In contrast to neural network models for relation extraction, our model learns relation vectors in an unsupervised way, which means that it can be used for measuring relational similarities and related tasks. Moreover, even in (distantly) supervised tasks (where we need to learn a classifier on top of the unsupervised relation vectors), our model has proven competitive with state-of-the-art neural network models. Compared to approaches that rely on averaging word vectors, our method is able to learn more faithful representations by focusing on the words that are most strongly related to the considered relationship.

Acknowledgments

This work was supported by ERC Starting Grant 637277. Experiments in this work were performed using the computational facilities of the Advanced Research Computing at Cardiff (ARCCA) Division, Cardiff University and the ICARUS computational facility from Information Services, at the University of Kent.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital libraries*. pages 85–94.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proc. IJCAI*. pages 2670–2676.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proc. EACL*. pages 23–32.

A. Bordes, J. Weston, R. Collobert, and Y. Bengio. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*. pages 172–183.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam .R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proc. AAAI*. pages 1306–1313.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ACE) program - tasks, data, and evaluation. In *Proc. LREC*.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proc. ACL*. pages 626–634.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proc. COLING*. pages 3519–3530.

Miao Fan, Kai Cao, Yifan He, and Ralph Grishman. 2015. Jointly embedding relations and mentions for knowledge population. In *Proc. RANLP*. pages 186–191.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proc. CoNLL*. pages 268–278.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proc. SemEval*. pages 33–38.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proc. NAACL-HLT*. pages 1367–1377.

Nitin Indurkhya and Fred J Damerau. 2010. *Handbook of natural language processing*, volume 2. CRC Press.

Shoaib Jameel and Steven Schockaert. 2016. D-GloVe: A feasible least squares model for estimating word embedding densities. In *Proc. COLING*. pages 1849–1860.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proc. *SEM*. pages 356–364.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: optimizing word embeddings for sentence representations. In *Proc. ACL*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. ICML*. pages 1188–1196.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proc. ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*. pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. ACL*. pages 1003–1011.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science* 34(8):1388–1429.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*. pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proc. ECML/PKDD*. pages 148–163.

Bryan Rink and Sanda Harabagiu. 2012. UTD: Determining relational similarity using lexical patterns. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. pages 413–418.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proc. NAACL-HLT*. pages 304–311.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proc. EMNLP*. pages 1201–1211.

Tim Van de Cruys. 2011. Two multivariate generalizations of pointwise mutual information. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*. pages 16–20.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proc. ACL*.

Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014a. Knowledge graph and text jointly embedding. In *EMNLP*. pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. pages 1112–1119.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proc. EMNLP*. pages 1366–1371.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. EMNLP*. pages 1785–1794.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proc. COLING*. pages 2335–2344.

Alisa Zhila, Wen-tau Yih, Christopher Meek, Geoffrey Zweig, and Tomas Mikolov. 2013. Combining heterogeneous models for measuring relational similarity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1000–1009.

Explicit Retrofitting of Distributional Word Vectors

Goran Glavaš
Data and Web Science Group
University of Mannheim
B6, 29, DE-68161 Mannheim
goran@informatik.uni-mannheim.de

Ivan Vulić
Language Technology Lab
University of Cambridge
9 West Road, Cambridge CB3 9DA
iv250@cam.ac.uk

Abstract

Semantic specialization of distributional word vectors, referred to as *retrofitting*, is a process of fine-tuning word vectors using external lexical knowledge in order to better embed some semantic relation. Existing retrofitting models integrate linguistic constraints directly into learning objectives and, consequently, specialize only the vectors of words from the constraints. In this work, in contrast, we transform external lexico-semantic relations into training examples which we use to learn an *explicit retrofitting model (ER)*. The ER model allows us to learn a global specialization function and specialize the vectors of words unobserved in the training data as well. We report large gains over original distributional vector spaces in (1) intrinsic word similarity evaluation and on (2) two downstream tasks – *lexical simplification* and *dialog state tracking*. Finally, we also successfully specialize vector spaces of new languages (i.e., unseen in the training data) by coupling ER with shared multilingual distributional vector spaces.

1 Introduction

Algebraic modeling of word vector spaces is one of the core research areas in modern Natural Language Processing (NLP) and its usefulness has been shown across a wide variety of NLP tasks (Collobert et al., 2011; Chen and Manning, 2014; Melamud et al., 2016). Commonly employed *distributional* models for word vector induction are based on the distributional hypothesis (Harris, 1954), i.e., they rely on word co-occurrences obtained from large text corpora (Mikolov et al., 2013b; Pennington et al., 2014; Levy and Goldberg, 2014a; Levy et al., 2015; Bojanowski et al., 2017).

The dependence on purely distributional knowledge results in a well-known tendency of fusing semantic similarity with other types of semantic relatedness (Hill et al., 2015; Schwartz et al., 2015) in the induced vector spaces. Consequently, the similarity between distributional vectors indicates just an abstract semantic association and not a precise semantic relation (Yih et al., 2012; Mohammad et al., 2013). For example, it is difficult to discern synonyms from antonyms in distributional spaces. This property has a particularly negative effect on NLP applications like text simplification and statistical dialog modeling, in which discerning semantic similarity from other types of semantic relatedness is pivotal to the system performance (Glavaš and Štajner, 2015; Faruqui et al., 2015; Mrkšić et al., 2016; Kim et al., 2016b).

A standard solution is to move beyond purely unsupervised learning of word representations, in a process referred to as *word vector space specialization* or *retrofitting*. Specialization models leverage external lexical knowledge from lexical resources, such as WordNet (Fellbaum, 1998), the Paraphrase Database (Ganitkevitch et al., 2013), or BabelNet (Navigli and Ponzetto, 2012), to *specialize* distributional spaces for a particular lexical relation, e.g., synonymy (Faruqui et al., 2015; Mrkšić et al., 2017) or hypernymy (Glavaš and Ponzetto, 2017). External constraints are commonly pairs of words between which a particular relation holds.

Existing specialization methods exploit the external linguistic constraints in two prominent ways: (1) *joint specialization* models modify the learning objective of the original distributional model by integrating the constraints into it (Yu and Dredze, 2014; Kiela et al., 2015; Nguyen et al., 2016, *inter alia*); (2) *post-processing* models fine-tune distributional vectors retroactively after training to satisfy the external constraints (Faruqui et al., 2015;

Mrkšić et al., 2017, *inter alia*). The latter, in general, outperform the former (Mrkšić et al., 2016). Retrofitting models can be applied to arbitrary distributional spaces but they suffer from a major limitation – they *locally* update only vectors of words present in the external constraints, whereas vectors of all other (unseen) words remain intact. In contrast, joint specialization models propagate the external signal to all words via the joint objective.

In this paper, we propose a new approach for specializing word vectors that unifies the strengths of both prior strategies, while mitigating their limitations. Same as retrofitting models, our novel framework, termed *explicit retrofitting (ER)*, is applicable to *arbitrary* distributional spaces. At the same time, the method learns an *explicit global specialization* function that can specialize vectors for *all* vocabulary words, similar as in joint models. Yet, unlike the joint models, ER does not require expensive re-training on large text corpora, but is directly applied on top of any pre-trained vector space. The key idea of ER is to directly learn a specialization function in a *supervised* setting, using lexical constraints as *training instances*. In other words, our model, implemented as a deep feed-forward neural architecture, learns a (non-linear) function which "translates" word vectors from the distributional space into the specialized space.

We show that the proposed ER approach yields considerable gains over distributional spaces in word similarity evaluation on standard benchmarks (Hill et al., 2015; Gerz et al., 2016), as well as in two downstream tasks – lexical simplification and dialog state tracking. Furthermore, we show that, by coupling the ER model with shared multilingual embedding spaces (Mikolov et al., 2013a; Smith et al., 2017), we can also specialize distributional spaces for languages unseen in the training data in a zero-shot language transfer setup. In other words, we show that an explicit retrofitting model trained with external constraints from one language can be successfully used to specialize the distributional space of another language.

2 Related Work

The importance of vector space specialization for downstream tasks has been observed, *inter alia*, for dialog state tracking (Mrkšić et al., 2017; Vulić et al., 2017b), spoken language understanding (Kim et al., 2016b,a), judging lexical entailment (Nguyen et al., 2017; Glavaš and Ponzetto, 2017; Vulić and Mrkšić, 2017), lexical contrast modeling (Nguyen et al., 2016), and cross-lingual transfer of lexical resources (Vulić et al., 2017a). A common goal pertaining to all retrofitting models is to pull the vectors of similar words (e.g., synonyms) closer together, while some models also push the vectors of dissimilar words (e.g., antonyms) further apart. The specialization methods fall into two categories: (1) joint specialization methods, and (2) post-processing (i.e., retrofitting) methods. Methods from both categories make use of similar lexical resources – they typically leverage WordNet (Fellbaum, 1998), FrameNet (Baker et al., 1998), the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015), morphological lexicons (Cotterell et al., 2016), or simple hand-crafted linguistic rules (Vulić et al., 2017b). In what follows, we discuss the two model categories.

Joint Specialization Models. These models integrate external constraints into the distributional training procedure of general word embedding algorithms such as CBOW, Skip-Gram (Mikolov et al., 2013b), or Canonical Correlation Analysis (Dhillon et al., 2015). They modify the prior or the regularization of the original objective (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015) or integrate the constraints directly into the, e.g., an SGNS- or CBOW-style objective (Liu et al., 2015; Ono et al., 2015; Bollegala et al., 2016; Osborne et al., 2016; Nguyen et al., 2016, 2017). Besides generally displaying lower performance compared to retrofitting methods (Mrkšić et al., 2016), these models are also tied to the distributional objective and any change of the underlying distributional model induces a change of the entire joint model. This makes them less versatile than the retrofitting methods.

Post-Processing Models. Models from the popularly termed *retrofitting* family inject lexical knowledge from external resources into arbitrary pre-trained word vectors (Faruqui et al., 2015; Jauhar et al., 2015; Rothe and Schütze, 2015; Wieting et al., 2015; Nguyen et al., 2016; Mrkšić et al., 2016). These models fine-tune the vectors of words present in the linguistic constraints to reflect the ground-truth lexical knowledge. While the large majority of specialization models from both classes operate only with similarity constraints, a line of recent work (Mrkšić et al., 2016; Mrkšić et al., 2017; Vulić et al., 2017b) demonstrates that knowledge about both similar and dissimilar words leads to improved performance in downstream tasks. The main shortcoming of the existing retrofitting models is their inability to specialize vectors of words unseen in external lexical resources.

Our explicit retrofitting framework brings together desirable properties of both model classes: (1) unlike joint models, it does not require adaptation to the underlying distributional model and expensive re-training, i.e., it is applicable to any pre-trained distributional space; (2) it allows for easy integration of both similarity and dissimilarity constraints into the specialization process; and (3) unlike post-processors, it specializes the full vocabulary of the original distributional space and not only vectors of words from external constraints.

3 Explicit Retrofitting

Our explicit retrofitting (ER) approach, illustrated by Figure 1a, consists of two major components: (1) an algorithm for preparing training instances from external lexical constraints, and (2) a supervised specialization model, based on a deep feed-forward neural network. This network, shown in Figure 1b learns a non-linear global specialization function from the training instances.

3.1 From Constraints to Training Instances

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, $\mathbf{x}_i \in \mathbb{R}^d$ be the d-dimensional distributional vector space that we want to specialize (with $V = \{w_i\}_{i=1}^{N}$ referring to the associated vocabulary) and let $\mathbf{X}' = \{\mathbf{x}'_i\}_{i=1}^{N}$ be the corresponding specialized vector space that we seek to obtain through explicit retrofitting. Let $\mathbf{C} = \{(w_i, w_j, r)_l\}_{l=1}^{L}$ be the set of L linguistic constraints from an external lexical resource, each consisting of a pair of vocabulary words w_i and w_j and a semantic relation r that holds between them. The most recent state-of-the-art retrofitting work (Mrkšić et al., 2017; Vulić et al., 2017b) suggests that using both similarity and dissimilarity constraints leads to better performance compared to using only similarity constraints. Therefore, we use synonymy and antonymy relations from external resources, i.e., $r_l \in \{ant, syn\}$. Let g be the function measuring the distance between words w_i and w_j based on their vector representations. The algorithm for preparing training instances from constraints is guided by the following assumptions:

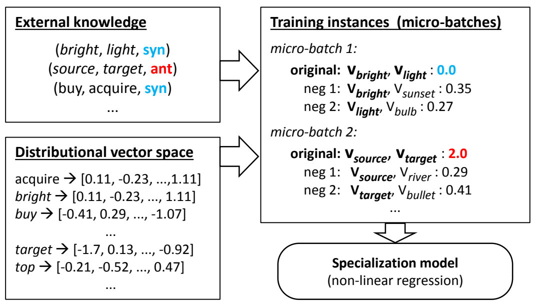
1. All synonymy pairs (w_i, w_j, syn) should have a minimal possible distance score in the specialized space, i.e., $g(\mathbf{x}'_i, \mathbf{x}'_j) = g_{min}$;[1]
2. All antonymy pairs (w_i, w_j, ant) should have a maximal distance in the specialized space, i.e., $g(\mathbf{x}'_i, \mathbf{x}'_j) = g_{max}$;[2]
3. The distances $g(\mathbf{x}'_i, \mathbf{x}'_k)$ in the specialized space between some word w_i and all other words w_k that are not synonyms or antonyms of w_i should be in the interval (g_{min}, g_{max}).

Our goal is to discern semantic similarity from semantic relatedness by comparing, in the specialized space, the distances between word pairs $(w_i, w_j, r) \in \mathbf{C}$ with distances that words w_i and w_j from those pairs have with other vocabulary words w_m. It is intuitive to enforce that the synonyms are as close as possible and antonyms as far as possible. However, we do not know what the distances between non-synonymous and non-antonymous words $g(\mathbf{x}'_i, \mathbf{x}_m)$ in the specialized space should look like. This is why, for all other words, similar to (Faruqui et al., 2016; Mrkšić et al., 2017), we assume that the distances in the specialized space for all word pairs not found in C should stay the same as in the distributional space: $g(\mathbf{x}'_i, \mathbf{x}'_m) = g(\mathbf{x}_i, \mathbf{x}_m)$. This way we preserve the useful semantic content available in the original distributional space.

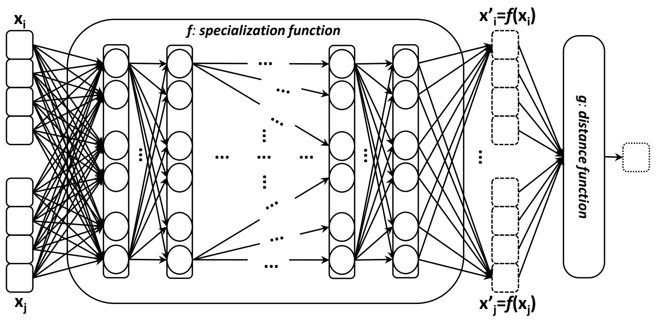
In downstream tasks most errors stem from vectors of semantically related words (e.g., *car – driver*) being as similar as vectors of semantically similar words (e.g., *car – automobile*). To anticipate this, we compare the distances of pairs $(w_i, w_j, r) \in \mathbf{C}$ with the distances for pairs (w_i, w_m) and (w_j, w_n), where w_m and w_n are *negative examples*: the vocabulary words that are *most similar* to w_i and w_j, respectively, in the original distributional space $\mathbf{X}$. Concretely, for each constraint $(w_i, w_j, r) \in \mathbf{C}$ we retrieve (1) K vocabulary words $\{w_m^k\}_{k=1}^{K}$ that are closest in the input distributional space (according to the distance function g) to the word w_i and (2) K vocabulary words $\{w_n^k\}_{k=1}^{K}$ that are closest to the word w_j. We then create, for each constraint $(w_i, w_j, r) \in \mathbf{C}$, a corresponding set M (termed *micro-batch*) of $2K+1$ embedding pairs coupled with a corresponding distance in the input distributional space:

[1]The minimal distance value is $g_{min} = 0$ for, e.g., cosine distance or Euclidean distance.

[2]While some distance functions do have a theoretical maximum (e.g., $g_{max} = 2$ for cosine distance), others (e.g., Euclidean distance) may be theoretically unbounded. For unbounded distance measures, we propose using the maximal distance between any two words from the vocabulary as g_{max}.


External knowledge
(bright, light, syn)
(source, target, ant)
(buy, acquire, syn)
...
Distributional vector space
acquire → [0.11, -0.23, ...,1.11]
bright → [0.11, -0.23, ..., 1.11]
buy → [-0.41, 0.29, ..., -1.07]
...
target → [-1.7, 0.13, ..., -0.92]
top → [-0.21, -0.52, ..., 0.47]
...
Training instances (micro-batches)
micro-batch 1:
original: v_bright, v_light : 0.0
neg 1: v_bright, v_sunset : 0.35
neg 2: v_light, v_bulb : 0.27
micro-batch 2:
original: v_source, v_target : 2.0
neg 1: v_source, v_river : 0.29
neg 2: v_target, v_bullet : 0.41
...
Specialization model
(non-linear regression)

(a) Illustration of the explicit retrofitting approach



(b) Supervised specialization model

Figure 1: **(a)** High-level illustration of the explicit retrofitting approach: lexical constraints, i.e., pairs of synonyms and antonyms, are transformed into respective micro-batches, which are then used to train the supervised specialization model. **(b)** The low-level implementation of the specialization model, combining the non-linear embedding specialization function f, defined as the deep fully-connected feed-forward network, with the distance metric g, measuring the distance between word vectors after their specialization.

$$
\begin{aligned}
M(w_i, w_j, r) = \{(\mathbf{x}_i, \mathbf{x}_j, g_r)\} \; \cup \\
\{(\mathbf{x}_i, \mathbf{x}_m^k, g(\mathbf{x}_i, \mathbf{x}_m^k))\}_{k=1}^{K} \; \cup \\
\{(\mathbf{x}_j, \mathbf{x}_n^k, g(\mathbf{x}_j, \mathbf{x}_n^k))\}_{k=1}^{K}
\end{aligned} \quad (1)
$$

with $g_r = g_{min}$ if $r = syn$; $g_r = g_{max}$ if $r = ant$.

3.2 Non-Linear Specialization Function

Our retrofitting framework learns a global *explicit specialization function* which, when applied on a distributional vector space, transforms it into a space that better captures semantic similarity, i.e., discerns similarity from all other types of semantic relatedness. We seek the optimal parameters θ of the parametrized function $f(\mathbf{x}; \theta) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ (where d is the dimensionality of the input space). The specialized embedding $\mathbf{x'}_i$ of the word w_i is then obtained as $\mathbf{x'}_i = f(\mathbf{x}_i; \theta)$. The specialized space $\mathbf{X'}$ is obtained by transforming distributional vectors of *all* vocabulary words, $\mathbf{X'} = f(\mathbf{X}; \theta)$.

We define the specialization function f to be a multi-layer fully-connected feed-forward network with H hidden layers and non-linear activations ϕ. The illustration of this network is given in Figure 1b. The i-th hidden layer is defined with a weight matrix $\mathbf{W}^i$ and a bias vector $\mathbf{b}^i$:

$$
h^i(\mathbf{x}; \theta_i) = \phi\left(h^{i-1}(\mathbf{x}; \theta_{i-1})\mathbf{W}^i + \mathbf{b}^i\right) \quad (2)
$$

where θ_i is the subset of network's parameters up to the i-th layer. Note that in this notation, $\mathbf{x} = h^0(\mathbf{x}; \emptyset)$ and $\mathbf{x'} = f(\mathbf{x}, \theta) = h^H(\mathbf{x}; \theta)$. Let d_h be the size of the hidden layers. The network's parameters are then as follows: $\mathbf{W}^1 \in \mathbb{R}^{d \times d_h}$; $\mathbf{W}^i \in \mathbb{R}^{d_h \times d_h}$, $i \in \{2, \dots, H-1\}$; $\mathbf{W}^H \in \mathbb{R}^{d_h \times d}$; $\mathbf{b}^i \in \mathbb{R}^{d_h}$, $i \in \{1, \dots, H-1\}$; $\mathbf{b}^H \in \mathbb{R}^d$.

3.3 Optimization Objectives

We feed the micro-batches consisting of $2K+1$ training instances to the specialization model (see Section 3.1). Each training instance consists of a pair of distributional (i.e., unspecialized) embedding vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and a score g denoting the desired distance between the specialized vectors $\mathbf{x'}_i$ and $\mathbf{x'}_j$ of corresponding words w_i and w_j.

Mean Square Distance Objective (ER-MSD). Let our training batch consist of N training instances, $\{(\mathbf{x}_1^i, \mathbf{x}_2^i, g^i)\}_{i=1}^N$. The simplest objective function is then the difference between the desired and obtained distances of specialized vectors:

$$
J_{MSD} = \sum_{i=1}^{N} \left(g(f(\mathbf{x}_1^i), f(\mathbf{x}_2^i)) - g^i\right)^2 \quad (3)
$$

By minimizing the MSD objective we simply force the specialization model to produce a specialized embedding space $\mathbf{X'}$ in which distances between all synonyms amount to g_{min}, distances between all antonyms amount to g_{max} and distances between all other word pairs remain the same as in the original space. The MSD objective does not leverage negative examples: it only indirectly enforces that synonym (or antonym) pairs (w_i, w_j) have smaller (or larger) distances than corresponding non-constraint word pairs (w_i, w_k) and (w_j, w_k).

Contrastive Objective (ER-CNT). An alternative to MSD is to *directly* contrast the distances of constraint pairs (i.e., antonyms and synonyms) with the distances of their corresponding negative examples, i.e., the pairs from their respective micro-batch (cf. Eq. (1) in Section 3.1). Such an objective should directly enforce that the similarity scores for synonyms (antonyms) (w_i, w_j) are larger (or smaller, for antonyms) than for pairs (w_i, w_k) and (w_j, w_k) involving the same words w_i and w_j, respectively. Let S and A be the sets of micro-batches created from synonymy and antonymy constraints. Let $M_s = \{(\mathbf{x}_1^i, \mathbf{x}_2^i, g^i)\}_{i=1}^{2K+1}$ be one micro-batch created from one synonymy constraint and let M_a be the analogous micro-batch created from one antonymy constraint. Let us then assume that the first triple (i.e., for $i = 1$) in every micro-batch corresponds to the constraint pair and the remaining $2K$ triples (i.e., for $i \in \{2, \dots, 2K+1\}$) to respective non-constraint word pairs. We then define the contrastive objective as follows:

$$J_{CNT} = \sum_{M_s \in S} \sum_{i=2}^{2K+1} \left((g^i - g_{min}) - ({g'}^i - {g'}^1) \right)^2 + \sum_{M_a \in A} \sum_{i=2}^{2K+1} \left((g_{max} - g^i) - ({g'}^1 - {g'}^i) \right)^2$$

where g' is a short-hand notation for the distance between vectors in the specialized space, i.e., $g'(\mathbf{x}_1, \mathbf{x}_2) = g(\mathbf{x}'_1, \mathbf{x}'_2) = g(f(\mathbf{x}_1), f(\mathbf{x}_2))$.

Topological Regularization. Because the distributional space $\mathbf{X}$ already contains useful semantic information, we want our specialized space $\mathbf{X}'$ to move similar words closer together and dissimilar words further apart, but without disrupting the overall topology of $\mathbf{X}$. To this end, we define an additional regularization objective that measures the distance between the original vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ and their specialized counterparts $\mathbf{x}'_1 = f(\mathbf{x}_1)$ and $\mathbf{x}'_2 = f(\mathbf{x}_2)$, for all examples in the training set:

$$J_{REG} = \sum_{i=1}^{N} g(\mathbf{x}_1^i, f(\mathbf{x}_1^i)) + g(\mathbf{x}_2^i, f(\mathbf{x}_2^i)) \tag{4}$$

We minimize the final objective function $J' = J + \lambda J_{REG}$. J is either J_{MSD} or J_{CNT} and λ is the regularization factor which determines how strictly we retain the topology of the original space.

4 Experimental Setup

Distributional Vectors. In order to estimate the robustness of the proposed explicit retrofitting procedure, we experiment with three different publicly available and widely used collections of pre-trained distributional vectors for English: (1) SGNS-W2 – vectors trained on the Wikipedia dump from the Polyglot project (Al-Rfou et al., 2013) using the Skip-Gram algorithm with Negative Sampling (SGNS) (Mikolov et al., 2013b) by Levy and Goldberg (2014b), using the context windows of size 2; (2) GLOVE-CC – vectors trained with the GloVe (Pennington et al., 2014) model on the Common Crawl; and (3) FASTTEXT – vectors trained on Wikipedia with a variant of SGNS that builds word vectors by summing the vectors of their constituent character n-grams (Bojanowski et al., 2017).

Linguistic Constraints. We experiment with the sets of linguistic constraints used in prior work (Zhang et al., 2014; Ono et al., 2015). These constraints, extracted from WordNet (Fellbaum, 1998) and Roget's Thesaurus (Kipfer, 2009), comprise a total of 1,023,082 synonymy word pairs and 380,873 antonymy word pairs.

Although this seems like a large number of linguistic constraints, there is only 57,320 unique words in all synonymy and antonymy constraints combined, and not all of these words are found in the dictionary of the pre-trained distributional vector space. For example, only 15.3% of the words from constraints are found in the whole vocabulary of SGNS-W2 embeddings. Similarly, we find only 13.3% and 14.6% constraint words among the 200K most frequent words from the GLOVE-CC and FASTTEXT vocabularies, respectively. This low coverage emphasizes the core limitation of current retrofitting methods, being able to specialize only the vectors of words seen in the external constraints, and the need for our global ER method which can specialize all word vectors from the distributional space.

ER Model Configuration. In all experiments, we set the distance function g to cosine distance: $g(\mathbf{x}_1, \mathbf{x}_2) = 1 - (\mathbf{x}_1 \cdot \mathbf{x}_2 / (\|\mathbf{x}_1\| \|\mathbf{x}_2\|))$ and use the hyperbolic tangent as activation, $\phi = \tanh$. For each constraint (w_i, w_j), we create $K = 4$ corresponding negative examples for both w_i and w_j, resulting in micro-batches with $2K + 1 = 9$ training instances.[3] We separate 10% of the created micro-batches as the validation set. We then tune the hyper-parameter values, the number of hidden layers $H = 5$ and their size $d_h = 1000$, and the

[3]For $K < 4$ we observed significant performance drop. Setting $K > 4$ resulted in negligible performance gains but significantly increased the model training time.

topological regularization factor $\lambda = 0.3$ by minimizing the model's objective J' on the validation set. We train the model in mini-batches, each containing $N_b = 100$ constraints (i.e., 900 training instances, see above), using the Adam optimizer (Kingma and Ba, 2015) with initial learning rate set to 10^{-4}. We use the loss on the validation set as the early stopping criteria.

5 Results and Discussion

5.1 Word Similarity

Evaluation Setup. We first evaluate the quality of the explicitly retrofitted embedding spaces intrinsically, on two word similarity benchmarks: SimLex-999 dataset (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016), a recent dataset containing human similarity ratings for 3,500 verb pairs.[4] We use Spearman's ρ rank correlation between gold and predicted word pair scores as the evaluation metric. We evaluate the specialized embedding spaces in two settings. In the first setting, termed *lexically disjoint*, we remove from our training set all linguistic constraints that contain any of the words found in SimLex or SimVerb. This way, we effectively evaluate the model's ability to generalize the specialization function to *unseen* words. In the second setting (*lexical overlap*) we retain the constraints containing SimLex or SimVerb words in the training set. For comparison, we also report performance of the state-of-the-art local retrofitting model ATTRACT-REPEL (Mrkšić et al., 2017), which is able to specialize only the words from the linguistic constraints.

Results. The results with our ER model applied to three distributional spaces are shown in Table 1. The scores suggest that the proposed ER model is universally useful and robust. The ER-specialized spaces outperform original distributional spaces across the board, for both objective functions. The results in the *lexically disjoint* setting are especially indicative of the improvements achieved by the ER. For example, we achieve a correlation gain of 18% for the GLOVE-CC vectors on SimLex using a specialization function learned *without* seeing a single constraint with any SimLex word.

[4]Other word similarity datasets such as MEN (Bruni et al., 2014) or WordSim-353 (Finkelstein et al., 2002) conflate the concepts of true semantic similarity and semantic relatedness in a broader sense. In contrast, SimLex and SimVerb explicitly discern between the two, with pairs of semantically related but not similar words (e.g. *car* and *wheel*) having low ratings.

In the *lexical overlap* setting, we observe substantial gains only for GLOVE-CC. The modest gains in this setting with FASTTEXT and SGNS-W2 in fact strengthen the impression that the ER model learns a *general* specialization function, i.e., it does not "overfit" to words from linguistic constraints. The ER model with the contrastive objective (ER-CNT) yields better performance on average than the one using the simpler square distance objective (ER-MSD). This is expected, given that the contrastive objective enforces the model to distinguish pairs of semantically (dis)similar words from pairs of semantically related words.

Finally, the post-processing ATTRACT-REPEL model based on local vector updates seems to substantially outperform the ER method in this task. The gap is especially visible for FASTTEXT and SGNS-W2 vectors. However, since ATTRACT-REPEL specializes only words seen in linguistic constraints,[5] its performance crucially depends on the coverage of test set words in the constraints. ATTRACT-REPEL excels on the intrinsic evaluation as the constraints cover 99.2% of SimLex words and 99.9% of SimVerb words. However, its usefulness is less pronounced in real-life downstream scenarios in which such high coverage cannot be guaranteed, as demonstrated in Section 5.3.

Analysis. We examine in more detail the performance of the ER model with respect to (1) the type of constraints used for training the model: synonyms and antonyms, only synonyms, or only antonyms and (2) the extent to which we retain the topology of the original distributional space (i.e., with respect to the value of the topological regularization factor λ). All reported results were obtained by specializing the GLOVE-CC distributional space in the *lexically disjoint* setting (i.e., employed constraints did not contain any of the SimLex or SimVerb words).

In Table 2 we show the specialization performance of the ER-CNT models ($H = 5$, $\lambda = 0.3$), using different types of constraints on SimLex-999 (SL) and SimVerb-3500 (SV). We compare the standard model, which exploits both synonym and antonym pairs for creating training instances, with the models employing only synonym and only antonym constraints, respectively. Clearly, we obtain the best specialization when combining synonyms and antonyms. Note, however, that using

[5]This is why ATTRACT-REPEL cannot be applied in the *lexically disjoint* setting: the scores simply stay the same.

	Setting: *lexically disjoint*						Setting: *lexical overlap*					
	GLOVE-CC		FASTTEXT		SGNS-W2		GLOVE-CC		FASTTEXT		SGNS-W2	
	SL	SV	SL	SV	SL	SV	SL	SV	SL	SV	SL	SV
Distributional (**X**)	.407	.280	.383	.247	.414	.272	.407	.280	.383	.247	.414	.272
ATTRACT-REPEL	.407	.280	.383	.247	.414	.272	**.690**	**.578**	**.629**	**.502**	**.658**	**.544**
ER-Specialized ($\mathbf{X}' = f(\mathbf{X})$)												
ER-MSD	.483	.345	.429	**.275**	**.445**	.302	.500	.358	**.445**	.284	**.469**	.323
ER-CNT	**.582**	**.439**	**.433**	.272	.435	**.329**	**.623**	**.519**	.419	**.335**	.449	**.355**

Table 1: Spearman's ρ correlation scores for three standard English distributional vectors spaces on English SimLex-999 (SL) and SimVerb-3500 (SV), using explicit retrofitting models with two different objective functions (ER-MSD and ER-CNT, cf. Section 3.3).

Constraints (ER-CNT model)	SL	SV
Synonyms only	.465	.339
Antonyms only	.451	.317
Synonyms + Antonyms	**.582**	**.439**

Table 2: Performance (ρ) on SL and SV for ER-CNT models trained with different constraints.

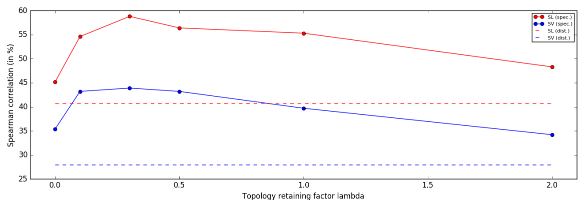


Figure 2: Specialization performance on SimLex-999 (blue line) and SimVerb-3500 (red line) for ER models with different topology regularization factors λ. Dashed lines indicate performance levels of the distributional (i.e., unspecialized) space.

only synonyms or only antonyms also improves over the original distributional space.

Next, in Figure 2 we depict the specialization performance (on SimLex and SimVerb) of the ER models with different values of the topology regularization factor λ (H fixed to 5). The best performance for is obtained for $\lambda = 0.3$. Smaller lambda values overly distort the original distributional space, whereas larger lambda values dampen the specialization effects of linguistic constraints.

5.2 Language Transfer

Readily available large collections of synonymy and antonymy word pairs do not exist for many languages. This is why we also investigate *zero-shot specialization*: we test if it is possible, with the help of cross-lingual word embeddings, to transfer the specialization knowledge learned from English constraints to languages without any training data.

Evaluation Setup. We use the mapping model of Smith et al. (2017) to induce a multilingual vec-

Model	German	Italian	Croatian
Distributional (**X**)	.407	.360	.249
ER-Specialized ($\mathbf{X}'$)			
ER-MSD	.415	.406	.287
ER-CNT	**.533**	**.448**	**.315**

Table 3: Spearman's ρ correlation scores for German, Italian, and Croatian embeddings in the transfer setup: the vectors are specialized using the models trained on English constraints and evaluated on respective language-specific SimLex-999 variants.

tor space[6] containing word vectors of three other languages – German, Italian, and Croatian – along with the English vectors.[7] Concretely, we map the Italian CBOW vectors (Dinu et al., 2015), German FastText vectors trained on German Wikipedia (Bojanowski et al., 2017), and Croatian Skip-Gram vectors trained on HrWaC corpus (Ljubešić and Erjavec, 2011) to the GLOVE-CC English space. We create the translation pairs needed to learn the projections by automatically translating 4,000 most frequent English words to all three other languages with Google Translate. We then employ the ER model trained to specialize the GLOVE-CC space using the full set of English constraints, to specialize the distributional spaces of other languages. We evaluate the quality of the specialized spaces on the respective SimLex-999 dataset for each language (Leviant and Reichart, 2015; Mrkšić et al., 2017).

Results. The results are provided in Table 3. They indicate that the ER models can substantially improve (e.g., by 13% for German vector space) over distributional spaces also in the language transfer setup without seeing a single constraint in the target language. These transfer results hold promise to support vector space specialization

[6]This model was chosen for its ease of use, readily available implementation, and strong comparative results (see (Ruder et al., 2017)). For more details we refer the reader to the original paper and the survey.

[7]The choice of languages was determined by the availability of the language-specific SimLex-999 variants.

even for resource-lean languages. The more sophisticated contrastive ER-CNT model variant again outperforms the simpler ER-MSD variant, and it does so for all three languages, which is consistent with the findings from the monolingual English experiments (see Table 1).

5.3 Downstream Tasks

We now evaluate the impact of our global ER method on two downstream tasks in which differentiating semantic similarity from semantic relatedness is particularly important: lexical text simplification (LS) and dialog state tracking (DST).

5.3.1 Lexical Text Simplification

Lexical simplification aims to replace complex words – used less frequently and known to fewer speakers – with their simpler synonyms that fit into the context, that is, without changing the meaning of the original text. Because retaining the meaning of the original text is a strict requirement, complex words need to be replaced with semantically similar words, whereas replacements with semantically related words (e.g., replacing *"pilot"* with *"airplane"* in *"Ferrari's pilot won the race"*) produce incorrect text which is more difficult to comprehend.

Simplification Using Distributional Vectors. We use the LIGHT-LS lexical simplification algorithm of Glavaš and Štajner (2015) which makes the word replacement decisions primarily based on semantic similarities between words in a distributional vector space.[8] For each word in the input text LIGHT-LS retrieves most similar replacement candidates from the vector space. The candidates are then ranked according to several measures of simplicity and fitness for the context. Finally, the replacement is made if the top-ranked candidate is estimated to be simpler than the original word. By plugging-in vector spaces specialized by the ER model into LIGHT-LS, we hope to generate true synonymous candidates more frequently than with the unspecialized distributional space.

Evaluation Setup. We evaluate LIGHT-LS on the LS dataset crowdsourced by Horn et al. (2014). For each indicated complex word Horn et al. (2014) collected 50 manual simplifications. We use two evaluation metrics from prior work (Horn et al., 2014; Glavaš and Štajner, 2015) to quantify the quality and frequency of word replacements: (1)

[8]The Light-LS implementation is available at: https://bitbucket.org/gg42554/embesimp

	GLOVE-CC		FASTTEXT		SGNS-W2	
Emb. space	A	C	A	C	A	C
Distributional	66.0	**94.0**	57.8	84.0	56.0	79.1
Specialized						
ATTRACT-REPEL	67.6	87.0	69.8	89.4	64.4	86.7
ER-CNT	**73.8**	93.0	**71.2**	**93.2**	**68.4**	**92.3**

Table 4: Lexical simplification performance with explicit retrofitting applied on three input spaces.

accurracy (A) is the number of correct simplifications made (i.e., when the replacement made by the system is found in the list of manual replacements) divided by the total number of indicated complex words; and (2) *change* (C) is the percentage of indicated complex words that were replaced by the system (regardless of whether the replacement was correct). We plug into LIGHT-LS both unspecialized and specialized variants of three previously used English embedding spaces: GLOVE-CC, FASTTEXT, and SGNS-W2. Additionally, we again evaluate specializations of the same spaces produced by the state-of-the-art local retrofitting model ATTRACT-REPEL (Mrkšić et al., 2017).

Results and Analysis. The results with LIGHT-LS are summarized in Table 4. ER-CNT model yields considerable gains over unspecialized spaces for both metrics. This suggests that the ER-specialized embedding spaces allow LIGHT-LS to generate true synonymous candidate replacements more often than with unspecialized spaces, and also verifies the importance of specialization for the LS task. Our ER-CNT model now also yields better results than ATTRACT-REPEL in a real-world downstream task. Only 59.6 % of all indicated complex words and manual replacement candidates from the LS dataset are now covered by the linguistic constraints. This accentuates the need to specialize the full distributional space in downstream applications as done by the ER model, while ATTRACT-REPEL is limited to local vector updates only of words seen in the constraints. By learning a global specialization function the proposed ER models seem more resilient to the observed drop in coverage of test words by linguistic constraints. Table 5 shows example substitutions of LIGHT-LS when using different embedding spaces: original GLOVE-CC space and its specializations obtained with ER-CNT and ATTRACT-REPEL.

5.3.2 Dialog State Tracking

Finally, we also evaluate the importance of explicit retrofitting in a downstream language understand-

Text	GloVe-CC	Attract-Repel	ER-CNT
Wrestlers portrayed a ***villain*** *or a hero as they followed a series of events that built tension*	*character*	*protagonist*	*demon*
This large version number jump was due to a feeling that a version 1.0 with no major missing ***pieces*** *was imminent.*	*ones*	*songs*	*parts*
The storm continued, crossing North Carolina , and ***retained*** *its strength until June 20 when it became extratropical near Newfoundland*	*lost*	*preserved*	*preserved*
Tibooburra has an ***arid****, desert climate with temperatures soaring above 40 Celsius in summer, often reaching as high as 47 degrees Celsius.*	*subtropical*	*humid*	*dry*

Table 5: Examples of lexical simplifications performed with the Light-LS tool when using different embedding spaces. The target word to be simplified is in bold.

GloVe-CC embedding vectors	JGA
Distributional ($\mathbf{X}$)	.797
Specialized ($\mathbf{X}' = f(\mathbf{X})$)	
Attract-Repel	.817
ER-CNT	.816

Table 6: DST performance of GloVe-CC embeddings specialized using explicit retrofitting.

ing task, namely dialog state tracking (DST) (Henderson et al., 2014; Williams et al., 2016). A DST model is typically the first component of a dialog system pipeline (Young, 2010), tasked with capturing user's goals and updating the dialog state at each dialog turn. Similarly as in lexical simplification, discerning similarity from relatedness is crucial in DST (e.g., a dialog system should not recommend an *"expensive pub in the south"* when asked for a *"cheap bar in the east"*).

Evaluation Setup. To evaluate the impact of specialized word vectors on DST, we employ the Neural Belief Tracker (NBT), a DST model that makes inferences purely based on pre-trained word vectors (Mrkšić et al., 2017).[9] NBT composes word embeddings into intermediate utterance and context representations. For full model details, we refer the reader to the original paper. Following prior work, our DST evaluation is based on the Wizard-of-Oz (WOZ) v2.0 dataset (Wen et al., 2017; Mrkšić et al., 2017) which contains 1,200 dialogs (600 training, 200 validation, and 400 test dialogs). We evaluate performance of the distributional and specialized GloVe-CC embeddings and report it in terms of *joint goal accuracy* (JGA), a standard DST evaluation metric. All reported results are averages over 5 runs of the NBT model.

Results. We show DST performance in Table 6. The DST results tell a similar story like word similarity and lexical simplification results – the ER model substantially improves over the distributional space. With linguistic specialization constraints covering 57% of words from the WOZ dataset, ER model's performance is on a par with the Attract-Repel specialization. This further confirms our hypothesis that the importance of learning a global specialization for the full vocabulary in downstream tasks grows with the drop of the test word coverage by specialization constraints.

[9]https://github.com/nmrksic/neural-belief-tracker

6 Conclusion

We presented a novel method for specializing word embeddings to better discern similarity from other types of semantic relatedness. Unlike existing retrofitting models, which directly update vectors of words from external constraints, we use the constraints as training examples to learn an explicit specialization function, implemented as a deep feed-forward neural network. Our global specialization approach resolves the well-known inability of retrofitting models to specialize vectors of words unseen in the constraints. We demonstrated the effectiveness of the proposed model on word similarity benchmarks, and in two downstream tasks: lexical simplification and dialog state tracking. We also showed that it is possible to transfer the specialization to languages without linguistic constraints.

In future work, we will investigate explicit retrofitting methods for asymmetric relations like hypernymy and meronymy. We also intend to apply the method to other downstream tasks and to investigate the zero-shot language transfer of the specialization function for more language pairs.

ER code is publicly available at: `https://github.com/codogogo/explirefit`.

Acknowledgments

Ivan Vulić is supported by the ERC Consolidator Grant LEXICAL (no. 648909).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of ACL*, pages 86–90.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of ECML-PKDD*, pages 132–148.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *Proceedings of AAAI*, pages 2690–2696.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

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

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*, pages 1651–1660.

Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR: Workshop Papers*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of NAACL-HLT*, pages 634–643.

Christiane Fellbaum. 1998. *WordNet*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Goran Glavaš and Simone Paolo Ponzetto. 2017. Dual tensor model for detecting asymmetric lexico-semantic relations. In *Proceedings of EMNLP*, pages 1758–1768.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of ACL*, pages 63–68.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014. The Second Dialog State Tracking Challenge. In *Proceedings of SIGDIAL*, pages 263–272.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *Proceedings of the ACL*, pages 458–463.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*, pages 683–693.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*, pages 2044–2048.

Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier. 2016a. Adjusting word embeddings with semantic intensity orders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 62–69.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016b. Intent detection using semantically enriched word embeddings. In *Proceedings of SLT*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR (Conference Track)*.

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus (3rd Edition)*. Philip Lief Group.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *CoRR*, abs/1508.00106.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Nikola Ljubešić and Tomaž Erjavec. 2011. hrWaC and slWaC: Compiling web corpora for croatian and slovene. In *Proceedings of TSD*, pages 395–402.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *Proceedings of NAACL-HLT*, pages 1030–1040.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint, CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2013. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*, pages 233–243.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of ACL*, pages 454–459.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of NAACL-HLT*, pages 984–989.

Dominique Osborne, Shashi Narayan, and Shay Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL*, 4:417–430.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL*, pages 425–430.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*, pages 1793–1803.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *CoRR*, abs/1706.04902.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*, pages 258–267.

Samuel L. Smith, David H.P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR (Conference Track)*.

Ivan Vulić and Nikola Mrkšić. 2017. Specialising word vectors for lexical entailment. *CoRR*, abs/1710.06371.

Ivan Vulić, Nikola Mrkšić, and Anna Korhonen. 2017a. Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of EMNLP*, pages 2536–2548.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017b. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*, pages 56–68.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The Dialog State Tracking Challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of CIKM*, pages 1219–1228.

Wen-tau Yih, Geoffrey Zweig, and John C. Platt. 2012. Polarity inducing latent semantic analysis. In *EMNLP-CoNLL*, pages 1212–1222.

Steve Young. 2010. Cognitive User Interfaces. *IEEE Signal Processing Magazine*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, pages 545–550.

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of EMNLP*, pages 1522–1531.

Unsupervised Neural Machine Translation with Weight Sharing

Zhen Yang[1,2], **Wei Chen**[1] , **Feng Wang**[1,2]*, **Bo Xu**[1]
[1]Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{yangzhen2014, wei.chen.media, feng.wang, xubo}@ia.ac.cn

Abstract

Unsupervised neural machine translation (NMT) is a recently proposed approach for machine translation which aims to train the model without using any labeled data. The models proposed for unsupervised NMT often use only one shared encoder to map the pairs of sentences from different languages to a shared-latent space, which is weak in keeping the unique and internal characteristics of each language, such as the style, terminology, and sentence structure. To address this issue, we introduce an extension by utilizing two independent encoders but sharing some partial weights which are responsible for extracting high-level representations of the input sentences. Besides, two different generative adversarial networks (GANs), namely the local GAN and global GAN, are proposed to enhance the cross-language translation. With this new approach, we achieve significant improvements on English-German, English-French and Chinese-to-English translation tasks.

1 Introduction

Neural machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014), directly applying a single neural network to transform the source sentence into the target sentence, has now reached impressive performance (Shen et al., 2015; Wu et al., 2016; Johnson et al., 2016; Gehring et al., 2017; Vaswani et al., 2017). The NMT typically consists of two sub neural networks. The encoder network reads and encodes the source sentence into a context vector, and the decoder network generates the target sentence iteratively based on the context vector. NMT can be studied in supervised and unsupervised learning settings. In the supervised setting, bilingual corpora is available for training the NMT model. In the unsupervised setting, we only have two independent monolingual corpora with one for each language and there is no bilingual training example to provide alignment information for the two languages. Due to lack of alignment information, the unsupervised NMT is considered more challenging. However, this task is very promising, since the monolingual corpora is usually easy to be collected.

Motivated by recent success in unsupervised cross-lingual embeddings (Artetxe et al., 2016; Zhang et al., 2017b; Conneau et al., 2017), the models proposed for unsupervised NMT often assume that a pair of sentences from two different languages can be mapped to a same latent representation in a shared-latent space (Lample et al., 2017; Artetxe et al., 2017b). Following this assumption, Lample et al. (2017) use a single encoder and a single decoder for both the source and target languages. The encoder and decoder, acting as a standard auto-encoder (AE), are trained to reconstruct the inputs. And Artetxe et al. (2017b) utilize a shared encoder but two independent decoders. With some good performance, they share a glaring defect, i.e., only one encoder is shared by the source and target languages. Although the shared encoder is vital for mapping sentences from different languages into the shared-latent space, it is weak in keeping the uniqueness and internal characteristics of each language, such as the style, terminology and sentence structure. Since each language has its own characteristics, the source and target languages should be encoded and learned independently. Therefore, we conjecture that the shared encoder may be a factor limit-

[1]Feng Wang is the corresponding author of this paper

ing the potential translation performance.

In order to address this issue, we extend the encoder-shared model, i.e., the model with one shared encoder, by leveraging two independent encoders with each for one language. Similarly, two independent decoders are utilized. For each language, the encoder and its corresponding decoder perform an AE, where the encoder generates the latent representations from the perturbed input sentences and the decoder reconstructs the sentences from the latent representations. To map the latent representations from different languages to a shared-latent space, we propose the weight-sharing constraint to the two AEs. Specifically, we share the weights of the last few layers of two encoders that are responsible for extracting high-level representations of input sentences. Similarly, we share the weights of the first few layers of two decoders. To enforce the shared-latent space, the word embeddings are used as a reinforced encoding component in our encoders. For cross-language translation, we utilize the back-translation following (Lample et al., 2017). Additionally, two different generative adversarial networks (GAN) (Yang et al., 2017), namely the local and global GAN, are proposed to further improve the cross-language translation. We utilize the local GAN to constrain the source and target latent representations to have the same distribution, whereby the encoder tries to fool a local discriminator which is simultaneously trained to distinguish the language of a given latent representation. We apply the global GAN to finetune the corresponding generator, i.e., the composition of the encoder and decoder of the other language, where a global discriminator is leveraged to guide the training of the generator by assessing how far the generated sentence is from the true data distribution [1]. In summary, we mainly make the following contributions:

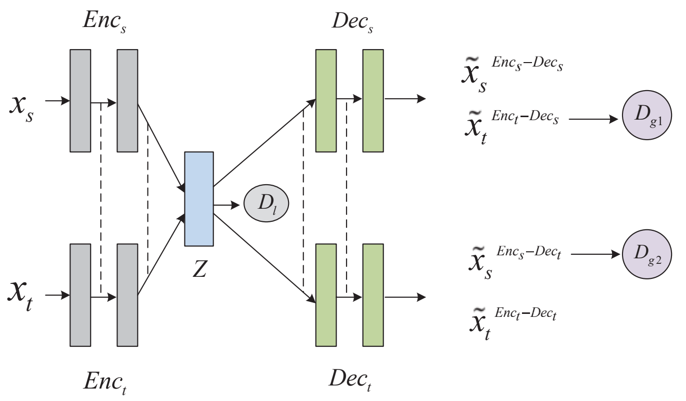
- We propose the weight-sharing constraint to unsupervised NMT, enabling the model to utilize an independent encoder for each language. To enforce the shared-latent space, we also propose the embedding-reinforced encoders and two different GANs for our model.
- We conduct extensive experiments on English-German, English-French and Chinese-to-English translation tasks. Experimental results show that the proposed approach consistently achieves great success.
- Last but not least, we introduce the directional self-attention to model temporal order information for the proposed model. Experimental results reveal that it deserves more efforts for researchers to investigate the temporal order information within self-attention layers of NMT.

[1]The code that we utilized to train and evaluate our models can be found at https://github.com/ZhenYangIACAS/unsupervised-NMT

2 Related Work

Several approaches have been proposed to train NMT models without direct parallel corpora. The scenario that has been widely investigated is one where two languages have little parallel data between them but are well connected by one pivot language. The most typical approach in this scenario is to independently translate from the source language to the pivot language and from the pivot language to the target language (Saha et al., 2016; Cheng et al., 2017). To improve the translation performance, Johnson et al. (2016) propose a multilingual extension of a standard NMT model and they achieve substantial improvement for language pairs without direct parallel training data.

Recently, motivated by the success of cross-lingual embeddings, researchers begin to show interests in exploring the more ambitious scenario where an NMT model is trained from monolingual corpora only. Lample et al. (2017) and Artetxe et al. (2017b) simultaneously propose an approach for this scenario, which is based on pre-trained cross lingual embeddings. Lample et al. (2017) utilizes a single encoder and a single decoder for both languages. The entire system is trained to reconstruct its perturbed input. For cross-lingual translation, they incorporate back-translation into the training procedure. Different from (Lample et al., 2017), Artetxe et al. (2017b) use two independent decoders with each for one language. The two works mentioned above both use a single shared encoder to guarantee the shared latent space. However, a concomitant defect is that the shared encoder is weak in keeping the uniqueness of each language. Our work also belongs to this more ambitious scenario, and to the best of our knowledge, we are one among the first endeavors to investigate how to train an NMT model with monolingual corpora only.



Enc_s
Dec_s
x_s
x_t
Z
D_l
$\tilde{x}_s^{Enc_s-Dec_s}$
$\tilde{x}_t^{Enc_t-Dec_s}$
D_{g1}
$\tilde{x}_s^{Enc_s-Dec_t}$
D_{g2}
$\tilde{x}_t^{Enc_t-Dec_t}$
Enc_t
Dec_t

Figure 1: The architecture of the proposed model. We implement the shared-latent space assumption using a weight sharing constraint where the connection of the last few layers in Enc_s and Enc_t are tied (illustrated with dashed lines) and the connection of the first few layers in Dec_s and Dec_t are tied. $\tilde{x}_s^{Enc_s-Dec_s}$ and $\tilde{x}_t^{Enc_t-Dec_t}$ are self-reconstructed sentences in each language. $\tilde{x}_s^{Enc_s-Dec_t}$ is the translated sentence from source to target and $\tilde{x}_t^{Enc_t-Dec_s}$ is the translation in reversed direction. D_l is utilized to assess whether the hidden representation of the encoder is from the source or target language. D_{g1} and D_{g2} are used to evaluate whether the translated sentences are realistic for each language respectively. Z represents the shared-latent space.

3 The Approach

3.1 Model Architecture

The model architecture, as illustrated in figure 1, is based on the AE and GAN. It consists of seven sub networks: including two encoders Enc_s and Enc_t, two decoders Dec_s and Dec_t, the local discriminator D_l, and the global discriminators D_{g1} and D_{g2}. For the encoder and decoder, we follow the newly emerged Transformer (Vaswani et al., 2017). Specifically, the encoder is composed of a stack of four identical layers [2]. Each layer consists of a multi-head self-attention and a simple position-wise fully connected feed-forward network. The decoder is also composed of four identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sublayer, which performs multi-head attention over the output of the encoder stack. For more details about the multi-head self-attention layer, we refer the reader to (Vaswani et al., 2017). We implement the local discriminator as a multi-layer perceptron and implement the global discriminator based on the convolutional neural network (CNN). Several ways exist to interpret the roles of the sub networks are summarised in table 1. The proposed system has several striking components , which are critical either for the system to be trained in an unsupervised manner or for improving the translation performance.

[2]The layer number is selected according to our preliminary experiment, which is presented in appendix **??**.

Networks	Roles
$\{Enc_s, Dec_s\}$	AE for source language
$\{Enc_t, Dec_t\}$	AE for target language
$\{Enc_s, Dec_t\}$	translation $source \rightarrow target$
$\{Enc_t, Dec_s\}$	translation $target \rightarrow source$
$\{Enc_s, D_l\}$	1st local GAN (GAN_{l1})
$\{Enc_t, D_l\}$	2nd local GAN (GAN_{l2})
$\{Enc_t, Dec_s, D_{g1}\}$	1st global GAN (GAN_{g1})
$\{Enc_s, Dec_t, D_{g2}\}$	2nd global GAN (GAN_{g2})

Table 1: Interpretation of the roles for the subnetworks in the proposed system.

Directional self-attention Compared to recurrent neural network, a disadvantage of the simple self-attention mechanism is that the temporal order information is lost. Although the Transformer applies the positional encoding to the sequence before processed by the self-attention, how to model temporal order information within an attention is still an open question. Following (Shen et al., 2017), we build the encoders in our model on the directional self-attention which utilizes the positional masks to encode temporal order information into attention output. More concretely, two positional masks, namely the forward mask M^f and backward mask M^b, are calculated as:

$$M_{ij}^f = \begin{cases} 0 & i < j \\ -\infty & otherwise \end{cases} \quad (1)$$

$$M_{ij}^b = \begin{cases} 0 & i > j \\ -\infty & otherwise \end{cases} \quad (2)$$

With the forward mask M^f, the later token only makes attention connections to the early tokens in the sequence, and vice versa with the backward mask. Similar to (Zhou et al., 2016; Wang et al., 2017), we utilize a self-attention network to process the input sequence in forward direction. The output of this layer is taken by an upper self-attention network as input, processed in the reverse direction.

Weight sharing Based on the shared-latent space assumption, we apply the weight sharing constraint to relate the two AEs. Specifically, we share the weights of the last few layers of the Enc_s and Enc_t, which are responsible for extracting high-level representations of the input sentences. Similarly, we also share the first few layers of the Dec_s and Dec_t, which are expected to decode high-level representations that are vital for reconstructing the input sentences. Compared to (Cheng et al., 2016; Saha et al., 2016) which use the fully shared encoder, we only share partial weights for the encoders and decoders. In the proposed model, the independent weights of the two encoders are expected to learn and encode the hidden features about the internal characteristics of each language, such as the terminology, style, and sentence structure. The shared weights are utilized to map the hidden features extracted by the independent weights to the shared-latent space.

Embedding reinforced encoder We use pretrained cross-lingual embeddings in the encoders that are kept fixed during training. And the fixed embeddings are used as a reinforced encoding component in our encoder. Formally, given the input sequence embedding vectors $E = \{e_1, \ldots, e_t\}$ and the initial output sequence of the encoder stack $H = \{h_1, \ldots, h_t\}$, we compute H_r as:

$$H_r = g \odot H + (1 - g) \odot E \quad (3)$$

where H_r is the final output sequence of the encoder which will be attended by the decoder (In Transformer, H is the final output of the encoder), g is a gate unit and computed as:

$$g = \sigma(W_1 E + W_2 H + b) \quad (4)$$

where W_1, W_2 and b are trainable parameters and they are shared by the two encoders. The motivation behind is twofold. Firstly, taking the fixed cross-lingual embedding as the other encoding component is helpful to reinforce the shared-latent space. Additionally, from the point of multi-channel encoders (Xiong et al., 2017), providing encoding components with different levels of composition enables the decoder to take pieces of source sentence at varying composition levels suiting its own linguistic structure.

3.2 Unsupervised Training

Based on the architecture proposed above, we train the NMT model with the monolingual corpora only using the following four strategies:

Denoising auto-encoding Firstly, we train the two AEs to reconstruct their inputs respectively. In this form, each encoder should learn to compose the embeddings of its corresponding language and each decoder is expected to learn to decompose this representation into its corresponding language. Nevertheless, without any constraint, the AE quickly learns to merely copy every word one by one, without capturing any internal structure of the language involved. To address this problem, we utilize the same strategy of denoising AE (Vincent et al., 2008) and add some noise to the input sentences (Hill et al., 2016; Artetxe et al., 2017b). To this end, we shuffle the input sentences randomly. Specifically, we apply a random permutation ε to the input sentence, verifying the condition:

$$|\varepsilon(i) - i| \leq \min(k([\frac{steps}{s}] + 1), n), \forall i \in \{1, n\} \quad (5)$$

where n is the length of the input sentence, $steps$ is the global steps the model has been updated, k and s are the tunable parameters which can be set by users beforehand. This way, the system needs to learn some useful structure of the involved languages to be able to recover the correct word order. In practice, we set $k = 2$ and $s = 100000$.

Back-translation In spite of denoising auto-encoding, the training procedure still involves a single language at each time, without considering our final goal of mapping an input sentence from the source/target language to the target/source language. For the cross language training, we utilize the back-translation approach for our unsupervised training procedure. Back-translation has shown its great effectiveness on improving NMT model with monolingual data and has been widely investigated by (Sennrich et al., 2015a; Zhang and Zong, 2016). In our approach, given an input sentence in a given language, we apply the corresponding encoder and the decoder of the other language to translate it to the other language [3]. By combining the translation with its original sentence, we get a pseudo-parallel corpus which is utilized to train the model to reconstruct the original sentence from its translation.

Local GAN Although the weight sharing constraint is vital for the shared-latent space assumption, it alone does not guarantee that the corresponding sentences in two languages will have the same or similar latent code. To further enforce the shared-latent space, we train a discriminative neural network, referred to as the local discriminator, to classify between the encoding of source sentences and the encoding of target sentences. The local discriminator, implemented as a multi-layer perceptron with two hidden layers of size 256, takes the output of the encoder, i.e., H_r calculated as equation 3, as input, and produces a binary prediction about the language of the input sentence. The local discriminator is trained to predict the language by minimizing the following cross-entropy loss:

$$\begin{aligned} L_{D_l}(\theta_{D_l}) = \\ - \mathbb{E}_{x \in x_s}[\log p(f = s|Enc_s(x))] \\ - \mathbb{E}_{x \in x_t}[\log p(f = t|Enc_t(x))] \end{aligned} \tag{6}$$

where θ_{D_l} represents the parameters of the local discriminator and $f \in \{s, t\}$. The encoders are trained to fool the local discriminator:

$$\begin{aligned} L_{Enc_s}(\theta_{Enc_s}) = \\ - \mathbb{E}_{x \in x_s}[\log p(f = t|Enc_s(x))] \end{aligned} \tag{7}$$

$$\begin{aligned} L_{Enc_t}(\theta_{Enc_t}) = \\ - \mathbb{E}_{x \in x_t}[\log p(f = s|Enc_t(x))] \end{aligned} \tag{8}$$

where θ_{Enc_s} and θ_{Enc_t} are the parameters of the two encoders.

Global GAN We apply the global GANs to fine tune the whole model so that the model is able to generate sentences undistinguishable from the true data, i.e., sentences in the training corpus. Different from the local GANs which updates the parameters of the encoders locally, the global GANs are utilized to update the whole parameters of the proposed model, including the parameters of encoders and decoders. The proposed model has two global GANs: GAN_{g1} and GAN_{g2}. In GAN_{g1}, the Enc_t and Dec_s act as the generator, which generates the sentence $\tilde{x}_t$ [4] from x_t. The D_{g1}, implemented based on CNN, assesses whether the generated sentence $\tilde{x}_t$ is the true target-language sentence or the generated sentence. The global discriminator aims to distinguish among the true sentences and generated sentences, and it is trained to minimize its classification error rate. During training, the D_{g1} feeds back its assessment to fine-tune the encoder Enc_t and decoder Dec_s. Since the machine translation is a sequence generation problem, following (Yang et al., 2017), we leverage policy gradient reinforcement training to back-propagate the assessment. We apply a similar processing to GAN_{g2} (The details about the architecture of the global discriminator and the training procedure of the global GANs can be seen in appendix **??** and **??**).

There are two stages in the proposed unsupervised training. In the first stage, we train the proposed model with denoising auto-encoding, back-translation and the local GANs, until no improvement is achieved on the development set. Specifically, we perform one batch of denoising auto-encoding for the source and target languages, one batch of back-translation for the two languages, and another batch of local GAN for the two languages. In the second stage, we fine tune the proposed model with the global GANs.

4 Experiments and Results

We evaluate the proposed approach on English-German, English-French and Chinese-to-English translation tasks [5]. We firstly describe the datasets, pre-processing and model hyper-parameters we used, then we introduce the baseline systems, and finally we present our experimental results.

4.1 Data Sets and Preprocessing

In English-German and English-French translation, we make our experiments comparable with previous work by using the datasets from the

[3]Since the quality of the translation shows little effect on the performance of the model (Sennrich et al., 2015a), we simply use greedy decoding for speed.

[4]The $\tilde{x}_t$ is $\tilde{x}_t^{Enc_t - Dec_s}$ in figure 1. We omit the superscript for simplicity.

[5]The reason that we do not conduct experiments on English-to-Chinese translation is that we do not get public test sets for English-to-Chinese.

WMT 2014 and WMT 2016 shared tasks respectively. For Chinese-to-English translation, we use the datasets from LDC, which has been widely utilized by previous works (Tu et al., 2017; Zhang et al., 2017a).

WMT14 English-French Similar to (Lample et al., 2017), we use the full training set of 36M sentence pairs and we lower-case them and remove sentences longer than 50 words, resulting in a parallel corpus of about 30M pairs of sentences. To guarantee no exact correspondence between the source and target monolingual sets, we build monolingual corpora by selecting English sentences from 15M random pairs, and selecting the French sentences from the complementary set. Sentences are encoded with byte-pair encoding (Sennrich et al., 2015b), which has an English vocabulary of about 32000 tokens, and French vocabulary of about 33000 tokens. We report results on $newstest2014$.

WMT16 English-German We follow the same procedure mentioned above to create monolingual training corpora for English-German translation, and we get two monolingual training data of 1.8M sentences each. The two languages share a vocabulary of about 32000 tokens. We report results on $newstest2016$.

LDC Chinese-English For Chinese-to-English translation, our training data consists of 1.6M sentence pairs randomly extracted from LDC corpora [6]. Since the data set is not big enough, we just build the monolingual data set by randomly shuffling the Chinese and English sentences respectively. In spite of the fact that some correspondence between examples in these two monolingual sets may exist, we never utilize this alignment information in our training procedure (see Section 3.2). Both the Chinese and English sentences are encoded with byte-pair encoding. We get an English vocabulary of about 34000 tokens, and Chinese vocabulary of about 38000 tokens. The results are reported on $NIST02$.

Since the proposed system relies on the pre-trained cross-lingual embeddings, we utilize the monolingual corpora described above to train the embeddings for each language independently by using word2vec (Mikolov et al., 2013). We then apply the public implementation [7] of the method proposed by (Artetxe et al., 2017a) to map these embeddings to a shared-latent space [8].

[6]LDC2002L27, LDC2002T01, LDC2002E18, LDC2003E07, LDC2004T08, LDC2004E12, LDC2005T10

[7]https://github.com/artetxem/vecmap

4.2 Model Hyper-parameters and Evaluation

Following the base model in (Vaswani et al., 2017), we set the dimension of word embedding as 512, dropout rate as 0.1 and the head number as 8. We use beam search with a beam size of 4 and length penalty $\alpha = 0.6$. The model is implemented in TensorFlow (Abadi et al., 2015) and trained on up to four K80 GPUs synchronously in a multi-GPU setup on a single machine.

For model selection, we stop training when the model achieves no improvement for the tenth evaluation on the development set, which is comprised of 3000 source and target sentences extracted randomly from the monolingual training corpora. Following (Lample et al., 2017), we translate the source sentences to the target language, and then translate the resulting sentences back to the source language. The quality of the model is then evaluated by computing the BLEU score over the original inputs and their reconstructions via this two-step translation process. The performance is finally averaged over two directions, i.e., from source to target and from target to source. BLEU (Papineni et al., 2002) is utilized as the evaluation metric. For Chinese-to-English, we apply the script *mteval-v11b.pl* to evaluate the translation performance. For English-German and English-French, we evaluate the translation performance with the script *multi-belu.pl* [9].

4.3 Baseline Systems

Word-by-word translation (WBW) The first baseline we consider is a system that performs word-by-word translations using the inferred bilingual dictionary. Specifically, it translates a sentence word-by-word, replacing each word with its nearest neighbor in the other language.

Lample et al. (2017) The second baseline is a previous work that uses the same training and testing sets with this paper. Their model belongs to the standard attention-based encoder-decoder framework, which implements the encoder using a bidirectional long short term memory network (LSTM) and implements the decoder using a simple forward LSTM. They apply one single encoder and

[8]The configuration we used to run these open-source toolkits can be found in appendix **??**

[9]https://github.com/moses-smt/mosesdecoder/blob/617e8c8/scripts/generic/multi-bleu.perl;mteval-v11b.pl

	en-de	de-en	en-fr	fr-en	zh-en
Supervised	24.07	26.99	30.50	30.21	40.02
Word-by-word	5.85	9.34	3.60	6.80	5.09
Lample et al. (2017)	9.64	13.33	15.05	14.31	-
The proposed approach	**10.86**	**14.62**	**16.97**	**15.58**	**14.52**

Table 2: The translation performance on English-German, English-French and Chinese-to-English test sets. The results of (Lample et al., 2017) are copied directly from their paper. We do not present the results of (Artetxe et al., 2017b) since we use different training sets.

decoder for the source and target languages.

Supervised training We finally consider exactly the same model as ours, but trained using the standard cross-entropy loss on the original parallel sentences. This model can be viewed as an upper bound for the proposed unsupervised model.

4.4 Results and Analysis

4.4.1 Number of weight-sharing layers

We firstly investigate how the number of weight-sharing layers affects the translation performance. In this experiment, we vary the number of weight-sharing layers in the AEs from 0 to 4. Sharing one layer in AEs means sharing one layer for the encoders and in the meanwhile, sharing one layer for the decoders. The BLEU scores of English-to-German, English-to-French and Chinese-to-English translation tasks are reported in figure 2. Each curve corresponds to a different translation task and the x-axis denotes the number of weight-sharing layers for the AEs. We find that the number of weight-sharing layers shows much effect on the translation performance. And the best translation performance is achieved when only one layer is shared in our system. When all of the four layers are shared, i.e., only one shared encoder is utilized, we get poor translation performance in all of the three translation tasks. This verifies our conjecture that the shared encoder is detrimental to the performance of unsupervised NMT especially for the translation tasks on distant language pairs. More concretely, for the related language pair translation, i.e., English-to-French, the encoder-shared model achieves -0.53 BLEU points decline than the best model where only one layer is shared. For the more distant language pair English-to-German, the encoder-shared model achieves more significant decline, i.e., -0.85 BLEU points decline. And for the most distant language pair Chinese-to-English, the decline is as large as -1.66 BLEU points. We explain this as that the more distant the language pair is, the more different characteristics they have. And the shared encoder is weak in keeping the unique characteristic of each language. Additionally, we also notice that using two completely independent encoders, i.e., setting the number of weight-sharing layers as 0, results in poor translation performance too. This confirms our intuition that the shared layers are vital to map the source and target latent representations to a shared-latent space. In the rest of our experiments, we set the number of weight-sharing layer as 1.

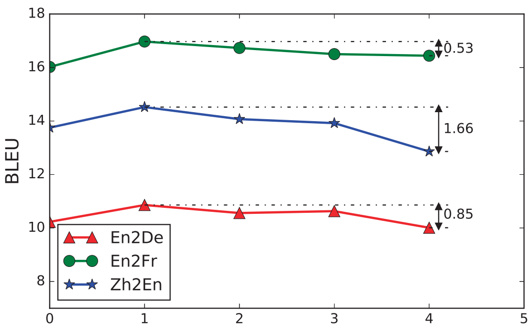


Figure 2: The effects of the weight-sharing layer number on English-to-German, English-to-French and Chinese-to-English translation tasks.

4.4.2 Translation results

Table 2 shows the BLEU scores on English-German, English-French and English-to-Chinese test sets. As it can be seen, the proposed approach obtains significant improvements than the word-by-word baseline system, with at least +5.01 BLEU points in English-to-German translation and up to +13.37 BLEU points in English-to-French translation. This shows that the proposed model only trained with monolingual data effec-

	en-de	de-en	en-fr	fr-en	zh-en
Without weight sharing	10.23	13.84	16.02	14.82	13.75
Without embedding-reinforced encoder	10.45	14.17	16.55	15.27	14.10
Without directional self-attention	10.60	14.21	16.82	15.30	14.29
Without local GANs	10.51	14.35	16.40	15.07	14.12
Without Global GANs	10.34	14.05	16.19	15.21	14.09
Full model	**10.86**	**14.62**	**16.97**	**15.58**	**14.52**

Table 3: Ablation study on English-German, English-French and Chinese-to-English translation tasks. Without weight sharing means no layers are shared in the two AEs.

tively learns to use the context information and the internal structure of each language. Compared to the work of (Lample et al., 2017), our model also achieves up to +1.92 BLEU points improvement on English-to-French translation task. We believe that the unsupervised NMT is very promising. However, there is still a large room for improvement compared to the supervised upper bound. The gap between the supervised and unsupervised model is as large as 12.3-25.5 BLEU points depending on the language pair and translation direction.

4.4.3 Ablation study

To understand the importance of different components of the proposed system, we perform an ablation study by training multiple versions of our model with some missing components: the local GANs, the global GANs, the directional self-attention, the weight-sharing, the embedding-reinforced encoders, etc. Results are reported in table 3. We do not test the the importance of the auto-encoding, back-translation and the pre-trained embeddings because they have been widely tested in (Lample et al., 2017; Artetxe et al., 2017b). Table 3 shows that the best performance is obtained with the simultaneous use of all the tested elements. The most critical component is the weight-sharing constraint, which is vital to map sentences of different languages to the shared-latent space. The embedding-reinforced encoder also brings some improvement on all of the translation tasks. When we remove the directional self-attention, we get up to -0.3 BLEU points decline. This indicates that it deserves more efforts to investigate the temporal order information in self-attention mechanism. The GANs also significantly improve the translation performance of our system. Specifically, the global GANs achieve improvement up to +0.78 BLEU points on English-to-French translation and the local GANs also obtain improvement up to +0.57 BLEU points on English-to-French translation. This reveals that the proposed model benefits a lot from the cross-domain loss defined by GANs.

5 Conclusion and Future work

The models proposed recently for unsupervised NMT use a single encoder to map sentences from different languages to a shared-latent space. We conjecture that the shared encoder is problematic for keeping the unique and inherent characteristic of each language. In this paper, we propose the weight-sharing constraint in unsupervised NMT to address this issue. To enhance the cross-language translation performance, we also propose the embedding-reinforced encoders, local GAN and global GAN into the proposed system. Additionally, the directional self-attention is introduced to model the temporal order information for our system.

We test the proposed model on English-German, English-French and Chinese-to-English translation tasks. The experimental results reveal that our approach achieves significant improvement and verify our conjecture that the shared encoder is really a bottleneck for improving the unsupervised NMT. The ablation study shows that each component of our system achieves some improvement for the final translation performance.

Unsupervised NMT opens exciting opportunities for the future research. However, there is still a large room for improvement compared to the supervised NMT. In the future, we would like to investigate how to utilize the monolingual data more effectively, such as incorporating the language model and syntactic information into unsupervised NMT. Besides, we decide to make more efforts to explore how to reinforce the temporal order information for the proposed model.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002102, and Beijing Engineering Research Center under Grant No. Z171100002217015. We would like to thank Xu Shuang for her preparing data used in this work. Additionally, we also want to thank Jiaming Xu, Suncong Zheng and Wenfu Wang for their invaluable discussions on this work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems .

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Conference on Empirical Methods in Natural Language Processing*. pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *Meeting of the Association for Computational Linguistics*. pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised neural machine translation .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928* .

Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, Wei Xu, Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint training for pivot-based neural machine translation. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*. pages 3974–3980.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2017. Word translation without parallel data .

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning .

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *TACL* .

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. *EMNLP* pages 1700–1709.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Association for Computational Linguistics* pages 311–318.

Amrita Saha, Mitesh M Khapra, Sarath Chandar, Janarthanan Rajendran, and Kyunghyun Cho. 2016. A correlational encoder decoder architecture for pivot based sequence generation. *arXiv preprint arXiv:1606.04754* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *Computer Science* .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding .

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* pages 3104–3112.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2017. Learning to remember translation history with a continuous cache .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need .

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 1096–1103.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. *arXiv preprint arXiv:1705.00861* .

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Hao Xiong, Zhongjun He, Xiaoguang Hu, and Hua Wu. 2017. Multi-channel encoder for neural machine translation. *arXiv preprint arXiv:1712.02109* .

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving neural machine translation with conditional sequence generative adversarial nets .

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017a. Prior knowledge integration for neural machine translation using posterior regularization. In *Meeting of the Association for Computational Linguistics*. pages 1514–1523.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*. pages 1535–1545.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Adversarial training for unsupervised bilingual lexicon induction. In *Meeting of the Association for Computational Linguistics*. pages 1959–1970.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199* .

Triangular Architecture for Rare Language Translation

Shuo Ren[1,2*], Wenhu Chen[3], Shujie Liu[4], Mu Li[4], Ming Zhou[4] and Shuai Ma[1,2]

[1]SKLSDE Lab, Beihang University, Beijing, China
[2]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing, China
[3]University of California, Santa Barbara, CA, USA
[4]Microsoft Research in Asia, Beijing, China

Abstract

Neural Machine Translation (NMT) performs poor on the low-resource language pair (X, Z), especially when Z is a rare language. By introducing another rich language Y, we propose a novel triangular training architecture (TA-NMT) to leverage bilingual data (Y, Z) (may be small) and (X, Y) (can be rich) to improve the translation performance of low-resource pairs. In this triangular architecture, Z is taken as the intermediate latent variable, and translation models of Z are jointly optimized with a unified bidirectional EM algorithm under the goal of maximizing the translation likelihood of (X, Y). Empirical results demonstrate that our method significantly improves the translation quality of rare languages on MultiUN and IWSLT2012 datasets, and achieves even better performance combining back-translation methods.

1 Introduction

In recent years, Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014) has achieved remarkable performance on many translation tasks (Jean et al., 2015; Sennrich et al., 2016; Wu et al., 2016; Sennrich et al., 2017). Being an end-to-end architecture, an NMT system first encodes the input sentence into a sequence of real vectors, based on which the decoder generates the target sequence word by word with the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). During training, NMT systems are optimized to maximize the translation probability of a given language pair with the Maximum Likelihood Estimation (MLE) method, which requires large bilingual data to fit the large parameter space. Without adequate data, which is common especially when it comes to a rare language, NMT usually falls short on low-resource language pairs (Zoph et al., 2016).

In order to deal with the data sparsity problem for NMT, exploiting monolingual data (Sennrich et al., 2015; Zhang and Zong, 2016; Cheng et al., 2016; Zhang et al., 2018; He et al., 2016) is the most common method. With monolingual data, the back-translation method (Sennrich et al., 2015) generates pseudo bilingual sentences with a target-to-source translation model to train the source-to-target one. By extending back-translation, source-to-target and target-to-source translation models can be jointly trained and boost each other (Cheng et al., 2016; Zhang et al., 2018). Similar to joint training (Cheng et al., 2016; Zhang et al., 2018), dual learning (He et al., 2016) designs a reinforcement learning framework to better capitalize on monolingual data and jointly train two models.

Instead of leveraging monolingual data (X or Z) to enrich the low-resource bilingual pair (X, Z), in this paper, we are motivated to introduce another rich language Y, by which additionally acquired bilingual data (Y, Z) and (X, Y) can be exploited to improve the translation performance of (X, Z). This requirement is easy to satisfy, especially when Z is a rare language but X is not. Under this scenario, (X, Y) can be a rich-resource pair and provide much bilingual data, while (Y, Z) would also be a low-resource pair mostly because Z is rare. For example, in the dataset IWSLT2012, there are only 112.6K bilingual sentence pairs of English-Hebrew, since Hebrew is a rare language. If French is introduced as the third language, we can have another low-resource bilingual data of French-Hebrew (116.3K sentence pairs), and easily-acquired bilingual data

[*]Contribution during internship at MSRA.

of the rich-resource pair English-French.

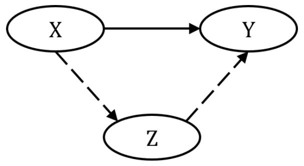


Figure 1: Triangular architecture for rare language translation. The solid lines mean rich-resource and the dash lines mean low-resource. X, Y and Z are three different languages.

With the introduced rich language Y, in this paper, we propose a novel triangular architecture (TA-NMT) to exploit the additional bilingual data of (Y, Z) and (X, Y), in order to get better translation performance on the low-resource pair (X, Z), as shown in Figure 1. In this architecture, (Y, Z) is used for training another translation model to score the translation model of (X, Z), while (X, Y) is used to provide large bilingual data with favorable alignment information.

Under the motivation to exploit the rich-resource pair (X, Y), instead of modeling $X \Rightarrow Z$ directly, our method starts from modeling the translation task $X \Rightarrow Y$ while taking Z as a latent variable. Then, we decompose $X \Rightarrow Y$ into two phases for training two translation models of low-resource pairs ((X, Z) and (Y, Z)) respectively. The first translation model generates a sequence in the hidden space of Z from X, based on which the second one generates the translation in Y. These two models can be optimized jointly with an Expectation Maximization (EM) framework with the goal of maximizing the translation probability $p(y|x)$. In this framework, the two models can boost each other by generating pseudo bilingual data for model training with the weights scored from the other. By reversing the translation direction of $X \Rightarrow Y$, our method can be used to train another two translation models $p(z|y)$ and $p(x|z)$. Therefore, the four translation models ($p(z|x)$, $p(x|z)$, $p(z|y)$ and $p(y|z)$) of the rare language Z can be optimized jointly with our proposed unified bidirectional EM algorithm.

Experimental results on the MultiUN and IWSLT2012 datasets demonstrate that our method can achieve significant improvements for rare languages translation. By incorporating back-translation (a method leveraging more monolingual data) into our method, TA-NMT can achieve even further improvements.

Our contributions are listed as follows:

- We propose a novel triangular training architecture (TA-NMT) to effectively tackle the data sparsity problem for rare languages in NMT with an EM framework.
- Our method can exploit two additional bilingual datasets at both the model and data levels by introducing another rich language.
- Our method is a unified bidirectional EM algorithm, in which four translation models on two low-resource pairs are trained jointly and boost each other.

2 Method

As shown in Figure 1, our method tries to leverage (X, Y) (a rich-resource pair) and (Y, Z) to improve the translation performance of low-resource pair (X, Z), during which translation models of (X, Z) and (Y, Z) can be improved jointly.

Instead of directly modeling the translation probabilities of low-resource pairs, we model the rich-resource pair translation $X \Rightarrow Y$, with the language Z acting as a bridge to connect X and Y. We decompose $X \Rightarrow Y$ into two phases for training two translation models. The first model $p(z|x)$ generates the latent translation in Z from the input sentence in X, based on which the second one $p(y|z)$ generate the final translation in language Y. Following the standard EM procedure (Borman, 2004) and Jensen's inequality, we derive the lower bound of $p(y|x)$ over the whole training data D as follows:

$$\begin{aligned}
&L(\Theta; D) \\
&= \sum_{(x,y)\in D} \log p(y|x) \\
&= \sum_{(x,y)\in D} \log \sum_z p(z|x)p(y|z) \\
&= \sum_{(x,y)\in D} \log \sum_z Q(z)\frac{p(z|x)p(y|z)}{Q(z)} \\
&\geq \sum_{(x,y)\in D} \sum_z Q(z) \log \frac{p(z|x)p(y|z)}{Q(z)} \\
&\doteq \mathcal{L}(Q)
\end{aligned} \tag{1}$$

where Θ is the model parameters set of $p(z|x)$ and $p(y|z)$, and $Q(z)$ is an arbitrary posterior distribution of z. We denote the lower-bound in the last but one line as $\mathcal{L}(Q)$. Note that we use an approximation that $p(y|x,z) \approx p(y|z)$ due to the semantic equivalence of parallel sentences x and y.

In the following subsections, we will first propose our EM method in subsection 2.1 based on the lower-bound derived above. Next, we will extend our method to two directions and give our unified bidirectional EM training in subsection 2.2. Then, in subsection 2.3, we will discuss more training details of our method and present our algorithm in the form of pseudo codes.

2.1 EM Training

To maximize $L(\Theta; D)$, the EM algorithm can be leveraged to maximize its lower bound $\mathcal{L}(Q)$. In the E-step, we calculate the expectation of the variable z using current estimate for the model, namely find the posterior distribution $Q(z)$. In the M-step, with the expectation $Q(z)$, we maximize the lower bound $\mathcal{L}(Q)$. Note that conditioned on the observed data and current model, the calculation of $Q(z)$ is intractable, so we choose $Q(z) = p(z|x)$ approximately.

M-step: In the M-step, we maximize the lower bound $\mathcal{L}(Q)$ w.r.t model parameters given $Q(z)$. By substituting $Q(z) = p(z|x)$ into $\mathcal{L}(Q)$, we can get the M-step as follows:

$$\begin{aligned}\Theta_{y|z} &= \arg\max_{\Theta_{y|z}} \mathcal{L}(Q) \\ &= \arg\max_{\Theta_{y|z}} \sum_{(x,y)\in D} \sum_{z} p(z|x) \log p(y|z) \\ &= \arg\max_{\Theta_{y|z}} \sum_{(x,y)\in D} E_{z\sim p(z|x)} \log p(y|z)\end{aligned} \tag{2}$$

E-step: The approximate choice of $Q(z)$ brings in a gap between $\mathcal{L}(Q)$ and $L(\Theta; D)$, which can be minimized in the E-step with Generalized EM method (McLachlan and Krishnan, 2007). According to Bishop (2006), we can write this gap explicitly as follows:

$$\begin{aligned}L(\Theta; D) - \mathcal{L}(Q) &= \sum_{z} Q(z) \log \frac{Q(z)}{p(z|y)} \\ &= KL(Q(z)||p(z|y)) \\ &= KL(p(z|x)||p(z|y))\end{aligned} \tag{3}$$

where $KL(\cdot)$ is the KullbackLeibler divergence, and the approximation that $p(z|x,y) \approx p(z|y)$ is also used above.

In the E-step, we minimize the gap between $\mathcal{L}(Q)$ and $L(\Theta; D)$ as follows:

$$\Theta_{z|x} = \arg\min_{\Theta_{z|x}} KL(p(z|x)||p(z|y)) \tag{4}$$

To sum it up, the E-step optimizes the model $p(z|x)$ by minimizing the gap between $\mathcal{L}(Q)$ and $L(\Theta; D)$ to get a better lower bound $\mathcal{L}(Q)$. This lower bound is then maximized in the M-step to optimize the model $p(y|z)$. Given the new model $p(y|z)$, the E-step tries to optimize $p(z|x)$ again to find a new lower bound, with which the M-step is re-performed. This iteration process continues until the models converge, which is guaranteed by the convergence of the EM algorithm.

2.2 Unified Bidirectional Training

The model $p(z|y)$ is used as an approximation of $p(z|x,y)$ in the E-step optimization (Equation 3). Due to the low resource property of the language pair (Y, Z), $p(z|y)$ cannot be well trained. To solve this problem, we can jointly optimize $p(x|z)$ and $p(z|y)$ similarly by maximizing the reverse translation probability $p(x|y)$.

We now give our unified bidirectional generalized EM procedures as follows:

- Direction of $X \Rightarrow Y$

  E: Optimize $\Theta_{z|x}$.

  $$\arg\min_{\Theta_{z|x}} KL(p(z|x)||p(z|y)) \tag{5}$$

  M: Optimize $\Theta_{y|z}$.

  $$\arg\max_{\Theta_{y|z}} \sum_{(x,y)\in D} E_{z\sim p(z|x)} \log p(y|z) \tag{6}$$

- Direction of $Y \Rightarrow X$

  E: Optimize $\Theta_{z|y}$.

  $$\arg\min_{\Theta_{z|y}} KL(p(z|y)||p(z|x)) \tag{7}$$

  M: Optimize $\Theta_{x|z}$.

  $$\arg\max_{\Theta_{x|z}} \sum_{(x,y)\in D} E_{z\sim p(z|y)} \log p(x|z) \tag{8}$$

Based on the above derivation, the whole architecture of our method can be illustrated in Figure 2, where the dash arrows denote the direction of $p(y|x)$, in which $p(z|x)$ and $p(y|z)$ are trained jointly with the help of $p(z|y)$, while the solid ones denote the direction of $p(x|y)$, in which $p(z|y)$ and $p(x|z)$ are trained jointly with the help of $p(z|x)$.

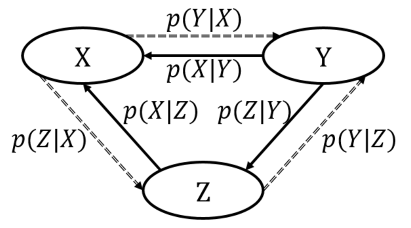


Figure 2: Triangular Learning Architecture for Low-Resource NMT

2.3 Training Details

A major difficulty in our unified bidirectional training is the exponential search space of the translation candidates, which could be addressed by either sampling (Shen et al., 2015; Cheng et al., 2016) or mode approximation (Kim and Rush, 2016). In our experiments, we leverage the sampling method and simply generate the top target sentence for approximation.

In order to perform gradient descend training, the parameter gradients for Equations 5 and 7 are formulated as follows:

$$\begin{aligned}
&\nabla_{\Theta_{z|x}} KL(p(z|x)||p(z|y)) \\
&= E_{z\sim p(z|x)} \log \frac{p(z|x)}{p(z|y)} \nabla_{\Theta_{z|x}} \log p(z|x) \\
&\nabla_{\Theta_{z|y}} KL(p(z|y)||p(z|x)) \\
&= E_{z\sim p(z|y)} \log \frac{p(z|y)}{p(z|x)} \nabla_{\Theta_{z|y}} \log p(z|y)
\end{aligned} \tag{9}$$

Similar to reinforcement learning, models $p(z|x)$ and $p(z|y)$ are trained using samples generated by the models themselves. According to our observation, some samples are noisy and detrimental to the training process. One way to tackle this is to filter out the bad ones using some additional metrics (BLEU, etc.). Nevertheless, in our settings, BLEU scores cannot be calculated during training due to the absence of the golden targets (z is generated based on x or y from the rich-resource pair (x, y)). Therefore we choose IBM model1 scores to weight the generated translation candidates, with the word translation probabilities calculated based on the given bilingual data (the low-resource pair (x, z) or (y, z)). Additionally, to stabilize the training process, the pseudo samples generated by model $p(z|x)$ or $p(z|y)$ are mixed with true bilingual samples in the same mini-batch with the ratio of 1-1. The whole training procedure is described in the following Algorithm 1, where the 5th and 9th steps are generating pseudo data.

Algorithm 1 Training low-resource translation models with the triangular architecture

```
Input: Rich-resource bilingual data $(x, y)$; low-resource bilingual data $(x, z)$ and $(y, z)$
Output: Parameters $\Theta_{z|x}$, $\Theta_{y|z}$, $\Theta_{z|y}$ and $\Theta_{x|z}$
 1: Pre-train $p(z|x), p(z|y), p(x|z), p(y|z)$
 2: while not convergence do
 3:     Sample $(x, y), (x^*, z^*), (y^*, z^*) \in D$
 4:             $\triangleright X \Rightarrow Y$: Optimize $\Theta_{z|x}$ and $\Theta_{y|z}$
 5:     Generate $z'$ from $p(z'|x)$ and build the training batches $B_1 = (x, z') \cup (x^*, z^*)$, $B_2 = (y, z') \cup (y^*, z^*)$
 6:     E-step: update $\Theta_{z|x}$ with $B_1$ (Equation 5)
 7:     M-step: update $\Theta_{y|z}$ with $B_2$ (Equation 6)
 8:             $\triangleright Y \Rightarrow X$: Optimize $\Theta_{z|y}$ and $\Theta_{x|z}$
 9:     Generate $z'$ from $p(z'|y)$ and build the training batches $B_3 = (y, z') \cup (y^*, z^*)$, $B_4 = (x, z') \cup (x^*, z^*)$
10:     E-step: update $\Theta_{z|y}$ with $B_3$ (Equation 7)
11:     M-step: update $\Theta_{x|z}$ with $B_4$ (Equation 8)
12: end while
13: return $\Theta_{z|x}$, $\Theta_{y|z}$, $\Theta_{z|y}$ and $\Theta_{x|z}$
```

3 Experiments

3.1 Datasets

In order to verify our method, we conduct experiments on two multilingual datasets. The one is MultiUN (Eisele and Chen, 2010), which is a collection of translated documents from the United Nations, and the other is IWSLT2012 (Cettolo et al., 2012), which is a set of multilingual transcriptions of TED talks. As is mentioned in section 1, our method is compatible with methods exploiting monolingual data. So we also find some extra monolingual data of rare languages in both datasets and conduct experiments incorporating back-translation into our method.

MultiUN: English-French (EN-FR) bilingual data are used as the rich-resource pair (X, Y). Arabic (AR) and Spanish (ES) are used as two simulated rare languages Z. We randomly choose subsets of bilingual data of (X, Z) and (Y, Z) in the original dataset to simulate low-resource situations, and make sure there is no overlap in Z between chosen data of (X, Z) and (Y, Z).

IWSLT2012[1]**:** English-French is used as the rich-resource pair (X, Y), and two rare languages Z are Hebrew (HE) and Romanian (RO) in our

[1]https://wit3.fbk.eu/mt.php?release=2012-02-plain

Pair	MultiUN		IWSLT2012	
	Lang	Size	Lang	Size
(X, Y)	EN-FR	9.9 M	EN-FR [3]	7.9 M
(X, Z)	EN-AR	116 K	EN-HE	112.6 K
(Y, Z)	FR-AR	116 K	FR-HE	116.3 K
mono Z	AR	3 M	HE	512.5 K
(X, Z)	EN-ES	116 K	EN-RO [4]	467.3 K
(Y, Z)	FR-ES	116 K	FR-RO	111.6 K
mono Z	ES	3 M	RO	885.0 K

Table 1: training data size of each language pair.

choice. Note that in this dataset, low-resource pairs (X, Z) and (Y, Z) are severely overlapped in Z. In addition, English-French bilingual data from WMT2014 dataset are also used to enrich the rich-resource pair. We also use additional English-Romanian bilingual data from Europarlv7 dataset (Koehn, 2005). The monolingual data of Z (HE and RO) are taken from the web[2].

In both datasets, all sentences are filtered within the length of 5 to 50 after tokenization. Both the validation and the test sets are 2,000 parallel sentences sampled from the bilingual data, with the left as training data. The size of training data of all language pairs are shown in Table 1.

3.2 Baselines

We compare our method with four baseline systems. The first baseline is the **RNNSearch** model (Bahdanau et al., 2014), which is a sequence-to-sequence model with attention mechanism trained with given small-scale bilingual data. The trained translation models are also used as pre-trained models for our subsequent training processes.

The second baseline is **PBSMT** (Koehn et al., 2003), which is a phrase-based statistical machine translation system. PBSMT is known to perform well on low-resource language pairs, so we want to compare it with our proposed method. And we use the public available implementation of Moses[5] for training and test in our experiments.

The third baseline is a teacher-student alike method (Chen et al., 2017). For the sake of brevity, we will denote it as **T-S**. The process is illustrated in Figure 3. We treat this method as a second baseline because it can also be regarded as a method exploiting (Y, Z) and (X, Y) to improve the translation of (X, Z) if we regard (X, Z) as the zero-resource pair and $p(x|y)$ as the teacher model when training $p(z|x)$ and $p(x|z)$.

The fourth baseline is back-translation (Sennrich et al., 2015). We will denote it as **Back-Trans**. More concretely, to train the model $p(z|x)$, we use extra monolingual Z described in Table 1 to do back-translation; to train the model $p(x|z)$, we use monolingual X taken from (X, Y). Procedures for training $p(z|y)$ and $p(y|z)$ are similar. This method use extra monolingual data of Z compared with our TA-NMT method. But we can incorporate it into our method.

Method	Resources
PBSMT	(X, Z), (Y, Z)
RNNSearch	(X, Z), (Y, Z)
T-S	(X, Z), (Y, Z), (X, Y)
BackTrans	(X, Z), (Y, Z), (X, Y), mono Z
TA-NMT	(X, Z), (Y, Z), (X, Y)
TA-NMT(GI)	(X, Z), (Y, Z), (X, Y), mono Z

Table 2: Resources that different methods use

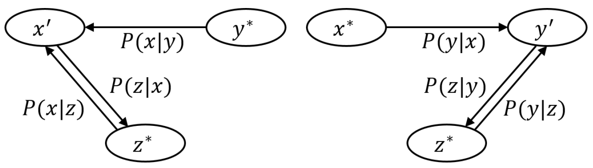


Figure 3: A teacher-student alike method for low-resource translation. For training $p(z|x)$ and $p(x|z)$, we mix the true pair $(y^*, z^*) \in D$ with the pseudo pair (x', z^*) generated by teacher model $p(x'|y^*)$ in the same mini-batch. The training procedure of $p(z|y)$ and $p(y|z)$ is similar.

3.3 Overall Results

Experimental results on both datasets are shown in Table 3 and 4 respectively, in which **RNNSearch**, **PBSMT**, **T-S** and **BackTrans** are four baselines. **TA-NMT** is our proposed method, and **TA-NMT(GI)** is our method incorporating back-translation as good initialization. For the purpose of clarity and a fair comparison, we list the resources that different methods exploit in Table 2.

From Table 3 on MultiUN, the performance of RNNSearch is relatively poor. As is expected, PBSMT performs better than RNNSearch on low-resource pairs by the average of 1.78 BLEU. The T-S method which can doubling the training data

[2]https://github.com/ajinkyakulkarni14/TED-Multilingual-Parallel-Corpus
[3]together with WMT2014
[4]together with Europarlv7
[5]http://www.statmt.org/moses/

Method	EN2AR (X⇒Z)	AR2EN (Z⇒X)	FR2AR (Y⇒Z)	AR2FR (Z⇒Y)	Ave	EN2ES (X⇒Z)	ES2EN (Z⇒X)	FR2ES (Y⇒Z)	ES2FR (Z⇒Y)	Ave
RNNSearch	18.03	31.40	13.42	22.04	21.22	38.77	36.51	32.92	33.05	35.31
PBSMT	19.44	30.81	15.27	23.65	22.29	38.47	36.64	34.99	33.98	36.02
T-S	19.02	32.47	14.59	23.53	22.40	39.75	38.02	33.67	34.04	36.57
BackTrans	22.19	32.02	15.85	23.57	23.73	42.27	38.42	35.81	34.25	37.76
TA-NMT	20.59	33.22	14.64	24.45	23.23	40.85	39.06	34.52	34.39	37.21
TA-NMT(GI)	**23.16**	**33.64**	**16.50**	**25.07**	**24.59**	**42.63**	**39.53**	**35.87**	**35.21**	**38.31**

Table 3: Test BLEU on MultiUN Dataset.

Method	EN2HE (X⇒Z)	HE2EN (Z⇒X)	FR2HE (Y⇒Z)	HE2FR (Z⇒Y)	Ave	EN2RO (X⇒Z)	RO2EN (Z⇒X)	FR2RO (Y⇒Z)	RO2FR (Z⇒Y)	Ave
RNNSearch	17.94	28.32	11.86	21.67	19.95	31.44	40.63	17.34	25.20	28.65
PBSMT	17.39	28.05	12.77	21.87	20.02	31.51	39.98	18.13	25.47	28.77
T-S	17.97	28.42	12.04	21.99	20.11	31.80	40.86	17.94	25.69	29.07
BackTrans	18.69	28.55	12.31	21.63	20.20	32.18	41.03	18.19	25.30	29.18
TA-NMT	19.19	29.28	12.76	22.62	20.96	33.65	41.93	18.53	26.35	30.12
TA-NMT(GI)	**19.90**	**29.94**	**13.54**	**23.25**	**21.66**	**34.41**	**42.61**	**19.30**	**26.53**	**30.71**

Table 4: Test BLEU on IWSLT Dataset.

for both (X, Z) and (Y, Z) by generating pseudo data from each other, leads up to 1.1 BLEU points improvement on average over RNNSearch. Compared with T-S, our method gains a further improvement of about 0.9 BLEU on average, because our method can better leverage the rich-resource pair (X, Y). With extra large monolingual Z introduced, BackTrans can improve the performance of $p(z|x)$ and $p(z|y)$ significantly compared with all the methods without monolingual Z. However TA-NMT is comparable with or even better than BackTrans for $p(x|z)$ and $p(y|z)$ because both of the methods leverage resources from rich-resource pair (X, Y), but BackTrans does not use the alignment information it provides. Moreover, with back-translation as good initialization, further improvement is achieved by TA-NMT(GI) of about 0.7 BLEU on average over BackTrans.

In Table 4, we can draw the similar conclusion. However, different from MultiUN, in the EN-FR-HE group of IWSLT, (X, Z) and (Y, Z) are severely overlapped in Z. Therefore, T-S cannot improve the performance obviously (only about 0.2 BLEU) on RNNSearch because it fails to essentially double training data via the teacher model. As for EN-FR-RO, with the additionally introduced EN-RO data from Europarlv7, which has no overlap in RO with FR-RO, T-S can improve the average performance more than the EN-FR-HE group. TA-NMT outperforms T-S by 0.93 BLEU on average. Note that even though BackTrans uses extra monolingual Z, the improvements are not so obvious as the former dataset, the reason for which we will delve into in the next subsection. Again, with back-translation as good initialization, TA-NMT(GI) can get the best result.

Note that BLEU scores of TA-NMT are lower than BackTrans in the directions of X⇒Z and Y⇒Z. The reason is that the resources used by these two methods are different, as shown in Table 2. To do back translation in two directions (e.g., X⇒Z and Z⇒X), we need monolingual data from both sides (e.g., X and Z), however, in TA-NMT, the monolingual data of Z is not necessary. Therefore, in the translation of X⇒Z or Y⇒Z, BackTrans uses additional monolingual data of Z while TA-NMT does not, that is why BackTrans outperforms TA-NMT in these directions. Our method can leverage back translation as a good initialization, aka TA-NMT(GI) , and outperforms BackTrans on all translation directions.

The average test BLEU scores of different methods in each data group (EN-FR-AR, EN-FR-ES, EN-FR-HE, and EN-FR-RO) are listed in the column **Ave** of the tables for clear comparison.

3.4 The Effect of Extra Monolingual Data

Comparing the results of BackTrans and TA-NMT(GI) on both datasets, we notice the improvements of both methods on IWSLT are not as significant as MultiUN. We speculate the reason is the relatively less amount of monolingual Z we use in the experiments on IWSLT as shown in Table 1. So we conduct the following experiment to verify the conjecture by changing the scale of monolingual Arabic data in the MultiUN dataset, of which the data utilization rates are set to 0%, 10%, 30%, 60% and 100% respectively. Then we compare the performance of BackTrans and TA-NMT(GI) in the EN-FR-AR group. As Figure 4 shows, the amount of monolingual Z actually has a big effect on the results, which can also verify our conjecture above upon the less significant improvement of BackTrans and TA-NMT(GI) on IWSLT. In addition, even with poor "good-initialization", TA-NMT(GI) still get the best results.

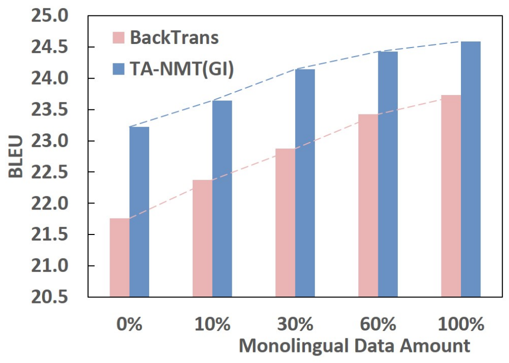


Figure 4: Test BLEU of the EN-FR-AR group performed by BackTrans and TA-NMT(GI) with different amount of monolingual Arabic data.

3.5 EM Training Curves

To better illustrate the behavior of our method, we print the training curves in both the M-steps and E-steps of TA-NMT and TA-NMT(GI) in Figure 5 above. The chosen models printed in this figure are EN2AR and AR2FR on MultiUN, and EN2RO and RO2FR on IWLST.

From Figure 5, we can see that the two low-resource translation models are improved nearly simultaneously along with the training process, which verifies our point that two weak models could boost each other in our EM framework. Notice that at the early stage, the performance of all models stagnates for several iterations, especially of TA-NMT. The reason could be that the pseudo bilingual data and the true training data are heterogeneous, and it may take some time for the models to adapt to a new distribution which both models agree. Compared with TA-NMT, TA-NMT(GI) are more stable, because the models may have adapted to a mixed distribution of heterogeneous data in the preceding back-translation phase.

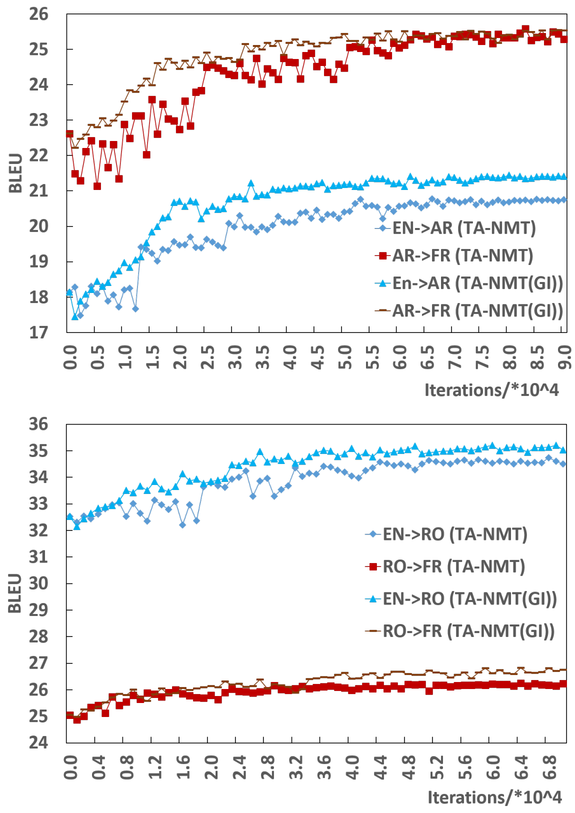


Figure 5: BLEU curves on validation sets during the training processes of TA-NMT and TA-NMT(GI). (Top: EN2AR (the E-step) and AR2FR (the M-step); Bottom: EN2RO (the E-step) and RO2FR (the M-step))

3.6 Reinforcement Learning Mechanism in Our Method

As shown in Equation 9, the E-step actually works as a reinforcement learning (RL) mechanism. Models $p(z|x)$ and $p(z|y)$ generate samples by themselves and receive rewards to update their parameters. Note that the reward here is described by the log terms in Equation 9, which is derived from our EM algorithm rather than defined artificially. In Table 5, we do a case study of the EN2ES translation sampled by $p(z|x)$ as well as its time-step rewards during the E-step.

In the first case, the best translation of "political" is "políticos". When the model $p(z|x)$ generates an inaccurate one "políticas", it receives a negative reward (-0.01), with which the model parameters will be updated accordingly. In the sec-

Source	in concluding , poverty eradication requires political will and commitment .
Output	en (0.66) conclusión (0.80) , (0.14) la (0.00) erradicación (1.00) de (0.40) la (0.00) pobreza (0.90) requiere (0.10) voluntad (1.00) y (0.46) compromiso (0.90) políticas (-0.01) . (1.00)
Reference	en conclusión , la erradicación de la pobreza necesita la voluntad y compromiso políticos .
Source	visit us and get to know and love berlin !
Output	visita (0.00) y (0.05) se (0.00) a (0.17) saber (0.00) y (0.04) a (0.01) berlín (0.00) ! (0.00)
Reference	visítanos y llegar a saber y amar a berlín .
Source	legislation also provides an important means of recognizing economic , social and cultural rights at the domestic level .
Output	la (1.00) legislación (0.34) tambin (1.00) constituye (0.60) un (1.00) medio (0.22) importante (0.74) de (0.63) reconocer (0.21) los (0.01) derechos (0.01) econmicos (0.03) , (0.01) sociales (0.02) y (0.01) culturales (1.00) a (0.00) nivel (0.40) nacional (1.00) . (0.03)
Reference	la legislación también constituye un medio importante de reconocer los derechos económicos , iales y culturales a nivel nacional .

Table 5: English to Spanish translation sampled in the E-step as well as its time-step rewards.

ond case, the output misses important words and is not fluent. Rewards received by the model $p(z|x)$ are zero for nearly all tokens in the output, leading to an invalid updating. In the last case, the output sentence is identical to the human reference. The rewards received are nearly all positive and meaningful, thus the RL rule will update the parameters to encourage this translation candidate.

4 Related Work

NMT systems, relying heavily on the availability of large bilingual data, result in poor translation quality for low-resource pairs (Zoph et al., 2016). This low-resource phenomenon has been observed in much preceding work. A very common approach is exploiting monolingual data of both source and target languages (Sennrich et al., 2015; Zhang and Zong, 2016; Cheng et al., 2016; Zhang et al., 2018; He et al., 2016).

As a kind of data augmentation technique, exploiting monolingual data can enrich the training data for low-resource pairs. Sennrich et al. (2015) propose back-translation, exploits the monolingual data of the target side, which is then used to generate pseudo bilingual data via an additional target-to-source translation model. Different from back-translation, Zhang and Zong (2016) propose two approaches to use source-side monolingual data, of which the first is employing a self-learning algorithm to generate pseudo data, while the second is using two NMT models to predict the translation and to reorder the source-side monolingual sentences. As an extension to these two methods, Cheng et al. (2016) and Zhang et al. (2018) combine two translation directions and propose a training framework to jointly optimize the source-to-target and target-to-source translation models. Similar to joint training, He et al. (2016) propose a dual learning framework with a reinforcement learning mechanism to better leverage monolingual data and make two translation models promote each other. All of these methods are concentrated on exploiting either the monolingual data of the source and target language or both of them.

Our method takes a different angle but is compatible with existing approaches, we propose a novel triangular architecture to leverage two additional language pairs by introducing a third rich language. By combining our method with existing approaches such as back-translation, we can make a further improvement.

Another approach for tackling the low-resource translation problem is multilingual neural machine translation (Firat et al., 2016), where different encoders and decoders for all languages with a shared attention mechanism are trained. This method tends to exploit the network architecture to relate low-resource pairs. Our method is different from it, which is more like a training method rather than network modification.

5 Conclusion

In this paper, we propose a triangular architecture (TA-NMT) to effectively tackle the problem of low-resource pairs translation with a unified bidirectional EM framework. By introducing another rich language, our method can better exploit the additional language pairs to enrich the original low-resource pair. Compared with the RNNSearch (Bahdanau et al., 2014), a teacher-student alike method (Chen et al., 2017) and the back-translation (Sennrich et al., 2015) on the same data level, our method achieves significant improvement on the MutiUN and IWSLT2012 datasets. Note that our method can be combined with methods exploiting monolingual data for NMT low-resource problem such as back-translation and make further improvements.

In the future, we may extend our architecture to other scenarios, such as totally unsupervised training with no bilingual data for the rare language.

Acknowledgments

We thank Zhirui Zhang and Shuangzhi Wu for useful discussions. This work is supported in part by NSFC U1636210, 973 Program 2014CB340300, and NSFC 61421003.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

Sean Borman. 2004. The expectation maximization algorithm-a short tutorial. *Submitted for publication*, pages 1–9.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, volume 261, page 268.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. *arXiv preprint arXiv:1705.00753*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*.

Andreas Eisele and Yu Chen. 2010. Multiun: A multilingual corpus from united nation documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA).

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *WMT@ EMNLP*, pages 134–140.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Geoffrey McLachlan and Thriyambakam Krishnan. 2007. *The EM algorithm and extensions*, volume 382. John Wiley & Sons.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural mt systems for wmt17. *arXiv preprint arXiv:1708.00726*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*, pages 1535–1545.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. In *AAAI*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates

Taku Kudo
Google, Inc.
`taku@google.com`

Abstract

Subword units are an effective way to alleviate the open vocabulary problems in neural machine translation (NMT). While sentences are usually converted into unique subword sequences, subword segmentation is potentially ambiguous and multiple segmentations are possible even with the same vocabulary. The question addressed in this paper is whether it is possible to harness the segmentation ambiguity as a noise to improve the robustness of NMT. We present a simple regularization method, subword regularization, which trains the model with multiple subword segmentations probabilistically sampled during training. In addition, for better subword sampling, we propose a new subword segmentation algorithm based on a unigram language model. We experiment with multiple corpora and report consistent improvements especially on low resource and out-of-domain settings.

1 Introduction

Neural Machine Translation (NMT) models (Bahdanau et al., 2014; Luong et al., 2015; Wu et al., 2016; Vaswani et al., 2017) often operate with fixed word vocabularies, as their training and inference depend heavily on the vocabulary size. However, limiting vocabulary size increases the amount of unknown words, which makes the translation inaccurate especially in an open vocabulary setting.

A common approach for dealing with the open vocabulary issue is to break up rare words into subword units (Schuster and Nakajima, 2012; Chitnis and DeNero, 2015; Sennrich et al., 2016; Wu et al., 2016). Byte-Pair-Encoding (BPE) (Sennrich et al., 2016) is a de facto standard subword segmentation algorithm applied to many NMT systems and achieving top translation quality in several shared tasks (Denkowski and Neubig, 2017; Nakazawa et al., 2017). BPE segmentation gives a good balance between the vocabulary size and the decoding efficiency, and also sidesteps the need for a special treatment of unknown words.

Subwords (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_/world	320 585 356 137 7 12295

Table 1: Multiple subword sequences encoding the same sentence "Hello World"

BPE encodes a sentence into a unique subword sequence. However, a sentence can be represented in multiple subword sequences even with the same vocabulary. Table 1 illustrates an example. While these sequences encode the same input "Hello World", NMT handles them as completely different inputs. This observation becomes more apparent when converting subword sequences into id sequences (right column in Table 1). These variants can be viewed as a spurious ambiguity, which might not always be resolved in decoding process. At training time of NMT, multiple segmentation candidates will make the model robust to noise and segmentation errors, as they can indirectly help the model to learn the compositionality of words, e.g., "books" can be decomposed into "book" + "s".

In this study, we propose a new regularization method for open-vocabulary NMT, called **subword regularization**, which employs multiple subword segmentations to make the NMT model accurate and robust. Subword regularization consists of the following two sub-contributions:

- We propose a simple NMT training algorithm to integrate multiple segmentation candidates. Our approach is implemented as an on-the-fly data sampling, which is not specific to NMT architecture. Subword regularization can be applied to any NMT system without changing the model structure.

- We also propose a new subword segmentation algorithm based on a language model, which provides multiple segmentations with probabilities. The language model allows to emulate the noise generated during the segmentation of actual data.

Empirical experiments using multiple corpora with different sizes and languages show that subword regularization achieves significant improvements over the method using a single subword sequence. In addition, through experiments with out-of-domain corpora, we show that subword regularization improves the robustness of the NMT model.

2 Neural Machine Translation with multiple subword segmentations

2.1 NMT training with on-the-fly subword sampling

Given a source sentence X and a target sentence Y, let $\mathbf{x} = (x_1, \ldots, x_M)$ and $\mathbf{y} = (y_1, \ldots, y_N)$ be the corresponding subword sequences segmented with an underlying subword segmenter, e.g., BPE. NMT models the translation probability $P(Y|X) = P(\mathbf{y}|\mathbf{x})$ as a target language sequence model that generates target subword y_n conditioning on the target history $y_{<n}$ and source input sequence $\mathbf{x}$:

$$P(\mathbf{y}|\mathbf{x};\theta) = \prod_{n=1}^{N} P(y_n|\mathbf{x}, y_{<n};\theta), \qquad (1)$$

where θ is a set of model parameters. A common choice to predict the subword y_n is to use a recurrent neural network (RNN) architecture. However, note that subword regularization is not specific to this architecture and can be applicable to other NMT architectures without RNN, e.g., (Vaswani et al., 2017; Gehring et al., 2017).

NMT is trained using the standard maximum likelihood estimation, i.e., maximizing the log-likelihood $\mathcal{L}(\theta)$ of a given parallel corpus $D =$

$$\begin{aligned} \{\langle X^{(s)}, Y^{(s)}\rangle\}_{s=1}^{|D|} &= \{\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)}\rangle\}_{s=1}^{|D|}, \\ \theta_{MLE} &= \arg\max_{\theta} \mathcal{L}(\theta) \\ where,\ \mathcal{L}(\theta) &= \sum_{s=1}^{|D|} \log P(\mathbf{y}^{(s)}|\mathbf{x}^{(s)};\theta). \quad (2) \end{aligned}$$

We here assume that the source and target sentences X and Y can be segmented into multiple subword sequences with the segmentation probabilities $P(\mathbf{x}|X)$ and $P(\mathbf{y}|Y)$ respectively. In subword regularization, we optimize the parameter set θ with the marginalized likelihood as (3).

$$\mathcal{L}_{marginal}(\theta) = \sum_{s=1}^{|D|} \mathbb{E}_{\substack{\mathbf{x}\sim P(\mathbf{x}|X^{(s)}) \\ \mathbf{y}\sim P(\mathbf{y}|Y^{(s)})}} [\log P(\mathbf{y}|\mathbf{x};\theta)] \quad (3)$$

Exact optimization of (3) is not feasible as the number of possible segmentations increases exponentially with respect to the sentence length. We approximate (3) with finite k sequences sampled from $P(\mathbf{x}|X)$ and $P(\mathbf{y}|Y)$ respectively.

$$\begin{gathered} \mathcal{L}_{marginal}(\theta) \cong \frac{1}{k^2} \sum_{s=1}^{|D|} \sum_{i=1}^{k} \sum_{j=1}^{k} \log P(\mathbf{y}_j|\mathbf{x}_i;\theta) \\ \mathbf{x}_i \sim P(\mathbf{x}|X^{(s)}), \quad \mathbf{y}_j \sim P(\mathbf{y}|Y^{(s)}). \end{gathered} \quad (4)$$

For the sake of simplicity, we use $k = 1$. Training of NMT usually uses an online training for efficiency, in which the parameter θ is iteratively optimized with respect to the smaller subset of D (mini-batch). When we have a sufficient number of iterations, subword sampling is executed via the data sampling of online training, which yields a good approximation of (3) even if $k = 1$. It should be noted, however, that the subword sequence is sampled on-the-fly for each parameter update.

2.2 Decoding

In the decoding of NMT, we only have a raw source sentence X. A straightforward approach for decoding is to translate from the best segmentation $\mathbf{x}^*$ that maximizes the probability $P(\mathbf{x}|X)$, i.e., $\mathbf{x}^* = argmax_{\mathbf{x}} P(\mathbf{x}|X)$. Additionally, we can use the n-best segmentations of $P(\mathbf{x}|X)$ to incorporate multiple segmentation candidates. More specifically, given n-best segmentations $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we choose the best translation $\mathbf{y}^*$ that maximizes the following score.

$$score(\mathbf{x}, \mathbf{y}) = \log P(\mathbf{y}|\mathbf{x})/|\mathbf{y}|^{\lambda}, \qquad (5)$$

where $|\mathbf{y}|$ is the number of subwords in $\mathbf{y}$ and $\lambda \in \mathbb{R}^+$ is the parameter to penalize shorter sentences. λ is optimized with the development data.

In this paper, we call these two algorithms **one-best decoding** and **n-best decoding** respectively.

3 Subword segmentations with language model

3.1 Byte-Pair-Encoding (BPE)

Byte-Pair-Encoding (BPE) (Sennrich et al., 2016; Schuster and Nakajima, 2012) is a subword segmentation algorithm widely used in many NMT systems[1]. BPE first splits the whole sentence into individual characters. The most frequent[2] adjacent pairs of characters are then consecutively merged until reaching a desired vocabulary size. Subword segmentation is performed by applying the same merge operations to the test sentence.

An advantage of BPE segmentation is that it can effectively balance the vocabulary size and the step size (the number of tokens required to encode the sentence). BPE trains the merged operations only with a frequency of characters. Frequent substrings will be joined early, resulting in common words remaining as one unique symbol. Words consisting of rare character combinations will be split into smaller units, e.g., substrings or characters. Therefore, only with a small fixed size of vocabulary (usually 16k to 32k), the number of required symbols to encode a sentence will not significantly increase, which is an important feature for an efficient decoding.

One downside is, however, that BPE is based on a greedy and deterministic symbol replacement, which can not provide multiple segmentations with probabilities. It is not trivial to apply BPE to the subword regularization that depends on segmentation probabilities $P(\mathbf{x}|X)$.

3.2 Unigram language model

In this paper, we propose a new subword segmentation algorithm based on a unigram language model, which is capable of outputing multiple subword segmentations with probabilities. The unigram language model makes an assumption that each subword occurs independently, and consequently, the probability of a subword sequence $\mathbf{x} = (x_1, \ldots, x_M)$ is formulated as the product of the subword occurrence probabilities $p(x_i)$[3]:

$$P(\mathbf{x}) = \prod_{i=1}^{M} p(x_i), \tag{6}$$

$$\forall i \; x_i \in \mathcal{V}, \; \sum_{x \in \mathcal{V}} p(x) = 1,$$

where $\mathcal{V}$ is a pre-determined vocabulary. The most probable segmentation $\mathbf{x}^*$ for the input sentence X is then given by

$$\mathbf{x}^* = \operatorname*{arg\,max}_{\mathbf{x} \in \mathcal{S}(X)} P(\mathbf{x}), \tag{7}$$

where $\mathcal{S}(X)$ is a set of segmentation candidates built from the input sentence X. $\mathbf{x}^*$ is obtained with the Viterbi algorithm (Viterbi, 1967).

If the vocabulary $\mathcal{V}$ is given, subword occurrence probabilities $p(x_i)$ are estimated via the EM algorithm that maximizes the following marginal likelihood $\mathcal{L}$ assuming that $p(x_i)$ are hidden variables.

$$\mathcal{L} = \sum_{s=1}^{|D|} \log(P(X^{(s)})) = \sum_{s=1}^{|D|} \log\Big(\sum_{\mathbf{x} \in \mathcal{S}(X^{(s)})} P(\mathbf{x}) \Big)$$

In the real setting, however, the vocabulary set $\mathcal{V}$ is also unknown. Because the joint optimization of vocabulary set and their occurrence probabilities is intractable, we here seek to find them with the following iterative algorithm.

1. Heuristically make a reasonably big seed vocabulary from the training corpus.
2. Repeat the following steps until $|\mathcal{V}|$ reaches a desired vocabulary size.
   (a) Fixing the set of vocabulary, optimize $p(x)$ with the EM algorithm.
   (b) Compute the $loss_i$ for each subword x_i, where $loss_i$ represents how likely the likelihood $\mathcal{L}$ is reduced when the subword x_i is removed from the current vocabulary.
   (c) Sort the symbols by $loss_i$ and keep top η % of subwords (η is 80, for example). Note that we always keep the subwords consisting of a single character to avoid out-of-vocabulary.

[1]Strictly speaking, wordpiece model (Schuster and Nakajima, 2012) is different from BPE. We consider wordpiece as a variant of BPE, as it also uses an incremental vocabulary generation with a different loss function.

[2]Wordpiece model uses a likelihood instead of frequency.

[3]Target sequence $\mathbf{y} = (y_1, \ldots, y_N)$ can also be modeled similarly.

There are several ways to prepare the seed vocabulary. The natural choice is to use the union of all characters and the most frequent substrings in the corpus[4]. Frequent substrings can be enumerated in $O(T)$ time and $O(20T)$ space with the Enhanced Suffix Array algorithm (Nong et al., 2009), where T is the size of the corpus. Similar to (Sennrich et al., 2016), we do not consider subwords that cross word boundaries.

As the final vocabulary $\mathcal{V}$ contains all individual characters in the corpus, character-based segmentation is also included in the set of segmentation candidates $\mathcal{S}(X)$. In other words, subword segmentation with the unigram language model can be seen as a probabilsitic mixture of characters, subwords and word segmentations.

3.3 Subword sampling

Subword regularization samples one subword segmentation from the distribution $P(\mathbf{x}|X)$ for each parameter update. A straightforward approach for an approximate sampling is to use the l-best segmentations. More specifically, we first obtain l-best segmentations according to the probability $P(\mathbf{x}|X)$. l-best search is performed in linear time with the Forward-DP Backward-A* algorithm (Nagata, 1994). One segmentation $\mathbf{x}_i$ is then sampled from the multinomial distribution $P(\mathbf{x}_i|X) \cong P(\mathbf{x}_i)^{\alpha} / \sum_{i=1}^{l} P(\mathbf{x}_i)^{\alpha}$, where $\alpha \in \mathbb{R}^{+}$ is the hyperparameter to control the smoothness of the distribution. A smaller α leads to sample $\mathbf{x}_i$ from a more uniform distribution. A larger α tends to select the Viterbi segmentation.

Setting $l \rightarrow \infty$, in theory, allows to take all possible segmentations into account. However, it is not feasible to increase l explicitly as the number of candidates increases exponentially with respect to the sentence length. In order to exactly sample from all possible segmentations, we use the Forward-Filtering and Backward-Sampling algorithm (FFBS) (Scott, 2002), a variant of the dynamic programming originally introduced by Bayesian hidden Markov model training. In FFBS, all segmentation candidates are represented in a compact lattice structure, where each node denotes a subword. In the first pass, FFBS computes a set of forward probabilities for all subwords in the lattice, which provide the probability of ending up in any particular subword w. In the second pass, traversing the nodes in the lattice from the end of the sentence to the beginning of the sentence, subwords are recursively sampled for each branch according to the forward probabilities.

[4]It is also possible to run BPE with a sufficient number of merge operations.

3.4 BPE vs. Unigram language model

BPE was originally introduced in the data compression literature (Gage, 1994). BPE is a variant of dictionary (substitution) encoder that incrementally finds a set of symbols such that the total number of symbols for encoding the text is minimized. On the other hand, the unigram language model is reformulated as an entropy encoder that minimizes the total code length for the text. According to Shannon's coding theorem, the optimal code length for a symbol s is $-\log p_s$, where p_s is the occurrence probability of s. This is essentially the same as the segmentation strategy of the unigram language model described as (7).

BPE and the unigram language model share the same idea that they encode a text using fewer bits with a certain data compression principle (dictionary vs. entropy). Therefore, we expect to see the same benefit as BPE with the unigram language model. However, the unigram language model is more flexible as it is based on a probabilistic language model and can output multiple segmentations with their probabilities, which is an essential requirement for subword regularization.

4 Related Work

Regularization by noise is a well studied technique in deep neural networks. A well-known example is dropout (Srivastava et al., 2014), which randomly turns off a subset of hidden units during training. Dropout is analyzed as an ensemble training, where many different models are trained on different subsets of the data. Subword regularization trains the model on different data inputs randomly sampled from the original input sentences, and thus is regarded as a variant of ensemble training.

The idea of noise injection has previously been used in the context of Denoising Auto-Encoders (DAEs) (Vincent et al., 2008), where noise is added to the inputs and the model is trained to reconstruct the original inputs. There are a couple of studies that employ DAEs in natural language processing.

(Lample et al., 2017; Artetxe et al., 2017) independently propose DAEs in the context of sequence-to-sequence learning, where they randomly alter the word order of the input sentence and the model is trained to reconstruct the original sentence. Their technique is applied to an unsupervised machine translation to make the encoder truly learn the compositionality of input sentences.

Word dropout (Iyyer et al., 2015) is a simple approach for a bag-of-words representation, in which the embedding of a certain word sequence is simply calculated by averaging the word embeddings. Word dropout randomly drops words from the bag before averaging word embeddings, and consequently can see $2^{|X|}$ different token sequences for each input X.

(Belinkov and Bisk, 2017) explore the training of character-based NMT with a synthetic noise that randomly changes the order of characters in a word. (Xie et al., 2017) also proposes a robust RNN language model that interpolates random unigram language model.

The basic idea and motivation behind subword regularization are similar to those of previous work. In order to increase the robustness, they inject noise to input sentences by randomly changing the internal representation of sentences. However, these previous approaches often depend on heuristics to generate synthetic noises, which do not always reflect the real noises on training and inference. In addition, these approaches can only be applied to source sentences (encoder), as they irreversibly rewrite the surface of sentences. Subword regularization, on the other hand, generates synthetic subword sequences with an underlying language model to better emulate the noises and segmentation errors. As subword regularization is based on an invertible conversion, we can safely apply it both to source and target sentences.

Subword regularization can also be viewed as a data augmentation. In subword regularization, an input sentence is converted into multiple invariant sequences, which is similar to the data augmentation for image classification tasks, for example, random flipping, distorting, or cropping.

There are several studies focusing on segmentation ambiguities in language modeling. Latent Sequence Decompositions (LSDs) (Chan et al., 2016) learns the mapping from the input and the output by marginalizing over all possible segmentations. LSDs and subword regularization do not assume a predetermined segmentation for a sentence, and take multiple segmentations by a similar marginalization technique. The difference is that subword regularization injects the multiple segmentations with a separate language model through an on-the-fly subword sampling. This approach makes the model simple and independent from NMT architectures.

Lattice-to-sequence models (Su et al., 2017; Sperber et al., 2017) are natural extension of sequence-to-sequence models, which represent inputs uncertainty through lattices. Lattice is encoded with a variant of TreeLSTM (Tai et al., 2015), which requires changing the model architecture. In addition, while subword regularization is applied both to source and target sentences, lattice-to-sequence models do not handle target side ambiguities.

A mixed word/character model (Wu et al., 2016) addresses the out-of-vocabulary problem with a fixed vocabulary. In this model, out-of-vocabulary words are not collapsed into a single UNK symbol, but converted into the sequence of characters with special prefixes representing the positions in the word. Similar to BPE, this model also encodes a sentence into a unique fixed sequence, thus multiple segmentations are not taken into account.

5 Experiments

5.1 Setting

We conducted experiments using multiple corpora with different sizes and languages. Table 2 summarizes the evaluation data we used [5] [6] [7] [8] [9] [10]. IWSLT15/17 and KFTT are relatively small corpora, which include a wider spectrum of languages with different linguistic properties. They can evaluate the language-agnostic property of subword regularization. ASPEC and WMT14 (en↔de) are medium-sized corpora. WMT14 (en↔cs) is a rather big corpus consisting of more than 10M parallel sentences.

We used GNMT (Wu et al., 2016) as the implementation of the NMT system for all experiments. We generally followed the settings and training procedure described in (Wu et al., 2016), however, we changed the settings according to the

[5]IWSLT15: http://workshop2015.iwslt.org/

[6]IWSLT17: http://workshop2017.iwslt.org/

[7]KFTT: http://www.phontron.com/kftt/

[8]ASPEC: http://lotus.kuee.kyoto-u.ac.jp/ASPEC/

[9]WMT14: http://statmt.org/wmt14/

[10]WMT14(en↔de) uses the same setting as (Wu et al., 2016).

corpus size. Table 2 shows the hyperparameters we used in each experiment. As common settings, we set the dropout probability to be 0.2. For parameter estimation, we used a combination of Adam (Kingma and Adam, 2014) and SGD algorithms. Both length normalization and converge penalty parameters are set to 0.2 (see section 7 in (Wu et al., 2016)). We set the decoding beam size to 4.

The data was preprocessed with Moses tokenizer before training subword models. It should be noted, however, that Chinese and Japanese have no explicit word boundaries and Moses tokenizer does not segment sentences into words, and hence subword segmentations are trained almost from unsegmented raw sentences in these languages.

We used the case sensitive BLEU score (Papineni et al., 2002) as an evaluation metric. As the output sentences are not segmented in Chinese and Japanese, we segment them with characters and KyTea[11] for Chinese and Japanese respectively before calculating BLEU scores.

BPE segmentation is used as a baseline system. We evaluate three test systems with different sampling strategies: (1) Unigram language model-based subword segmentation without subword regularization ($l = 1$), (2) with subword regularization ($l = 64,\ \alpha = 0.1$) and (3) ($l = \infty,\ \alpha = 0.2/0.5$) 0.2: IWSLT, 0.5: others. These sampling parameters were determined with preliminary experiments. $l = 1$ is aimed at a pure comparison between BPE and the unigram language model. In addition, we compare one-best decoding and n-best decoding (See section 2.2). Because BPE is not able to provide multiple segmentations, we only evaluate one-best decoding for BPE. Consequently, we compare 7 systems ($1 + 3 \times 2$) for each language pair.

5.2 Main Results

Table 3 shows the translation experiment results.

First, as can be seen in the table, BPE and unigram language model without subword regularization ($l = 1$) show almost comparable BLEU scores. This is not surprising, given that both BPE and the unigram language model are based on data compression algorithms.

We can see that subword regularization ($l > 1$) boosted BLEU scores quite impressively (+1 to 2 points) in all language pairs except for WMT14 (en→cs) dataset. The gains are larger especially in lower resource settings (IWSLT and KFTT). It can be considered that the positive effects of data augmentation with subword regularization worked better in lower resource settings, which is a common property of other regularization techniques.

As for the sampling algorithm, ($l = \infty\ \alpha = 0.2/0.5$) slightly outperforms ($l = 64,\ \alpha = 0.1$) on IWSLT corpus, but they show almost comparable results on larger data set. Detailed analysis is described in Section 5.5.

On top of the gains with subword regularization, n-best decoding yields further improvements in many language pairs. However, we should note that the subword regularization is mandatory for n-best decoding and the BLEU score is degraded in some language pairs without subword regularization ($l = 1$). This result indicates that the decoder is more confused for multiple segmentations when they are not explored at training time.

5.3 Results with out-of-domain corpus

To see the effect of subword regularization on a more open-domain setting, we evaluate the systems with out-of-domain in-house data consisting of multiple genres: Web, patents and query logs. Note that we did not conduct the comparison with KFTT and ASPEC corpora, as we found that the domains of these corpora are too specific[12], and preliminary evaluations showed extremely poor BLEU scores (less than 5) on out-of-domain corpora.

Table 4 shows the results. Compared to the gains obtained with the standard in-domain evaluations in Table 3, subword regularization achieves significantly larger improvements (+2 points) in every domain of corpus. An interesting observation is that we have the same level of improvements even on large training data sets (WMT14), which showed marginal or small gains with the in-domain data. This result strongly supports our claim that subword regularization is more useful for open-domain settings.

5.4 Comparison with other segmentation algorithms

Table 5 shows the comparison on different segmentation algorithms: word, character, mixed word/character (Wu et al., 2016), BPE

[11] http://www.phontron.com/kytea

[12] KFTT focuses on Wikipedia articles related to Kyoto, and ASPEC is a corpus of scientific paper domain. Therefore, it is hard to translate out-of-domain texts.

Corpus	Language pair	Size of sentences: train	dev	test	Parameters: #vocab (Enc/Dec shared)	#dim of LSTM, embedding	#layers of LSTM (Enc+Dec)
IWSLT15	en ↔ vi	133k	1553	1268	16k	512	2+2
	en ↔ zh	209k	887	1261	16k	512	2+2
IWSLT17	en ↔ fr	232k	890	1210	16k	512	2+2
	en ↔ ar	231k	888	1205	16k	512	2+2
KFTT	en ↔ ja	440k	1166	1160	8k	512	6+6
ASPEC	en ↔ ja	2M	1790	1812	16k	512	6+6
WMT14	en ↔ de	4.5M	3000	3003	32k	1024	8+8
	en ↔ cs	15M	3000	3003	32k	1024	8+8

Table 2: Details of evaluation data set

Corpus	Language pair	baseline (BPE)	Proposed (one-best decoding) $l = 1$	$l = 64$ $\alpha = 0.1$	$l = \infty$ α=0.2/0.5	Proposed (n-best decoding, n=64) $l = 1$	$l = 64$ $\alpha = 0.1$	$l = \infty$ α=0.2/0.5
IWSLT15	en → vi	25.61	25.49	27.68*	27.71*	25.33	28.18*	28.48*
	vi → en	22.48	22.32	24.73*	26.15*	22.04	24.66*	26.31*
	en → zh	16.70	16.90	19.36*	20.33*	16.73	20.14*	21.30*
	zh → en	15.76	15.88	17.79*	16.95*	16.23	17.75*	17.29*
IWSLT17	en → fr	35.53	35.39	36.70*	36.36*	35.16	37.60*	37.01*
	fr → en	33.81	33.74	35.57*	35.54*	33.69	36.07*	36.06*
	en → ar	13.01	13.04	14.92*	15.55*	12.29	14.90*	15.36*
	ar → en	25.98	27.09*	28.47*	29.22*	27.08*	29.05*	29.29*
KFTT	en → ja	27.85	28.92*	30.37*	30.01*	28.55*	31.46*	31.43*
	ja → en	21.37	21.46	22.33*	22.04*	21.37	22.47*	22.64*
ASPEC	en → ja	40.62	40.66	41.24*	41.23*	40.86	41.55*	41.87*
	ja → en	26.51	26.76	27.08*	27.14*	27.49*	27.75*	27.89*
WMT14	en → de	24.53	24.50	25.04*	24.74	22.73	25.00*	24.57
	de → en	28.01	28.65*	28.83*	29.39*	28.24	29.13*	29.97*
	en → cs	25.25	25.54	25.41	25.26	24.88	25.49	25.38
	cs → en	28.78	28.84	29.64*	29.41*	25.77	29.23*	29.15*

Table 3: Main Results (BLEU(%)) (l: sampling size in SR, α: smoothing parameter). * indicates statistically significant difference ($p < 0.05$) from baselines with bootstrap resampling (Koehn, 2004). The same mark is used in Table 4 and 6.

(Sennrich et al., 2016) and our unigram model with or without subword regularization. The BLEU scores of word, character and mixed word/character models are cited from (Wu et al., 2016). As German is a morphologically rich language and needs a huge vocabulary for word models, subword-based algorithms perform a gain of more than 1 BLEU point than word model. Among subword-based algorithms, the unigram language model with subword regularization achieved the best BLEU score (25.04), which demonstrates the effectiveness of multiple subword segmentations.

5.5 Impact of sampling hyperparameters

Subword regularization has two hyperparameters: l: size of sampling candidates, α: smoothing constant. Figure 1 shows the BLEU scores of various hyperparameters on IWSLT15 (en → vi) dataset.

First, we can find that the peaks of BLEU scores against smoothing parameter α are different depending on the sampling size l. This is expected, because $l = \infty$ has larger search space than $l = 64$, and needs to set α larger to sample sequences close to the Viterbi sequence $\mathbf{x}^*$.

Another interesting observation is that $\alpha = 0.0$ leads to performance drops especially on $l = \infty$. When $\alpha = 0.0$, the segmentation probability $P(\mathbf{x}|X)$ is virtually ignored and one segmentation is uniformly sampled. This result suggests that biased sampling with a language model is helpful to emulate the real noise in the actual translation.

In general, larger l allows a more aggressive regularization and is more effective for low resource settings such as IWSLT. However, the estimation of α is more sensitive and performance becomes even worse than baseline when α is extremely small. To weaken the effect of regularization and avoid selecting invalid parameters, it might be more reasonable to use $l = 64$ for high resource languages.

Domain (size)	Corpus	Language pair	Baseline (BPE)	Proposed (SR)
Web (5k)	IWSLT15	en → vi	13.86	17.36*
		vi → en	7.83	11.69*
		en → zh	9.71	13.85*
		zh → en	5.93	8.13*
	IWSLT17	en → fr	16.09	20.04*
		fr → en	14.77	19.99*
	WMT14	en → de	22.71	26.02*
		de → en	26.42	29.63*
		en → cs	19.53	21.41*
		cs → en	25.94	27.86*
Patent (2k)	WMT14	en → de	15.63	25.76*
		de → en	22.74	32.66*
		en → cs	16.70	19.38*
		cs → en	23.20	25.30*
Query (2k)	IWSLT15	en → zh	9.30	12.47*
		zh → en	14.94	19.99*
	IWSLT17	en → fr	10.79	10.99
		fr → en	19.01	23.96*
	WMT14	en → de	25.93	29.82*
		de → en	26.24	30.90*

Table 4: Results with out-of-domain corpus
($l = \infty$, $\alpha = 0.2$: IWSLT15/17, $l = 64$, $\alpha = 0.1$: others, one-best decding)

Model	BLEU
Word	23.12
Character (512 nodes)	22.62
Mixed Word/Character	24.17
BPE	24.53
Unigram w/o SR ($l = 1$)	24.50
Unigram w/ SR ($l = 64$, $\alpha = 0.1$)	25.04

Table 5: Comparison of different segmentation algorithms (WMT14 en→de)

Although we can see in general that the optimal hyperparameters are roughly predicted with the held-out estimation, it is still an open question how to choose the optimal size l in subword sampling.

5.6 Results with single side regularization

Table 6 summarizes the BLEU scores with subword regularization either on source or target sentence to figure out which components (encoder or decoder) are more affected. As expected, we can see that the BLEU scores with single side regularization are worse than full regularization. However, it should be noted that single side regularization still has positive effects. This result implies that subword regularization is not only helpful for encoder-decoder architectures, but applicable to other NLP tasks that only use an either encoder or decoder, including text classification (Iyyer et al., 2015) and image caption generation (Vinyals et al., 2015).

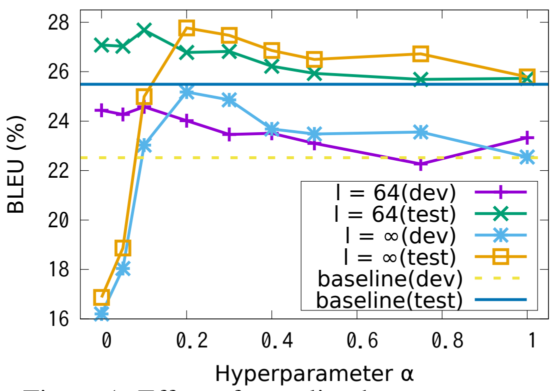


Figure 1: Effect of sampling hyperparameters

Regularization type	en→vi	vi→en	en→ar	ar→en
No reg. (baseline)	25.49	22.32	13.04	27.09
Source only	26.00	23.09*	13.46	28.16*
Target only	26.10	23.62*	14.34*	27.89*
Source and target	27.68*	24.73*	14.92*	28.47*

Table 6: Comparison on different regularization strategies (IWSLT15/17, $l = 64$, $\alpha = 0.1$)

6 Conclusions

In this paper, we presented a simple regularization method, **subword regularization**[13], for NMT, with no change to the network architecture. The central idea is to virtually augment training data with on-the-fly subword sampling, which helps to improve the accuracy as well as robustness of NMT models. In addition, for better subword sampling, we propose a new subword segmentation algorithm based on the unigram language model. Experiments on multiple corpora with different sizes and languages show that subword regularization leads to significant improvements especially on low resource and open-domain settings.

Promising avenues for future work are to apply subword regularization to other NLP tasks based on encoder-decoder architectures, e.g., dialog generation (Vinyals and Le, 2015) and automatic summarization (Rush et al., 2015). Compared to machine translation, these tasks do not have enough training data, and thus there could be a large room for improvement with subword regularization. Additionally, we would like to explore the application of subword regularization for machine learning, including Denoising Auto Encoder (Vincent et al., 2008) and Adversarial Training (Goodfellow et al., 2015).

[13]Implementation is available at `https://github.com/google/sentencepiece`

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXive preprint arXiv:1710.11041* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXive preprint arXiv:1711.02173* .

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. 2016. Latent sequence decompositions. *arXiv preprint arXiv:1610.03035* .

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proc. of EMNLP*. pages 2088–2093.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. *Proc. of Workshop on Neural Machine Translation* .

Philip Gage. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* .

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proc. of ICLR*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Diederik P Kingma and Jimmy Ba Adam. 2014. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXive preprint arXiv:1711.00043* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc of EMNLP*.

Masaaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm. In *Proc. of COLING*.

Toshiaki Nakazawa, Shohei Higashiyama, Chenchen Ding, Hideya Mino, Isao Goto, Hideto Kazawa, Yusuke Oda, Graham Neubig, and Sadao Kurohashi. 2017. Overview of the 4th workshop on asian translation. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*. pages 1–54.

Ge Nong, Sen Zhang, and Wai Hong Chan. 2009. Linear suffix array construction by almost pure induced-sorting. In *Proc. of DCC*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proc. of EMNLP*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *Proc. of ICASSP*.

Steven L Scott. 2002. Bayesian methods for hidden markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proc. of EMNLP*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1).

Jinsong Su, Zhixing Tan, De yi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI*. pages 3302–3308.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proc. of ACL* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXive preprint arXiv:1706.03762* .

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory* 13(2):260–269.

Yonghui Wu, Mike Schuster, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *Proc. of ICLR*.

The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation

Mia Xu Chen * **Orhan Firat** * **Ankur Bapna** *

Melvin Johnson **Wolfgang Macherey** **George Foster** **Llion Jones** **Niki Parmar**

Noam Shazeer **Ashish Vaswani** **Jakob Uszkoreit** **Lukasz Kaiser**

Mike Schuster **Zhifeng Chen** **Yonghui Wu** **Macduff Hughes**

miachen,orhanf,ankurbpn,yonghui@google.com

Google AI

Abstract

The past year has witnessed rapid advances in sequence-to-sequence (seq2seq) modeling for Machine Translation (MT). The classic RNN-based approaches to MT were first out-performed by the convolutional seq2seq model, which was then out-performed by the more recent Transformer model. Each of these new approaches consists of a fundamental architecture accompanied by a set of modeling and training techniques that are in principle applicable to other seq2seq architectures. In this paper, we tease apart the new architectures and their accompanying techniques in two ways. First, we identify several key modeling and training techniques, and apply them to the RNN architecture, yielding a new RNMT+ model that outperforms all of the three fundamental architectures on the benchmark WMT'14 English→French and English→German tasks. Second, we analyze the properties of each fundamental seq2seq architecture and devise new hybrid architectures intended to combine their strengths. Our hybrid models obtain further improvements, outperforming the RNMT+ model on both benchmark datasets.

1 Introduction

In recent years, the emergence of seq2seq models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) has revolutionized the field of MT by replacing traditional phrase-based approaches with neural machine translation (NMT) systems based on the encoder-decoder paradigm. In the first architectures that surpassed the quality of phrase-based MT, both the encoder and decoder were implemented as Recurrent Neural Networks (RNNs), interacting via a soft-attention mechanism (Bahdanau et al., 2015). The RNN-based NMT approach, or RNMT, was quickly established as the de-facto standard for NMT, and gained rapid adoption into large-scale systems in industry, e.g. Baidu (Zhou et al., 2016), Google (Wu et al., 2016), and Systran (Crego et al., 2016).

Following RNMT, convolutional neural network based approaches (LeCun and Bengio, 1998) to NMT have recently drawn research attention due to their ability to fully parallelize training to take advantage of modern fast computing devices. such as GPUs and Tensor Processing Units (TPUs) (Jouppi et al., 2017). Well known examples are ByteNet (Kalchbrenner et al., 2016) and ConvS2S (Gehring et al., 2017). The ConvS2S model was shown to outperform the original RNMT architecture in terms of quality, while also providing greater training speed.

Most recently, the Transformer model (Vaswani et al., 2017), which is based solely on a self-attention mechanism (Parikh et al., 2016) and feed-forward connections, has further advanced the field of NMT, both in terms of translation quality and speed of convergence.

In many instances, new architectures are accompanied by a novel set of techniques for performing training and inference that have been carefully optimized to work in concert. This 'bag of tricks' can be crucial to the performance of a proposed architecture, yet it is typically under-documented and left for the enterprising researcher to discover in publicly released code (if any) or through anecdotal evidence. This is not simply a problem for reproducibility; it obscures the central scientific question of how much of the observed gains come from the new architecture

* Equal contribution.

and how much can be attributed to the associated training and inference techniques. In some cases, these new techniques may be broadly applicable to other architectures and thus constitute a major, though implicit, contribution of an architecture paper. Clearly, they need to be considered in order to ensure a fair comparison across different model architectures.

In this paper, we therefore take a step back and look at which techniques and methods contribute significantly to the success of recent architectures, namely ConvS2S and Transformer, and explore applying these methods to other architectures, including RNMT models. In doing so, we come up with an enhanced version of RNMT, referred to as RNMT+, that significantly outperforms all individual architectures in our setup. We further introduce new architectures built with different components borrowed from RNMT+, ConvS2S and Transformer. In order to ensure a fair setting for comparison, all architectures were implemented in the same framework, use the same pre-processed data and apply no further post-processing as this may confound bare model performance.

Our contributions are three-fold:

1. In ablation studies, we quantify the effect of several modeling improvements (including multi-head attention and layer normalization) as well as optimization techniques (such as synchronous replica training and label-smoothing), which are used in recent architectures. We demonstrate that these techniques are applicable across different model architectures.

2. Combining these improvements with the RNMT model, we propose the new RNMT+ model, which significantly outperforms all fundamental architectures on the widely-used WMT'14 En$\rightarrow$Fr and En$\rightarrow$De benchmark datasets. We provide a detailed model analysis and comparison of RNMT+, ConvS2S and Transformer in terms of model quality, model size, and training and inference speed.

3. Inspired by our understanding of the relative strengths and weaknesses of individual model architectures, we propose new model architectures that combine components from the RNMT+ and the Transformer model, and achieve better results than both individual architectures.

We quickly note two prior works that provided empirical solutions to the difficulty of training NMT architectures (specifically RNMT). In (Britz et al., 2017) the authors systematically explore which elements of NMT architectures have a significant impact on translation quality. In (Denkowski and Neubig, 2017) the authors recommend three specific techniques for strengthening NMT systems and empirically demonstrated how incorporating those techniques improves the reliability of the experimental results.

2 Background

In this section, we briefly discuss the commmonly used NMT architectures.

2.1 RNN-based NMT Models - RNMT

RNMT models are composed of an encoder RNN and a decoder RNN, coupled with an attention network. The encoder summarizes the input sequence into a set of vectors while the decoder conditions on the encoded input sequence through an attention mechanism, and generates the output sequence one token at a time.

The most successful RNMT models consist of stacked RNN encoders with one or more bidirectional RNNs (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005), and stacked decoders with unidirectional RNNs. Both encoder and decoder RNNs consist of either LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) or GRU units (Cho et al., 2014), and make extensive use of residual (He et al., 2015) or highway (Srivastava et al., 2015) connections.

In Google-NMT (GNMT) (Wu et al., 2016), the best performing RNMT model on the datasets we consider, the encoder network consists of one bi-directional LSTM layer, followed by 7 uni-directional LSTM layers. The decoder is equipped with a single attention network and 8 uni-directional LSTM layers. Both the encoder and the decoder use residual skip connections between consecutive layers.

In this paper, we adopt GNMT as the starting point for our proposed RNMT+ architecture.

2.2 Convolutional NMT Models - ConvS2S

In the most successful convolutional sequence-to-sequence model (Gehring et al., 2017), both the encoder and decoder are constructed by stacking multiple convolutional layers, where each layer contains 1-dimensional convolutions followed by a gated linear units (GLU) (Dauphin et al., 2016). Each decoder layer computes a separate dot-product attention by using the current decoder layer output and the final encoder layer outputs. Positional embeddings are used to provide explicit positional information to the model. Following the practice in (Gehring et al., 2017), we scale the gradients of the encoder layers to stabilize training. We also use residual connections across each convolutional layer and apply weight normalization (Salimans and Kingma, 2016) to speed up convergence. We follow the public ConvS2S codebase[1] in our experiments.

2.3 Conditional Transformation-based NMT Models - Transformer

The Transformer model (Vaswani et al., 2017) is motivated by two major design choices that aim to address deficiencies in the former two model families: (1) Unlike RNMT, but similar to the ConvS2S, the Transformer model avoids any sequential dependencies in both the encoder and decoder networks to maximally parallelize training. (2) To address the limited context problem (limited receptive field) present in ConvS2S, the Transformer model makes pervasive use of self-attention networks (Parikh et al., 2016) so that each position in the current layer has access to information from all other positions in the previous layer.

The Transformer model still follows the encoder-decoder paradigm. Encoder transformer layers are built with two sub-modules: (1) a self-attention network and (2) a feed-forward network. Decoder transformer layers have an additional cross-attention layer sandwiched between the self-attention and feed-forward layers to attend to the encoder outputs.

There are two details which we found very important to the model's performance: (1) Each sub-layer in the transformer (i.e. self-attention, cross-attention, and the feed-forward sub-layer) follows a strict computation sequence: *normalize* $\rightarrow$ *transform* $\rightarrow$ *dropout* $\rightarrow$ *residual-add*. (2) In addition to per-layer normalization, the final encoder output is again normalized to prevent a blow up after consecutive residual additions.

In this paper, we follow the latest version of the Transformer model in the Tensor2Tensor[2] codebase.

2.4 A Theory-Based Characterization of NMT Architectures

From a theoretical point of view, RNNs belong to the most expressive members of the neural network family (Siegelmann and Sontag, 1995)[3]. Possessing an infinite Markovian structure (and thus an infinite receptive fields) equips them to model sequential data (Elman, 1990), especially natural language (Grefenstette et al., 2015) effectively. In practice, RNNs are notoriously hard to train (Hochreiter, 1991; Bengio et al., 1994; Hochreiter et al., 2001), confirming the well known dilemma of trainability versus expressivity.

Convolutional layers are adept at capturing local context and local correlations by design. A fixed and narrow receptive field for each convolutional layer limits their capacity when the architecture is shallow. In practice, this weakness is mitigated by stacking more convolutional layers (e.g. 15 layers as in the ConvS2S model), which makes the model harder to train and demands meticulous initialization schemes and carefully designed regularization techniques.

The transformer network is capable of approximating arbitrary squashing functions (Hornik et al., 1989), and can be considered a strong feature extractor with extended receptive fields capable of linking salient features from the entire sequence. On the other hand, lacking a memory component (as present in the RNN models) prevents the network from modeling a state space, reducing its theoretical strength as a sequence model, thus it requires additional positional information (e.g. sinusoidal positional encodings).

Above theoretical characterizations will drive our explorations in the following sections.

3 Experiment Setup

We train our models on the standard WMT'14 En$\rightarrow$Fr and En$\rightarrow$De datasets that comprise 36.3M and 4.5M sentence pairs, respectively. Each sentence was encoded into a sequence of sub-word units obtained by first tokenizing the sentence with the Moses tokenizer, then splitting tokens into sub-word units (also known as "wordpieces") using the approach described in (Schuster and Nakajima, 2012).

[1]https://github.com/facebookresearch/fairseq-py

[2]https://github.com/tensorflow/tensor2tensor

[3]Assuming that data complexity is satisfied.

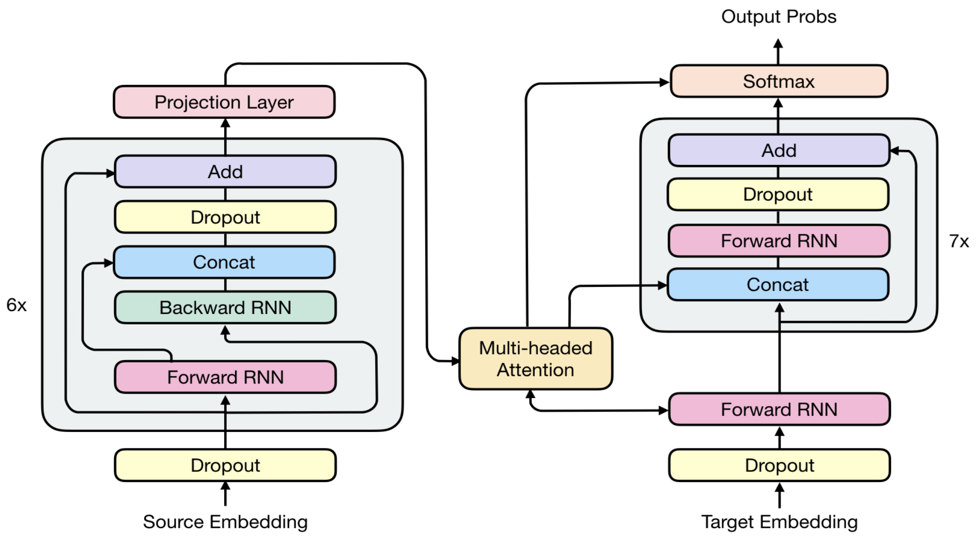


Figure 1: Model architecture of RNMT+. On the left side, the encoder network has 6 bidirectional LSTM layers. At the end of each bidirectional layer, the outputs of the forward layer and the backward layer are concatenated. On the right side, the decoder network has 8 unidirectional LSTM layers, with the first layer used for obtaining the attention context vector through multi-head additive attention. The attention context vector is then fed directly into the rest of the decoder layers as well as the softmax layer.

We use a shared vocabulary of 32K sub-word units for each source-target language pair. No further manual or rule-based post processing of the output was performed beyond combining the sub-word units to generate the targets. We report all our results on newstest 2014, which serves as the test set. A combination of newstest 2012 and newstest 2013 is used for validation.

To evaluate the models, we compute the BLEU metric on tokenized, true-case output.[4] For each training run, we evaluate the model every 30 minutes on the dev set. Once the model converges, we determine the best window based on the average dev-set BLEU score over 21 consecutive evaluations. We report the mean test score and standard deviation over the selected window. This allows us to compare model architectures based on their mean performance after convergence rather than individual checkpoint evaluations, as the latter can be quite noisy for some models.

To enable a fair comparison of architectures, we use the same pre-processing and evaluation methodology for all our experiments. We refrain from using checkpoint averaging (exponential moving averages of parameters) (Junczys-Dowmunt et al., 2016) or checkpoint ensembles (Jean et al., 2015; Chen et al., 2017) to focus on evaluating the performance of individual models.

4 RNMT+

4.1 Model Architecture of RNMT+

The newly proposed RNMT+ model architecture is shown in Figure 1. Here we highlight the key architectural choices that are different between the RNMT+ model and the GNMT model. There are 6 bidirectional LSTM layers in the encoder instead of 1 bidirectional LSTM layer followed by 7 unidirectional layers as in GNMT. For each bidirectional layer, the outputs of the forward layer and the backward layer are concatenated before being fed into the next layer. The decoder network consists of 8 unidirectional LSTM layers similar to the GNMT model. Residual connections are added to the third layer and above for both the encoder and decoder. Inspired by the Transformer model, per-gate layer normalization (Ba et al., 2016) is applied within each LSTM cell. Our empirical results show that layer normalization greatly stabilizes training. No non-linearity is applied to the LSTM output. A projection layer is added to the encoder final output.[5] Multi-head additive attention is used instead of the single-head attention in the GNMT model. Similar to GNMT, we use the

[4]This procedure is used in the literature to which we compare (Gehring et al., 2017; Wu et al., 2016).

[5]Additional projection aims to reduce the dimensionality of the encoder output representations to match the decoder stack dimension.

bottom decoder layer and the final encoder layer output after projection for obtaining the recurrent attention context. In addition to feeding the attention context to all decoder LSTM layers, we also feed it to the softmax by concatenating it with the layer input. This is important for both the quality of the models with multi-head attention and the stability of the training process.

Since the encoder network in RNMT+ consists solely of bi-directional LSTM layers, model parallelism is not used during training. We compensate for the resulting longer per-step time with increased data parallelism (more model replicas), so that the overall time to reach convergence of the RNMT+ model is still comparable to that of GNMT.

We apply the following regularization techniques during training.

- **Dropout:** We apply dropout to both embedding layers and each LSTM layer output before it is added to the next layer's input. Attention dropout is also applied.

- **Label Smoothing:** We use uniform label smoothing with an uncertainty=0.1 (Szegedy et al., 2015). Label smoothing was shown to have a positive impact on both Transformer and RNMT+ models, especially in the case of RNMT+ with multi-head attention. Similar to the observations in (Chorowski and Jaitly, 2016), we found it beneficial to use a larger beam size (e.g. 16, 20, etc.) during decoding when models are trained with label smoothing.

- **Weight Decay:** For the WMT'14 En→De task, we apply L2 regularization to the weights with $\lambda = 10^{-5}$. Weight decay is only applied to the En→De task as the corpus is smaller and thus more regularization is required.

We use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-6}$ and vary the learning rate according to this schedule:

$$lr = 10^{-4} \cdot \min\left(1 + \frac{t \cdot (n-1)}{np}, n, n \cdot (2n)^{\frac{s-nt}{e-s}}\right) \quad (1)$$

Here, t is the current step, n is the number of concurrent model replicas used in training, p is the number of warmup steps, s is the start step of the exponential decay, and e is the end step of the decay. Specifically, we first increase the learning rate linearly during the number of warmup steps, keep it a constant until the decay start step s, then exponentially decay until the decay end step e, and keep it at $5 \cdot 10^{-5}$ after the decay ends. This learning rate schedule is motivated by a similar schedule that was successfully applied in training the Resnet-50 model with a very large batch size (Goyal et al., 2017).

In contrast to the asynchronous training used for GNMT (Dean et al., 2012), we train RNMT+ models with synchronous training (Chen et al., 2016). Our empirical results suggest that when hyper-parameters are tuned properly, synchronous training often leads to improved convergence speed and superior model quality.

To further stabilize training, we also use adaptive gradient clipping. We discard a training step completely if an anomaly in the gradient norm value is detected, which is usually an indication of an imminent gradient explosion. More specifically, we keep track of a moving average and a moving standard deviation of the $\log$ of the gradient norm values, and we abort a step if the norm of the gradient exceeds four standard deviations of the moving average.

4.2 Model Analysis and Comparison

In this section, we compare the results of RNMT+ with ConvS2S and Transformer.

All models were trained with synchronous training. RNMT+ and ConvS2S were trained with 32 NVIDIA P100 GPUs while the Transformer Base and Big models were trained using 16 GPUs.

For RNMT+, we use sentence-level cross-entropy loss. Each training batch contained 4096 sentence pairs (4096 source sequences and 4096 target sequences). For ConvS2S and Transformer models, we use token-level cross-entropy loss. Each training batch contained 65536 source tokens and 65536 target tokens. For the GNMT baselines on both tasks, we cite the largest BLEU score reported in (Wu et al., 2016) without reinforcement learning.

Table 1 shows our results on the WMT'14 En→Fr task. Both the Transformer Big model and RNMT+ outperform GNMT and ConvS2S by about 2 BLEU points. RNMT+ is slightly better than the Transformer Big model in terms of its mean BLEU score. RNMT+ also yields a much lower standard deviation, and hence we observed much less fluctuation in the training curve. It takes approximately 3 days for the Transformer

Base model to converge, while both RNMT+ and the Transformer Big model require about 5 days to converge. Although the batching schemes are quite different between the Transformer Big and the RNMT+ model, they have processed about the same amount of training samples upon convergence.

Model	Test BLEU	Epochs	Training Time
GNMT	38.95	-	-
ConvS2S [7]	39.49 ± 0.11	62.2	438h
Trans. Base	39.43 ± 0.17	20.7	90h
Trans. Big [8]	40.73 ± 0.19	8.3	120h
RNMT+	41.00 ± 0.05	8.5	120h

Table 1: Results on WMT14 En→Fr. The numbers before and after '±' are the mean and standard deviation of test BLEU score over an evaluation window. Note that Transformer models are trained using 16 GPUs, while ConvS2S and RNMT+ are trained using 32 GPUs.

Table 2 shows our results on the WMT'14 En→De task. The Transformer Base model improves over GNMT and ConvS2S by more than 2 BLEU points while the Big model improves by over 3 BLEU points. RNMT+ further outperforms the Transformer Big model and establishes a new state of the art with an averaged value of 28.49. In this case, RNMT+ converged slightly faster than the Transformer Big model and maintained much more stable performance after convergence with a very small standard deviation, which is similar to what we observed on the En-Fr task.

Table 3 summarizes training performance and model statistics. The Transformer Base model is the fastest model in terms of training speed. RNMT+ is slower to train than the Transformer Big model on a per-GPU basis. However, since the RNMT+ model is quite stable, we were able to offset the lower per-GPU throughput with higher concurrency by increasing the number of model replicas, and hence the overall time to convergence was not slowed down much. We also computed the number of floating point operations (FLOPs) in the model's forward path as well as the number of total parameters for all architectures (cf. Table 3). RNMT+ requires fewer FLOPs than the Transformer Big model, even though both models have a comparable number of parameters.

[6]Since the ConvS2S model convergence is very slow we did not explore further tuning on En→Fr, and validated our implementation on En→De.

[7]The BLEU scores for Transformer model are slightly lower than those reported in (Vaswani et al., 2017) due to four differences:

1) We report the mean test BLEU score using the strategy described in section 3.

2) We did not perform checkpoint averaging since it would be inconsistent with our evaluation for other models.

3) We avoided any manual post-processing, like unicode normalization using Moses replace-unicode-punctuation.perl or output tokenization using Moses tokenizer.perl, to rule out its effect on the evaluation. We observed a significant BLEU increase (about 0.6) on applying these post processing techniques.

4) In (Vaswani et al., 2017), reported BLEU scores are calculated using mteval-v13a.pl from Moses, which re-tokenizes its input.

Model	Test BLEU	Epochs	Training Time
GNMT	24.67	-	-
ConvS2S	25.01 ±0.17	38	20h
Trans. Base	27.26 ± 0.15	38	17h
Trans. Big	27.94 ± 0.18	26.9	48h
RNMT+	28.49 ± 0.05	24.6	40h

Table 2: Results on WMT14 En→De. Note that Transformer models are trained using 16 GPUs, while ConvS2S and RNMT+ are trained using 32 GPUs.

Model	Examples/s	FLOPs	Params
ConvS2S	80	15.7B	263.4M
Trans. Base	160	6.2B	93.3M
Trans. Big	50	31.2B	375.4M
RNMT+	30	28.1B	378.9M

Table 3: Performance comparison. Examples/s are normalized by the number of GPUs used in the training job. FLOPs are computed assuming that source and target sequence length are both 50.

5 Ablation Experiments

In this section, we evaluate the importance of four main techniques for both the RNMT+ and the Transformer Big models. We believe that these techniques are universally applicable across different model architectures, and should always be employed by NMT practitioners for best performance.

We take our best RNMT+ and Transformer Big models and remove each one of these techniques independently. By doing this we hope to learn two things about each technique: (1) How much does it affect the model performance? (2) How useful is it for stable training of other techniques and hence the final model?

Model	RNMT+	Trans. Big
Baseline	41.00	40.73
- Label Smoothing	40.33	40.49
- Multi-head Attention	40.44	39.83
- Layer Norm.	*	*
- Sync. Training	39.68	*

Table 4: Ablation results of RNMT+ and the Transformer Big model on WMT'14 En $\rightarrow$ Fr. We report average BLEU scores on the test set. An asterisk '*' indicates an unstable training run (training halts due to non-finite elements).

From Table 4 we draw the following conclusions about the four techniques:

- **Label Smoothing** We observed that label smoothing improves both models, leading to an average increase of 0.7 BLEU for RNMT+ and 0.2 BLEU for Transformer Big models.

- **Multi-head Attention** Multi-head attention contributes significantly to the quality of both models, resulting in an average increase of 0.6 BLEU for RNMT+ and 0.9 BLEU for Transformer Big models.

- **Layer Normalization** Layer normalization is most critical to stabilize the training process of either model, especially when multi-head attention is used. Removing layer normalization results in unstable training runs for both models. Since by design, we remove one technique at a time in our ablation experiments, we were unable to quantify how much layer normalization helped in either case. To be able to successfully train a model without layer normalization, we would have to adjust other parts of the model and retune its hyper-parameters.

- **Synchronous training** Removing synchronous training has different effects on RNMT+ and Transformer. For RNMT+, it results in a significant quality drop, while for the Transformer Big model, it causes the model to become unstable. We also notice that synchronous training is only successful when coupled with a tailored learning rate schedule that has a warmup stage at the beginning (cf. Eq. 1 for RNMT+ and Eq. 2 for Transformer). For RNMT+, removing this warmup stage during synchronous training causes the model to become unstable.

6 Hybrid NMT Models

In this section, we explore hybrid architectures that shed some light on the salient behavior of each model family. These hybrid models outperform the individual architectures on both benchmark datasets and provide a better understanding of the capabilities and limitations of each model family.

6.1 Assessing Individual Encoders and Decoders

In an encoder-decoder architecture, a natural assumption is that the role of an encoder is to build feature representations that can best encode the meaning of the source sequence, while a decoder should be able to process and interpret the representations from the encoder and, at the same time, track the current target history. Decoding is inherently auto-regressive, and keeping track of the state information should therefore be intuitively beneficial for conditional generation.

We set out to study which family of encoders is more suitable to extract rich representations from a given input sequence, and which family of decoders can make the best of such rich representations. We start by combining the encoder and decoder from different model families. Since it takes a significant amount of time for a ConvS2S model to converge, and because the final translation quality was not on par with the other models, we focus on two types of hybrids only: Transformer encoder with RNMT+ decoder and RNMT+ encoder with Transformer decoder.

Encoder	Decoder	En$\rightarrow$Fr Test BLEU
Trans. Big	Trans. Big	40.73 ± 0.19
RNMT+	RNMT+	41.00 ± 0.05
Trans. Big	RNMT+	$\mathbf{41.12 \pm 0.16}$
RNMT+	Trans. Big	39.92 ± 0.21

Table 5: Results for encoder-decoder hybrids.

From Table 5, it is clear that the Transformer encoder is better at encoding or feature extraction than the RNMT+ encoder, whereas RNMT+ is better at decoding or conditional language modeling, confirming our intuition that a stateful decoder is beneficial for conditional language generation.

6.2 Assessing Encoder Combinations

Next, we explore how the features extracted by an encoder can be further enhanced by incorporating additional information. Specifically, we investigate the combination of transformer layers with RNMT+ layers in the same encoder block to build even richer feature representations. We exclusively use RNMT+ decoders in the following architectures since stateful decoders show better performance according to Table 5.

We study two mixing schemes in the encoder (see Fig. 2):

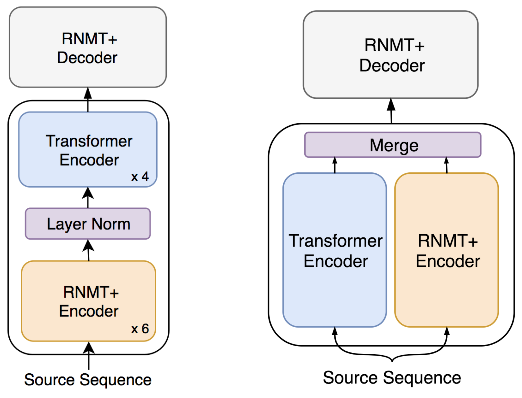
(1) *Cascaded Encoder*: The cascaded encoder aims at combining the representational power of RNNs and self-attention. The idea is to enrich a set of stateful representations by cascading a feature extractor with a focus on vertical mapping, similar to (Pascanu et al., 2013; Devlin, 2017). Our best performing cascaded encoder involves fine tuning transformer layers stacked on top of a pre-trained frozen RNMT+ encoder. Using a pre-trained encoder avoids optimization difficulties while significantly enhancing encoder capacity. As shown in Table 6, the cascaded encoder improves over the Transformer encoder by more than 0.5 BLEU points on the WMT'14 En→Fr task. This suggests that the Transformer encoder is able to extract richer representations if the input is augmented with sequential context.

(2) *Multi-Column Encoder*: As illustrated in Fig. 2b, a multi-column encoder merges the outputs of several independent encoders into a single combined representation. Unlike a cascaded encoder, the multi-column encoder enables us to investigate whether an RNMT+ decoder can distinguish information received from two different channels and benefit from its combination. A crucial operation in a multi-column encoder is therefore how different sources of information are merged into a unified representation. Our best multi-column encoder performs a simple concatenation of individual column outputs.

The model details and hyperparameters of the above two encoders are described in Appendix A.5 and A.6. As shown in Table 6, the multi-column encoder followed by an RNMT+ decoder achieves better results than the Transformer and the RNMT model on both WMT'14 benchmark tasks.

Model	En→Fr BLEU	En→De BLEU
Trans. Big	40.73 ± 0.19	27.94 ± 0.18
RNMT+	41.00 ± 0.05	28.49 ± 0.05
Cascaded	**41.67** ± **0.11**	28.62 ± 0.06
MultiCol	41.66 ± 0.11	**28.84** ± **0.06**

Table 6: Results for hybrids with cascaded encoder and multi-column encoder.



(a) Cascaded Encoder (b) Multi-Column Encoder

Figure 2: Vertical and horizontal mixing of Transformer and RNMT+ components in an encoder.

7 Conclusion

In this work we explored the efficacy of several architectural and training techniques proposed in recent studies on seq2seq models for NMT. We demonstrated that many of these techniques are broadly applicable to multiple model architectures. Applying these new techniques to RNMT models yields RNMT+, an enhanced RNMT model that significantly outperforms the three fundamental architectures on WMT'14 En→Fr and En→De tasks. We further presented several hybrid models developed by combining encoders and decoders from the Transformer and RNMT+ models, and empirically demonstrated the superiority of the Transformer encoder and the RNMT+ decoder in comparison with their counterparts. We then enhanced the encoder architecture by horizontally and vertically mixing components borrowed from these architectures, leading to hybrid architectures that obtain further improvements over RNMT+.

We hope that our work will motivate NMT researchers to further investigate generally applicable training and optimization techniques, and that our exploration of hybrid architectures will open paths for new architecture search efforts for NMT.

Our focus on a standard single-language-pair translation task leaves important open questions to be answered: How do our new architectures compare in multilingual settings, i.e., modeling an *interlingua*? Which architecture is more efficient and powerful in processing finer grained inputs and outputs, e.g., characters or bytes? How transferable are the representations learned by the different architectures to other tasks? And what are the characteristic errors that each architecture makes, e.g., linguistic plausibility?

Acknowledgments

We would like to thank the entire Google Brain Team and Google Translate Team for their foundational contributions to this project. We would also like to thank the entire Tensor2Tensor development team for their useful inputs and discussions.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450. http://arxiv.org/abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*. http://arxiv.org/abs/1409.0473.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.* 5(2):157–166.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1442–1451. https://www.aclweb.org/anthology/D17-1151.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *CoRR* abs/1710.03282. http://arxiv.org/abs/1710.03282.

Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Józefowicz. 2016. Revisiting distributed synchronous SGD. *CoRR* abs/1604.00981. http://arxiv.org/abs/1604.00981.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*. http://arxiv.org/abs/1406.1078.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *CoRR* abs/1612.02695. http://arxiv.org/abs/1612.02695.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. Systran's pure neural machine translation systems. *CoRR* abs/1610.05540. http://arxiv.org/abs/1610.05540.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR* abs/1612.08083. http://arxiv.org/abs/1612.08083.

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, MarcAurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large scale distributed deep networks. In *NIPS*.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, pages 18–27. http://www.aclweb.org/anthology/W17-3203.

Jacob Devlin. 2017. Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the cpu. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2820–2825. http://aclweb.org/anthology/D17-1300.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14(2):179 – 211.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR* abs/1705.03122. http://arxiv.org/abs/1705.03122.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12(10):2451–2471.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He.

2017. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR* abs/1706.02677. http://arxiv.org/abs/1706.02677.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602 – 610. IJCNN 2005.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1828–1836. http://dl.acm.org/citation.cfm?id=2969442.2969444.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR* abs/1512.03385. http://arxiv.org/abs/1512.03385.

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München* 91:1.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jrgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

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

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2(5):359 – 366.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, and et al. 2017. In-datacenter performance analysis of a tensor processing unit. *CoRR* abs/1704.04760. http://arxiv.org/abs/1704.04760.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The amu-uedin submission to the wmt16 news translation task: Attention-based nmt models as feature functions in phrase-based smt. *arXiv preprint arXiv:1605.04809* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Conference on Empirical Methods in Natural Language Processing*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR* abs/1610.10099. http://arxiv.org/abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Yann LeCun and Yoshua Bengio. 1998. The handbook of brain theory and neural networks. MIT Press, Cambridge, MA, USA, chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. http://dl.acm.org/citation.cfm?id=303568.303704.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *CoRR* abs/1312.6026. http://arxiv.org/abs/1312.6026.

Tim Salimans and Diederik P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *CoRR* abs/1602.07868. http://arxiv.org/abs/1602.07868.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing* .

H.T. Siegelmann and E.D. Sontag. 1995. On the computational power of neural nets. *Journal of Computer and System Sciences* 50(1):132 – 150.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR* abs/1505.00387.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *CoRR* abs/1512.00567. http://arxiv.org/abs/1512.00567.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR* abs/1606.04199. http://arxiv.org/abs/1606.04199.

Ultra-Fine Entity Typing

Eunsol Choi[†] **Omer Levy**[†] **Yejin Choi**[†♯] **Luke Zettlemoyer**[†]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♯]Allen Institute for Artificial Intelligence, Seattle WA
{eunsol,omerlevy,yejin,lsz}@cs.washington.edu

Abstract

We introduce a new entity typing task: given a sentence with an entity mention, the goal is to predict a set of free-form phrases (e.g. skyscraper, songwriter, or criminal) that describe appropriate types for the target entity. This formulation allows us to use a new type of distant supervision at large scale: head words, which indicate the type of the noun phrases they appear in. We show that these ultra-fine types can be crowd-sourced, and introduce new evaluation sets that are much more diverse and fine-grained than existing benchmarks. We present a model that can predict open types, and is trained using a multitask objective that pools our new head-word supervision with prior supervision from entity linking. Experimental results demonstrate that our model is effective in predicting entity types at varying granularity; it achieves state of the art performance on an existing fine-grained entity typing benchmark, and sets baselines for our newly-introduced datasets.[1]

1 Introduction

Entities can often be described by very fine grained types. Consider the sentences "Bill robbed John. He was arrested." The noun phrases "John," "Bill," and "he" have very specific types that can be inferred from the text. This includes the facts that "Bill" and "he" are both likely "criminal" due to the "robbing" and "arresting," while "John" is more likely a "victim" because he was "robbed." Such fine-grained types (victim, criminal) are important for context-sensitive tasks such as coreference resolution and question answering (e.g. "Who was the victim?"). Inferring such types for each mention (John, he) is not possible given current typing models that only predict relatively coarse types and only consider named entities.

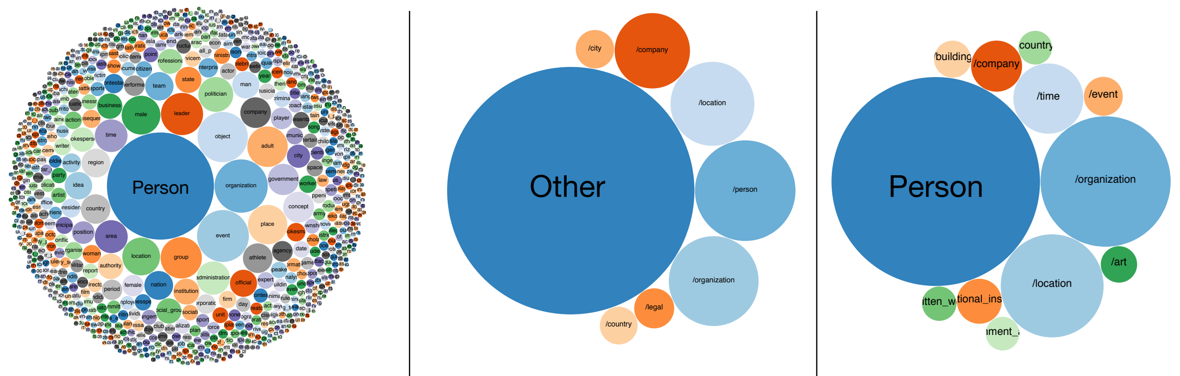
[1]Our data and model can be downloaded from: http://nlp.cs.washington.edu/entity_type

Sentence with Target Entity	Entity Types
During the Inca Empire, {**the Inti Raymi**} was the most important of four ceremonies celebrated in Cusco.	event, festival, **ritual, custom, ceremony, party, celebration**
{**They**} have been asked to appear in court to face the charge.	person, **accused, suspect, defendant**
Ban praised Rwanda's commitment to the UN and its role in {**peacemaking operations**}.	event, **plan, mission, action**

Table 1: Examples of entity mentions and their annotated types, as annotated in our dataset. The entity mentions are bold faced and in the curly brackets. The bold blue types do not appear in existing fine-grained type ontologies.

To address this challenge, we present a new task: given a sentence with a target entity mention, predict free-form noun phrases that describe appropriate types for the role the target entity plays in the sentence. Table 1 shows three examples that exhibit a rich variety of types at different granularities. Our task effectively subsumes existing fine-grained named entity typing formulations due to the use of a very large type vocabulary and the fact that we predict types for all noun phrases, including named entities, nominals, and pronouns.

Incorporating fine-grained entity types has improved entity-focused downstream tasks, such as relation extraction (Yaghoobzadeh et al., 2017a), question answering (Yavuz et al., 2016), query analysis (Balog and Neumayer, 2012), and coreference resolution (Durrett and Klein, 2014). These systems used a relatively coarse type ontology. However, manually designing the ontology is a challenging task, and it is difficult to cover all pos-



a) Our Dataset b) OntoNotes c) FIGER

Figure 1: A visualization of all the labels that cover 90% of the data, where a bubble's size is proportional to the label's frequency. Our dataset is much more diverse and fine grained when compared to existing datasets (OntoNotes and FIGER), in which the top 5 types cover 70-80% of the data.

sible concepts even within a limited domain. This can be seen empirically in existing datasets, where the label distribution of fine-grained entity typing datasets is heavily skewed toward coarse-grained types. For instance, annotators of the OntoNotes dataset (Gillick et al., 2014) marked about half of the mentions as "other," because they could not find a suitable type in their ontology (see Figure 1 for a visualization and Section 2.2 for details).

Our more open, ultra-fine vocabulary, where types are free-form noun phrases, alleviates the need for hand-crafted ontologies, thereby greatly increasing overall type coverage. To better understand entity types in an unrestricted setting, we crowdsource a new dataset of 6,000 examples. Compared to previous fine-grained entity typing datasets, the label distribution in our data is substantially more *diverse* and *fine-grained*. Annotators easily generate a wide range of types and can determine with 85% agreement if a type generated by another annotator is appropriate. Our evaluation data has over 2,500 unique types, posing a challenging learning problem.

While our types are harder to predict, they also allow for a new form of contextual distant supervision. We observe that text often contains cues that explicitly match a mention to its type, in the form of the mention's head word. For example, "the incumbent chairman of the African Union" is a type of "chairman." This signal complements the supervision derived from linking entities to knowledge bases, which is context-oblivious. For example, "Clint Eastwood" can be described with dozens of types, but context-sensitive typing would prefer "director" instead of "mayor" for the sentence "Clint Eastwood won 'Best Director' for Million Dollar Baby."

We combine head-word supervision, which provides ultra-fine type labels, with traditional signals from entity linking. Although the problem is more challenging at finer granularity, we find that mixing fine and coarse-grained supervision helps significantly, and that our proposed model with a multitask objective exceeds the performance of existing entity typing models. Lastly, we show that head-word supervision can be used for previous formulations of entity typing, setting the new state-of-the-art performance on an existing fine-grained NER benchmark.

2 Task and Data

Given a sentence and an entity mention e within it, the task is to predict a set of natural-language phrases T that describe the type of e. The selection of T is context sensitive; for example, in "Bill Gates has donated billions to eradicate malaria," Bill Gates should be typed as "philanthropist" and not "inventor." This distinction is important for context-sensitive tasks such as coreference resolution and question answering (e.g. "Which philanthropist is trying to prevent malaria?").

We annotate a dataset of about 6,000 mentions via crowdsourcing (Section 2.1), and demonstrate that using an large type vocabulary substantially increases annotation coverage and diversity over existing approaches (Section 2.2).

2.1 Crowdsourcing Entity Types

To capture multiple domains, we sample sentences from Gigaword (Parker et al., 2011), OntoNotes (Hovy et al., 2006), and web articles (Singh et al., 2012). We select entity mentions by taking maximal noun phrases from a constituency parser (Manning et al., 2014) and mentions from a coreference resolution system (Lee et al., 2017).

We provide the sentence and the target entity mention to five crowd workers on Mechanical Turk, and ask them to annotate the entity's type. To encourage annotators to generate fine-grained types, we require at least one general type (e.g. person, organization, location) and two specific types (e.g. doctor, fish, religious institute), from a type vocabulary of about 10K frequent noun phrases. We use WordNet (Miller, 1995) to expand these types automatically by generating all their synonyms and hypernyms based on the most common sense, and ask five different annotators to validate the generated types. Each pair of annotators agreed on 85% of the binary validation decisions (i.e. whether a type is suitable or not) and 0.47 in Fleiss's κ. To further improve consistency, the final type set contained only types selected by at least 3/5 annotators. Further crowdsourcing details are available in the supplementary material.

Our collection process focuses on precision. Thus, the final set is diverse but not comprehensive, making evaluation non-trivial (see Section 5).

2.2 Data Analysis

We collected about 6,000 examples. For analysis, we classified each type into three disjoint bins:

- 9 **general** types: person, location, object, organization, place, entity, object, time, event
- 121 **fine-grained** types, mapped to fine-grained entity labels from prior work (Ling and Weld, 2012; Gillick et al., 2014) (e.g. film, athlete)
- 10,201 **ultra-fine** types, encompassing every other label in the type space (e.g. detective, lawsuit, temple, weapon, composer)

On average, each example has 5 labels: 0.9 general, 0.6 fine-grained, and 3.9 ultra-fine types. Among the 10,000 ultra-fine types, 2,300 unique types were actually found in the 6,000 crowdsourced examples. Nevertheless, our distant supervision data (Section 3) provides positive training examples for every type in the entire vocabulary, and our model (Section 4) can and does predict from a 10K type vocabulary. For example, the model correctly predicts "television network" and "archipelago" for some mentions, even though that type never appears in the 6,000 crowdsourced examples.

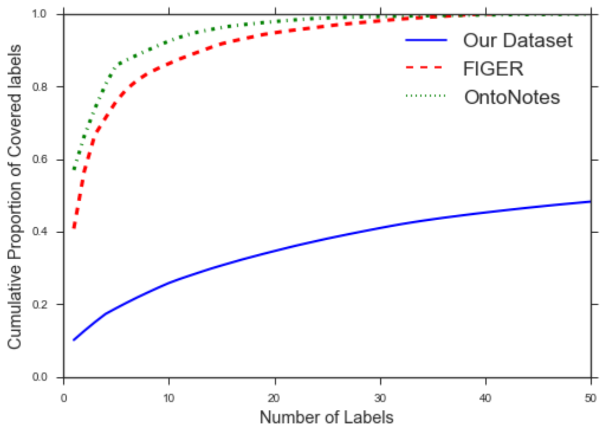


Figure 2: The label distribution across different evaluation datasets. In existing datasets, the top 4 or 7 labels cover over 80% of the labels. In ours, the top 50 labels cover less than 50% of the data.

Improving Type Coverage We observe that prior fine-grained entity typing datasets are heavily focused on coarse-grained types. To quantify our observation, we calculate the distribution of types in FIGER (Ling and Weld, 2012), OntoNotes (Gillick et al., 2014), and our data. For examples with multiple types ($|T| > 1$), we counted each type $1/|T|$ times.

Figure 2 shows the percentage of labels covered by the top N labels in each dataset. In previous enitity typing datasets, the distribution of labels is highly skewed towards the top few labels. To cover 80% of the examples, FIGER requires only the top 7 types, while OntoNotes needs only 4; our dataset requires 429 different types.

Figure 1 takes a deeper look by visualizing the types that cover 90% of the data, demonstrating the diversity of our dataset. It is also striking that more than half of the examples in OntoNotes are classified as "other," perhaps because of the limitation of its predefined ontology.

Improving Mention Coverage Existing datasets focus mostly on named entity mentions, with the exception of OntoNotes, which contained nominal expressions. This has implications on the transferability of FIGER/OntoNotes-based models to tasks such as coreference resolution, which need to analyze all types of entity mentions (pronouns, nominal expressions, and named entity

Source	Example Sentence	Labels	Size	Prec.
Head Words	**Western powers that brokered the proposed deal in Vienna** are likely to balk, said Valerie Lincy, a researcher with the Wisconsin Project.	power	20M	80.4%
	Alexis Kaniaris, CEO of the organizing company Europartners, explained, speaking in a radio program in **national radio station NET**.	radio, station, radio_station		
Entity Linking + Definitions	**Toyota** recalled more than 8 million vehicles globally over sticky pedals that can become entrapped in floor mats.	manufacturer	2.7M	77.7%
Entity Linking + KB	Iced Earth's musical style is influenced by many traditional heavy metal groups such as **Black Sabbath**.	person, artist, actor, author, musician	2.5M	77.6%

Table 2: Distant supervision examples and statistics. We extracted the headword and Wikipedia definition supervision from Gigaword and Wikilink corpora. KB-based supervision is mapped from prior work, which used Wikipedia and news corpora.

mentions). Our new dataset provides a well-rounded benchmark with roughly 40% pronouns, 38% nominal expressions, and 22% named entity mentions. The case of pronouns is particularly interesting, since the mention itself provides little information.

3 Distant Supervision

Training data for fine-grained NER systems is typically obtained by linking entity mentions and drawing their types from knowledge bases (KBs). This approach has two limitations: recall can suffer due to KB incompleteness (West et al., 2014), and precision can suffer when the selected types do not fit the context (Ritter et al., 2011). We alleviate the recall problem by mining entity mentions that were linked to Wikipedia in HTML, and extract relevant types from their encyclopedic definitions (Section 3.1). To address the precision issue (context-insensitive labeling), we propose a new source of distant supervision: automatically extracted nominal head words from raw text (Section 3.2). Using head words as a form of distant supervision provides fine-grained information about named entities and nominal mentions. While a KB may link "the 44th president of the United States" to many types such as author, lawyer, and professor, head words provide only the type "president", which is relevant in the context.

We experiment with the new distant supervision sources as well as the traditional KB supervision. Table 2 shows examples and statistics for each source of supervision. We annotate 100 examples from each source to estimate the noise and usefulness in each signal (precision in Table 2).

3.1 Entity Linking

For KB supervision, we leveraged training data from prior work (Ling and Weld, 2012; Gillick et al., 2014) by manually mapping their ontology to our 10,000 noun type vocabulary, which covers 130 of our labels (general and fine-grained).[2] Section 6 defines this mapping in more detail.

To improve both entity and type coverage of KB supervision, we use definitions from Wikipedia. We follow Shnarch et al. () who observed that the first sentence of a Wikipedia article often states the entity's type via an "is a" relation; for example, "Roger Federer is a Swiss professional tennis player." Since we are using a large type vocabulary, we can now mine this typing information.[3] We extracted descriptions for 3.1M entities which contain 4,600 unique type labels such as "competition," "movement," and "village."

We bypass the challenge of automatically linking entities to Wikipedia by exploiting existing hyperlinks in web pages (Singh et al., 2012), following prior work (Ling and Weld, 2012; Yosef et al., 2012). Since our heuristic extraction of types from the definition sentence is somewhat noisy, we use a more conservative entity linking policy[4] that yields a signal with similar overall accuracy to KB-linked data.

[2] Data from: https://github.com/shimaokasonse/NFGEC

[3] We extract types by applying a dependency parser (Manning et al., 2014) to the definition sentence, and taking nouns that are dependents of a copular edge or connected to nouns linked to copulars via appositive or conjunctive edges.

[4] Only link if the mention contains the Wikipedia entity's name *and* the entity's name contains the mention's head.

3.2 Contextualized Supervision

Many nominal entity mentions include detailed type information within the mention itself. For example, when describing Titan V as "the newly-released graphics card", the head words and phrases of this mention ("graphics card" and "card") provide a somewhat noisy, but very easy to gather, context-sensitive type signal.

We extract nominal head words with a dependency parser (Manning et al., 2014) from the Gigaword corpus as well as the Wikilink dataset. To support multiword expressions, we included nouns that appear next to the head if they form a phrase in our type vocabulary. Finally, we lowercase all words and convert plural to singular.

Our analysis reveals that this signal has a comparable accuracy to the types extracted from entity linking (around 80%). Many errors are from the parser, and some errors stem from idioms and transparent heads (e.g. "parts of capital" labeled as "part"). While the headword is given as an input to the model, with heavy regularization and multitasking with other supervision sources, this supervision helps encode the context.

4 Model

We design a model for predicting sets of types given a mention in context. The architecture resembles the recent neural AttentiveNER model (Shimaoka et al., 2017), while improving the sentence and mention representations, and introducing a new multitask objective to handle multiple sources of supervision. The hyperparameter settings are listed in the supplementary material.

Context Representation Given a sentence $x_1, \ldots, x_n$, we represent each token x_i using a pre-trained word embedding w_i. We concatenate an additional location embedding l_i which indicates whether x_i is before, inside, or after the mention. We then use $[x_i; l_i]$ as an input to a bidirectional LSTM, producing a contextualized representation h_i for each token; this is different from the architecture of Shimaoka et al. 2017, who used two separate bidirectional LSTMs on each side of the mention. Finally, we represent the context c as a weighted sum of the contextualized token representations using MLP-based attention:

$$a_i = \text{SoftMax}_i(v_a \cdot \text{relu}(W_a h_i))$$

Where W_a and v_a are the parameters of the attention mechanism's MLP, which allows interaction between the forward and backward directions of the LSTM before computing the weight factors.

Mention Representation We represent the mention m as the concatenation of two items: (a) a character-based representation produced by a CNN on the entire mention span, and (b) a weighted sum of the pre-trained word embeddings in the mention span computed by attention, similar to the mention representation in a recent coreference resolution model (Lee et al., 2017). The final representation is the concatenation of the context and mention representations: $r = [c; m]$.

Label Prediction We learn a type label embedding matrix $W_t \in \mathbb{R}^{n \times d}$ where n is the number of labels in the prediction space and d is the dimension of r. This matrix can be seen as a combination of three sub matrices, $W_{general}, W_{fine}, W_{ultra}$, each of which contains the representations of the general, fine, and ultra-fine types respectively. We predict each type's probability via the sigmoid of its inner product with r: $y = \sigma(W_t r)$. We predict every type t for which $y_t > 0.5$, or $\arg\max y_t$ if there is no such type.

Multitask Objective The distant supervision sources provide partial supervision for ultra-fine types; KBs often provide more general types, while head words usually provide only ultra-fine types, without their generalizations. In other words, the absence of a type at a different level of abstraction does not imply a negative signal; e.g. when the head word is "inventor", the model should not be discouraged to predict "person".

Prior work used a customized hinge loss (Abhishek et al., 2017) or max margin loss (Ren et al., 2016a) to improve robustness to noisy or incomplete supervision. We propose a multitask objective that reflects the characteristic of our training dataset. Instead of updating all labels for each example, we divide labels into three bins (general, fine, and ultra-fine), and update labels only in bin containing at least one positive label. Specifically, the training objective is to minimize J where t is the target vector at each granularity:

$$\begin{aligned} J_{\text{all}} = {} & J_{\text{general}} \cdot \mathbb{1}_{\text{general}}(t) \\ & + J_{\text{fine}} \cdot \mathbb{1}_{\text{fine}}(t) \\ & + J_{\text{ultra}} \cdot \mathbb{1}_{\text{ultra}}(t) \end{aligned}$$

Where $\mathbb{1}_{\text{category}}(t)$ is an indicator function that checks if t contains a type in the category, and

Model	Dev				Test			
	MRR	P	R	F1	MRR	P	R	F1
AttentiveNER	0.221	**53.7**	15.0	23.5	0.223	**54.2**	15.2	23.7
Our Model	**0.229**	48.1	**23.2**	**31.3**	**0.234**	47.1	**24.2**	**32.0**

Table 3: Performance of our model and AttentiveNER (Shimaoka et al., 2017) on the new entity typing benchmark, using same training data. We show results for both development and test sets.

Train Data	Total				General (1918)			Fine (1289)			Ultra-Fine (7594)		
	MRR	P	R	F1	P	R	F1	P	R	F1	P	R	F1
All	**0.229**	48.1	**23.2**	**31.3**	60.3	61.6	61.0	40.4	**38.4**	**39.4**	42.8	8.8	14.6
– Crowd	0.173	40.1	14.8	21.6	53.7	45.6	49.3	20.8	18.5	19.6	54.4	4.6	8.4
– Head	0.220	**50.3**	19.6	28.2	58.8	**62.8**	60.7	**44.4**	29.8	35.6	**46.2**	4.7	8.5
– EL	0.225	48.4	22.3	30.6	**62.2**	60.1	**61.2**	40.3	26.1	31.7	41.4	**9.9**	**16.0**

Table 4: Results on the development set for different type granularity and for different supervision data with our model. In each row, we remove a single source of supervision. Entity linking (EL) includes supervision from both KB and Wikipedia definitions. The numbers in the first row are example counts for each type granularity.

J_{category} is the category-specific logistic regression objective:

$$J = -\sum_i t_i \cdot \log(y_i) + (1 - t_i) \cdot \log(1 - y_i)$$

5 Evaluation

Experiment Setup The crowdsourced dataset (Section 2.1) was randomly split into train, development, and test sets, each with about 2,000 examples. We use this relatively small manually-annotated training set (*Crowd* in Table 4) alongside the two distant supervision sources: entity linking (KB and Wikipedia definitions) and head words. To combine supervision sources of different magnitudes (2K crowdsourced data, 4.7M entity linking data, and 20M head words), we sample a batch of equal size from each source at each iteration. We reimplement the recent AttentiveNER model (Shimaoka et al., 2017) for reference.[5]

We report macro-averaged precision, recall, and F1, and the average mean reciprocal rank (MRR).

Results Table 3 shows the performance of our model and our reimplementation of AttentiveNER. Our model, which uses a multitask objective to learn finer types without punishing more general types, shows recall gains at the cost of drop in precision. The MRR score shows that our model is slightly better than the baseline at ranking correct types above incorrect ones.

Table 4 shows the performance breakdown for different type granularity and different supervision. Overall, as seen in previous work on fine-grained NER literature (Gillick et al., 2014; Ren et al., 2016a), finer labels were more challenging to predict than coarse grained labels, and this issue is exacerbated when dealing with ultra-fine types. All sources of supervision appear to be useful, with crowdsourced examples making the biggest impact. Head word supervision is particularly helpful for predicting ultra-fine labels, while entity linking improves fine label prediction. The low general type performance is partially because of nominal/pronoun mentions (e.g. "it"), and because of the large type inventory (sometimes "location" and "place" are annotated interchangeably).

Analysis We manually analyzed 50 examples from the development set, four of which we present in Table 5. Overall, the model was able to generate accurate general types and a diverse set of type labels. Despite our efforts to annotate a comprehensive type set, the gold labels still miss many potentially correct labels (example (a): "man" is reasonable but counted as incorrect). This makes the precision estimates lower than the actual performance level, with about half the precision errors belonging to this category. Real precision errors include predicting co-hyponyms (example (b): "accident" instead of "attack"), and types that

[5] We use the AttentiveNER model with no engineered features or hierarchical label encoding (as a hierarchy is not clear in our label setting) and let it predict from the same label space, training with the same supervision data.

(a)	Example Annotation	Bruguera said {he} had problems with his left leg and had grown tired early during the match . **person, athlete, player, adult, male, contestant**
	Prediction	**person, athlete, player, adult, male, contestant**, defendant, man
(b)	Example Annotation	{The explosions} occurred on the night of October 7 , against the Hilton Taba and campsites used by Israelis in Ras al-Shitan. **event** *calamity, attack, disaster*
	Prediction	**event,** accident
(c)	Example Annotation	Similarly , Enterprise was considered for refit to replace Challenger after {the latter} was destroyed , but Endeavour was built from structural spares instead . *object, spacecraft, rocket, thing, vehicle, shuttle*
	Prediction	event
(d)	Context Annotation	" There is a wealth of good news in this report , and I 'm particularly encouraged by the progress {we} are making against AIDS , " HHS Secretary Donna Shalala said in a statement. **government, group,** *organization,hospital,administration,socialist*
	Prediction	**government, group**, person

Table 5: Example and predictions from our best model on the development set. Entity mentions are marked with curly brackets, the correct predictions are boldfaced, and the missing labels are italicized and written in red.

may be true, but are not supported by the context.

We found that the model often abstained from predicting any fine-grained types. Especially in challenging cases as in example (c), the model predicts only general types, explaining the low recall numbers (28% of examples belong to this category). Even when the model generated correct fine-grained types as in example (d), the recall was often fairly low since it did not generate a complete set of related fine-grained labels.

Estimating the performance of a model in an incomplete label setting and expanding label coverage are interesting areas for future work. Our task also poses a potential modeling challenge; sometimes, the model predicts two incongruous types (e.g. "location" and "person"), which points towards modeling the task as a joint set prediction task, rather than predicting labels individually. We provide sample outputs on the project website.

6 Improving Existing Fine-Grained NER with Better Distant Supervision

We show that our model and distant supervision can improve performance on an existing fine-grained NER task. We chose the widely-used OntoNotes (Gillick et al., 2014) dataset which includes nominal and named entity mentions.[6]

[6]While we were inspired by FIGER (Ling and Weld, 2012), the dataset presents technical difficulties. The test set has only 600 examples, and the development set was labeled with distant supervision, not manual annotation. We therefore focus our evaluation on OntoNotes.

Augmenting the Training Data The original OntoNotes training set (ONTO in Tables 6 and 7) is extracted by linking entities to a KB. We supplement this dataset with our two new sources of distant supervision: Wikipedia definition sentences (WIKI) and head word supervision (HEAD) (see Section 3). To convert the label space, we manually map a single noun from our natural-language vocabulary to each formal-language type in the OntoNotes ontology. 77% of OntoNote's types directly correspond to suitable noun labels (e.g. "doctor" to "/person/doctor"), whereas the other cases were mapped with minimal manual effort (e.g. "musician" to "person/artist/music", "politician" to "/person/political_figure"). We then expand these labels according to the ontology to include their hypernyms ("/person/political_figure" will also generate "/person"). Lastly, we create negative examples by assigning the "/other" label to examples that are not mapped to the ontology. The augmented dataset contains 2.5M/0.6M new positive/negative examples, of which 0.9M/0.1M are from Wikipedia definition sentences and 1.6M/0.5M from head words.

Experiment Setup We compare performance to other published results and to our reimplementation of AttentiveNER (Shimaoka et al., 2017). We also compare models trained with different sources of supervision. For this dataset, we did not use our multitask objective (Section 4), since expanding types to include their ontological hypernyms largely eliminates the partial supervision as-

	Acc.	Ma-F1	Mi-F1
AttentiveNER++	51.7	70.9	64.9
AFET (Ren et al., 2016a)	55.1	71.1	64.7
LNR (Ren et al., 2016b)	57.2	71.5	66.1
Ours (ONTO+WIKI+HEAD)	**59.5**	**76.8**	**71.8**

Table 6: Results on the OntoNotes fine-grained entity typing test set. The first two models (AttentiveNER++ and AFET) use only KB-based supervision. LNR uses a filtered version of the KB-based training set. Our model uses all our distant supervision sources.

Model	Training Data			Performance		
	ONTO	WIKI	HEAD	Acc.	MaF1	MiF1
Attn. NER	✓			46.5	63.3	58.3
	✓	✓	✓	53.7	72.8	68.0
Ours	✓			41.7	64.2	59.5
	✓	✓		48.5	67.6	63.6
	✓		✓	57.9	73.0	66.9
		✓	✓	60.1	75.0	68.7
	✓	✓	✓	**61.6**	**77.3**	**71.8**

Table 7: Ablation study on the OntoNotes fine-grained entity typing development. The second row isolates dataset improvements, while the third row isolates the model.

sumption. Following prior work, we report macro- and micro-averaged F1 score, as well as accuracy (exact set match).

Results Table 6 shows the overall performance on the test set. Our combination of model and training data shows a clear improvement from prior work, setting a new state-of-the art result.[7]

In Table 7, we show an ablation study. Our new supervision sources improve the performance of both the AttentiveNER model and our own. We observe that every supervision source improves performance in its own right. Particularly, the naturally-occurring head-word supervision seems to be the prime source of improvement, increasing performance by about 10% across all metrics.

Predicting Miscellaneous Types While analyzing the data, we observed that over half of the mentions in OntoNotes' development set were annotated only with the miscellaneous type ("/other"). For both models in our evaluation, detecting the miscellaneous category is substantially easier than producing real types (94% F1 vs. 58% F1 with our best model). We provide further details of this analysis in the supplementary material.

7 Related Work

Fine-grained NER has received growing attention, and is used in many applications (Gupta et al., 2017; Ren et al., 2017; Yaghoobzadeh et al., 2017b; Raiman and Raiman, 2018). Researchers studied typing in varied contexts, including mentions in specific sentences (as we consider) (Ling and Weld, 2012; Gillick et al., 2014; Yogatama et al., 2015; Dong et al., 2015; Schutze et al., 2017), corpus-level prediction (Yaghoobzadeh and Schütze, 2016), and lexicon level (given only a noun phrase with no context) (Yao et al., 2013).

Recent work introduced fine-grained type ontologies (Rabinovich and Klein, 2017; Murty et al., 2017; Corro et al., 2015), defined using Wikipedia categories (100), Freebase types (1K) and WordNet senses (16K). However, they focus on named entities, and data has been challenging to gather, often approximating gold annotations with distant supervision. In contrast, (1) our ontology contains any frequent noun phrases that depicts a type, (2) our task goes beyond named entities, covering every noun phrase (even pronouns), and (3) we provide crowdsourced annotations which provide context-sensitive, fine grained type labels.

Contextualized fine-grained entity typing is related to selectional preference (Resnik, 1996; Pantel et al., 2007; Zapirain et al., 2013; de Cruys, 2014), where the goal is to induce semantic generalizations on the type of arguments a predicate prefers. Rather than focusing on predicates, we condition on the entire sentence to deduce the arguments' types, which allows us to capture more nuanced types. For example, not every type that fits "**He** played the violin in his room" is also suitable for "**He** played the violin in the Carnegie Hall". Entity typing here can be connected to argument finding in semantic role labeling.

To deal with noisy distant supervision for KB population and entity typing, researchers used multi-instance multi-label learning (Surdeanu et al., 2012; Yaghoobzadeh et al., 2017b) or custom losses (Abhishek et al., 2017; Ren et al., 2016a). Our multitask objective handles noisy supervision by pooling different distant supervision sources across different levels of granularity.

[7] We did not compare to a system from (Yogatama et al., 2015), which reports slightly higher test number (72.98 micro F1) as they used a different, unreleased test set.

8 Conclusion

Using virtually unrestricted types allows us to expand the standard KB-based training methodology with typing information from Wikipedia definitions and naturally-occurring head-word supervision. These new forms of distant supervision boost performance on our new dataset as well as on an existing fine-grained entity typing benchmark. These results set the first performance levels for our evaluation dataset, and suggest that the data will support significant future work.

Acknowledgement

The research was supported in part the ARO (W911NF-16-1-0121) the NSF (IIS-1252835, IIS-1562364), and an Allen Distinguished Investigator Award. We would like to thank the reviewers for constructive feedback. Also thanks to Yotam Eshel and Noam Cohen for providing the Wikilink dataset. Special thanks to the members of UW NLP for helpful discussions and feedback.

References

Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of European Chapter of Association for Computational Linguistics*.

Krisztian Balog and Robert Neumayer. 2012. Hierarchical target type identification for entity-oriented queries. In *Proceedings of the Conference on Information and Knowledge Management*.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of Empirical Methods in Natural Language Processing*.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. In *Transactions of the Association for Computational Linguistics*.

Daniel Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *CoRR*, abs/1412.1820.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2671–2680.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of Association for the Advancement of Artificial Intelligence*. Citeseer.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Shikhar Murty, Patrick Verga, Luke Vilnis, and Andrew McCallum. 2017. Finer grained entity typing with typenet. In *AKBC Workshop*.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard H. Hovy. 2007. Isp: Learning inferential selectional preferences. In *Proceedings of North American Chapter of the Association for Computational Linguistics*.

Robert Parker, David Graff, David Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition (ldc2011t07). In *Linguistic Data Consortium*.

Maxim Rabinovich and Dan Klein. 2017. Fine-grained entity typing with high-multiplicity assignments. In *Proceedings of Association for Computational Linguistics (ACL)*.

Jonathan Raiman and Olivier Raiman. 2018. Deeptype: Multilingual entity linking by neural type system evolution. In *Association for the Advancement of Artificial Intelligence*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings Empirical Methods in Natural Language Processing*.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of Knowledge Discovery and Data Mining*.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of World Wide Web Conference*.

Philip Resnik. 1996. Selectional constraints: an information-theoretic model and its computational realization. *Cognition*, 61 1-2:127–59.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Hinrich Schutze, Ulli Waltinger, and Sanjeev Karn. 2017. End-to-end trainable attentive decoder for hierarchical entity classification. In *Proceedings of European Chapter of Association for Computational Linguistics*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the European Chapter of Association for Computational Linguistics (ACL)*.

Eyal Shnarch, Libby Barak, and Ido Dagan. Extracting lexical reference rules from wikipedia. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of World Wide Web Conference*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017a. Noise mitigation for neural entity typing and relation extraction. *In Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, abs/1612.07495.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017b. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of European Chapter of Association for Computational Linguistics*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Corpus-level fine-grained entity typing using contextual information. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Automatic KnowledgeBase Construction Workshop at the Conference on Information and Knowledge Management*.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of Association for Computational Linguistics (ACL)*.

M Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of the International Conference on Computational Linguistics*.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez i Villodre, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39:631–663.

Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking

Shikhar Murty*
UMass Amherst
smurty@cs.umass.edu

Patrick Verga*
UMass Amherst
pat@cs.umass.edu

Luke Vilnis
UMass Amherst
luke@cs.umass.edu

Irena Radovanovic
Chan Zuckerberg Initiative
iradovanovic@chanzuckerberg.com

Andrew McCallum
UMass Amherst
mccallum@cs.umass.edu

Abstract

Extraction from raw text to a knowledge base of entities and fine-grained types is often cast as prediction into a flat set of entity and type labels, neglecting the rich hierarchies over types and entities contained in curated ontologies. Previous attempts to incorporate hierarchical structure have yielded little benefit and are restricted to shallow ontologies. This paper presents new methods using real and complex bilinear mappings for integrating hierarchical information, yielding substantial improvement over flat predictions in entity linking and fine-grained entity typing, and achieving new state-of-the-art results for end-to-end models on the benchmark FIGER dataset. We also present two new human-annotated datasets containing wide and deep hierarchies which we will release to the community to encourage further research in this direction: *MedMentions*, a collection of PubMed abstracts in which 246k mentions have been mapped to the massive UMLS ontology; and *TypeNet*, which aligns Freebase types with the WordNet hierarchy to obtain nearly 2k entity types. In experiments on all three datasets we show substantial gains from hierarchy-aware training.

1 Introduction

Identifying and understanding entities is a central component in knowledge base construction (Roth et al., 2015) and essential for enhancing downstream tasks such as relation extraction (Yaghoobzadeh et al., 2017b), question answering (Das et al., 2017; Welbl et al., 2017) and search (Dalton et al., 2014). This has led to considerable research in automatically identifying entities in text, predicting their types, and linking them to existing structured knowledge sources.

Current state-of-the-art models encode a textual mention with a neural network and classify the mention as being an instance of a fine grained type or entity in a knowledge base. Although in many cases the types and their entities are arranged in a hierarchical ontology, most approaches ignore this structure, and previous attempts to incorporate hierarchical information yielded little improvement in performance (Shimaoka et al., 2017). Additionally, existing benchmark entity typing datasets only consider small label sets arranged in very shallow hierarchies. For example, FIGER (Ling and Weld, 2012), the *de facto* standard fine grained entity type dataset, contains only 113 types in a hierarchy only two levels deep.

In this paper we investigate models that explicitly integrate hierarchical information into the embedding space of entities and types, using a hierarchy-aware loss on top of a deep neural network classifier over textual mentions. By using this additional information, we learn a richer, more robust representation, gaining statistical efficiency when predicting similar concepts and aiding the classification of rarer types. We first validate our methods on the narrow, shallow type system of FIGER, out-performing state-of-the-art methods not incorporating hand-crafted features and matching those that do.

To evaluate on richer datasets and stimulate further research into hierarchical entity/typing prediction with larger and deeper ontologies, we introduce two new human annotated datasets. The first is *MedMentions*, a collection of PubMed ab-

*equal contribution
Data and code for experiments: https://github.com/MurtyShikhar/Hierarchical-Typing

stracts in which 246k concept mentions have been annotated with links to the Unified Medical Language System (UMLS) ontology (Bodenreider, 2004), an order of magnitude more annotations than comparable datasets. UMLS contains over 3.5 million concepts in a hierarchy having average depth 14.4. Interestingly, UMLS does not distinguish between types and entities (an approach we heartily endorse), and the technical details of linking to such a massive ontology lead us to refer to our MedMentions experiments as entity linking. Second, we present *TypeNet*, a curated mapping from the Freebase type system into the WordNet hierarchy. TypeNet contains over 1900 types with an average depth of 7.8.

In experimental results, we show improvements with a hierarchically-aware training loss on each of the three datasets. In entity-linking MedMentions to UMLS, we observe a 6% relative increase in accuracy over the base model. In experiments on entity-typing from Wikipedia into TypeNet, we show that incorporating the hierarchy of types and including a hierarchical loss provides a dramatic 29% relative increase in MAP. Our models even provide benefits for shallow hierarchies allowing us to match the state-of-art results of Shimaoka et al. (2017) on the FIGER (GOLD) dataset without requiring hand-crafted features.

We will publicly release the TypeNet and MedMentions datasets to the community to encourage further research in truly fine-grained, hierarchical entity-typing and linking.

2 New Corpora and Ontologies

2.1 MedMentions

Over the years researchers have constructed many large knowledge bases in the biomedical domain (Apweiler et al., 2004; Davis et al., 2008; Chatraryamontri et al., 2017). Many of these knowledge bases are specific to a particular sub-domain encompassing a few particular types such as genes and diseases (Piñero et al., 2017).

UMLS (Bodenreider, 2004) is particularly comprehensive, containing over 3.5 million concepts (UMLS does not distinguish between entities and types) defining their relationships and a curated hierarchical ontology. For example *LETM1 Protein* IS-A *Calcium Binding Protein* IS-A *Binding Protein* IS-A *Protein* IS-A *Genome Encoded Entity*. This fact makes UMLS particularly well suited for methods explicitly exploiting hierarchical structure.

Accurately linking textual biological entity mentions to an existing knowledge base is extremely important but few richly annotated resources are available. Even when resources do exist, they often contain no more than a few thousand annotated entity mentions which is insufficient for training state-of-the-art neural network entity linkers. State-of-the-art methods must instead rely on string matching between entity mentions and canonical entity names (Leaman et al., 2013; Wei et al., 2015; Leaman and Lu, 2016). To address this, we constructed MedMentions, a new, large dataset identifying and linking entity mentions in PubMed abstracts to specific UMLS concepts. Professional annotators exhaustively annotated UMLS entity mentions from 3704 PubMed abstracts, resulting in 246,000 linked mention spans. The average depth in the hierarchy of a concept from our annotated set is 14.4 and the maximum depth is 43.

MedMentions contains an order of magnitude more annotations than similar biological entity linking PubMed datasets (Doğan et al., 2014; Wei et al., 2015; Li et al., 2016). Additionally, these datasets contain annotations for only one or two entity types (genes or chemicals and disease etc.). MedMentions instead contains annotations for a wide diversity of entities linking to UMLS. Statistics for several other datasets are in Table 1 and further statistics are in 2.

Dataset	mentions	unique entities
MedMentions	246,144	25,507
BCV-CDR	28,797	2,356
NCBI Disease	6,892	753
BCII-GN Train	6,252	1,411
NLM Citation GIA	1,205	310

Table 1: Statistics from various biological entity linking data sets from scientific articles. NCBI Disease (Doğan et al., 2014) focuses exclusively on disease entities. BCV-CDR (Li et al., 2016) contains both chemicals and diseases. BCII-GN and NLM (Wei et al., 2015) both contain genes.

Statistic	Train	Dev	Test
#Abstracts	2,964	370	370
#Sentences	28,457	3,497	3,268
#Mentions	199,977	24,026	22,141
#Entities	22,416	5,934	5,521

Table 2: MedMentions statistics.

2.2 TypeNet

TypeNet is a new dataset of hierarchical entity types for extremely fine-grained entity typing. TypeNet was created by manually aligning Freebase types (Bollacker et al., 2008) to noun synsets from the WordNet hierarchy (Fellbaum, 1998), naturally producing a hierarchical type set.

To construct TypeNet, we first consider all Freebase types that were linked to more than 20 entities. This is done to eliminate types that are either very specific or very rare. We also remove all Freebase API types, e.g. the [/freebase, /dataworld, /schema, /atom, /scheme, and /topics] domains.

For each remaining Freebase type, we generate a list of candidate WordNet synsets through a substring match. An expert annotator then attempted to map the Freebase type to one or more synsets in the candidate list with a *parent-of*, *child-of* or *equivalence* link by comparing the definitions of each synset with example entities of the Freebase type. If no match was found, the annotator manually formulated queries for the online WordNet API until an appropriate synset was found. See Table 9 for an example annotation.

Two expert annotators independently aligned each Freebase type before meeting to resolve any conflicts. The annotators were conservative with assigning equivalence links resulting in a greater number of *child-of* links. The final dataset contained 13 *parent-of*, 727 *child-of*, and 380 *equivalence* links. Note that some Freebase types have multiple *child-of* links to WordNet, making TypeNet, like WordNet, a directed acyclic graph. We then took the union of each of our annotated Freebase types, the synset that they linked to, and any ancestors of that synset.

Typeset	Count	Depth	Gold KB links
CoNLL-YAGO	4	1	Yes
OntoNotes 5.0	19	1	No
Gillick et al. (2014)	88	3	Yes
Figer	112	2	Yes
Hyena	505	9	No
Freebase	2k	2	Yes
WordNet	16k	14	No
TypeNet*	1,941	14	Yes

Table 3: Statistics from various type sets. TypeNet is the largest type hierarchy with a gold mapping to KB entities. *The entire WordNet could be added to TypeNet increasing the total size to 17k types.

We also added an additional set of 614 *FB* $\rightarrow$ *FB* links 4. This was done by computing conditional probabilities of Freebase types given other Freebase types from a collection of 5 million randomly chosen Freebase entities. The conditional probability P($t_2 \mid t_1$) of a Freebase type t_2 given another Freebase type t_1 was calculated as $\frac{\#(t_1,t_2)}{\#t_1}$. Links with a conditional probability less than or equal to 0.7 were discarded. The remaining links were manually verified by an expert annotator and valid links were added to the final dataset, preserving acyclicity.

Freebase Types	1081
WordNet Synsets	860
child-of links	727
equivalence links	380
parent-of links	13
Freebase-Freebase links	614

Table 4: Stats for the final TypeNet dataset. child-of, parent-of, and equivalence links are from Freebase types $\rightarrow$ WordNet synsets.

3 Model

3.1 Background: Entity Typing and Linking

We define a textual mention m as a sentence with an identified entity. The goal is then to classify m with one or more labels. For example, we could take the sentence m = "*Barack Obama is the President of the United States.*" with the identified entity string **Barack Obama**. In the task of *entity linking*, we want to map m to a specific entity in a knowledge base such as "m/02mjmr" in Freebase. In *mention-level typing*, we label m with one or more types from our type system T such as t^m = {president, leader, politician} (Ling and Weld, 2012; Gillick et al., 2014; Shimaoka et al., 2017). In *entity-level typing*, we instead consider a bag of mentions B_e which are all linked to the same entity. We label B_e with t^e, the set of all types expressed in all $m \in B_e$ (Yao et al., 2013; Neelakantan and Chang, 2015; Verga et al., 2017; Yaghoobzadeh et al., 2017a).

3.2 Mention Encoder

Our model converts each mention m to a d dimensional vector. This vector is used to classify the type or entity of the mention. The basic model depicted in Figure 1 concatenates the averaged word embeddings of the mention string with the output of a convolutional neural network (CNN). The

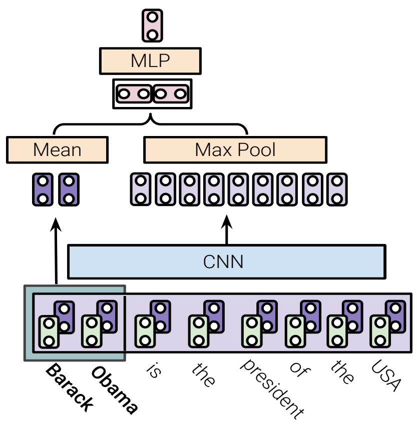


Figure 1: Sentence encoder for all our models. The input to the CNN consists of the concatenation of position embeddings with word embeddings. The output of the CNN is concatenated with the mean of mention surface form embeddings, and then passed through a 2 layer MLP.

word embeddings of the mention string capture global, context independent semantics while the CNN encodes a context dependent representation.

3.2.1 Token Representation

Each sentence is made up of s tokens which are mapped to d_w dimensional word embeddings. Because sentences may contain mentions of more than one entity, we explicitly encode a distinguished mention in the text using position embeddings which have been shown to be useful in state of the art relation extraction models (dos Santos et al., 2015; Lin et al., 2016) and machine translation (Vaswani et al., 2017). Each word embedding is concatenated with a d_p dimensional learned position embedding encoding the token's relative distance to the target entity. Each token within the distinguished mention span has position 0, tokens to the left have a negative distance from $[-s, 0)$, and tokens to the right of the mention span have a positive distance from $(0, s]$. We denote the final sequence of token representations as M.

3.2.2 Sentence Representation

The embedded sequence M is then fed into our context encoder. Our context encoder is a single layer CNN followed by a $\tanh$ non-linearity to produce C. The outputs are max pooled across time to get a final context embedding, m_{CNN}.

$$c_i = \tanh(b + \sum_{j=0}^{w} W[j]M[i - \lfloor \frac{w}{2} \rfloor + j])$$

$$m_{\text{CNN}} = \max_{0 \le i \le n-w+1} c_i$$

Each $W[j] \in \mathbb{R}^{d \times d}$ is a CNN filter, the bias $b \in \mathbb{R}^d$, $M[i] \in \mathbb{R}^d$ is a token representation, and the max is taken pointwise. In all of our experiments we set $w = 5$.

In addition to the contextually encoded mention, we create a global mention encoding, m_{G}, by averaging the word embeddings of the tokens within the mention span.

The final mention representation m_{F} is constructed by concatenating m_{CNN} and m_{G} and applying a two layer feed-forward network with $\tanh$ non-linearity (see Figure 1):

$$m_{\text{F}} = W_2 \tanh(W_1 \begin{bmatrix} m_{\text{SFM}} \\ m_{\text{CNN}} \end{bmatrix} + b_1) + b_2$$

4 Training

4.1 Mention-Level Typing

Mention level entity typing is treated as multi-label prediction. Given the sentence vector m_{F}, we compute a score for each type in typeset T as:

$$y_j = \mathbf{t_j}^{\top} m_{\text{F}}$$

where $\mathbf{t_j}$ is the embedding for the j^{th} type in T and y_j is its corresponding score. The mention is labeled with t^m, a binary vector of all types where $t^m_j = 1$ if the j^{th} type is in the set of gold types for m and 0 otherwise. We optimize a multi-label binary cross entropy objective:

$$\mathcal{L}_{\text{type}}(m) = -\sum_j t^m_j \log y_j + (1 - t^m_j) \log(1 - y_j)$$

4.2 Entity-Level Typing

In the absence of mention-level annotations, we instead must rely on distant supervision (Mintz et al., 2009) to noisily label all mentions of entity e with all types belonging to e. This procedure inevitably leads to noise as not all mentions of an entity express each of its known types. To alleviate this noise, we use multi-instance multi-label learning (MIML) (Surdeanu et al., 2012) which operates over *bags* rather than mentions. A bag of mentions $B_e = \{m^1, m^2, \ldots, m^n\}$ is the set of all mentions belonging to entity e. The bag is labeled with t^e, a binary vector of all types where $t^e_j = 1$ if the j^th type is in the set of gold types for e and 0 otherwise.

For every entity, we subsample k mentions from its bag of mentions. Each mention is then encoded independently using the model described in Section 3.2 resulting in a bag of vectors. Each of the k sentence vectors m^i_{F} is used to compute a score for each type in t^e:

$$y^i_j = \mathbf{t_j}^\top m^i_{\mathrm{F}}$$

where $\mathbf{t_j}$ is the embedding for the j^th type in t^e and y^i is a vector of logits corresponding to the i^th mention. The final bag predictions are obtained using element-wise LogSumExp pooling across the k logit vectors in the bag to produce entity level logits y:

$$y = \log \sum_i \exp(y^i)$$

We use these final bag level predictions to optimize a multi-label binary cross entropy objective:

$$\mathcal{L}_{\text{type}}(B_e) = -\sum_j t^e_j \log y_j + (1 - t^e_j)\log(1 - y_j)$$

4.3 Entity Linking

Entity linking is similar to mention-level entity typing with a single correct class per mention. Because the set of possible entities is in the millions, linking models typically integrate an alias table mapping entity mentions to a set of possible candidate entities. Given a large corpus of entity linked data, one can compute conditional probabilities from mention strings to entities (Spitkovsky and Chang, 2012). In many scenarios this data is unavailable. However, knowledge bases such as UMLS contain a canonical string name for each of its curated entities. State-of-the-art biological entity linking systems tend to operate on various string edit metrics between the entity mention string and the set of canonical entity strings in the existing structured knowledge base (Leaman et al., 2013; Wei et al., 2015).

For each mention in our dataset, we generate 100 candidate entities $e_c = (e_1, e_2, \ldots, e_{100})$ each with an associated string similarity score csim. See Appendix A.5.1 for more details on candidate generation. We generate the sentence representation m_{F} using our encoder and compute a similarity score between m_{F} and the learned embedding e of each of the candidate entities. This score and string cosine similarity csim are combined via a learned linear combination to generate our final score. The final prediction at test time $\hat{e}$ is the maximally similar entity to the mention.

$$\phi(m, e) = \alpha\, e^\top m_{\mathrm{F}} + \beta\, \mathrm{csim}(m, e)$$
$$\hat{e} = \operatorname*{argmax}_{e \in e_c} \phi(m, e)$$

We optimize this model by multinomial cross entropy over the set of candidate entities and correct entity e.

$$\mathcal{L}_{\text{link}}(m, e_c) = -\,\phi(m, e) + \log \sum_{e' \in e_c} \exp \phi(m, e')$$

5 Encoding Hierarchies

Both entity typing and entity linking treat the label space as prediction into a flat set. To explicitly incorporate the structure between types/entities into our training, we add an additional loss. We consider two methods for modeling the hierarchy of the embedding space: real and complex bilinear maps, which are two of the state-of-the-art knowledge graph embedding models.

5.1 Hierarchical Structure Models

Bilinear: Our standard bilinear model scores a hypernym link between (c_1, c_2) as:

$$s(c_1, c_2) = c_1{}^\top A c_2$$

where A $\in \mathbb{R}^{d\times d}$ is a learned real-valued non-diagonal matrix and c_1 is the child of c_2 in the hierarchy. This model is equivalent to RESCAL (Nickel et al., 2011) with a single IS-A relation type. The type embeddings are the same whether used on the left or right side of the relation. We merge this with the base model by using the parameter A as an additional map before type/entity scoring.

Complex Bilinear: We also experiment with a complex bilinear map based on the ComplEx model (Trouillon et al., 2016), which was shown to have strong performance predicting the hypernym relation in WordNet, suggesting suitability for asymmetric, transitive relations such as those in our type hierarchy. ComplEx uses complex valued vectors for types, and diagonal complex matrices for relations, using Hermitian inner products (taking the complex conjugate of the second argument, equivalent to treating the right-hand-side type embedding to be the complex conjugate of the left hand side), and finally taking the real part of the score[1]. The score of a hypernym link between (c_1, c_2) in the ComplEx model is defined as:

$$\begin{aligned}
s(c_1, c_2) &= \text{Re}(< c_1, r_{\text{IS-A}}, c_2 >) \\
&= \text{Re}(\sum_k c_{1k} r_k \bar{c}_{2k}) \\
&= \langle \text{Re}(c_1), \text{Re}(r_{\text{IS-A}}), \text{Re}(c_2) \rangle \\
&+ \langle \text{Re}(c_1), \text{Im}(r_{\text{IS-A}}), \text{Im}(c_2) \rangle \\
&+ \langle \text{Im}(c_1), \text{Re}(r_{\text{IS-A}}), \text{Im}(c_2) \rangle \\
&- \langle \text{Im}(c_1), \text{Im}(r_{\text{IS-A}}), \text{Re}(c_2) \rangle
\end{aligned}$$

where c_1, c_2 and $r_{\text{IS-A}}$ are complex valued vectors representing c_1, c_2 and the IS-A relation respectively. $\text{Re}(z)$ represents the real component of z and $\text{Im}(z)$ is the imaginary component. As noted in Trouillon et al. (2016), the above function is antisymmetric when $r_{\text{IS-A}}$ is purely imaginary.

Since entity/type embeddings are complex vectors, in order to combine it with our base model, we also need to represent mentions with complex vectors for scoring. To do this, we pass the output of the mention encoder through two different affine transformations to generate a real and imaginary component:

$$\begin{aligned}
\text{Re}(m_{\text{F}}) &= W_{\text{real}} m_{\text{F}} + b_{\text{real}} \\
\text{Im}(m_{\text{F}}) &= W_{\text{img}} m_{\text{F}} + b_{\text{img}}
\end{aligned}$$

where m_{F} is the output of the mention encoder, and W_{real}, $W_{\text{img}} \in \mathbb{R}^{d \times d}$ and b_{real}, $b_{\text{img}} \in \mathbb{R}^d$.

5.2 Training with Hierarchies

Learning a hierarchy is analogous to learning embeddings for nodes of a knowledge graph with a single hypernym/IS-A relation. To train these embeddings, we sample (c_1, c_2) pairs, where each pair is a positive link in our hierarchy. For each positive link, we sample a set N of n negative links. We encourage the model to output high scores for positive links, and low scores for negative links via a binary cross entropy (BCE) loss:

$$\begin{aligned}
\mathcal{L}_{\text{struct}} &= -\log \sigma(s(c_{1i}, c_{2i})) \\
&\quad + \sum_N \log(1 - \sigma(s(c_{1i}, c'_{2i}))) \\
\mathcal{L} &= \mathcal{L}_{\text{type/link}} + \gamma \mathcal{L}_{\text{struct}}
\end{aligned}$$

[1]This step makes the scoring function technically not bilinear, as it commutes with addition but not complex multiplication, but we term it *bilinear* for ease of exposition.

where $s(c_1, c_2)$ is the score of a link (c_1, c_2), and $\sigma(\cdot)$ is the logistic sigmoid. The weighting parameter γ is $\in \{0.1, 0.5, 0.8, 1, 2.0, 4.0\}$. The final loss function that we optimize is $\mathcal{L}$.

6 Experiments

We perform three sets of experiments: mention-level entity typing on the benchmark dataset FIGER, entity-level typing using Wikipedia and TypeNet, and entity linking using MedMentions.

6.1 Models

CNN: Each mention is encoded using the model described in Section 3.2. The resulting embedding is used for classification into a flat set labels. Specific implementation details can be found in Appendix A.2.
CNN+Complex: The CNN+Complex model is equivalent to the CNN model but uses complex embeddings and Hermitian dot products.
Transitive: This model does not add an additional hierarchical loss to the training objective (unless otherwise stated). We add additional labels to each entity corresponding to the transitive closure, or the union of all ancestors of its known types. This provides a rich additional learning signal that greatly improves classification of specific types.
Hierarchy: These models add an explicit hierarchical loss to the training objective, as described in Section 5, using either complex or real-valued bilinear mappings, and the associated parameter sharing.

6.2 Mention-Level Typing in FIGER

To evaluate the efficacy of our methods we first compare against the current state-of-art models of Shimaoka et al. (2017). The most widely used type system for fine-grained entity typing is FIGER which consists of 113 types organized in a 2 level hierarchy. For training, we use the publicly available W2M data (Ren et al., 2016) and optimize the mention typing loss function defined in Section-4.1 with the additional hierarchical loss where specified. For evaluation, we use the manually annotated FIGER (GOLD) data by Ling and Weld (2012). See Appendix A.2 and A.3 for specific implementation details.

6.2.1 Results

In Table 5 we see that our base CNN models (CNN and CNN+Complex) match LSTM models of Shimaoka et al. (2017) and Gupta et al. (2017), the

Model	Acc	Macro F1	Micro F1
Ling and Weld (2012)	47.4	69.2	65.5
Shimaoka et al. (2017) †	55.6	75.1	71.7
Gupta et al. (2017)†	57.7	72.8	72.1
Shimaoka et al. (2017)‡	**59.6**	**78.9**	**75.3**
CNN	57.0	75.0	72.2
+ hierarchy	58.4	76.3	73.6
CNN+Complex	57.2	75.3	72.9
+ hierarchy	**59.7**	**78.3**	**75.4**

Table 5: Accuracy and Macro/Micro F1 on FIGER (GOLD). † is an LSTM model. ‡ is an attentive LSTM along with additional hand crafted features.

Model	Low Data	Full Data
CNN	51.72	68.15
+ hierarchy	54.82	75.56
+ transitive	57.68	77.21
+ hierarchy + transitive	58.74	**78.59**
CNN+Complex	50.51	69.83
+ hierarchy	55.30	72.86
+ transitive	53.71	72.18
+ hierarchy + transitive	**58.81**	77.21

Table 6: MAP of entity-level typing in Wikipedia data using TypeNet. The second column shows results using 5% of the total data. The last column shows results using the full set of 344,246 entities.

Model	original	normalized
mention tfidf	61.09	74.66
CNN	67.42	82.40
+ hierarchy	67.73	82.77
CNN+Complex	67.23	82.17
+ hierarchy	**68.34**	**83.52**

Table 7: Accuracy on entity linking in MedMentions. Maximum recall is 81.82% because we use an imperfect alias table to generate candidates. Normalized scores consider only mentions which contain the gold entity in the candidate set. Mention tfidf is $csim$ from Section 4.3.

previous state-of-the-art for models without hand-crafted features. When incorporating structure into our models, we gain 2.5 points of accuracy in our CNN+Complex model, matching the overall state of the art attentive LSTM that relied on hand-crafted features from syntactic parses, topic models, and character n-grams. The structure can help our model predict lower frequency types which is a similar role played by hand-crafted features.

6.3 Entity-Level Typing in TypeNet

Next we evaluate our models on entity-level typing in TypeNet using Wikipedia. For each entity, we follow the procedure outlined in Section 4.2. We predict labels for each instance in the entity's bag and aggregate them into entity-level predictions using $\mathrm{LogSumExp}$ pooling. Each type is assigned a predicted score by the model. We then rank these scores and calculate average precision for each of the types in the test set, and use these scores to calculate mean average precision (MAP). We evaluate using MAP instead of accuracy which is standard in large knowledge base link prediction tasks (Verga et al., 2017; Trouillon et al., 2016). These scores are calculated only over Freebase types, which tend to be lower in the hierarchy. This is to avoid artificial score inflation caused by trivial predictions such as 'entity.' See Appendix A.4 for more implementation details.

6.3.1 Results

Table 6 shows the results for entity level typing on our Wikipedia TypeNet dataset. We see that both the basic CNN and the CNN+Complex models perform similarly with the CNN+Complex model doing slightly better on the full data regime. We also see that both models get an improvement when adding an explicit hierarchy loss, even before adding in the transitive closure. The transitive closure itself gives an additional increase in performance to both models. In both of these cases, the basic CNN model improves by a greater amount than CNN+Complex. This could be a result of the complex embeddings being more difficult to optimize and therefore more susceptible to variations in hyperparameters. When adding in both the transitive closure and the explicit hierarchy loss, the performance improves further. We observe similar trends when training our models in a lower data regime with ~150,000 examples, or about 5% of the total data.

In all cases, we note that the baseline models that do not incorporate any hierarchical information (neither the transitive closure nor the hierarchy loss) perform ~9 MAP worse, demonstrating the benefits of incorporating structure information.

6.4 MedMentions Entity Linking with UMLS

In addition to entity typing, we evaluate our model's performance on an entity linking task using MedMentions, our new PubMed / UMLS dataset described in Section 2.1.

6.4.1 Results

Table 7 shows results for baselines and our proposed variant with additional hierarchical loss. None of these models incorporate transitive clo-

Tips and Pitfalls in **Direct Ligation** of Large Spontaneous Splenorenal Shunt during Liver Transplantation Patients with large spontaneous splenorenal shunt . . .
baseline: Direct [Direct → General Modifier → Qualifier → Property or Attribute]
+hierarchy: Ligature (correct) [Ligature → Surgical Procedures → medical treatment approach]
A novel approach for selective chemical functionalization and localized assembly of one-dimensional **nanostructures**.
baseline: Structure [Structure → order or structure → general epistemology]
+hierarchy: Nanomaterials (correct) [Nanomaterials → Nanoparticle Complex → Drug or Chemical by Structure]
Gcn5 is recruited onto the **il-2** promoter by interacting with the NFAT in T cells upon TCR stimulation .
baseline: Interleukin-27 [Interleukin-27 → IL2 → Interleukin Gene]
+hierarchy: IL2 Gene (correct) [IL2 Gene → Interleukin Gene]

Table 8: Example predictions from MedMentions. Each example shows the sentence with entity mention span in bold. **Baseline**, shows the predicted entity and its ancestors of a model not incorporating structure. Finally, **+hierarchy** shows the prediction and ancestors for a model which explicitly incorporates the hierarchical structure information.

sure information, due to difficulty incorporating it in our candidate generation, which we leave to future work. The *Normalized* metric considers performance only on mentions with an alias table hit; all models have 0 accuracy for mentions otherwise. We also report the overall score for comparison in future work with improved candidate generation. We see that incorporating structure information results in a 1.1% reduction in absolute error, corresponding to a ~6% reduction in relative error on this large-scale dataset.

Table 8 shows qualitative predictions for models with and without hierarchy information incorporated. Each example contains the sentence (with target entity in bold), predictions for the baseline and hierarchy aware models, and the ancestors of the predicted entity. In the first and second example, the baseline model becomes extremely dependent on TFIDF string similarities when the gold candidate is rare ($\leq$ 10 occurrences). This shows that modeling the structure of the entity hierarchy helps the model disambiguate rare entities. In the third example, structure helps the model understand the hierarchical nature of the labels and prevents it from predicting an entity that is overly specific (e.g predicting Interleukin-27 rather than the correct and more general entity IL2 Gene).

Note that, in contrast with the previous tasks, the complex hierarchical loss provides a significant boost, while the real-valued bilinear model does not. A possible explanation is that UMLS is a far larger/deeper ontology than even TypeNet, and the additional ability of complex embeddings to model intricate graph structure is key to realizing gains from hierarchical modeling.

7 Related Work

By directly linking a large set of mentions and typing a large set of entities with respect to a new ontology and corpus, and our incorporation of structural learning between the many entities and types in our ontologies of interest, our work draws on many different but complementary threads of research in information extraction, knowledge base population, and completion.

Our structural, hierarchy-aware loss between types and entities draws on research in Knowledge Base Inference such as Jain et al. (2018), Trouillon et al. (2016) and Nickel et al. (2011). Combining KB completion with hierarchical structure in knowledge bases has been explored in (Dalvi et al., 2015; Xie et al., 2016). Recently, Wu et al. (2017) proposed a hierarchical loss for text classification.

Linking mentions to a flat set of entities, often in Freebase or Wikipedia, is a long-standing task in NLP (Bunescu and Pasca, 2006; Cucerzan, 2007; Durrett and Klein, 2014; Francis-Landau et al., 2016). Typing of mentions at varying levels of granularity, from CoNLL-style named entity recognition (Tjong Kim Sang and De Meulder, 2003), to the more fine-grained recent approaches (Ling and Weld, 2012; Gillick et al., 2014; Shimaoka et al., 2017), is also related to our task. A few prior attempts to incorporate a very shallow hierarchy into fine-grained entity typing have not lead to significant or consistent improvements (Gillick et al., 2014; Shimaoka et al., 2017).

The knowledge base Yago (Suchanek et al., 2007) includes integration with WordNet and type hierarchies have been derived from its type system (Yosef et al., 2012). Del Corro et al. (2015) use manually crafted rules and patterns (Hearst patterns (Hearst, 1992), appositives, etc) to automatically match entity types to Wordnet synsets.

Recent work has moved towards unifying these two highly related tasks by improving entity linking by simultaneously learning a fine grained entity type predictor (Gupta et al., 2017). Learning hierarchical structures or transitive relations between concepts has been the subject of much recent work (Vilnis and McCallum, 2015; Vendrov et al., 2016; Nickel and Kiela, 2017)

We draw inspiration from all of this prior work, and contribute datasets and models to address previous challenges in jointly modeling the structure of large-scale hierarchical ontologies and mapping textual mentions into an extremely fine-grained space of entities and types.

8 Conclusion

We demonstrate that explicitly incorporating and modeling hierarchical information leads to increased performance in experiments on entity typing and linking across three challenging datasets. Additionally, we introduce two new human-annotated datasets: MedMentions, a corpus of 246k mentions from PubMed abstracts linked to the UMLS knowledge base, and TypeNet, a new hierarchical fine-grained entity typeset an order of magnitude larger and deeper than previous datasets.

While this work already demonstrates considerable improvement over non-hierarchical modeling, future work will explore techniques such as Box embeddings (Vilnis et al., 2018) and Poincaré embeddings (Nickel and Kiela, 2017) to represent the hierarchical embedding space, as well as methods to improve recall in the candidate generation process for entity linking. Most of all, we are excited to see new techniques from the NLP community using the resources we have presented.

9 Acknowledgements

We thank Nicholas Monath, Haw-Shiuan Chang and Emma Strubell for helpful comments on early drafts of the paper. Creation of the MedMentions corpus is supported and managed by the Meta team at the Chan Zuckerberg Initiative. A pre-release of the dataset is available at `http://github.com/chanzuckerberg/MedMentions`. This work was supported in part by the Center for Intelligent Information Retrieval and the Center for Data Science, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction., and in part by the National Science Foundation under Grant No. IIS-1514053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. 2004. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl_1):D115–D119.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Razvan C Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Eacl*, volume 6, pages 9–16.

Andrew Chatr-aryamontri, Rose Oughtred, Lorrie Boucher, Jennifer Rust, Christie Chang, Nadine K Kolas, Lara O'Donnell, Sara Oster, Chandra Theesfeld, Adnane Sellam, et al. 2017. The biogrid interaction database: 2017 update. *Nucleic acids research*, 45(D1):D369–D379.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374. ACM.

Bhavana Dalvi, Einat Minkov, Partha P Talukdar, and William W Cohen. 2015. Automatic gloss finding for a knowledge base using ontological constraints. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 369–378. ACM.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 2: Short Papers), pages 358–365, Vancouver, Canada. Association for Computational Linguistics.

Allan Peter Davis, Cynthia G Murphy, Cynthia A Saraceni-Richards, Michael C Rosenstein, Thomas C Wiegers, and Carolyn J Mattingly. 2008. Comparative toxicogenomics database: a knowledgebase and discovery tool for chemical–gene–disease networks. *Nucleic acids research*, 37(suppl_1):D786–D792.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 1256–1261.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *CoRR*, abs/1412.1820.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2671–2680, Copenhagen, Denmark. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Prachi Jain, Shikhar Murty, Mausam, and Soumen Chakrabarti. 2018. Mitigating the effect of out-of-vocabulary entity pairs in matrix factorization for knowledge base inference. In *The 27th International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. 2013. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917.

Robert Leaman and Zhiyong Lu. 2016. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics*, 32(18):2839–2846.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525, Denver, Colorado. Association for Computational Linguistics.

Maximilian Nickel and Douwe Kiela. 2017. Poincar\'e embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

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

Janet Piñero, Àlex Bravo, Núria Queralt-Rosinach, Alba Gutiérrez-Sacristán, Jordi Deu-Pons, Emilio Centeno, Javier García-García, Ferran Sanz, and Laura I Furlong. 2017. Disgenet: a comprehensive platform integrating information on human disease-associated genes and variants. *Nucleic acids research*, 45(D1):D833–D839.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1825–1834.

Benjamin Roth, Nicholas Monath, David Belanger, Emma Strubell, Patrick Verga, and Andrew McCallum. 2015. Building knowledge bases with universal schema: Cold start and slot-filling approaches.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing ACL*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1271–1280, Valencia, Spain. Association for Computational Linguistics.

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for english wikipedia concepts.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the International Conference on World Wide Web (WWW)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Conference on Advances in Neural Information Processing (NIPS)*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. *ICLR*.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 613–622, Valencia, Spain. Association for Computational Linguistics.

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. 2018. Probabilistic embedding of knowledge graphs with box lattice measures. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. *ICLR*.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2015. Gnormplus: an integrative approach for tagging genes, gene families, and protein domains. *BioMed research international*, 2015.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481*.

Cinna Wu, Mark Tygert, and Yann LeCun. 2017. Hierarchical loss for classification. *CoRR*, abs/1709.01062.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, pages 2965–2971.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017a. Corpus-level fine-grained entity typing. *arXiv preprint arXiv:1708.02275*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017b. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1183–1194, Valencia, Spain. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 79–84. ACM.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

A Supplementary Materials

A.1 TypeNet Construction

Freebase type: musical_chord
Example entities: psalms_chord, power_chord harmonic_seventh_chord
chord.n.01: a straight line connecting two points on a curve
chord.n.02: a combination of three or more notes that blend harmoniously when sounded together
musical.n.01: a play or film whose action and dialogue is interspersed with singing and dancing

Table 9: Example given to TypeNet annotators. Here, the Freebase type to be linked is musical_chord. This type is annotated in Freebase belonging to the entities psalms_chord, harmonic_seventh_chord, and power_chord. Below the list of example entities are candidate WordNet synsets obtained by substring matching between the Freebase type and all WordNet synsets. The correctly aligned synset is chord.n.02 shown in bold.

A.2 Model Implementation Details

For all of our experiments, we use pretrained 300 dimensional word vectors from Pennington et al. (2014). These embeddings are fixed during training. The type vectors and entity vectors are all 300 dimensional vectors initialized using Glorot initialization (Glorot and Bengio, 2010). The number of negative links for hierarchical training $n \in \{16, 32, 64, 128, 256\}$.

For regularization, we use dropout (Srivastava et al., 2014) with p $\in \{0.5, 0.75, 0.8\}$ on the sentence encoder output and L2 regularize all learned parameters with $\lambda \in \{$1e-5, 5e-5, 1e-4$\}$. All our parameters are optimized using Adam (Kingma and Ba, 2014) with a learning rate of 0.001. We tune our hyper-parameters via grid search and early stopping on the development set.

A.3 FIGER Implementation Details

To train our models, we use the mention typing loss function defined in Section-5. For models with structure training, we additionally add in the hierarchical loss, along with a weight that is obtained by tuning on the dev set. We follow the same inference time procedure as Shimaoka et al. (2017) For each mention, we first assign the type with the largest probability according to the logits, and then assign additional types based on the condition that their corresponding probability be greater than 0.5.

A.4 Wikipedia Data and Implementation Details

At train time, each training example randomly samples an entity bag of 10 mentions. At test time we classify bags of 20 mentions of an entity. The dataset contains a total of 344,246 entities mapped to the 1081 Freebase types from TypeNet. We consider all sentences in Wikipedia between 10 and 50 tokens long. Tokenization and sentence splitting was performed using NLTK (Loper and Bird, 2002). From these sentences, we considered all entities annotated with a cross-link in Wikipedia that we could link to Freebase and assign types in TypeNet. We then split the data by entities into a 90-5-5 train, dev, test split.

A.5 UMLS Implementation details

We pre-process each string by lowercasing and removing stop words. We consider ngrams from size 1 to 5 and keep the top 100,000 features and the final vectors are L2 normalized. For each mention, In our experiments we consider the top 100 most similar entities as the candidate set.

A.5.1 Candidate Generation Details

Each mention and each canonical entity string in UMLS are mapped to TFIDF character ngram vectors. We pre-process each string by lowercasing and removing stop words. We consider ngrams from size 1 to 5 and keep the top 100,000 features and the final vectors are L2 normalized. For each mention, we calculate the cosine similarity, csim, between the mention string and each canonical entity string. In our experiments we consider the top 100 most similar entities as the candidate set.

Improving Knowledge Graph Embedding Using Simple Constraints

Boyang Ding[1,2], **Quan Wang**[1,2,3*], **Bin Wang**[1,2], **Li Guo**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]State Key Laboratory of Information Security, Chinese Academy of Sciences
{dingboyang,wangquan,wangbin,guoli}@iie.ac.cn

Abstract

Embedding knowledge graphs (KGs) into continuous vector spaces is a focus of current research. Early works performed this task via simple models developed over KG triples. Recent attempts focused on either designing more complicated triple scoring models, or incorporating extra information beyond triples. This paper, by contrast, investigates the potential of using very simple constraints to improve KG embedding. We examine *non-negativity constraints* on entity representations and *approximate entailment constraints* on relation representations. The former help to learn compact and interpretable representations for entities. The latter further encode regularities of logical entailment between relations into their distributed representations. These constraints impose prior beliefs upon the structure of the embedding space, without negative impacts on efficiency or scalability. Evaluation on WordNet, Freebase, and DBpedia shows that our approach is simple yet surprisingly effective, significantly and consistently outperforming competitive baselines. The constraints imposed indeed improve model interpretability, leading to a substantially increased structuring of the embedding space. Code and data are available at `https://github.com/iieir-km/ComplEx-NNE_AER`.

1 Introduction

The past decade has witnessed great achievements in building web-scale knowledge graphs (KGs), *e.g.*, Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2015), and Google's Knowledge Vault (Dong et al., 2014). A typical KG is a multi-relational graph composed of entities as nodes and relations as different types of edges, where each edge is represented as a triple of the form (*head entity*, *relation*, *tail entity*). Such KGs contain rich structured knowledge, and have proven useful for many NLP tasks (Wasserman-Pritsker et al., 2015; Hoffmann et al., 2011; Yang and Mitchell, 2017).

Recently, the concept of *knowledge graph embedding* has been presented and quickly become a hot research topic. The key idea there is to embed components of a KG (*i.e.*, entities and relations) into a continuous vector space, so as to simplify manipulation while preserving the inherent structure of the KG. Early works on this topic learned such vectorial representations (*i.e.*, embeddings) via just simple models developed over KG triples (Bordes et al., 2011, 2013; Jenatton et al., 2012; Nickel et al., 2011). Recent attempts focused on either designing more complicated triple scoring models (Socher et al., 2013; Bordes et al., 2014; Wang et al., 2014; Lin et al., 2015b; Xiao et al., 2016; Nickel et al., 2016b; Trouillon et al., 2016; Liu et al., 2017), or incorporating extra information beyond KG triples (Chang et al., 2014; Zhong et al., 2015; Lin et al., 2015a; Neelakantan et al., 2015; Guo et al., 2015; Luo et al., 2015b; Xie et al., 2016a,b; Xiao et al., 2017). See (Wang et al., 2017) for a thorough review.

This paper, by contrast, investigates the potential of using very simple constraints to improve the KG embedding task. Specifically, we examine two types of constraints: (i) *non-negativity constraints* on entity representations and (ii) *approximate entailment constraints* over relation representations. By using the former, we learn compact representations for entities, which would naturally induce sparsity and interpretability (Murphy et al., 2012). By using the latter, we further encode regularities of logical entailment between relations into their

[*]Corresponding author: Quan Wang.

distributed representations, which might be advantageous to downstream tasks like link prediction and relation extraction (Rocktäschel et al., 2015; Guo et al., 2016). These constraints impose prior beliefs upon the structure of the embedding space, and will help us to learn more predictive embeddings, without significantly increasing the space or time complexity.

Our work has some similarities to those which integrate logical background knowledge into KG embedding (Rocktäschel et al., 2015; Wang et al., 2015; Guo et al., 2016, 2018). Most of such works, however, need grounding of first-order logic rules. The grounding process could be time and space inefficient especially for complicated rules. To avoid grounding, Demeester et al. (2016) tried to model rules using only relation representations. But their work creates vector representations for entity pairs rather than individual entities, and hence fails to handle unpaired entities. Moreover, it can only incorporate strict, hard rules which usually require extensive manual effort to create. Minervini et al. (2017b) proposed adversarial training which can integrate first-order logic rules without grounding. But their work, again, focuses on strict, hard rules. Minervini et al. (2017a) tried to handle uncertainty of rules. But their work assigns to different rules a same confidence level, and considers only equivalence and inversion of relations, which might not always be available in a given KG.

Our approach differs from the aforementioned works in that: (i) it imposes constraints directly on entity and relation representations without grounding, and can easily scale up to large KGs; (ii) the constraints, *i.e.*, non-negativity and approximate entailment derived automatically from statistical properties, are quite universal, requiring no manual effort and applicable to almost all KGs; (iii) it learns an individual representation for each entity, and can successfully make predictions between unpaired entities.

We evaluate our approach on publicly available KGs of WordNet, Freebase, and DBpedia as well. Experimental results indicate that our approach is simple yet surprisingly effective, achieving significant and consistent improvements over competitive baselines, but without negative impacts on efficiency or scalability. The non-negativity and approximate entailment constraints indeed improve model interpretability, resulting in a substantially increased structuring of the embedding space.

The remainder of this paper is organized as follows. We first review related work in Section 2, and then detail our approach in Section 3. Experiments and results are reported in Section 4, followed by concluding remarks in Section 5.

2 Related Work

Recent years have seen growing interest in learning distributed representations for entities and relations in KGs, a.k.a. KG embedding. Early works on this topic devised very simple models to learn such distributed representations, solely on the basis of triples observed in a given KG, *e.g.*, TransE which takes relations as translating operations between head and tail entities (Bordes et al., 2013), and RESCAL which models triples through bilinear operations over entity and relation representations (Nickel et al., 2011). Later attempts roughly fell into two groups: (i) those which tried to design more complicated triple scoring models, *e.g.*, the TransE extensions (Wang et al., 2014; Lin et al., 2015b; Ji et al., 2015), the RESCAL extensions (Yang et al., 2015; Nickel et al., 2016b; Trouillon et al., 2016; Liu et al., 2017), and the (deep) neural network models (Socher et al., 2013; Bordes et al., 2014; Shi and Weninger, 2017; Schlichtkrull et al., 2017; Dettmers et al., 2018); (ii) those which tried to integrate extra information beyond triples, *e.g.*, entity types (Guo et al., 2015; Xie et al., 2016b), relation paths (Neelakantan et al., 2015; Lin et al., 2015a), and textual descriptions (Xie et al., 2016a; Xiao et al., 2017). Please refer to (Nickel et al., 2016a; Wang et al., 2017) for a thorough review of these techniques. In this paper, we show the potential of using very simple constraints (*i.e.*, non-negativity constraints and approximate entailment constraints) to improve KG embedding, without significantly increasing the model complexity.

A line of research related to ours is KG embedding with logical background knowledge incorporated (Rocktäschel et al., 2015; Wang et al., 2015; Guo et al., 2016, 2018). But most of such works require grounding of first-order logic rules, which is time and space inefficient especially for complicated rules. To avoid grounding, Demeester et al. (2016) proposed lifted rule injection, and Minervini et al. (2017b) investigated adversarial training. Both works, however, can only handle strict, hard rules which usually require extensive effort to create. Minervini et al. (2017a) tried to handle uncertainty of background knowledge. But their work considers only equivalence and inversion between relations, which might not always be available in a given KG. Our approach, in contrast, imposes constraints directly on entity and relation representations without grounding. And the constraints used are quite universal, requiring no manual effort and applicable to almost all KGs.

Non-negativity has long been a subject studied in various research fields. Previous studies reveal that non-negativity could naturally induce sparsity and, in most cases, better interpretability (Lee and Seung, 1999). In many NLP-related tasks, non-negativity constraints are introduced to learn more interpretable word representations, which capture the notion of semantic composition (Murphy et al., 2012; Luo et al., 2015a; Fyshe et al., 2015). In this paper, we investigate the ability of non-negativity constraints to learn more accurate KG embeddings with good interpretability.

3 Our Approach

This section presents our approach. We first introduce a basic embedding technique to model triples in a given KG (§ 3.1). Then we discuss the non-negativity constraints over entity representations (§ 3.2) and the approximate entailment constraints over relation representations (§ 3.3). And finally we present the overall model (§ 3.4).

3.1 A Basic Embedding Model

We choose ComplEx (Trouillon et al., 2016) as our basic embedding model, since it is simple and efficient, achieving state-of-the-art predictive performance. Specifically, suppose we are given a KG containing a set of triples $\mathcal{O} = \{(e_i, r_k, e_j)\}$, with each triple composed of two entities $e_i, e_j \in \mathcal{E}$ and their relation $r_k \in \mathcal{R}$. Here $\mathcal{E}$ is the set of entities and $\mathcal{R}$ the set of relations. ComplEx then represents each entity $e \in \mathcal{E}$ as a complex-valued vector $\mathbf{e} \in \mathbb{C}^d$, and each relation $r \in \mathcal{R}$ a complex-valued vector $\mathbf{r} \in \mathbb{C}^d$, where d is the dimensionality of the embedding space. Each $\mathbf{x} \in \mathbb{C}^d$ consists of a real vector component $\text{Re}(\mathbf{x})$ and an imaginary vector component $\text{Im}(\mathbf{x})$, *i.e.*, $\mathbf{x} = \text{Re}(\mathbf{x}) + i\text{Im}(\mathbf{x})$. For any given triple $(e_i, r_k, e_j) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, a multi-linear dot product is used to score that triple, *i.e.*,

$$\begin{aligned}\phi(e_i, r_k, e_j) &\triangleq \text{Re}(\langle \mathbf{e}_i, \mathbf{r}_k, \bar{\mathbf{e}}_j \rangle) \\ &\triangleq \text{Re}(\textstyle\sum_\ell [\mathbf{e}_i]_\ell [\mathbf{r}_k]_\ell [\bar{\mathbf{e}}_j]_\ell), \end{aligned} \quad (1)$$

where $\mathbf{e}_i, \mathbf{r}_k, \mathbf{e}_j \in \mathbb{C}^d$ are the vectorial representations associated with e_i, r_k, e_j, respectively; $\bar{\mathbf{e}}_j$ is the conjugate of $\mathbf{e}_j$; $[\cdot]_\ell$ is the ℓ-th entry of a vector; and $\text{Re}(\cdot)$ means taking the real part of a complex value. Triples with higher $\phi(\cdot, \cdot, \cdot)$ scores are more likely to be true. Owing to the asymmetry of this scoring function, *i.e.*, $\phi(e_i, r_k, e_j) \neq \phi(e_j, r_k, e_i)$, ComplEx can effectively handle asymmetric relations (Trouillon et al., 2016).

3.2 Non-negativity of Entity Representations

On top of the basic ComplEx model, we further require entities to have non-negative (and bounded) vectorial representations. In fact, these distributed representations can be taken as feature vectors for entities, with latent semantics encoded in different dimensions. In ComplEx, as well as most (if not all) previous approaches, there is no limitation on the range of such feature values, which means that both positive and negative properties of an entity can be encoded in its representation. However, as pointed out by Murphy et al. (2012), it would be uneconomical to store all negative properties of an entity or a concept. For instance, to describe cats (a concept), people usually use positive properties such as cats are mammals, cats eat fishes, and cats have four legs, but hardly ever negative properties like cats are not vehicles, cats do not have wheels, or cats are not used for communication.

Based on such intuition, this paper proposes to impose non-negativity constraints on entity representations, by using which only positive properties will be stored in these representations. To better compare different entities on the same scale, we further require entity representations to stay within the hypercube of $[0, 1]^d$, as approximately Boolean embeddings (Kruszewski et al., 2015), *i.e.*,

$$\mathbf{0} \leq \text{Re}(\mathbf{e}), \text{Im}(\mathbf{e}) \leq \mathbf{1}, \quad \forall e \in \mathcal{E}, \quad (2)$$

where $\mathbf{e} \in \mathbb{C}^d$ is the representation for entity $e \in \mathcal{E}$, with its real and imaginary components denoted by $\text{Re}(\mathbf{e}), \text{Im}(\mathbf{e}) \in \mathbb{R}^d$; $\mathbf{0}$ and $\mathbf{1}$ are d-dimensional vectors with all their entries being 0 or 1; and $\geq, \leq, =$ denote the entry-wise comparisons throughout the paper whenever applicable. As shown by Lee and Seung (1999), non-negativity, in most cases, will further induce sparsity and interpretability.

3.3 Approximate Entailment for Relations

Besides the non-negativity constraints over entity representations, we also study approximate entailment constraints over relation representations. By approximate entailment, we mean an ordered pair of relations that the former approximately entails the latter, *e.g.*, `BornInCountry` and `Nationality`, stating that a person born in a country is very likely, but not necessarily, to have a nationality of that country. Each such relation pair is associated with a weight to indicate the confidence level of entailment. A larger weight stands for a higher level of confidence. We denote by $r_p \xrightarrow{\lambda} r_q$ the approximate entailment between relations r_p and r_q, with confidence level λ. This kind of entailment can be derived automatically from a KG by modern rule mining systems (Galárraga et al., 2015). Let $\mathcal{T}$ denote the set of all such approximate entailments derived beforehand.

Before diving into approximate entailment, we first explore the modeling of strict entailment, *i.e.*, entailment with infinite confidence level $\lambda = +\infty$. The strict entailment $r_p \rightarrow r_q$ states that if relation r_p holds then relation r_q must also hold. This entailment can be roughly modelled by requiring

$$\phi(e_i, r_p, e_j) \leq \phi(e_i, r_q, e_j), \quad \forall e_i, e_j \in \mathcal{E}, \tag{3}$$

where $\phi(\cdot,\cdot,\cdot)$ is the score for a triple predicted by the embedding model, defined by Eq. (1). Eq. (3) can be interpreted as follows: for any two entities e_i and e_j, if (e_i, r_p, e_j) is a true fact with a high score $\phi(e_i, r_p, e_j)$, then the triple (e_i, r_q, e_j) with an even higher score should also be predicted as a true fact by the embedding model. Note that given the non-negativity constraints defined by Eq. (2), a sufficient condition for Eq. (3) to hold, is to further impose

$$\mathrm{Re}(\mathbf{r}_p) \leq \mathrm{Re}(\mathbf{r}_q), \ \ \mathrm{Im}(\mathbf{r}_p) = \mathrm{Im}(\mathbf{r}_q), \tag{4}$$

where $\mathbf{r}_p$ and $\mathbf{r}_q$ are the complex-valued representations for r_p and r_q respectively, with the real and imaginary components denoted by $\mathrm{Re}(\cdot), \mathrm{Im}(\cdot) \in \mathbb{R}^d$. That means, when the constraints of Eq. (4) (along with those of Eq. (2)) are satisfied, the requirement of Eq. (3) (or in other words $r_p \rightarrow r_q$) will always hold. We provide a proof of sufficiency as supplementary material.

Next we examine the modeling of approximate entailment. To this end, we further introduce the confidence level λ and allow slackness in Eq. (4), which yields

$$\lambda\big(\mathrm{Re}(\mathbf{r}_p) - \mathrm{Re}(\mathbf{r}_q)\big) \leq \boldsymbol{\alpha}, \tag{5}$$

$$\lambda\big(\mathrm{Im}(\mathbf{r}_p) - \mathrm{Im}(\mathbf{r}_q)\big)^2 \leq \boldsymbol{\beta}. \tag{6}$$

Here $\boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}$ are slack variables, and $(\cdot)^2$ means an entry-wise operation. Entailments with higher confidence levels show less tolerance for violating the constraints. When $\lambda = +\infty$, Eqs. (5) – (6) degenerate to Eq. (4). The above analysis indicates that our approach can model entailment simply by imposing constraints over relation representations, without traversing all possible (e_i, e_j) entity pairs (*i.e.*, grounding). In addition, different confidence levels are encoded in the constraints, making our approach moderately tolerant of uncertainty.

3.4 The Overall Model

Finally, we combine together the basic embedding model of ComplEx, the non-negativity constraints on entity representations, and the approximate entailment constraints over relation representations. The overall model is presented as follows:

$$\begin{aligned} \min_{\Theta,\{\boldsymbol{\alpha},\boldsymbol{\beta}\}} \ & \sum_{\mathcal{D}^+\cup\mathcal{D}^-} \log\big(1 + \exp(-y_{ijk}\phi(e_i, r_k, e_j))\big) \\ & + \mu \sum\nolimits_{\mathcal{T}} \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\beta}) + \eta\|\Theta\|_2^2, \\ \text{s.t.} \ \ & \lambda\big(\mathrm{Re}(\mathbf{r}_p) - \mathrm{Re}(\mathbf{r}_q)\big) \leq \boldsymbol{\alpha}, \\ & \lambda\big(\mathrm{Im}(\mathbf{r}_p) - \mathrm{Im}(\mathbf{r}_q)\big)^2 \leq \boldsymbol{\beta}, \\ & \boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}, \quad \forall r_p \xrightarrow{\lambda} r_q \in \mathcal{T}, \\ & \mathbf{0} \leq \mathrm{Re}(\mathbf{e}), \mathrm{Im}(\mathbf{e}) \leq \mathbf{1}, \quad \forall e \in \mathcal{E}. \end{aligned} \tag{7}$$

Here, $\Theta \triangleq \{\mathbf{e} : e \in \mathcal{E}\} \cup \{\mathbf{r} : r \in \mathcal{R}\}$ is the set of all entity and relation representations; $\mathcal{D}^+$ and $\mathcal{D}^-$ are the sets of positive and negative training triples respectively; a positive triple is directly observed in the KG, *i.e.*, $(e_i, r_k, e_j) \in \mathcal{O}$; a negative triple can be generated by randomly corrupting the head or the tail entity of a positive triple, *i.e.*, (e_i', r_k, e_j) or (e_i, r_k, e_j'); $y_{ijk} = \pm 1$ is the label (positive or negative) of triple (e_i, r_k, e_j). In this optimization, the first term of the objective function is a typical logistic loss, which enforces triples to have scores close to their labels. The second term is the sum of slack variables in the approximate entailment constraints, with a penalty coefficient $\mu \geq 0$. The motivation is, although we allow slackness in those constraints we hope the total slackness to be small, so that the constraints can be better satisfied. The last term is L_2 regularization to avoid over-fitting, and $\eta \geq 0$ is the regularization coefficient.

To solve this optimization problem, the approximate entailment constraints (as well as the corresponding slack variables) are converted into penalty terms and added to the objective function, while the non-negativity constraints remain as they are. As such, the optimization problem of Eq. (7) can be rewritten as:

$$
\begin{aligned}
\min_{\Theta} \quad & \sum_{\mathcal{D}^+ \cup \mathcal{D}^-} \log\big(1+\exp(-y_{ijk}\phi(e_i, r_k, e_j))\big) \\
& + \mu \sum\nolimits_{\mathcal{T}} \lambda \mathbf{1}^\top \big[\mathrm{Re}(\mathbf{r}_p) - \mathrm{Re}(\mathbf{r}_q)\big]_+ \\
& + \mu \sum\nolimits_{\mathcal{T}} \lambda \mathbf{1}^\top \big(\mathrm{Im}(\mathbf{r}_p) - \mathrm{Im}(\mathbf{r}_q)\big)^2 + \eta \|\Theta\|_2^2, \\
\text{s.t.} \quad & \mathbf{0} \leq \mathrm{Re}(\mathbf{e}), \mathrm{Im}(\mathbf{e}) \leq \mathbf{1}, \quad \forall e \in \mathcal{E}, \qquad (8)
\end{aligned}
$$

where $[\mathbf{x}]_+ = \max(\mathbf{0}, \mathbf{x})$ with $\max(\cdot, \cdot)$ being an entry-wise operation. The equivalence between Eq. (7) and Eq. (8) is shown in the supplementary material. We use SGD in mini-batch mode as our optimizer, with AdaGrad (Duchi et al., 2011) to tune the learning rate. After each gradient descent step, we project (by truncation) real and imaginary components of entity representations into the hypercube of $[0, 1]^d$, to satisfy the non-negativity constraints.

While favouring a better structuring of the embedding space, imposing the additional constraints will not substantially increase model complexity. Our approach has a space complexity of $O(nd + md)$, which is the same as that of ComplEx. Here, n is the number of entities, m the number of relations, and $O(nd + md)$ to store a d-dimensional complex-valued vector for each entity and each relation. The time complexity (per iteration) of our approach is $O(sd+td+\bar{n}d)$, where s is the average number of triples in a mini-batch, $\bar{n}$ the average number of entities in a mini-batch, and t the total number of approximate entailments in $\mathcal{T}$. $O(sd)$ is to handle triples in a mini-batch, $O(td)$ penalty terms introduced by the approximate entailments, and $O(\bar{n}d)$ further the non-negativity constraints on entity representations. Usually there are much fewer entailments than triples, *i.e.*, $t \ll s$, and also $\bar{n} \leq 2s$.[1] So the time complexity of our approach is on a par with $O(sd)$, *i.e.*, the time complexity of ComplEx.

4 Experiments and Results

This section presents our experiments and results. We first introduce the datasets used in our experiments (§ 4.1). Then we empirically evaluate our approach in the link prediction task (§ 4.2). After that, we conduct extensive analysis on both entity representations (§ 4.3) and relation representations (§ 4.4) to show the interpretability of our model. Code and data used in the experiments are available at `https://github.com/iieir-km/ComplEx-NNE_AER`.

4.1 Datasets

The first two datasets we used are WN18 and FB15K, released by Bordes et al. (2013).[2] WN18 is a subset of WordNet containing 18 relations and 40,943 entities, and FB15K a subset of Freebase containing 1,345 relations and 14,951 entities. We create our third dataset from the mapping-based objects of core DBpedia.[3] We eliminate relations not included within the DBpedia ontology such as `HomePage` and `Logo`, and discard entities appearing less than 20 times. The final dataset, referred to as DB100K, is composed of 470 relations and 99,604 entities. Triples on each datasets are further divided into training, validation, and test sets, used for model training, hyperparameter tuning, and evaluation respectively. We follow the original split for WN18 and FB15K, and draw a split of 597,572/50,000/50,000 triples for DB100K.

We further use AMIE+ (Galárraga et al., 2015)[4] to extract approximate entailments automatically from the *training* set of each dataset. As suggested by Guo et al. (2018), we consider entailments with PCA confidence higher than 0.8.[5] As such, we extract 17 approximate entailments from WN18, 535 from FB15K, and 56 from DB100K. Table 1 gives some examples of these approximate entailments, along with their confidence levels. Table 2 further summarizes the statistics of the datasets.

4.2 Link Prediction

We first evaluate our approach in the link prediction task, which aims to predict a triple (e_i, r_k, e_j) with e_i or e_j missing, *i.e.*, predict e_i given (r_k, e_j) or predict e_j given (e_i, r_k).

Evaluation Protocol: We follow the protocol introduced by Bordes et al. (2013). For each test triple (e_i, r_k, e_j), we replace its head entity e_i with every entity $e_i' \in \mathcal{E}$, and calculate a score for the corrupted triple (e_i', r_k, e_j), *e.g.*, $\phi(e_i', r_k, e_j)$ defined by Eq. (1). Then we sort these scores in de-

[1]There will be at most $2s$ entities contained in s triples.

[2]`https://everest.hds.utc.fr/doku.php?id=en:smemlj12`

[3]`http://downloads.dbpedia.org/2016-10/core/`

[4]`https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/`

[5]PCA confidence is the confidence under the partial completeness assumption. See (Galárraga et al., 2015) for details.

hypernym^{-1} $\xrightarrow{1.00}$ hyponym
synset_domain_topic_of^{-1} $\xrightarrow{0.99}$ member_of_domain_topic
instance_hypernym^{-1} $\xrightarrow{0.98}$ instance_hyponym
/people/place_of_birth^{-1} $\xrightarrow{1.00}$ /location/people_born_here
/film/directed_by^{-1} $\xrightarrow{0.98}$ /director/film
/country/admin_divisions $\xrightarrow{0.91}$ /country/1st_level_divisions
owner $\xrightarrow{0.95}$ owning_company
child^{-1} $\xrightarrow{0.92}$ parent
distributing_company $\xrightarrow{0.92}$ distributing_label

Table 1: Approximate entailments extracted from WN18 (top), FB15K (middle), and DB100K (bottom), where r^{-1} means the inverse of relation r.

Dataset	# Ent	# Rel	# Train/Valid/Test			# Cons
WN18	40,943	18	141,442	5,000	5,000	17
FB15K	14,951	1,345	483,142	50,000	59,071	535
DB100K	99,604	470	597,572	50,000	50,000	56

Table 2: Statistics of datasets, where the columns respectively indicate the number of entities, relations, training/validation/test triples, and approximate entailments.

scending order, and get the rank of the correct entity e_i. During ranking, we remove corrupted triples that already exist in either the training, validation, or test set, *i.e.*, the *filtered* setting as described in (Bordes et al., 2013). This whole procedure is repeated while replacing the tail entity e_j. We report on the *test* set the mean reciprocal rank (MRR) and the proportion of correct entities ranked in the top n (HITS@N), with $n = 1, 3, 10$.

Comparison Settings: We compare the performance of our approach against a variety of KG embedding models developed in recent years. These models can be categorized into three groups:

- Simple embedding models that utilize triples alone without integrating extra information, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), HolE (Nickel et al., 2016b), ComplEx (Trouillon et al., 2016), and ANALOGY (Liu et al., 2017). Our approach is developed on the basis of ComplEx.

- Other extensions of ComplEx that integrate logical background knowledge in addition to triples, including RUGE (Guo et al., 2018) and ComplEx$^{\mathrm{R}}$ (Minervini et al., 2017a). The former requires grounding of first-order logic rules. The latter is restricted to relation equivalence and inversion, and assigns an identical confidence level to all different rules.

- Latest developments or implementations that achieve current state-of-the-art performance reported on the benchmarks of WN18 and FB15K, including R-GCN (Schlichtkrull et al., 2017), ConvE (Dettmers et al., 2018), and Single DistMult (Kadlec et al., 2017).[6] The first two are built based on neural network architectures, which are, by nature, more complicated than the simple models. The last one is a re-implementation of DistMult, generating 1000 to 2000 negative training examples per positive one, which leads to better performance but requires significantly longer training time.

We further evaluate our approach in two different settings: (i) ComplEx-NNE that imposes only the Non-Negativity constraints on Entity representations, *i.e.*, optimization Eq. (8) with $\mu = 0$; and (ii) ComplEx-NNE+AER that further imposes the Approximate Entailment constraints over Relation representations besides those non-negativity ones, *i.e.*, optimization Eq. (8) with $\mu > 0$.

Implementation Details: We compare our approach against all the three groups of baselines on the benchmarks of WN18 and FB15K. We directly report their original results on these two datasets to avoid re-implementation bias. On DB100K, the newly created dataset, we take the first two groups of baselines, *i.e.*, those simple embedding models and ComplEx extensions with logical background knowledge incorporated. We do not use the third group of baselines due to efficiency and complexity issues. We use the code provided by Trouillon et al. (2016)[7] for TransE, DistMult, and ComplEx, and the code released by their authors for ANALOGY[8] and RUGE[9]. We re-implement HolE and ComplEx$^{\mathrm{R}}$ so that all the baselines (as well as our approach) share the same optimization mode, *i.e.*, SGD with AdaGrad and gradient normalization, to facilitate a fair comparison.[10] We follow Trouillon et al. (2016) to adopt a ranking loss for TransE and a logistic loss for all the other methods.

[6] We do not consider Ensemble DistMult (Dettmers et al., 2018) which combines several different models together, to facilitate a fair comparison.

[7] https://github.com/ttrouill/complex

[8] https://github.com/quark0/ANALOGY

[9] https://github.com/iieir-km/RUGE

[10] An exception here is that ANALOGY uses asynchronous SGD with AdaGrad (Liu et al., 2017).

	WN18				FB15K			
		HITS@N				HITS@N		
	MRR	1	3	10	MRR	1	3	10
TransE (Bordes et al., 2013)	0.454	0.089	0.823	0.934	0.380	0.231	0.472	0.641
DistMult (Yang et al., 2015)	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824
HolE (Nickel et al., 2016b)	0.938	0.930	0.945	0.949	0.524	0.402	0.613	0.739
ComplEx (Trouillon et al., 2016)	0.941	0.936	0.945	0.947	0.692	0.599	0.759	0.840
ANALOGY (Liu et al., 2017)	0.942	0.939	0.944	0.947	0.725	0.646	0.785	0.854
RUGE (Guo et al., 2018)	—	—	—	—	0.768	0.703	0.815	0.865
ComplEx$^{\mathrm{R}}$ (Minervini et al., 2017a)	0.940	—	0.943	0.947	—	—	—	—
R-GCN (Schlichtkrull et al., 2017)	0.814	0.686	0.928	0.955	0.651	0.541	0.736	0.825
R-GCN+ (Schlichtkrull et al., 2017)	0.819	0.697	0.929	**0.964**	0.696	0.601	0.760	0.842
ConvE (Dettmers et al., 2018)	0.942	0.935	**0.947**	0.955	0.745	0.670	0.801	0.873
Single DistMult (Kadlec et al., 2017)	0.797	—	—	0.946	0.798	—	—	**0.893**
ComplEx-NNE (this work)	0.941	0.937	0.944	0.948	0.727*	0.659*	0.772*	0.845*
ComplEx-NNE+AER (this work)	**0.943**	**0.940**	0.945	0.948	**0.803***	**0.761***	**0.831***	0.874*

Table 3: Link prediction results on the test sets of WN18 and FB15K. Results for TransE and DistMult are taken from (Trouillon et al., 2016). Results for the other baselines are taken from the original papers. Missing scores not reported in the literature are indicated by "—". Best scores are highlighted in bold, and "$*$" indicates statistically significant improvements over ComplEx.

		HITS@N		
	MRR	1	3	10
TransE	0.111	0.016	0.164	0.270
DistMult	0.233	0.115	0.301	**0.448**
HolE	0.260	0.182	0.309	0.411
ComplEx	0.242	0.126	0.312	0.440
ANALOGY	0.252	0.143	0.323	0.427
RUGE	0.246	0.129	0.325	0.433
ComplEx$^{\mathrm{R}}$	0.253	0.167	0.294	0.420
ComplEx-NNE	0.298*	0.229*	0.330*	0.426
ComplEx-NNE+AER	**0.306***	**0.244***	**0.334***	0.418

Table 4: Link prediction results on the test set of DB100K, with best scores highlighted in bold, statistically significant improvements marked by "$*$".

Among those baselines, RUGE and ComplEx$^{\mathrm{R}}$ require additional logical background knowledge. RUGE makes use of soft rules, which are extracted by AMIE+ from the *training* sets. As suggested by Guo et al. (2018), length-1 and length-2 rules with PCA confidence higher than 0.8 are utilized. Note that our approach also makes use of AMIE+ rules with PCA confidence higher than 0.8. But it only considers entailments between a pair of relations, *i.e.*, length-1 rules. ComplEx$^{\mathrm{R}}$ takes into account equivalence and inversion between relations. We derive such axioms directly from our approximate entailments. If $r_p \xrightarrow{\lambda_1} r_q$ and $r_q \xrightarrow{\lambda_2} r_p$ with $\lambda_1, \lambda_2 > 0.8$, we think relations r_p and r_q are equivalent. And similarly, if $r_p^{-1} \xrightarrow{\lambda_1} r_q$ and $r_q^{-1} \xrightarrow{\lambda_2} r_p$ with $\lambda_1, \lambda_2 > 0.8$, we consider r_p as an inverse of r_q.

For all the methods, we create 100 mini-batches on each dataset, and conduct a grid search to find hyperparameters that maximize MRR on the validation set, with at most 1000 iterations over the training set. Specifically, we tune the embedding size $d \in \{100, 150, 200\}$, the L_2 regularization coefficient $\eta \in \{0.001, 0.003, 0.01, 0.03, 0.1\}$, the ratio of negative over positive training examples $\alpha \in \{2, 10\}$, and the initial learning rate $\gamma \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$. For TransE, we tune the margin of the ranking loss $\delta \in \{0.1, 0.2, 0.5, 1, 2, 5, 10\}$. Other hyperparameters of ANALOGY and RUGE are set or tuned according to the default settings suggested by their authors (Liu et al., 2017; Guo et al., 2018). After getting the best ComplEx model, we tune the relation constraint penalty of our approach ComplEx-NNE+AER (μ in Eq. (8)) in the range of $\{10^{-5}, 10^{-4}, \cdots, 10^4, 10^5\}$, with all its other hyperparameters fixed to their optimal configurations. We then directly set $\mu = 0$ to get the optimal ComplEx-NNE model. The weight of soft constraints in ComplEx$^{\mathrm{R}}$ is tuned in the same range as μ. The optimal configurations for our approach are: $d = 200$, $\eta = 0.03$, $\alpha = 10$, $\gamma = 1.0$, $\mu = 10$ on WN18; $d = 200$, $\eta{=}0.01$, $\alpha{=}10$, $\gamma = 0.5$, $\mu = 10^{-3}$ on FB15K; and $d = 150$, $\eta = 0.03$, $\alpha = 10$, $\gamma = 0.1$, $\mu = 10^{-5}$ on DB100K.

Experimental Results: Table 3 presents the results on the test sets of WN18 and FB15K, where the results for the baselines are taken directly from previous literature. Table 4 further provides the results on the test set of DB100K, with all the methods tuned and tested in (almost) the same setting. On all the datasets, we test statistical significance of the improvements achieved by ComplEx-NNE/ ComplEx-NNE+AER over ComplEx, by using a paired t-test. The reciprocal rank or HITS@N value with $n = 1, 3, 10$ for each test triple is used as paired data. The symbol "$*$" indicates a significance level of $p < 0.05$.

The results demonstrate that imposing the non-negativity and approximate entailment constraints indeed improves KG embedding. ComplEx-NNE and ComplEx-NNE+AER perform better than (or at least equally well as) ComplEx in almost all the metrics on all the three datasets, and most of the improvements are statistically significant (except those on WN18). More interestingly, just by introducing these simple constraints, ComplEx-NNE+ AER can beat very strong baselines, including the best performing basic models like ANALOGY, those previous extensions of ComplEx like RUGE or ComplEx$^{\mathrm{R}}$, and even the complicated developments or implementations like ConvE or Single DistMult. This demonstrates the superiority of our approach.

4.3 Analysis on Entity Representations

This section inspects how the structure of the entity embedding space changes when the constraints are imposed. We first provide the visualization of entity representations on DB100K. On this dataset each entity is associated with a single type label.[11] We pick 4 types `reptile`, `wine_region`, `species`, and `programming_language`, and randomly select 30 entities from each type. Figure 1 visualizes the representations of these entities learned by ComplEx and ComplEx-NNE+AER (real components only), with the optimal configurations determined by link prediction (see § 4.2 for details, applicable to all analysis hereafter). During the visualization, we normalize the real component of each entity by $[\tilde{\mathbf{x}}]_\ell = \frac{[\mathbf{x}]_\ell - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$, where $\min(\mathbf{x})$ or $\max(\mathbf{x})$ is the minimum or maximum entry of $\mathbf{x}$ respectively. We observe that after imposing the non-negativity constraints, ComplEx-NNE+AER indeed obtains compact and interpretable representations for entities. Each entity is represented by only a relatively small number of "active" dimensions. And entities

[11] `http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_wkd_uris_en.ttl.bz2`

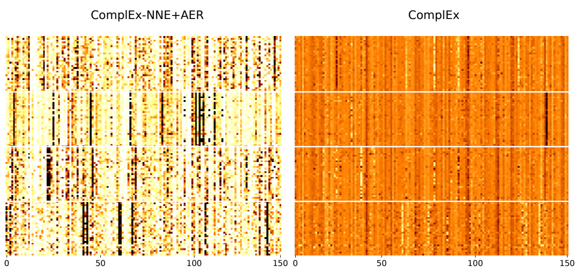


Figure 1: Visualization of real components of entity representations (rows) learned by ComplEx-NNE+AER (left) and ComplEx (right). From top to bottom, entities belong to type `reptile`, `wine_region`, `species`, and `programming_language` in turn. Values range from 0 (white) via 0.5 (orange) to 1 (black). Best viewed in color.

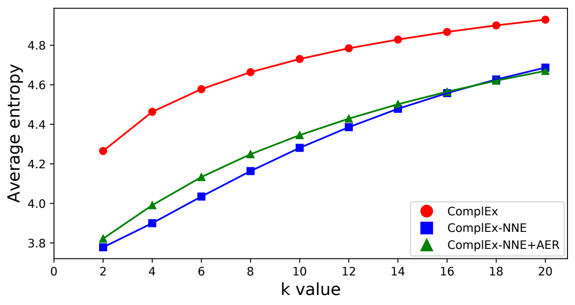


Figure 2: Average entropy over all dimensions of real components of entity representations learned by ComplEx (circles), ComplEx-NNE (squares), and ComplEx-NNE+AER (triangles) as K varies.

with the same type tend to activate the same set of dimensions, while entities with different types often get clearly different dimensions activated.

Then we investigate the semantic purity of these dimensions. Specifically, we collect the representations of all the entities on DB100K (real components only). For each dimension of these representations, top K percent of entities with the highest activation values on this dimension are picked. We can calculate the entropy of the type distribution of the entities selected. This entropy reflects diversity of entity types, or in other words, semantic purity. If all the K percent of entities have the same type, we will get the lowest entropy of zero (the highest semantic purity). On the contrary, if each of them has a distinct type, we will get the highest entropy (the lowest semantic purity). Figure 2 shows the average entropy over all dimensions of entity representations (real components only) learned by ComplEx, ComplEx-NNE, and ComplEx-NNE+

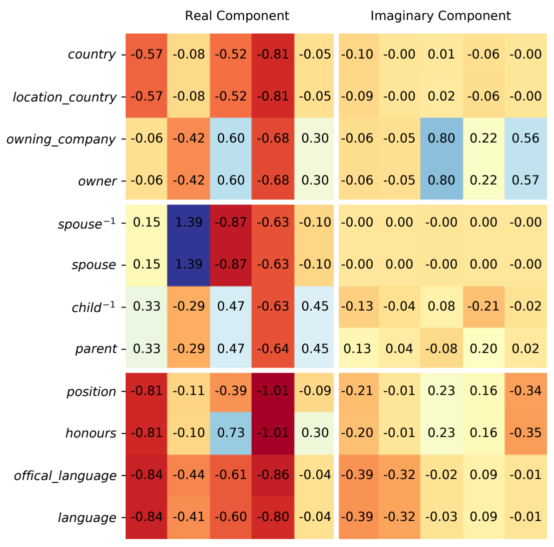


Figure 3: Visualization of relation representations learned by ComplEx-NNE+AER, with the top 4 relations from the equivalence class, the middle 4 the inversion class, and the bottom 4 others.

AER, as K varies. We can see that after imposing the non-negativity constraints, ComplEx-NNE and ComplEx-NNE+AER can learn entity representations with latent dimensions of consistently higher semantic purity. We have conducted the same analyses on imaginary components of entity representations, and observed similar phenomena. The results are given as supplementary material.

4.4 Analysis on Relation Representations

This section further provides a visual inspection of the relation embedding space when the constraints are imposed. To this end, we group relation pairs involved in the DB100K entailment constraints into 3 classes: equivalence, inversion, and others.[12] We choose 2 pairs of relations from each class, and visualize these relation representations learned by ComplEx-NNE+AER in Figure 3, where for each relation we randomly pick 5 dimensions from both its real and imaginary components. By imposing the approximate entailment constraints, these relation representations can encode logical regularities quite well. Pairs of relations from the first class (equivalence) tend to have identical representations $\mathbf{r}_p \approx \mathbf{r}_q$, those from the second class (inversion) complex conjugate representations $\mathbf{r}_p \approx \bar{\mathbf{r}}_q$; and the others representations that $\mathrm{Re}(\mathbf{r}_p) \leq \mathrm{Re}(\mathbf{r}_q)$ and $\mathrm{Im}(\mathbf{r}_p) \approx \mathrm{Im}(\mathbf{r}_q)$.

[12]Equivalence and inversion are detected using heuristics introduced in § 4.2 (implementation details). See the supplementary material for detailed properties of these three classes.

5 Conclusion

This paper investigates the potential of using very simple constraints to improve KG embedding. Two types of constraints have been studied: (i) the non-negativity constraints to learn compact, interpretable entity representations, and (ii) the approximate entailment constraints to further encode logical regularities into relation representations. Such constraints impose prior beliefs upon the structure of the embedding space, and will not significantly increase the space or time complexity. Experimental results on benchmark KGs demonstrate that our method is simple yet surprisingly effective, showing significant and consistent improvements over strong baselines. The constraints indeed improve model interpretability, yielding a substantially increased structuring of the embedding space.

Acknowledgments

We would like to thank all the anonymous reviewers for their insightful and valuable suggestions, which help to improve the quality of this paper. This work is supported by the National Key Research and Development Program of China (No. 2016QY03D0503) and the Fundamental Theory and Cutting Edge Technology Research Program of the Institute of Information Engineering, Chinese Academy of Sciences (No. Y7Z0261101).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. pages 1247–1250.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. pages 301–306.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1568–1579.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted rule injection for relation embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1389–1399.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. pages 1811–1818.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 601–610.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Alona Fyshe, Leila Wehbe, Partha P. Talukdar, Brian Murphy, and Tom M. Mitchell. 2015. A compositional and interpretable semantic space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 32–41.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal* 24(6):707–730.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 84–94.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 192–202.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. pages 4816–4823.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. pages 541–550.

Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R. Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*. pages 3167–3175.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via a dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 687–696.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 69–74.

German Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving Boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics* 3:375–388.

Daniel D. Lee and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401:788–791.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2015. DBpedia: A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal* 6(2):167–195.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. pages 2181–2187.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. pages 2168–2178.

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015a. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1687–1692.

Yuanfei Luo, Quan Wang, Bin Wang, and Li Guo. 2015b. Context-dependent knowledge graph embedding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1656–1661.

Pasquale Minervini, Luca Costabello, Emir Muñoz, Vít Nováček, and Pierre-Yves Vandenbussche. 2017a. Regularizing knowledge graph embeddings via equivalence and inversion axioms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. pages 668–683.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017b. Adversarial sets for regularising neural link predictors. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*. pages 1933–1950.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 156–166.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*. pages 809–816.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1119–1129.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv:1703.06103* .

Baoxu Shi and Tim Weninger. 2017. ProjE: Embedding projection for knowledge graph completion. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. pages 1236–1242.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*. pages 926–934.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*. pages 2071–2080.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29(12):2724–2743.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. pages 1859–1865.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. pages 1112–1119.

Evgenia Wasserman-Pritsker, William W. Cohen, and Einat Minkov. 2015. Learning to identify the best contexts for knowledge-based WSD. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1662–1667.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 2316–2325.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2017. SSP: Semantic space projection for knowledge graph embedding with text descriptions. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. pages 3104–3110.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016a. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. pages 2659–2665.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016b. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. pages 2965–2971.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. pages 1436–1446.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations*.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 267–272.

Towards Understanding the Geometry of Knowledge Graph Embeddings

Chandrahas
Indian Institute of Science
chandrahas@iisc.ac.in

Aditya Sharma
Indian Institute of Science
adityasharma@iisc.ac.in

Partha Talukdar
Indian Institute of Science
ppt@iisc.ac.in

Abstract

Knowledge Graph (KG) embedding has emerged as a very active area of research over the last few years, resulting in the development of several embedding methods. These KG embedding methods represent KG entities and relations as vectors in a high-dimensional space. Despite this popularity and effectiveness of KG embeddings in various tasks (e.g., link prediction), geometric understanding of such embeddings (i.e., arrangement of entity and relation vectors in vector space) is unexplored – we fill this gap in the paper. We initiate a study to analyze the geometry of KG embeddings and correlate it with task performance and other hyperparameters. To the best of our knowledge, this is the first study of its kind. Through extensive experiments on real-world datasets, we discover several insights. For example, we find that there are sharp differences between the geometry of embeddings learnt by different classes of KG embeddings methods. We hope that this initial study will inspire other follow-up research on this important but unexplored problem.

1 Introduction

Knowledge Graphs (KGs) are multi-relational graphs where nodes represent entities and typed-edges represent relationships among entities. Recent research in this area has resulted in the development of several large KGs, such as NELL (Mitchell et al., 2015), YAGO (Suchanek et al., 2007), and Freebase (Bollacker et al., 2008), among others. These KGs contain thousands of predicates (e.g., *person*, *city*, *mayorOf(person, city)*, etc.), and millions of triples involving such predicates, e.g., *(Bill de Blasio, mayorOf, New York City)*.

The problem of learning embeddings for Knowledge Graphs has received significant attention in recent years, with several methods being proposed (Bordes et al., 2013; Lin et al., 2015; Nguyen et al., 2016; Nickel et al., 2016; Trouillon et al., 2016). These methods represent entities and relations in a KG as vectors in high dimensional space. These vectors can then be used for various tasks, such as, link prediction, entity classification etc. Starting with TransE (Bordes et al., 2013), there have been many KG embedding methods such as TransH (Wang et al., 2014), TransR (Lin et al., 2015) and STransE (Nguyen et al., 2016) which represent relations as translation vectors from head entities to tail entities. These are *additive models*, as the vectors interact via addition and subtraction. Other KG embedding models, such as, DistMult (Yang et al., 2014), HolE (Nickel et al., 2016), and ComplEx (Trouillon et al., 2016) are *multiplicative* where entity-relation-entity triple likelihood is quantified by a multiplicative score function. All these methods employ a score function for distinguishing correct triples from incorrect ones.

In spite of the existence of many KG embedding methods, our understanding of the geometry and structure of such embeddings is very shallow. A recent work (Mimno and Thompson, 2017) analyzed the geometry of word embeddings. However, the problem of analyzing geometry of KG embeddings is still unexplored – we fill this important gap. In this paper, we analyze the geometry of such vectors in terms of their lengths and conicity, which, as defined in Section 4, describes their positions and orientations in the vector space. We later study the effects of model type and training hyperparameters on the geometry of KG embeddings and correlate geometry with performance.

We make the following contributions:

- We initiate a study to analyze the geometry of various Knowledge Graph (KG) embeddings. To the best of our knowledge, this is the first study of its kind. We also formalize various metrics which can be used to study geometry of a set of vectors.
- Through extensive analysis, we discover several interesting insights about the geometry of KG embeddings. For example, we find systematic differences between the geometries of embeddings learned by additive and multiplicative KG embedding methods.
- We also study the relationship between geometric attributes and predictive performance of the embeddings, resulting in several new insights. For example, in case of multiplicative models, we observe that for entity vectors generated with a fixed number of negative samples, lower conicity (as defined in Section 4) or higher average vector length lead to higher performance.

Source code of all the analysis tools developed as part of this paper is available at `https://github.com/malllabiisc/kg-geometry`. We are hoping that these resources will enable one to quickly analyze the geometry of any KG embedding, and potentially other embeddings as well.

2 Related Work

In spite of the extensive and growing literature on both KG and non-KG embedding methods, very little attention has been paid towards understanding the geometry of the learned embeddings. A recent work (Mimno and Thompson, 2017) is an exception to this which addresses this problem in the context of word vectors. This work revealed a surprising correlation between word vector geometry and the number of negative samples used during training. Instead of word vectors, in this paper we focus on understanding the geometry of KG embeddings. In spite of this difference, the insights we discover in this paper generalizes some of the observations in the work of (Mimno and Thompson, 2017). Please see Section 6.2 for more details.

Since KGs contain only positive triples, negative sampling has been used for training KG embeddings. Effect of the number of negative samples in KG embedding performance was studied by (Toutanova et al., 2015). In this paper, we study the effect of the number of negative samples on KG embedding geometry as well as performance.

In addition to the additive and multiplicative KG embedding methods already mentioned in Section 1, there is another set of methods where the entity and relation vectors interact via a neural network. Examples of methods in this category include NTN (Socher et al., 2013), CONV (Toutanova et al., 2015), ConvE (Dettmers et al., 2017), R-GCN (Schlichtkrull et al., 2017), ER-MLP (Dong et al., 2014) and ER-MLP-2n (Ravishankar et al., 2017). Due to space limitations, in this paper we restrict our scope to the analysis of the geometry of additive and multiplicative KG embedding models only, and leave the analysis of the geometry of neural network-based methods as part of future work.

3 Overview of KG Embedding Methods

For our analysis, we consider six representative KG embedding methods: TransE (Bordes et al., 2013), TransR (Lin et al., 2015), STransE (Nguyen et al., 2016), DistMult (Yang et al., 2014), HolE (Nickel et al., 2016) and ComplEx (Trouillon et al., 2016). We refer to TransE, TransR and STransE as *additive* methods because they learn embeddings by modeling relations as translation vectors from one entity to another, which results in vectors interacting via the addition operation during training. On the other hand, we refer to DistMult, HolE and ComplEx as *multiplicative* methods as they quantify the likelihood of a triple belonging to the KG through a multiplicative score function. The score functions optimized by these methods are summarized in Table 1.

Notation: Let $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph (KG) where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{T} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of triples stored in the graph. Most of the KG embedding methods learn vectors $\mathbf{e} \in \mathbb{R}^{d_e}$ for $e \in \mathcal{E}$, and $\mathbf{r} \in \mathbb{R}^{d_r}$ for $r \in \mathcal{R}$. Some methods also learn projection matrices $M_r \in \mathbb{R}^{d_r \times d_e}$ for relations. The correctness of a triple is evaluated using a model specific score function $\sigma : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$. For learning the embeddings, a loss function $\mathcal{L}(\mathcal{T}, \mathcal{T}'; \theta)$, defined over a set of positive triples $\mathcal{T}$, set of (sampled) negative triples $\mathcal{T}'$, and the parameters θ is optimized.

We use small italics characters (e.g., h, r) to represent entities and relations, and correspond-

Type	Model	Score Function $\sigma(h,r,t)$
Additive	TransE (Bordes et al., 2013)	$-\left\|\mathbf{h}+\mathbf{r}-\mathbf{t}\right\|_1$
	TransR (Lin et al., 2015)	$-\left\|M_r\mathbf{h}+\mathbf{r}-M_r\mathbf{t}\right\|_1$
	STransE (Nguyen et al., 2016)	$-\left\|M_r^1\mathbf{h}+\mathbf{r}-M_r^2\mathbf{t}\right\|_1$
Multiplicative	DistMult (Yang et al., 2014)	$\mathbf{r}^\top(\mathbf{h}\odot\mathbf{t})$
	HolE (Nickel et al., 2016)	$\mathbf{r}^\top(\mathbf{h}\star\mathbf{t})$
	ComplEx (Trouillon et al., 2016)	$\mathbf{Re}(\mathbf{r}^\top(\mathbf{h}\odot\bar{\mathbf{t}}))$

Table 1: Summary of various Knowledge Graph (KG) embedding methods used in the paper. Please see Section 3 for more details.

ing bold characters to represent their vector embeddings (e.g., $\mathbf{h}$, $\mathbf{r}$). We use bold capitalization (e.g., $\mathbf{V}$) to represent a set of vectors. Matrices are represented by capital italics characters (e.g., M).

3.1 Additive KG Embedding Methods

This is the set of methods where entity and relation vectors interact via additive operations. The score function for these models can be expressed as below

$$\sigma(h,r,t) = -\left\|M_r^1\mathbf{h}+\mathbf{r}-M_r^2\mathbf{t}\right\|_1 \qquad (1)$$

where $\mathbf{h},\mathbf{t} \in \mathbb{R}^{d_e}$ and $\mathbf{r} \in \mathbb{R}^{d_r}$ are vectors for head entity, tail entity and relation respectively. $M_r^1, M_r^2 \in \mathbb{R}^{d_r \times d_e}$ are projection matrices from entity space $\mathbb{R}^{d_e}$ to relation space $\mathbb{R}^{d_r}$.

TransE (Bordes et al., 2013) is the simplest additive model where the entity and relation vectors lie in same $d-$dimensional space, i.e., $d_e = d_r = d$. The projection matrices $M_r^1 = M_r^2 = \mathcal{I}_d$ are identity matrices. The relation vectors are modeled as translation vectors from head entity vectors to tail entity vectors. Pairwise ranking loss is then used to learn these vectors. Since the model is simple, it has limited capability in capturing many-to-one, one-to-many and many-to-many relations.

TransR (Lin et al., 2015) is another translation-based model which uses separate spaces for entity and relation vectors allowing it to address the shortcomings of TransE. Entity vectors are projected into a relation specific space using the corresponding projection matrix $M_r^1 = M_r^2 = M_r$. The training is similar to TransE.

STransE (Nguyen et al., 2016) is a generalization of TransR and uses different projection matrices for head and tail entity vectors. The training is similar to TransE. STransE achieves better performance than the previous methods but at the cost of more number of parameters.

Equation 1 is the score function used in STransE. TransE and TransR are special cases of STransE with $M_r^1 = M_r^2 = \mathcal{I}_d$ and $M_r^1 = M_r^2 = M_r$, respectively.

3.2 Multiplicative KG Embedding Methods

This is the set of methods where the vectors interact via multiplicative operations (usually dot product). The score function for these models can be expressed as

$$\sigma(h,r,t) = \mathbf{r}^\top f(\mathbf{h},\mathbf{t}) \qquad (2)$$

where $\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{F}^d$ are vectors for head entity, tail entity and relation respectively. $f(\mathbf{h},\mathbf{t}) \in \mathbb{F}^d$ measures compatibility of head and tail entities and is specific to the model. $\mathbb{F}$ is either real space $\mathbb{R}$ or complex space $\mathbb{C}$. Detailed descriptions of the models we consider are as follows.

DistMult (Yang et al., 2014) models entities and relations as vectors in $\mathbb{R}^d$. It uses an entry-wise product ($\odot$) to measure compatibility between head and tail entities, while using logistic loss for training the model.

$$\sigma_{DistMult}(h,r,t) = \mathbf{r}^\top(\mathbf{h}\odot\mathbf{t}) \qquad (3)$$

Since the entry-wise product in (3) is symmetric, DistMult is not suitable for asymmetric and anti-symmetric relations.

HolE (Nickel et al., 2016) also models entities and relations as vectors in $\mathbb{R}^d$. It uses circular correlation operator ($\star$) as compatibility function defined as

$$[\mathbf{h}\star\mathbf{t}]_k = \sum_{i=0}^{d-1} \mathbf{h}_i\mathbf{t}_{(k+i) \bmod d}$$

The score function is given as

$$\sigma_{HolE}(h,r,t) = \mathbf{r}^\top(\mathbf{h}\star\mathbf{t}) \qquad (4)$$

The circular correlation operator being asymmetric, can capture asymmetric and anti-symmetric relations, but at the cost of higher time complexity

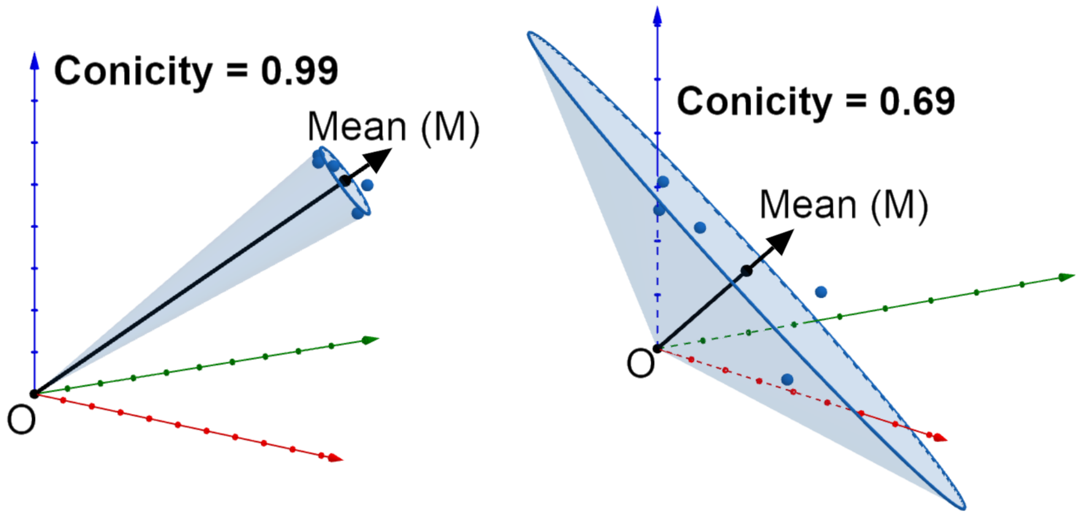


Figure 1: Comparison of high vs low Conicity. Randomly generated vectors are shown in blue with their sample mean vector M in black. Figure on the left shows the case when vectors lie in narrow cone resulting in high Conicity value. Figure on the right shows the case when vectors are spread out having relatively lower Conicity value. We skipped very low values of Conicity as it was difficult to visualize. The points are sampled from 3d Spherical Gaussian with mean (1,1,1) and standard deviation 0.1 (left) and 1.3 (right). Please refer to Section 4 for more details.

($\mathcal{O}(d \log d)$). For training, we use pairwise ranking loss.

ComplEx (Trouillon et al., 2016) represents entities and relations as vectors in $\mathbb{C}^d$. The compatibility of entity pairs is measured using entry-wise product between head and complex conjugate of tail entity vectors.

$$\sigma_{ComplEx}(h, r, t) = \mathbf{Re}(\mathbf{r}^\top(\mathbf{h} \odot \bar{\mathbf{t}})) \qquad (5)$$

In contrast to (3), using complex vectors in (5) allows ComplEx to handle symmetric, asymmetric and anti-symmetric relations using the same score function. Similar to DistMult, logistic loss is used for training the model.

4 Metrics

For our geometrical analysis, we first define a term **'alignment to mean'** (**ATM**) of a vector $\mathbf{v}$ belonging to a set of vectors $\mathbf{V}$, as the cosine similarity[1] between $\mathbf{v}$ and the mean of all vectors in $\mathbf{V}$.

$$\text{ATM}(\mathbf{v}, \mathbf{V}) = \text{cosine}\left(\mathbf{v}, \frac{1}{|\mathbf{V}|}\sum_{\mathbf{x}\in\mathbf{V}} \mathbf{x}\right)$$

We also define '**conicity**' of a set $\mathbf{V}$ as the mean ATM of all vectors in $\mathbf{V}$.

$$\text{Conicity}(\mathbf{V}) = \frac{1}{|\mathbf{V}|}\sum_{\mathbf{v}\in\mathbf{V}} \text{ATM}(\mathbf{v}, \mathbf{V})$$

[1]cosine$(u, v) = \frac{u^\top v}{\|u\|\|v\|}$

Dataset		FB15k	WN18
#Relations		1,345	18
#Entities		14,541	40,943
#Triples	Train	483,142	141,440
	Validation	50,000	5,000
	Test	59,071	5,000

Table 2: Summary of datasets used in the paper.

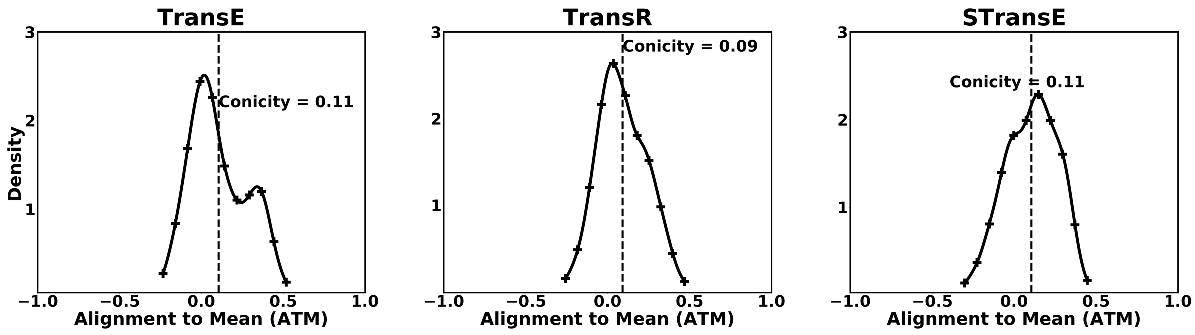
By this definition, a high value of Conicity($\mathbf{V}$) would imply that the vectors in $\mathbf{V}$ lie in a narrow cone centered at origin. In other words, the vectors in the set $\mathbf{V}$ are highly aligned with each other. In addition to that, we define the variance of ATM across all vectors in $\mathbf{V}$, as the '**vector spread**'(VS) of set $\mathbf{V}$,

$$\text{VS}(\mathbf{V}) = \frac{1}{|\mathbf{V}|}\sum_{\mathbf{v}\in\mathbf{V}} \Big(\text{ATM}(\mathbf{v}, \mathbf{V}) - \text{Conicity}(\mathbf{V})\Big)^2$$

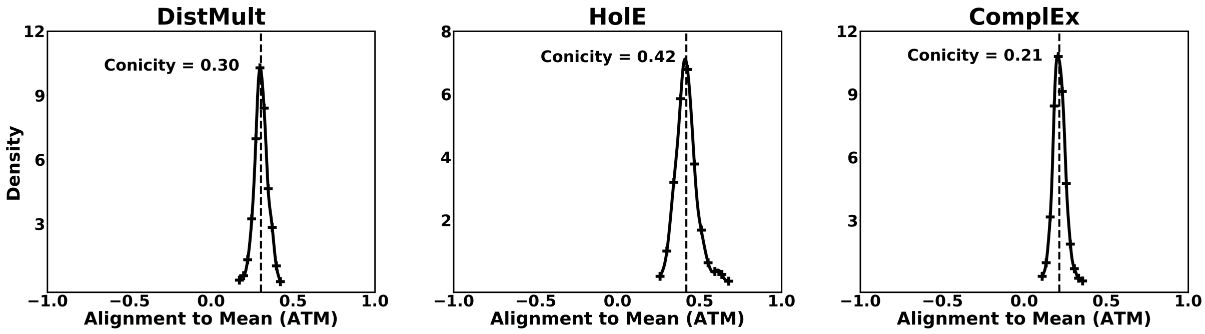
Figure 1 visually demonstrates these metrics for randomly generated 3-dimensional points. The left figure shows high Conicity and low vector spread while the right figure shows low Conicity and high vector spread.

We define the length of a vector $\mathbf{v}$ as L_2-norm of the vector $\|\mathbf{v}\|_2$ and **'average vector length'** (**AVL**) for the set of vectors $\mathbf{V}$ as

$$\text{AVL}(\mathbf{V}) = \frac{1}{|\mathbf{V}|}\sum_{\mathbf{v}\in\mathbf{V}} \|\mathbf{v}\|_2$$



(a) Additive Models



(b) Multiplicative Models

Figure 2: Alignment to Mean (ATM) vs Density plots for entity embeddings learned by various additive (top row) and multiplicative (bottom row) KG embedding methods. For each method, a plot averaged across entity frequency bins is shown. From these plots, we conclude that entity embeddings from additive models tend to have low (positive as well as negative) ATM and thereby low Conicity and high vector spread. Interestingly, this is reversed in case of multiplicative methods. Please see Section 6.1 for more details.

5 Experimental Setup

Datasets: We run our experiments on subsets of two widely used datasets, viz., Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995), called FB15k and WN18 (Bordes et al., 2013), respectively. We detail the characteristics of these datasets in Table 2.

Please note that while the results presented in Section 6 are on the FB15K dataset, we reach the same conclusions on WN18. The plots for our experiments on WN18 can be found in the Supplementary Section.

Hyperparameters: We experiment with multiple values of hyperparameters to understand their effect on the geometry of KG embeddings. Specifically, we vary the dimension of the generated vectors between $\{50, 100, 200\}$ and the number of negative samples used during training between $\{1, 50, 100\}$. For more details on algorithm specific hyperparameters, we refer the reader to the Supplementary Section.[2]

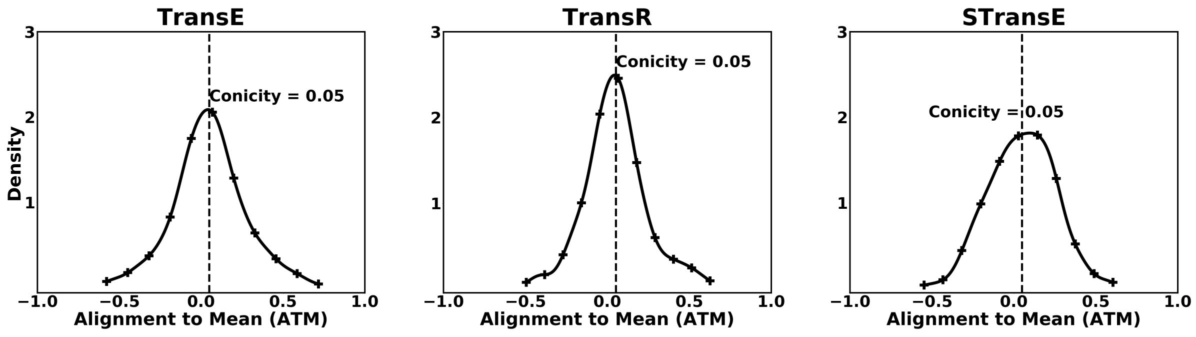
Frequency Bins: We follow (Mimno and Thompson, 2017) for entity and relation samples used in the analysis. Multiple bins of entities and relations are created based on their frequencies and 100 randomly sampled vectors are taken from each bin. These set of sampled vectors are then used for our analysis. For more information about sampling vectors, please refer to (Mimno and Thompson, 2017).

6 Results and Analysis

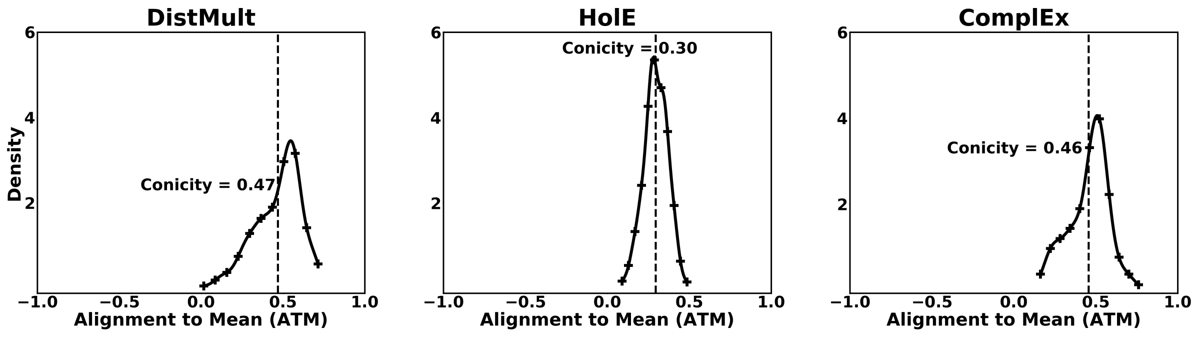
In this section, we evaluate the following questions.

- Does model type (e.g., additive vs multiplicative) have any effect on the geometry of embeddings? (Section 6.1)

[2]For training, we used codes from `https://github.com/Mrlyk423/Relation_Extraction` (TransE, TransR), `https://github.com/datquocnguyen/STransE` (STransE), `https://github.com/mnick/holographic-embeddings` (HolE) and `https://github.com/ttrouill/complex` (ComplEx and DistMult).



(a) Additive Models



(b) Multiplicative Models

Figure 3: Alignment to Mean (ATM) vs Density plots for relation embeddings learned by various additive (top row) and multiplicative (bottom row) KG embedding methods. For each method, a plot averaged across entity frequency bins is shown. Trends in these plots are similar to those in Figure 2. Main findings from these plots are summarized in Section 6.1.

- Does negative sampling have any effect on the embedding geometry? (Section 6.2)
- Does the dimension of embedding have any effect on its geometry? (Section 6.3)
- How is task performance related to embedding geometry? (Section 6.4)

In each subsection, we summarize the main findings at the beginning, followed by evidence supporting those findings.

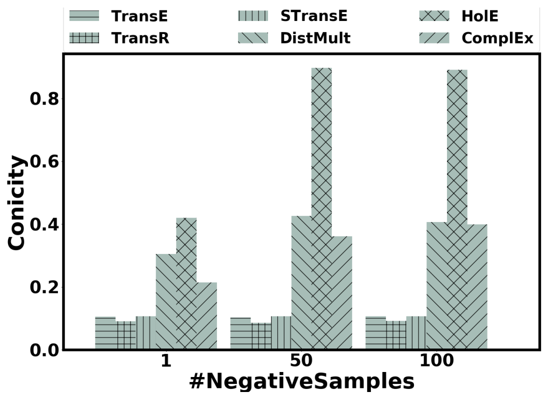
6.1 Effect of Model Type on Geometry

Summary of Findings:
Additive: Low conicity and high vector spread.
Multiplicative: High conicity and low vector spread.

In this section, we explore whether the type of the score function optimized during the training has any effect on the geometry of the resulting embedding. For this experiment, we set the number of negative samples to 1 and the vector dimension to 100 (we got similar results for 50-dimensional vectors). Figure 2 and Figure 3 show the distribution of ATMs of these sampled entity and relation vectors, respectively.[3]

Entity Embeddings: As seen in Figure 2, there is a stark difference between the geometries of entity vectors produced by additive and multiplicative models. The ATMs of all entity vectors produced by multiplicative models are positive with very low vector spread. Their high conicity suggests that they are not uniformly dispersed in the vector space, but lie in a narrow cone along the mean vector. This is in contrast to the entity vectors obtained from additive models which are both positive and negative with higher vector spread. From the lower values of conicity, we conclude that entity vectors from additive models are evenly dispersed in the vector space. This observation is also reinforced by looking at the high vector spread of additive models in comparison to that of multiplicative models. We also observed that additive models are sensitive to the frequency of entities, with high frequency bins having higher conicity than low frequency bins. However, no such pattern was observed for multiplicative models and

[3]We also tried using the *global* mean instead of mean of the sampled set for calculating cosine similarity in ATM, and got very similar results.



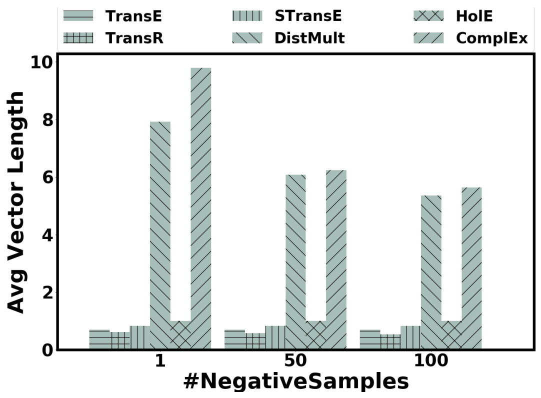


Figure 4: Conicity (left) and Average Vector Length (right) vs Number of negative samples for entity vectors learned using various KG embedding methods. In each bar group, first three models are additive, while the last three are multiplicative. Main findings from these plots are summarized in Section 6.2

conicity was consistently similar across frequency bins. For clarity, we have not shown different plots for individual frequency bins.

Relation Embeddings: As in entity embeddings, we observe a similar trend when we look at the distribution of ATMs for relation vectors in Figure 3. The conicity of relation vectors generated using additive models is almost zero across frequency bands. This coupled with the high vector spread observed, suggests that these vectors are scattered throughout the vector space. Relation vectors from multiplicative models exhibit high conicity and low vector spread, suggesting that they lie in a narrow cone centered at origin, like their entity counterparts.

6.2 Effect of Number of Negative Samples on Geometry

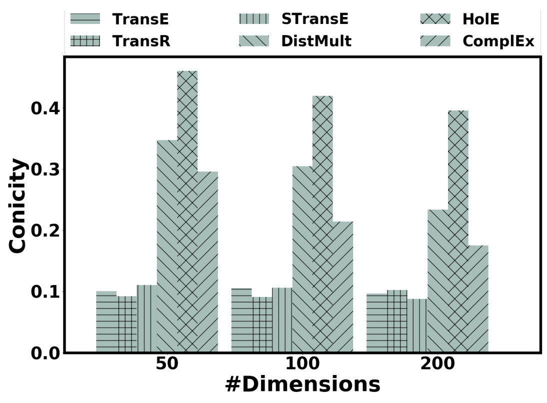
Summary of Findings:
Additive: Conicity and average length are invariant to changes in #NegativeSamples for both entities and relations.
Multiplicative: Conicity increases while average vector length decrease with increasing #NegativeSamples for entities. Conicity decreases, while average vector length remains constant (except HolE) for relations.

For experiments in this section, we keep the vector dimension constant at 100.

Entity Embeddings: As seen in Figure 4 (left), the conicity of entity vectors increases as the number of negative samples is increased for multiplicative models. In contrast, conicity of the entity vectors generated by additive models is unaffected by change in number of negative samples and they continue to be dispersed throughout the vector space. From Figure 4 (right), we observe that the average length of entity vectors produced by additive models is also invariant of any changes in number of negative samples. On the other hand, increase in negative sampling decreases the average entity vector length for all multiplicative models except HolE. The average entity vector length for HolE is nearly 1 for any number of negative samples, which is understandable considering it constrains the entity vectors to lie inside a unit ball (Nickel et al., 2016). This constraint is also enforced by the additive models: TransE, TransR, and STransE.

Relation Embeddings: Similar to entity embeddings, in case of relation vectors trained using additive models, the average length and conicity do not change while varying the number of negative samples. However, the conicity of relation vectors from multiplicative models decreases with increase in negative sampling. The average relation vector length is invariant for all multiplicative methods, except for HolE. We see a surprisingly big jump in average relation vector length for HolE going from 1 to 50 negative samples, but it does not change after that. Due to space constraints in the paper, we refer the reader to the Supplementary Section for plots discussing the effect of number of negative samples on geometry of relation vectors.

We note that the multiplicative score between two vectors may be increased by either increasing the alignment between the two vectors (i.e., increasing Conicity and reducing vector spread between them), or by increasing their lengths. It is interesting to note that we see exactly these effects in the geometry of multiplicative methods



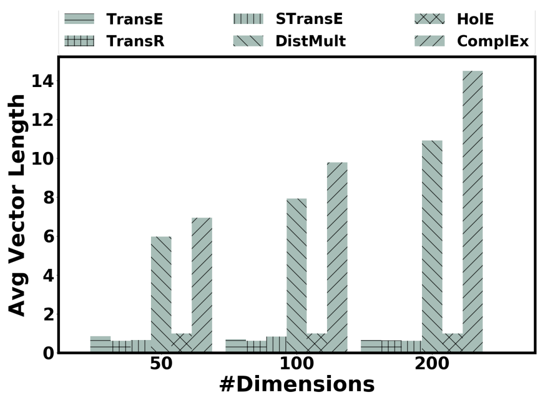


Figure 5: Conicity (left) and Average Vector Length (right) vs Number of Dimensions for entity vectors learned using various KG embedding methods. In each bar group, first three models are additive, while the last three are multiplicative. Main findings from these plots are summarized in Section 6.3.

analyzed above.

6.2.1 Correlation with Geometry of Word Embeddings

Our conclusions from the geometrical analysis of entity vectors produced by multiplicative models are similar to the results in (Mimno and Thompson, 2017), where increase in negative sampling leads to increased conicity of word vectors trained using the skip-gram with negative sampling (SGNS) method. On the other hand, additive models remain unaffected by these changes.

SGNS tries to maximize a score function of the form $\mathbf{w}^T \cdot \mathbf{c}$ for positive word context pairs, where $\mathbf{w}$ is the word vector and $\mathbf{c}$ is the context vector (Mikolov et al., 2013). This is very similar to the score function of multiplicative models as seen in Table 1. Hence, SGNS can be considered as a multiplicative model in the word domain.

Hence, we argue that our result on the increase in negative samples increasing the conicity of vectors trained using a multiplicative score function can be considered as a generalization of the one in (Mimno and Thompson, 2017).

6.3 Effect of Vector Dimension on Geometry

Summary of Findings:
Additive: Conicity and average length are invariant to changes in dimension for both entities and relations.
Multiplicative: Conicity decreases for both entities and relations with increasing dimension. Average vector length increases for both entities and relations, except for HolE entities.

Entity Embeddings: To study the effect of vector dimension on conicity and length, we set the number of negative samples to 1, while varying the vector dimension. From Figure 5 (left), we observe that the conicity for entity vectors generated by any additive model is almost invariant of increase in dimension, though STransE exhibits a slight decrease. In contrast, entity vector from multiplicative models show a clear decreasing pattern with increasing dimension.

As seen in Figure 5 (right), the average lengths of entity vectors from multiplicative models increase sharply with increasing vector dimension, except for HolE. In case of HolE, the average vector length remains constant at one. Deviation involving HolE is expected as it enforces entity vectors to fall within a unit ball. Similar constraints are enforced on entity vectors for additive models as well. Thus, the average entity vector lengths are not affected by increasing vector dimension for all additive models.

Relation Embeddings: We reach similar conclusion when analyzing against increasing dimension the change in geometry of relation vectors produced using these KG embedding methods. In this setting, the average length of relation vectors learned by HolE also increases as dimension is increased. This is consistent with the other methods in the multiplicative family. This is because, unlike entity vectors, the lengths of relation vectors of HolE are not constrained to be less than unit length. Due to lack of space, we are unable to show plots for relation vectors here, but the same can be found in the Supplementary Section.

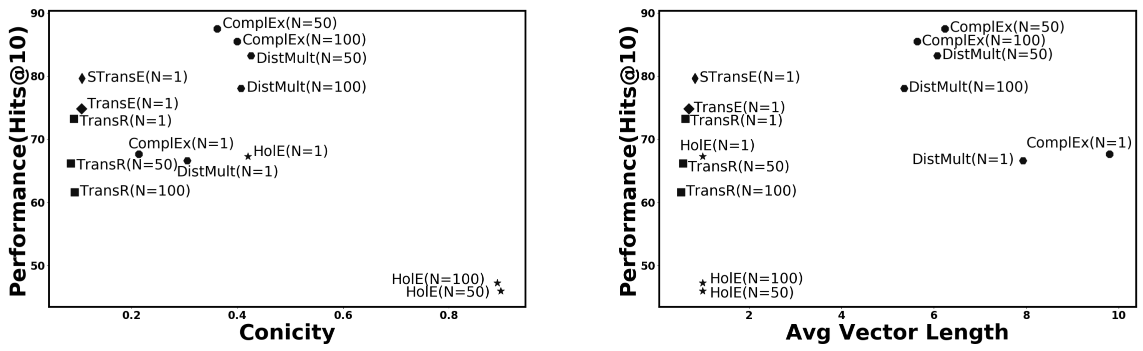


Figure 6: Relationship between Performance (HITS@10) on a link prediction task vs Conicity (left) and Avg. Vector Length (right). For each point, N represents the number of negative samples used. Main findings are summarized in Section 6.4.

6.4 Relating Geometry to Performance

Summary of Findings:
Additive: Neither entites nor relations exhibit correlation between geometry and performance.
Multiplicative: Keeping negative samples fixed, lower conicity or higher average vector length for entities leads to improved performance. No relationship for relations.

In this section, we analyze the relationship between geometry and performance on the Link prediction task, using the same setting as in (Bordes et al., 2013). Figure 6 (left) presents the effects of conicity of entity vectors on performance, while Figure 6 (right) shows the effects of average entity vector length.[4]

As we see from Figure 6 (left), for fixed number of negative samples, the multiplicative model with lower conicity of entity vectors achieves better performance. This performance gain is larger for higher numbers of negative samples (N). Additive models don't exhibit any relationship between performance and conicity, as they are all clustered around zero conicity, which is in-line with our observations in previous sections. In Figure 6 (right), for all multiplicative models except HolE, a higher average entity vector length translates to better performance, while the number of negative samples is kept fixed. Additive models and HolE don't exhibit any such patterns, as they are all clustered just below unit average entity vector length.

The above two observations for multiplicative models make intuitive sense, as lower conicity and higher average vector length would both translate to vectors being more dispersed in the space.

[4]A more focused analysis for multiplicative models is presented in Section 3 of Supplementary material.

We see another interesting observation regarding the high sensitivity of HolE to the number of negative samples used during training. Using a large number of negative examples (e.g., $N = 50$ or 100) leads to very high conicity in case of HolE. Figure 6 (right) shows that average entity vector length of HolE is always one. These two observations point towards HolE's entity vectors lying in a tiny part of the space. This translates to HolE performing poorer than all other models in case of high numbers of negative sampling.

We also did a similar study for relation vectors, but did not see any discernible patterns.

7 Conclusion

In this paper, we have initiated a systematic study into the important but unexplored problem of analyzing geometry of various Knowledge Graph (KG) embedding methods. To the best of our knowledge, this is the first study of its kind. Through extensive experiments on multiple real-world datasets, we are able to identify several insights into the geometry of KG embeddings. We have also explored the relationship between KG embedding geometry and its task performance. We have shared all our source code to foster further research in this area.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This work is supported in part by the Ministry of Human Resources Development (Government of India), Intel, Intuit, and by gifts from Google and Accenture.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. 2017. Convolutional 2D Knowledge Graph Embeddings. *ArXiv e-prints* .

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 601–610.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*. pages 2181–2187.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2863–2868.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of NAACL-HLT*. pages 460–466.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Srinivas Ravishankar, Chandrahas, and Partha Pratim Talukdar. 2017. Revisiting simple neural networks for learning representations of knowledge graphs. *6th Workshop on Automated Knowledge Base Construction (AKBC) at NIPS 2017* .

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *ArXiv e-prints* .

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*. pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Empirical Methods in Natural Language Processing (EMNLP)*. ACL Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. Citeseer, pages 1112–1119.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* .

A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss

Wan-Ting Hsu[1], **Chieh-Kai Lin**[1], **Ming-Ying Lee**[1], **Kerui Min**[2], **Jing Tang**[2], **Min Sun**[1]
[1] National Tsing Hua University, [2] Cheetah Mobile
{hsuwanting, axk51013, masonyl03}@gapp.nthu.edu.tw,
{minkerui, tangjing}@cmcm.com, sunmin@ee.nthu.edu.tw

Abstract

We propose a unified model combining the strength of extractive and abstractive summarization. On the one hand, a simple extractive model can obtain sentence-level attention with high ROUGE scores but less readable. On the other hand, a more complicated abstractive model can obtain word-level dynamic attention to generate a more readable paragraph. In our model, sentence-level attention is used to modulate the word-level attention such that words in less attended sentences are less likely to be generated. Moreover, a novel inconsistency loss function is introduced to penalize the inconsistency between two levels of attentions. By end-to-end training our model with the inconsistency loss and original losses of extractive and abstractive models, we achieve state-of-the-art ROUGE scores while being the most informative and readable summarization on the CNN/Daily Mail dataset in a solid human evaluation.

1 Introduction

Text summarization is the task of automatically condensing a piece of text to a shorter version while maintaining the important points. The ability to condense text information can aid many applications such as creating news digests, presenting search results, and generating reports. There are mainly two types of approaches: extractive and abstractive. Extractive approaches assemble summaries directly from the source text typically selecting one whole sentence at a time. In contrast, abstractive approaches can generate novel words and phrases not copied from the source text.

Original Article: McDonald's says...... The company says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week, in products including a new sandwich, as well as existing sandwiches, wraps and salads. It says the biggest change is the removal of sodium phosphates, which it said was used to keep the chicken moist, in favor of vegetable starch. The new recipe also does not use maltodextrin, which McDonald's said is generally used as a sugar to increase browning or as a carrier for seasoning. Jessica Foust, director of culinary innovation at McDonald's, said the changes were made because customers said they want 'simple, clean ingredients' they are familiar with...... And *Panera Bread has said* it plans to purge artificial colors, flavors and preservatives from its food by 2016......

Extractive Approach: **The company** says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week, in products including a new sandwich, as well as existing sandwiches, wraps and salads. It says the biggest change is the removal of sodium phosphates, **which it said was used to keep the chicken moist, in favor of vegetable starch**. The new recipe also does not use maltodextrin, which McDonald's said is generally used as a sugar to increase browning or as a carrier for seasoning.

Abstractive Approach: McDonald's says it expects the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week. The company says the changes were made because customers said they want 'simple, clean ingredients' they are familiar with. *McDonald's said* it plans to purge artificial colors, flavors and preservatives from its food by 2016.

Unified Approach: **McDonald's says it expects** the new 'Artisan Grilled Chicken' to be in its more than 14,300 U.S. stores by the end of next week, in products including a new sandwich, as well as existing sandwiches, wraps and salads. It says the biggest change is the removal of sodium phosphates. The new recipe also does not use maltodextrin, which McDonald's said is generally used as a sugar to increase browning or as a carrier for seasoning.

Figure 1: Comparison of extractive, abstractive, and our unified summaries on a news article. The extractive model picks most important but **incoherent or not concise** (see blue bold font) sentences. The abstractive summary is readable, concise but still *loses or mistakes some facts* (see red italics font). The final summary rewritten from fragments (see underline font) has the advantages from both extractive (importance) and abstractive advantage (**coherence** (see green bold font)).

Hence, abstractive summaries can be more coherent and concise than extractive summaries.

Extractive approaches are typically simpler. They output the probability of each sentence to be selected into the summary. Many earlier works on summarization (Cheng and Lapata, 2016; Nallapati et al., 2016a, 2017; Narayan et al., 2017; Yasunaga et al., 2017) focus on extractive summarization. Among them, Nallapati et al.

(2017) have achieved high ROUGE scores. On the other hand, abstractive approaches (Nallapati et al., 2016b; See et al., 2017; Paulus et al., 2017; Fan et al., 2017; Liu et al., 2017) typically involve sophisticated mechanism in order to paraphrase, generate unseen words in the source text, or even incorporate external knowledge. Neural networks (Nallapati et al., 2017; See et al., 2017) based on the attentional encoder-decoder model (Bahdanau et al., 2014) were able to generate abstractive summaries with high ROUGE scores but suffer from inaccurately reproducing factual details and an inability to deal with out-of-vocabulary (OOV) words. Recently, See et al. (2017) propose a pointer-generator model which has the abilities to copy words from source text as well as generate unseen words. Despite recent progress in abstractive summarization, extractive approaches (Nallapati et al., 2017; Yasunaga et al., 2017) and lead-3 baseline (i.e., selecting the first 3 sentences) still achieve strong performance in ROUGE scores.

We propose to explicitly take advantage of the strength of state-of-the-art extractive and abstractive summarization and introduced the following unified model. Firstly, we treat the probability output of each sentence from the extractive model (Nallapati et al., 2017) as sentence-level attention. Then, we modulate the word-level dynamic attention from the abstractive model (See et al., 2017) with sentence-level attention such that words in less attended sentences are less likely to be generated. In this way, extractive summarization mostly benefits abstractive summarization by mitigating spurious word-level attention. Secondly, we introduce a novel inconsistency loss function to encourage the consistency between two levels of attentions. The loss function can be computed without additional human annotation and has shown to ensure our unified model to be mutually beneficial to both extractive and abstractive summarization. On CNN/Daily Mail dataset, our unified model achieves state-of-the-art ROUGE scores and outperforms a strong extractive baseline (i.e., lead-3). Finally, to ensure the quality of our unified model, we conduct a solid human evaluation and confirm that our method significantly outperforms recent state-of-the-art methods in informativity and readability.

To summarize, our contributions are twofold:

- We propose a unified model combining sentence-level and word-level attentions to take advantage of both extractive and abstractive summarization approaches.
- We propose a novel inconsistency loss function to ensure our unified model to be mutually beneficial to both extractive and abstractive summarization. The unified model with inconsistency loss achieves the best ROUGE scores on CNN/Daily Mail dataset and outperforms recent state-of-the-art methods in informativity and readability on human evaluation.

2 Related Work

Text summarization has been widely studied in recent years. We first introduce the related works of neural-network-based extractive and abstractive summarization. Finally, we introduce a few related works with hierarchical attention mechanism.

Extractive summarization. Kågebäck et al. (2014) and Yin and Pei (2015) use neural networks to map sentences into vectors and select sentences based on those vectors. Cheng and Lapata (2016), Nallapati et al. (2016a) and Nallapati et al. (2017) use recurrent neural networks to read the article and get the representations of the sentences and article to select sentences. Narayan et al. (2017) utilize side information (i.e., image captions and titles) to help the sentence classifier choose sentences. Yasunaga et al. (2017) combine recurrent neural networks with graph convolutional networks to compute the salience (or importance) of each sentence. While some extractive summarization methods obtain high ROUGE scores, they all suffer from low readability.

Abstractive summarization. Rush et al. (2015) first bring up the abstractive summarization task and use attention-based encoder to read the input text and generate the summary. Based on them, Miao and Blunsom (2016) use a variational auto-encoder and Nallapati et al. (2016b) use a more powerful sequence-to-sequence model. Besides, Nallapati et al. (2016b) create a new article-level summarization dataset called CNN/Daily Mail by adapting DeepMind question-answering dataset (Hermann et al., 2015). Ranzato et al. (2015) change the traditional training method to directly optimize evaluation metrics (e.g., BLEU and ROUGE). Gu et al. (2016), See et al. (2017) and Paulus et al. (2017) combine pointer networks

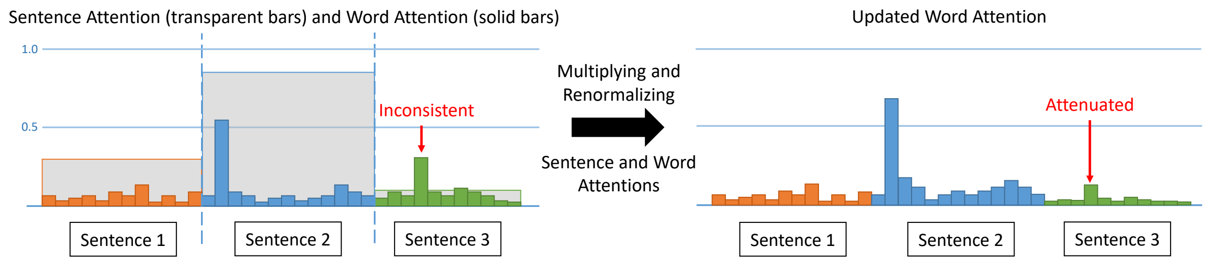


Figure 2: Our unified model combines the word-level and sentence-level attentions. Inconsistency occurs when word attention is high but sentence attention is low (see red arrow).

(Vinyals et al., 2015) into their models to deal with out-of-vocabulary (OOV) words. Chen et al. (2016) and See et al. (2017) restrain their models from attending to the same word to decrease repeated phrases in the generated summary. Paulus et al. (2017) use policy gradient on summarization and state out the fact that high ROUGE scores might still lead to low human evaluation scores. Fan et al. (2017) apply convolutional sequence-to-sequence model and design several new tasks for summarization. Liu et al. (2017) achieve high readability score on human evaluation using generative adversarial networks.

Hierarchical attention. Attention mechanism was first proposed by Bahdanau et al. (2014). Yang et al. (2016) proposed a hierarchical attention mechanism for document classification. We adopt the method of combining sentence-level and word-level attention in Nallapati et al. (2016b). However, their sentence attention is dynamic, which means it will be different for each generated word. Whereas our sentence attention is fixed for all generated words. Inspired by the high performance of extractive summarization, we propose to use fixed sentence attention.

Our model combines state-of-the-art extractive model (Nallapati et al., 2017) and abstractive model (See et al., 2017) by combining sentence-level attention from the former and word-level attention from the latter. Furthermore, we design an inconsistency loss to enhance the cooperation between the extractive and abstractive models.

3 Our Unified Model

We propose a unified model to combine the strength of both state-of-the-art extractor (Nallapati et al., 2017) and abstracter (See et al., 2017). Before going into details of our model, we first define the tasks of the extractor and abstracter.

Problem definition. The input of both extractor and abstracter is a sequence of words $\mathbf{w} = [w_1, w_2, ..., w_m, ...]$, where m is the word index. The sequence of words also forms a sequence of sentences $\mathbf{s} = [s_1, s_2, ..., s_n, ...]$, where n is the sentence index. The m^{th} word is mapped into the $n(m)^{th}$ sentence, where $n(\cdot)$ is the mapping function. The output of the extractor is the sentence-level attention $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_n, ...]$, where β_n is the probability of the n^{th} sentence been extracted into the summary. On the other hand, our attention-based abstractor computes word-level attention $\boldsymbol{\alpha}^t = \left[\alpha_1^t, \alpha_2^t, ..., \alpha_m^t, ...\right]$ dynamically while generating the t^{th} word in the summary. The output of the abstracter is the summary text $\mathbf{y} = \left[y^1, y^2, ..., y^t, ...\right]$, where y^t is t^{th} word in the summary.

In the following, we introduce the mechanism to combine sentence-level and word-level attentions in Sec. 3.1. Next, we define the novel inconsistency loss that ensures extractor and abstracter to be mutually beneficial in Sec. 3.2. We also give the details of our extractor in Sec. 3.3 and our abstracter in Sec. 3.4. Finally, our training procedure is described in Sec. 3.5.

3.1 Combining Attentions

Pieces of evidence (e.g., Vaswani et al. (2017)) show that attention mechanism is very important for NLP tasks. Hence, we propose to explicitly combine the sentence-level β_n and word-level α_m^t attentions by simple scalar multiplication and renormalization. The updated word attention $\hat{\alpha}_m^t$ is

$$\hat{\alpha}_m^t = \frac{\alpha_m^t \times \beta_{n(m)}}{\sum_m \alpha_m^t \times \beta_{n(m)}}. \quad (1)$$

The multiplication ensures that only when both word-level α_m^t and sentence-level β_n attentions are high, the updated word attention $\hat{\alpha}_m^t$ can be high. Since the sentence-level attention β_n from the extractor already achieves high ROUGE

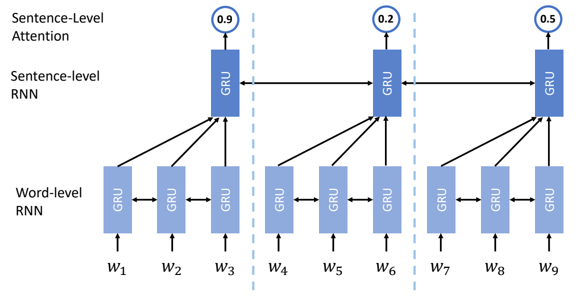


Figure 3: Architecture of the extractor. We treat the sigmoid output of each sentence as sentence-level attention $\in [0, 1]$.

scores, β_n intuitively modulates the word-level attention α^t_m to mitigate spurious word-level attention such that words in less attended sentences are less likely to be generated (see Fig. 2). As highlighted in Sec. 3.4, the word-level attention $\hat{\alpha}^t_m$ significantly affects the decoding process of the abstracter. Hence, an updated word-level attention is our key to improve abstractive summarization.

3.2 Inconsistency Loss

Instead of only leveraging the complementary nature between sentence-level and word-level attentions, we would like to encourage these two-levels of attentions to be mostly consistent to each other during training as an intrinsic learning target for free (i.e., without additional human annotation). Explicitly, we would like the sentence-level attention to be high when the word-level attention is high. Hence, we design the following inconsistency loss,

$$L_{inc} = -\frac{1}{T}\sum_{t=1}^{T}\log\left(\frac{1}{|\mathcal{K}|}\sum_{m\in\mathcal{K}}\alpha^t_m \times \beta_{n(m)}\right), \quad (2)$$

where $\mathcal{K}$ is the set of top K attended words and T is the number of words in the summary. This implicitly encourages the distribution of the word-level attentions to be sharp and sentence-level attention to be high. To avoid the degenerated solution for the distribution of word attention to be one-hot and sentence attention to be high, we include the original loss functions for training the extractor (L_{ext} in Sec. 3.3) and abstracter (L_{abs} and L_{cov} in Sec. 3.4). Note that Eq. 1 is the only part that the extractor is interacting with the abstracter. Our proposed inconsistency loss facilitates our end-to-end trained unified model to be mutually beneficial to both the extractor and abstracter.

3.3 Extractor

Our extractor is inspired by Nallapati et al. (2017). The main difference is that our extractor does not need to obtain the final summary. It mainly needs to obtain a short list of important sentences with a high recall to further facilitate the abstractor. We first introduce the network architecture and the loss function. Finally, we define our ground truth important sentences to encourage high recall.

Architecture. The model consists of a hierarchical bidirectional GRU which extracts sentence representations and a classification layer for predicting the sentence-level attention β_n for each sentence (see Fig. 3).

Extractor loss. The following sigmoid cross entropy loss is used,

$$L_{ext} = -\frac{1}{N}\sum_{n=1}^{N}(g_n \log\beta_n + (1-g_n)\log(1-\beta_n)), \quad (3)$$

where $g_n \in \{0, 1\}$ is the ground-truth label for the n^{th} sentence and N is the number of sentences. When $g_n = 1$, it indicates that the n^{th} sentence should be attended to facilitate abstractive summarization.

Ground-truth label. The goal of our extractor is to extract sentences with high informativity, which means the extracted sentences should contain information that is needed to generate an abstractive summary as much as possible. To obtain the ground-truth labels $\mathbf{g} = \{g_n\}_n$, first, we measure the informativity of each sentence s_n in the article by computing the ROUGE-L recall score (Lin, 2004) between the sentence s_n and the reference abstractive summary $\hat{\mathbf{y}} = \{\hat{y}^t\}_t$. Second, we sort the sentences by their informativity and select the sentence in the order of high to low informativity. We add one sentence at a time if the new sentence can increase the informativity of all the selected sentences. Finally, we obtain the ground-truth labels $\mathbf{g}$ and train our extractor by minimizing Eq. 3. Note that our method is different from Nallapati et al. (2017) who aim to extract a final summary for an article so they use ROUGE F-1 score to select ground-truth sentences; while we focus on high informativity, hence, we use ROUGE recall score to obtain as much information as possible with respect to the reference summary $\hat{\mathbf{y}}$.

3.4 Abstracter

The second part of our model is an abstracter that reads the article; then, generate a summary

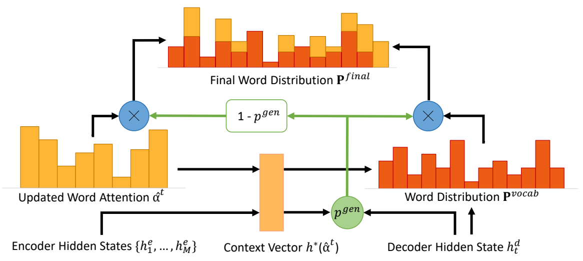

Final Word Distribution $\mathbf{P}^{final}$
1 - p^{gen}
Updated Word Attention $\hat{\alpha}^t$
Word Distribution $\mathbf{P}^{vocab}$
p^{gen}
Encoder Hidden States $\{h_1^e, \ldots, h_M^e\}$
Context Vector $h^*(\hat{\alpha}^t)$
Decoder Hidden State h_t^d

Figure 4: Decoding mechanism in the abstracter. In the decoder step t, our updated word attention $\hat{\boldsymbol{\alpha}}^t$ is used to generate context vector $h^*(\hat{\boldsymbol{\alpha}}^t)$. Hence, it updates the final word distribution $\mathbf{P}^{final}$.

word-by-word. We use the pointer-generator network proposed by See et al. (2017) and combine it with the extractor by combining sentence-level and word-level attentions (Sec. 3.1).

Pointer-generator network. The pointer-generator network (See et al., 2017) is a specially designed sequence-to-sequence attentional model that can generate the summary by copying words in the article or generating words from a fixed vocabulary at the same time. The model contains a bidirectional LSTM which serves as an encoder to encode the input words $\mathbf{w}$ and a unidirectional LSTM which serves as a decoder to generate the summary $\mathbf{y}$. For details of the network architecture, please refer to See et al. (2017). In the following, we describe how the updated word attention $\hat{\boldsymbol{\alpha}}^t$ affects the decoding process.

Notations. We first define some notations. h_m^e is the encoder hidden state for the m^{th} word. h_t^d is the decoder hidden state in step t. $h^*(\hat{\boldsymbol{\alpha}}^t) = \sum_m^M \hat{\alpha}_m^t \times h_m^e$ is the context vector which is a function of the updated word attention $\hat{\boldsymbol{\alpha}}^t$. $\mathbf{P}^{vocab}(h^*(\hat{\boldsymbol{\alpha}}^t))$ is the probability distribution over the fixed vocabulary before applying the copying mechanism.

$$\begin{aligned}&\mathbf{P}^{vocab}(h^*(\hat{\boldsymbol{\alpha}}^t)) \qquad (4)\\ &= \text{softmax}(W_2(W_1[h_t^d, h^*(\hat{\boldsymbol{\alpha}}^t)] + b_1) + b_2),\end{aligned}$$

where W_1, W_2, b_1 and b_2 are learnable parameters. $\mathbf{P}^{vocab} = \{P_w^{vocab}\}_w$ where $P_w^{vocab}(h^*(\hat{\boldsymbol{\alpha}}^t))$ is the probability of word w being decoded. $p^{gen}(h^*(\hat{\boldsymbol{\alpha}}^t)) \in [0, 1]$ is the generating probability (see Eq.8 in See et al. (2017)) and $1 - p^{gen}(h^*(\hat{\boldsymbol{\alpha}}^t))$ is the copying probability.

Final word distribution. $P_w^{final}(\hat{\boldsymbol{\alpha}}^t)$ is the final probability of word w being decoded (i.e., $y^t = w$). It is related to the updated word attention $\hat{\boldsymbol{\alpha}}^t$ as follows (see Fig. 4),

$$\begin{aligned}P_w^{final}(\hat{\boldsymbol{\alpha}}^t) &= p^{gen}(h^*(\hat{\boldsymbol{\alpha}}^t))P_w^{vocab}(h^*(\hat{\boldsymbol{\alpha}}^t)) \quad (5)\\ &+ (1 - p^{gen}(h^*(\hat{\boldsymbol{\alpha}}^t))) \sum_{m:w_m=w} \hat{\alpha}_m^t.\end{aligned}$$

Note that $\mathbf{P}^{final} = \{P_w^{final}\}_w$ is the probability distribution over the fixed vocabulary and out-of-vocabulary (OOV) words. Hence, OOV words can be decoded. Most importantly, it is clear from Eq. 5 that $P_w^{final}(\hat{\boldsymbol{\alpha}}^t)$ is a function of the updated word attention $\hat{\boldsymbol{\alpha}}^t$. Finally, we train the abstracter to minimize the negative log-likelihood:

$$L_{abs} = -\frac{1}{T}\sum_{t=1}^{T} \log P_{\hat{y}^t}^{final}(\hat{\boldsymbol{\alpha}}^t)\;, \qquad (6)$$

where $\hat{y}^t$ is the t^{th} token in the reference abstractive summary.

Coverage mechanism. We also apply coverage mechanism (See et al., 2017) to prevent the abstracter from repeatedly attending to the same place. In each decoder step t, we calculate the coverage vector $\mathbf{c}^t = \sum_{t'=0}^{t-1} \hat{\boldsymbol{\alpha}}^{t'}$ which indicates so far how much attention has been paid to every input word. The coverage vector $\mathbf{c}^t$ will be used to calculate word attention $\hat{\boldsymbol{\alpha}}^t$ (see Eq.11 in See et al. (2017)). Moreover, coverage loss L_{cov} is calculated to directly penalize the repetition in updated word attention $\hat{\boldsymbol{\alpha}}^t$:

$$L_{cov} = \frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} \min(\hat{\alpha}_m^t, \mathbf{c}_m^t)\;. \qquad (7)$$

The objective function for training the abstracter with coverage mechanism is the weighted sum of negative log-likelihood and coverage loss.

3.5 Training Procedure

We first pre-train the extractor by minimizing L_{ext} in Eq. 3 and the abstracter by minimizing L_{abs} and L_{cov} in Eq. 6 and Eq. 7, respectively. When pre-training, the abstracter takes ground-truth extracted sentences (i.e., sentences with $g_n = 1$) as input. To combine the extractor and abstracter, we proposed two training settings : (1) two-stages training and (2) end-to-end training.

Two-stages training. In this setting, we view the sentence-level attention $\boldsymbol{\beta}$ from the pre-trained extractor as hard attention. The extractor becomes a classifier to select sentences with high attention (i.e., $\beta_n >$ threshold). We simply combine the extractor and abstracter by feeding the extracted sentences to the abstracter. Note that we finetune the abstracter since the input text becomes extractive summary which is obtained from the extractor.

End-to-end training. For end-to-end training, the sentence-level attention $\boldsymbol{\beta}$ is soft attention and will be combined with the word-level attention $\boldsymbol{\alpha}^t$ as described in Sec. 3.1. We end-to-end train the extractor and abstracter by minimizing four loss functions: L_{ext}, L_{abs}, L_{cov}, as well as L_{inc} in Eq. 2. The final loss is as below:

$$L_{e2e} = \lambda_1 L_{ext} + \lambda_2 L_{abs} + \lambda_3 L_{cov} + \lambda_4 L_{inc}, \tag{8}$$

where λ_1, λ_2, λ_3, λ_4 are hyper-parameters. In our experiment, we give L_{ext} a bigger weight (e.g., $\lambda_1 = 5$) when end-to-end training with L_{inc} since we found that L_{inc} is relatively large such that the extractor tends to ignore L_{ext}.

4 Experiments

We introduce the dataset and implementation details of our method evaluated in our experiments.

4.1 Dataset

We evaluate our models on the CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016b; See et al., 2017) which contains news stories in CNN and Daily Mail websites. Each article in this dataset is paired with one human-written multi-sentence summary. This dataset has two versions: *anonymized* and *non-anonymized*. The former contains the news stories with all the named entities replaced by special tokens (e.g., `@entity2`); while the latter contains the raw text of each news story. We follow See et al. (2017) and obtain the *non-anonymized* version of this dataset which has 287,113 training pairs, 13,368 validation pairs and 11,490 test pairs.

4.2 Implementation Details

We train our extractor and abstracter with 128-dimension word embeddings and set the vocabulary size to 50k for both source and target text. We follow Nallapati et al. (2017) and See et al. (2017) and set the hidden dimension to 200 and 256 for the extractor and abstracter, respectively. We use Adagrad optimizer (Duchi et al., 2011) and apply early stopping based on the validation set. In the testing phase, we limit the length of the summary to 120.

Pre-training. We use learning rate 0.15 when pre-training the extractor and abstracter. For the extractor, we limit both the maximum number of sentences per article and the maximum number of tokens per sentence to 50 and train the model for 27k iterations with the batch size of 64. For the abstracter, it takes ground-truth extracted sentences (i.e., sentences with $g_n = 1$) as input. We limit the length of the source text to 400 and the length of the summary to 100 and use the batch size of 16. We train the abstracter without coverage mechanism for 88k iterations and continue training for 1k iterations with coverage mechanism ($L_{abs} : L_{cov} = 1 : 1$).

Two-stages training. The abstracter takes extracted sentences with $\beta_n > 0.5$, where $\boldsymbol{\beta}$ is obtained from the pre-trained extractor, as input during two-stages training. We finetune the abstracter for 10k iterations.

End-to-end training. During end-to-end training, we will minimize four loss functions (Eq. 8) with $\lambda_1 = 5$ and $\lambda_2 = \lambda_3 = \lambda_4 = 1$. We set K to 3 for computing L_{inc}. Due to the limitation of the memory, we reduce the batch size to 8 and thus use a smaller learning rate 0.01 for stability. The abstracter here reads the whole article. Hence, we increase the maximum length of source text to 600. We end-to-end train the model for 50k iterations.

5 Results

Our unified model not only generates an abstractive summary but also extracts the important sentences in an article. Our goal is that both of the two types of outputs can help people to read and understand an article faster. Hence, in this section, we evaluate the results of our extractor in Sec. 5.1 and unified model in Sec. 5.2. Furthermore, in Sec. 5.3, we perform human evaluation and show that our model can provide a better abstractive summary than other baselines.

5.1 Results of Extracted Sentences

To evaluate whether our extractor obtains enough information for the abstracter, we use full-length ROUGE recall scores[1] between the extracted sentences and reference abstractive summary. High ROUGE recall scores can be obtained if the extracted sentences include more words or sequences overlapping with the reference abstractive summary. For each article, we select sentences with the sentence probabilities β greater than 0.5. We show the results of the ground-truth sentence labels (Sec. 3.3) and our models on the

[1]All our ROUGE scores are reported by the official ROUGE script. We use the `pyrouge` package. `https://pypi.org/project/pyrouge/0.1.3/`

Method	ROUGE-1	ROUGE-2	ROUGE-L
pre-trained	73.50	35.55	68.57
end2end w/o inconsistency loss	72.97	35.11	67.99
end2end w/ inconsistency loss	**78.40**	**39.45**	**73.83**
ground-truth labels	89.23	49.36	85.46

Table 1: ROUGE recall scores of the extracted sentences. *pre-trained* indicates the extractor trained on the ground-truth labels. *end2end* indicates the extractor after end-to-end training with the abstracter. Note that *ground-truth labels* show the upper-bound performance since the reference summary to calculate ROUGE-recall is abstractive. All our ROUGE scores have a 95% confidence interval with at most ±0.33.

Method	ROUGE-1	ROUGE-2	ROUGE-L
HierAttn (Nallapati et al., 2016b)*	32.75	12.21	29.01
DeepRL (Paulus et al., 2017)*	39.87	15.82	36.90
pointer-generator (See et al., 2017)	39.53	17.28	36.38
GAN (Liu et al., 2017)	39.92	17.65	36.71
two-stage (ours)	39.97	17.43	36.34
end2end w/o inconsistency loss (ours)	40.19	17.67	36.68
end2end w/ inconsistency loss (ours)	**40.68**	**17.97**	**37.13**
lead-3 (See et al., 2017)	40.34	17.70	36.57

Table 2: ROUGE F-1 scores of the generated abstractive summaries on the CNN/Daily Mail test set. Our two-stages model outperforms pointer-generator model on ROUGE-1 and ROUGE-2. In addition, our model trained end-to-end with inconsistency loss exceeds the lead-3 baseline. All our ROUGE scores have a 95% confidence interval with at most ±0.24. '*' indicates the model is trained and evaluated on the anonymized dataset and thus is not strictly comparable with ours.

test set of the CNN/Daily Mail dataset in Table 1. Note that the ground-truth extracted sentences can't get ROUGE recall scores of 100 because reference summary is abstractive and may contain some words and sequences that are not in the article. Our extractor performs the best when end-to-end trained with inconsistency loss.

5.2 Results of Abstractive Summarization

We use full-length ROUGE-1, ROUGE-2 and ROUGE-L F-1 scores to evaluate the generated summaries. We compare our models (two-stage and end-to-end) with state-of-the-art abstractive summarization models (Nallapati et al., 2016b; Paulus et al., 2017; See et al., 2017; Liu et al., 2017) and a strong lead-3 baseline which directly uses the first three article sentences as the summary. Due to the writing style of news articles, the most important information is often written at the beginning of an article which makes lead-3 a strong baseline. The results of ROUGE F-1 scores are shown in Table 2. We prove that with help of the extractor, our unified model can outperform pointer-generator (the third row in Table 2) even with two-stages training (the fifth row in Table 2). After end-to-end training without inconsistency loss, our method already achieves better ROUGE scores by cooperating with each other. Moreover, our model end-to-end trained with inconsistency loss achieves state-of-the-art ROUGE scores and exceeds lead-3 baseline.

In order to quantify the effect of inconsistency loss, we design a metric – inconsistency rate R_{inc} – to measure the inconsistency for each generated summary. For each decoder step t, if the word with maximum attention belongs to a sentence with low attention (i.e., $\beta_{n(\text{argmax}(\boldsymbol{\alpha}^t))} < \text{mean}(\boldsymbol{\beta})$), we define this step as an inconsistent step t_{inc}. The inconsistency rate R_{inc} is then defined as the percentage of the inconsistent steps in the summary.

$$R_{inc} = \frac{\text{Count}(t_{inc})}{T}, \tag{9}$$

where T is the length of the summary. The average inconsistency rates on test set are shown in Table 4. Our inconsistency loss significantly decrease R_{inc} from about 20% to 4%. An example of inconsistency improvement is shown in Fig. 5.

Method	informativity	conciseness	readability
DeepRL (Paulus et al., 2017)	3.23	2.97	2.85
pointer-generator (See et al., 2017)	3.18	3.36	3.47
GAN (Liu et al., 2017)	3.22	3.52	3.51
Ours	**3.58**	3.40	**3.70**
reference	3.43	**3.61**	3.62

Table 3: Comparing human evaluation results with state-of-the-art methods.

Method	avg. R_{inc}
w/o incon. loss	0.198
w/ incon. loss	0.042

Table 4: Inconsistency rate of our end-to-end trained model with and without inconsistency loss.

Without inconsistency loss:
If that was a tornado, it was one monster of one. Luckily, so far it looks like no one was hurt. With tornadoes touching down near Dallas on Sunday, **Ryan Shepard snapped a photo of a black cloud formation reaching down to the ground.** **He said it was a tornado. It wouldn't be an exaggeration to say it looked half a mile wide.** More like a mile, said Jamie Moore, head of emergency management in Johnson County, Texas. It could have been one the National Weather Service warned about in a tweet as severe thunderstorms drenched the area, causing street flooding. (...)

With inconsistency loss:
If that was a tornado, it was one monster of one. Luckily, so far it looks like no one was hurt. With **tornadoes touching down near Dallas on Sunday, Ryan Shepard snapped a photo of a black cloud formation reaching down to the ground.** He said it was a tornado. It wouldn't be an exaggeration to say it looked half a mile wide. More like a mile, said Jamie Moore, head of emergency management in Johnson County, Texas. **It could have been one the National Weather Service warned about in a tweet as** severe thunderstorms drenched the area, **causing street flooding.** (...)

Figure 5: Visualizing the consistency between sentence and word attentions on the original article. We highlight word (bold font) and sentence (underline font) attentions. We compare our methods trained with and without inconsistency loss. Inconsistent fragments (see red bold font) occur when trained without the inconsistency loss.

5.3 Human Evaluation

We perform human evaluation on Amazon Mechanical Turk (MTurk)[2] to evaluate the informativity, conciseness and readability of the summaries. We compare our best model (end2end with inconsistency loss) with pointer-generator (See et al., 2017), generative adversarial network (Liu et al., 2017) and deep reinforcement model (Paulus et al., 2017). For these three models, we use the test set outputs provided by the authors[3].

We randomly pick 100 examples in the test set. All generated summaries are re-capitalized and de-tokenized. Since Paulus et al. (2017) trained their model on anonymized data, we also recover the anonymized entities and numbers of their outputs.

We show the article and 6 summaries (reference summary, 4 generated summaries and a random summary) to each human evaluator. The random summary is a reference summary randomly picked from other articles and is used as a trap. We show the instructions of three different aspects as: (1) Informativity: how well does the summary capture the important parts of the article? (2) Conciseness: is the summary clear enough to explain everything without being redundant? (3) Readability: how well-written (fluent and grammatical) the summary is? The user interface of our human evaluation is shown in the supplementary material.

We ask the human evaluator to evaluate each summary by scoring the three aspects with 1 to 5 score (higher the better). We reject all the evaluations that score the informativity of the random summary as 3, 4 and 5. By using this trap mechanism, we can ensure a much better quality of our human evaluation. For each example, we first ask 5 human evaluators to evaluate. However, for those articles that are too long, which are always skipped by the evaluators, it is hard to collect 5 reliable evaluations. Hence, we collect at least 3 evaluations for every example. For each summary, we average the scores over different human evaluators.

The results are shown in Table 3. The reference summaries get the best score on conciseness since the recent abstractive models tend to copy sentences from the input articles. However, our model learns well to select important information and form complete sentences so we even get slightly better scores on informativity and readability than the reference summaries. We show a typical example of our model comparing with other state-of-

[2]`https://www.mturk.com/`
[3]`https://github.com/abisee/pointer-generator` and `https://likicode.com` for the first two. For DeepRL, we asked through email.

Original article (truncated):
A chameleon balances carefully on a branch, waiting calmly for its prey... except that if you look closely, you will see that this picture is not all that it seems. For the 'creature' poised to pounce is not a colourful species of lizard but something altogether more human. Featuring two carefully painted female models, it is a clever piece of sculpture designed to create an amazing illusion. It is the work of Italian artist Johannes Stoetter. Scroll down for video. Can you see us? Italian artist **Johannes Stoetter has painted two naked women to look like a chameleon. The 37-year-old has previously transformed his models into frogs and parrots** but this may be his most intricate and impressive piece to date. **Stoetter daubed water-based body paint on the naked models to create the multicoloured effect, then intertwined them to form the shape of a chameleon.** To complete the deception, the models rested on a bench painted to match their skin and held the green branch in the air beneath them. Stoetter can take weeks to plan one of his pieces and hours to paint it. Speaking about The Chameleon, he said: 'I worked about four days to design the motif bigger and paint it with colours. The body painting took me about six hours with the help of an assistant. I covered the hair with natural clay to make the heads look bald.' Camouflage job: A few finishing touches are applied to the two naked models to complete the transformation. 'There are different difficulties on different levels as in every work, but I think that my passion and love to my work is so big, that I figure out a way to deal with difficulties. My main inspirations are nature, my personal life-philosophy, every-day-life and people themselves.' However, the finished result existed only briefly before the models were able to get up and wash the paint off with just a video and some photographs to record it. (...)

Reference summary:
Johannes Stoetter's artwork features two carefully painted female models. The 37-year-old has previously transformed models into frogs and parrots. Daubed water-based body paint on naked models to create the effect. Completing the deception, models rested on bench painted to match skin.

DeepRL:
Italian artist Johannes Stoetter has painted female models to look like a chameleon. The 37-year-old has previously transformed his models into frogs and parrots but *this may be the work of Italian artist*. He has painted nude models and it is a clever piece of sculpture designed to create an amazing illusion. *It is work of artist Johannes Stoetter.*

GAN:
Italian artist Johannes Stoetter has painted two naked women to look like a chameleon. The 37-year-old has previously transformed his models into frogs and parrots *but this may be his most intricate and impressive piece to date*.

Pointer-generator:
Italian artist Johannes Stoetter has painted two naked women to look like a chameleon. *It is the work of Italian artist Johannes Stoetter.* Stoetter daubed water-based body paint on the naked models to create the multicoloured effect, then intertwined them to form the shape of a chameleon.

Our unified model (with inconsistency loss):
Johannes Stoetter has painted two naked women to look like a chameleon. The 37-year-old has previously transformed his models into frogs and parrots. Stoetter daubed water-based body paint on the naked models to create the multicoloured effect, then intertwined them to form the shape of a chameleon.

Figure 6: Typical Comparison. Our model attended at the most important information (blue bold font) matching well with the reference summary; while other state-of-the-art methods generate repeated or less important information (red italic font).

the-art methods in Fig. 6. More examples (5 using CNN/Daily Mail news articles and 3 using non-news articles as inputs) are provided in the supplementary material.

6 Conclusion

We propose a unified model combining the strength of extractive and abstractive summarization. Most importantly, a novel inconsistency loss function is introduced to penalize the inconsistency between two levels of attentions. The inconsistency loss enables extractive and abstractive summarization to be mutually beneficial. By end-to-end training of our model, we achieve the best ROUGE-recall and ROUGE while being the most informative and readable summarization on the CNN/Daily Mail dataset in a solid human evaluation.

Acknowledgments

We thank the support from Cheetah Mobile, National Taiwan University, and MOST 107-2634-F-007-007, 106-3114-E-007-004, 107-2633-E-002-001. We thank Yun-Zhu Song for assistance with useful survey and experiment on the task of abstractive summarization.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–494.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2017. Generative adversarial network for abstractive text summarization. In *Proceddings of the 2018 Association for the Advancement of Artificial Intelligence*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceddings of the 2017 Association for the Advancement of Artificial Intelligence*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016a. Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Shashi Narayan, Nikos Papasarantopoulos, Mirella Lapata, and Shay B Cohen. 2017. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the 2018 International Conference on Learning Representations*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1383–1389. AAAI Press.

Extractive Summarization with SWAP-NET: Sentences and Words from Alternating Pointer Networks

Aishwarya Jadhav
Indian Institute of Science
Bangalore, India
aishwaryaj@iisc.ac.in

Vaibhav Rajan
School of Computing
National University of Singapore
vaibhav.rajan@nus.edu.sg

Abstract

We present a new neural sequence-to-sequence model for extractive summarization called SWAP-NET (Sentences and Words from Alternating Pointer Networks). Extractive summaries comprising a salient subset of input sentences, often also contain important key words. Guided by this principle, we design SWAP-NET that models the interaction of key words and salient sentences using a new two-level pointer network based architecture. SWAP-NET identifies both salient sentences and key words in an input document, and then combines them to form the extractive summary. Experiments on large scale benchmark corpora demonstrate the efficacy of SWAP-NET that outperforms state-of-the-art extractive summarizers.

1 Introduction

Automatic summarization aims to shorten a text document while maintaining the salient information of the original text. The practical need for such systems is growing with the rapid and continuous increase in textual information sources in multiple domains.

Summarization tools can be broadly classified into two categories: extractive and abstractive. *Extractive* summarization selects parts of the input document to create its summary while *abstractive* summarization generates summaries that may have words or phrases not present in the input document. Abstractive summarization is clearly harder as methods have to address factual and grammatical errors that may be introduced and problems in utilizing external knowledge sources to obtain paraphrasing or generalization. Extractive summarizers obviate the need to solve these problems by selecting the most salient textual units (usually sentences) from the input documents. As a result, they generate summaries that are grammatically and semantically more accurate than those from abstractive methods. While they may have problems like incorrect or unclear referring expressions or lack of coherence, they are computationally simpler and more efficient to generate. Indeed, state-of-the-art extractive summarizers are comparable or often better in performance to competitive abstractive summarizers (see (Nallapati et al., 2017) for a recent empirical comparison).

Classical approaches to extractive summarization have relied on human-engineered features from the text that are used to score sentences in the input document and select the highest-scoring sentences. These include graph or constraint-optimization based approaches as well as classifier-based methods. A review of these approaches can be found in Nenkova et al. (2011). Some of these methods generate summaries from multiple documents. In this paper, we focus on single document summarization.

Modern approaches that show the best performance are based on end-to-end deep learning models that do not require human-crafted features. Neural models have tremendously improved performance in several difficult problems in NLP such as machine translation (Chen et al., 2017) and question-answering (Hao et al., 2017). Deep models with thousands of parameters require large, labeled datasets and for summarization this hurdle of labeled data was surmounted by Cheng and Lapata (2016), through the creation of a labeled dataset of news stories from CNN and Daily Mail consisting of around 280,000 documents and human-generated summaries.

Recurrent neural networks with *encoder-decoder* architecture (Sutskever et al., 2014) have been successful in a variety of NLP tasks where an encoder obtains representations of input sequences and a decoder generates target sequences. Attention mechanisms (Bahdanau et al., 2015) are used to model the effects of different loci in the input sequence during decoding. Pointer networks (Vinyals et al., 2015) use this mechanism to obtain target sequences wherein each decoding step is used to *point* to elements of the input sequence. This pointing ability has been effectively utilized by state-of-the-art extractive and abstractive summarizers (Cheng and Lapata, 2016; Nallapati et al., 2016; See et al., 2017).

In this work, we design SWAP-NET a new deep learning model for extractive summarization. Similar to previous models, we use an encoder-decoder architecture with attention mechanism to select important sentences. Our key contribution is to design an architecture that utilizes key words in the selection process. Salient sentences of a document, that are useful in summaries, often contain key words and, to our knowledge, none of the previous models have explicitly modeled this interaction. We model this interaction through a two-level encoder and decoder, one for words and the other for sentences. An attention-based mechanism, similar to that of Pointer Networks, is used to learn important words and sentences from labeled data. A switch mechanism is used to select between words and sentences during decoding and the final summary is generated using a combination of selected sentences and words. We demonstrate the efficacy of our model on the CNN/Daily Mail corpus where it outperforms state-of-the-art extractive summarizers. Our experiments also suggest that the semantic redundancy in SWAP-NET generated summaries is comparable to that of human-generated summaries.

2 Problem Formulation

Let D denote an input document, comprising of a sequence of N sentences: $s_1, \ldots, s_N$. Ignoring sentence boundaries, let $w_1, \ldots, w_n$ be the sequence of n words in document D. An extractive summary aims to obtain a subset of the input sentences that forms a salient summary.

We use the interaction between words and sentences in a document to predict important words and sentences. Let the target sequence of indices of important words and sentences be $V = v_1, \ldots, v_m$, where each index v_j can point to either a sentence or a word in an input document. We design a supervised sequence-to-sequence recurrent neural network model, SWAP-NET, that uses these target sequences (of sentences and words) to learn salient sentences and key words. Our objective is to find SWAP-NET model parameters M that maximize the probability $p(V|M, D) = \prod_j p(v_j|v_1, \ldots, v_{j-1}, M, D) = \prod_j p(v_j|v_{<j}, M, D)$. We omit M in the following to simplify notation. SWAP-NET predicts both key words and salient sentences, that are subsequently used for extractive summary generation.

3 Background

We briefly describe Pointer Networks (Vinyals et al., 2015). Our approach, detailed in the following sections, uses a similar attention mechanism.

Given a sequence of n vectors $X = x_1, \ldots. x_n$ and a sequence of indices $R = r_1, \ldots. r_m$, each between 1 and n, the Pointer Network is an encoder-decoder architecture trained to maximize $p(R|X;\theta) = \prod_{j=1}^{m} p_\theta(r_j|r_1, \ldots. r_{j-1}, X;\theta)$, where θ denotes the model parameters. Let the encoder and decoder hidden states be $(e_1, \ldots., e_n)$ and $(d_1, \ldots., d_m)$ respectively. The attention vector at each output step j is computed as follows:

$$u_i^j = v^T \tanh(W_e e_i + W_d d_j),\ i \in (1, \ldots, n)$$

$$\alpha_i^j = \text{softmax}(u_i^j),\ i \in (1, \ldots, n)$$

The softmax normalizes vector u^j to be an *attention* mask over inputs. In a pointer network, the same attention mechanism is used to select one of the n input vectors with the highest probability, at each decoding step, thus effectively *pointing* to an input:

$$p(r_j|r_1, \ldots. r_{j-1}, X) = \text{softmax}(u^j)$$

Here, v, W_d, and W_e are learnable parameters of the model.

4 SWAP-NET

We use an encoder-decoder architecture with an attention mechanism similar to that of Pointer Networks. To model the interaction between words and sentences in a document we use two encoders and decoders, one at the word level and the other at the sentence level. The sentence-level decoder learns to *point* to important sentences while the

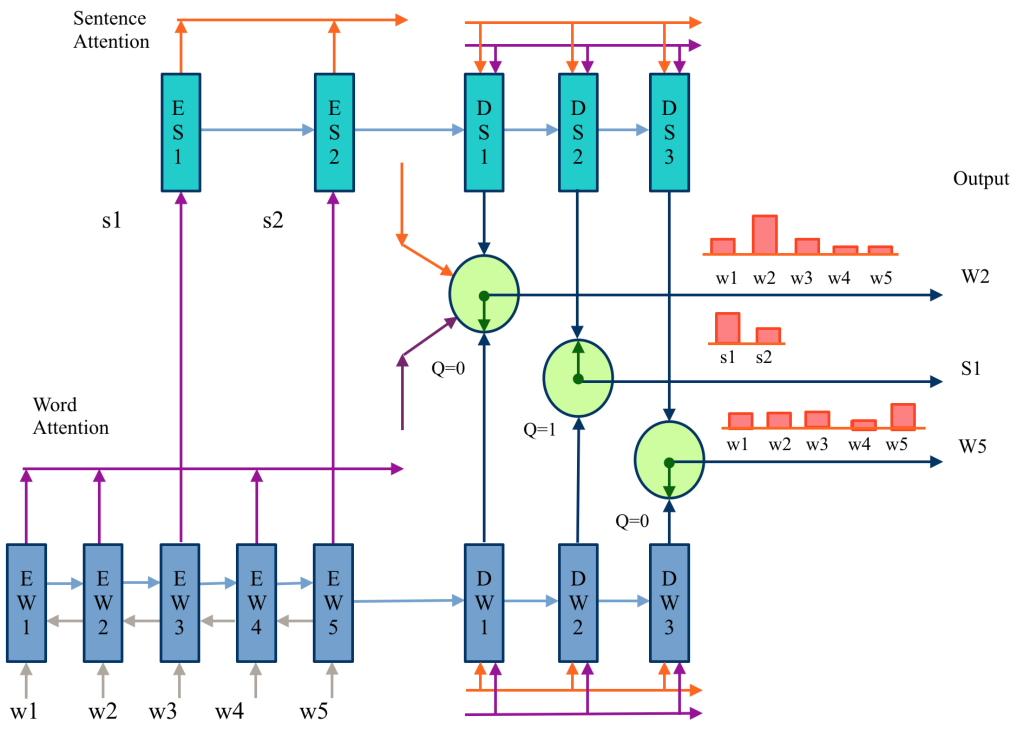


Figure 1: SWAP-NET architecture. EW: word encoder, ES: sentence encoder, DW: word decoder, DS: sentence decoder, Q: switch. Input document has words $[w1, \ldots, w5]$ and sentences $[s1, s2]$. Target sequence shown: $v_1 = w2, v_2 = s1, v_3 = w5$. Best viewed in color.

word-level decoder learns to *point* to important words. A switch mechanism is trained to select either a word or a sentence at each decoding step. The final summary is created using the output words and sentences. We now describe the details of the architecture.

4.1 Encoder

We use two encoders: a bi-directional LSTM at the word level and a LSTM at the sentence level. Each word w_i is represented by a K-dimensional embedding (e.g., via word2vec), denoted by x_i. The word embedding x_i is encoded as e_i using bi-directional LSTM for $i = 1, \ldots, n$. The vector output of BiLSTM at the end of a sentence is used to represent that entire sentence, which is further encoded by the sentence-level LSTM as $E_k = \text{LSTM}(e_{k^l}, E_{k-1})$, where k^l is the index of the last word in the k^{th} sentence in D and E_k is the hidden state at the k^{th} step of LSTM, for $k = 1, \ldots, N$. See figure 1.

4.2 Decoder

We use two decoders – a sentence-level and a word-level decoder, that are both LSTMs, with each decoder *pointing* to sentences and words respectively (similar to a pointer network). Thus, we can consider the output of each decoder step to be an index in the input sequence to the encoder. Let m be the number of steps in each decoder. Let $T_1, \ldots, T_m$ be the sequence of indices generated by the sentence-level decoder, where each index $T_j \in \{1, \ldots, N\}$; and let $t_1, \ldots, t_m$ be the sequence of indices generated by the word-level decoder, where each index $t_j \in \{1, \ldots, n\}$.

4.3 Network Details

At the j^{th} decoding step, we have to select a sentence or a word which is done through a binary switch Q_j that has two states $Q_j = 0$ and $Q_j = 1$ to denote word and sentence selection respectively. So, we first determine the switch probability $p(Q_j | v_{<j}, D)$. Let α^s_{kj} denote the probability of selecting the k^{th} input sentence at the j^{th} decoding step of sentence decoder:

$$\alpha^s_{kj} = p(T_j = k | v_{<j}, Q_j = 1, D),$$

and let α^w_{ij} denote the probability of selecting the i^{th} input word at the j^{th} decoding step of word decoder:

$$\alpha^w_{ij} = p(t_j = i | v_{<j}, Q_j = 0, D),$$

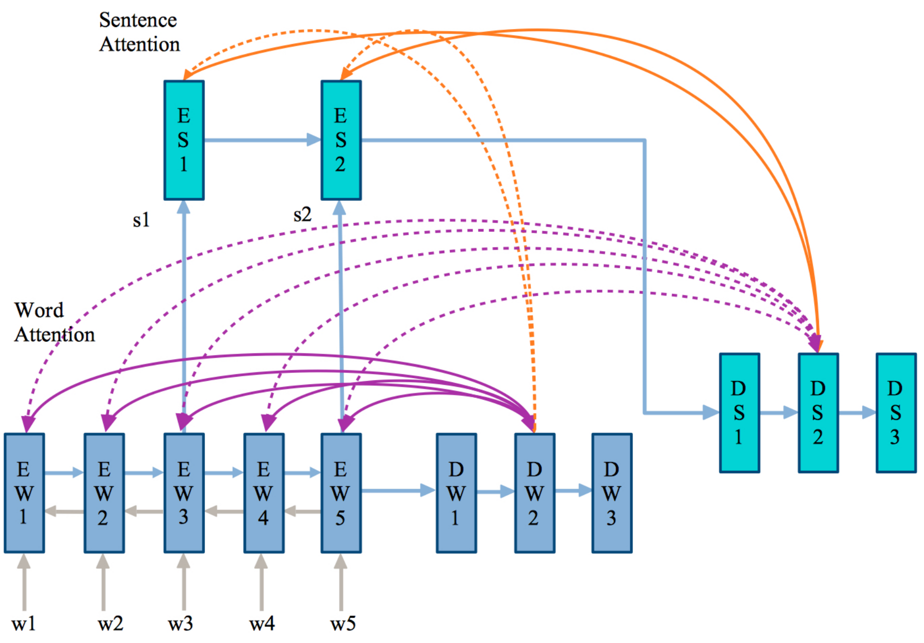


Figure 2: Illustration of word and sentence level attention in the second decoder step (Eq. 1 and Eq. 2). Purple: attention on words, Orange: attention on sentences, Unidirectional dotted arrows: attention from previous step, Bidirectional arrows: attention from previous and to present step. Best viewed in color.

both conditional on the corresponding switch selection. We set v_j based on the probability values:

$$v_j = \begin{cases} k = \arg\max_k p^s_{kj} & \text{if } \max_k p^s_{kj} > \max_i p^w_{ij} \\ i = \arg\max_i p^w_{ij} & \text{if } \max_i p^w_{ij} > \max_k p^s_{kj} \end{cases}$$

$$p^s_{kj} = \alpha^s_{kj} p(Q_j = 1 | v_{<j}, D),$$

$$p^w_{ij} = \alpha^w_{ij} p(Q_j = 0 | v_{<j}, D).$$

These probabilities are obtained through the attention weight vectors at the word and sentence levels and the switch probabilities:

$$\alpha^w_{ij} = \text{softmax}(v_t^T \phi(w_h h_j + w_t e_i)),$$

$$\alpha^s_{kj} = \text{softmax}(V_T^T \phi(W_H H_j + W_T E_k)).$$

Parameters v_t, w_h, w_t, V_T, W_H and W_T are trainable parameters. Parameters h_j and H_j are the hidden vectors at the j^{th} step of the word-level and sentence-level decoder respectively defined as:

$$h_j = LSTM(h_{j-1}, a_{j-1}, \phi(A_{j-1})) \qquad (1)$$

$$H_j = LSTM(H_{j-1}, A_{j-1}, \phi(a_{j-1})) \qquad (2)$$

where $a_j = \sum_{i=0}^{n} \alpha^w_{ij} e_i$, $A_j = \sum_{k=0}^{N} \alpha^s_{kj} E_k$. The non-linear transformation, ϕ (we choose tanh), is used to connect the word-level encodings to the sentence decoder and the sentence-level encodings to the word decoder. Specifically, the word-level decoder updates its state by considering a sum of sentence encodings, weighted by the attentions from the previous state and *mutatis mutandis* for the sentence-level decoder.

The switch probability $p(Q_j | v_{<j}, D)$ at the j^{th} decoding step is given by:

$$p(Q_j = 1 | v_{<j}, D) = \sigma(w_Q^T(H_{j-1}, A_{j-1}, \phi(h_{j-1}, a_{j-1})))$$

$$p(Q_j = 0 | v_{<j}, D) = 1 - p(Q_j = 1 | v_{<j}, D)$$

where w_Q is a trainable parameter and σ denotes the sigmoid function and ϕ is the chosen non-linear transformation (tanh).

During training the loss function l_j at j^{th} step is set to $l_j = -\log(p^s_{kj} q^s_j + p^w_{ij} q^w_j) - \log p(Q_j | v_{<j}, D)$. Note that at each decoding step, switch is either $q^w_j = 1, q^s_j = 0$ if the j^{th} output is a word or $q^w_j = 0, q^s_j = 1$ if the j^{th} output is a sentence. The switch probability is also considered in the loss function.

4.4 Summary Generation

Given a document whose summary is to be generated, its sentences and words are given as input to the trained encoder. At the j^{th} decoding step, either a sentence or a word is chosen based on the probability values α_{kj}^{s} and α_{ij}^{w} and the switch probability $p(Q_j|v_{<j}, D)$. We assign importance scores to the selected sentences based on their probability values during decoding as well as the probabilities of the selected words that are present in the selected sentences. Thus sentences with words selected by the decoder are given higher importance. Let the k^{th} input sentence s_k be selected at the j^{th} decoding step and i^{th} input word w_i be selected at the l^{th} decoding step. Then the importance of s_k is defined as

$$I(s_k) = \alpha_{kj}^{s} + \lambda \sum_{w_i \in s_k} \alpha_{il}^{w} \qquad (3)$$

In our experiments we choose $\lambda = 1$. The final summary consists of three sentences with the highest importance scores.

5 Related Work

Traditional approaches to extractive summarization rely on human-engineered features based on, for example, part of speech (Erkan and Radev, 2004) and term frequency (Nenkova et al., 2006). Sentences in the input document are scored using these features, ranked and then selected for the final summary. Methods used for extractive summarization include graph-based approaches (Mihalcea, 2005) and Integer Linear Programming (Gillick and Favre, 2009). There are many classifier-based approaches that select sentences for the extractive summary using methods such as Conditional Random Fields (Shen et al., 2007) and Hidden Markov models (Conroy and O'leary, 2001). A review of these classical approaches can be found in Nenkova et al. (2011).

End-to-end deep learning based neural models that can effectively learn from text data, without human-crafted features, have witnessed rapid development, resulting in improved performance in multiple areas such as machine translation (Chen et al., 2017) and question-answering (Hao et al., 2017), to name a few. Large labelled corpora based on news stories from CNN and Daily Mail, with human generated summaries have become available (Cheng and Lapata, 2016), that have spurred the use of deep learning models in summarization. Recurrent neural network based architectures have been designed for both extractive (Cheng and Lapata, 2016; Nallapati et al., 2017) and abstractive (See et al., 2017; Tan et al., 2017) summarization problems. Among these, the work of Cheng and Lapata (2016) and Nallapati et al. (2017) are closest to our work on extractive single-document summarization.

An encoder-decoder architecture with an attention mechanism similar to that of a pointer network is used by Cheng and Lapata (2016). Their hierarchical encoder uses a CNN at the word level leading to sentence representations that are used in an RNN to obtain document representations. They use a hierarchical attention model where the first level decoder predicts salient sentences used for an extractive summary and based on this output, the second step predicts keywords which are used for abstractive summarization. Thus they do not use key words for extractive summarization and for abstractive summarization they generate key words based on sentences predicted independently of key words. SWAP-NET, in contrast, is simpler using only two-level RNNs for word and sentence level representations in both the encoder and decoder. In our model we predict both words and sentences in such a way that their attentions interact with each other and generate extractive summaries considering both the attentions. By modeling the interaction between these key words and important sentences in our decoder architecture, we are able to extract sentences that are closer to the gold summaries.

SummaRuNNer, the method developed by Nallapati et al. (2017) is not similar to our method in its architecture but only in the aim of extractive summary generation. It does not use an encoder-decoder architecture; instead it is an RNN based binary classifier that decides whether or not to include a sentence in the summary. The RNN is multi-layered representing inputs, words, sentences and the final sentence labels. The decision of selecting a sentence at each step of the RNN is based on the content of the sentence, salience in the document, novelty with respect to previously selected sentences and other positional features. Their approach is considerably simpler than that of Cheng and Lapata (2016) but obtains summaries closer to the gold summaries, and additionally, facilitates interpretable visualization and training from abstractive summaries. Their experiments show improved performance over both abstractive and extractive summarizers from several previous models (Nallapati et al., 2017).

We note that several elements of our architecture have been introduced and used in earlier work. Pointer networks (Vinyals et al., 2015) used the attention mechanism of (Bahdanau et al., 2015) to solve combinatorial optimization problems. They have also been used to *point* to sentences in extractive (Cheng and Lapata, 2016) and abstractive (Nallapati et al., 2016; See et al., 2017) summarizers. The switch mechanism was introduced to incorporate rare or out-of-vocabulary words (Gulcehre et al., 2016) and are used in several summarizers (e.g. (Nallapati et al., 2016)). However, we use it to select between word and sentence level decoders in our model.

The importance of all the three interactions: (i) sentence-sentence, (ii) word-word and (iii) sentence-word, for summarization, have been studied by Wan et al. (2007) using graph-based approaches. In particular, they show that methods that account for saliency using both the following considerations perform better than methods that consider either one of them alone, and SWAP-NET is based on the same principles.

- A sentence should be salient if it is heavily linked with other salient sentences, and a word should be salient if it is heavily linked with other salient words.
- A sentence should be salient if it contains many salient words, and a word should be salient if it appears in many salient sentences.

6 Data and Experiments

6.1 Experimental Settings

In our experiments the maximum number of words per document is limited to 800, and the maximum number of sentences per document to 50 (padding is used to maintain the length of word sequences). We also use the symbols <GO> and <EOS> to indicate start and end of prediction by decoders. The total vocabulary size is 150,000 words.

We use word embeddings of dimension 100 pretrained using word2vec (Mikolov et al., 2013) on the training dataset. We fix the LSTM hidden state size at 200. We use a batch size of 16 and the ADAM optimizer (Kingma and Ba, 2015) with parameters: learning rate = 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$ to train SWAP-NET. We employ gradient clipping to regularize our model and an early stopping criterion based on the validation loss.

During training we find that SWAP-NET learns to predict important sentences faster than to predict words. To speed up learning of word probabilities, we add the term $-\log \alpha^w_{ij}$ to our loss function l_j in the final iterations of training. It is possible to get the same sentence or word in multiple (usually consecutive) decoding steps. In that case, in Eq. 3 we consider the maximum value of alpha obtained across these steps and calculate maximum scores of distinct sentences and words.

We select 3 top scoring sentences for the summary, as there are 3.11 sentences on average in the gold summary of the training set (similar to settings used by others, e.g., (Narayan et al., 2017)).

6.2 Baselines

Two state-of-the-art methods for extractive summarization are **SummaRuNNer** (Nallapati et al., 2017) and **NN**, the neural summarizer of Cheng and Lapata (2016). SummaRuNNer can also provide extractive summaries while being trained abstractively (Nallapati et al., 2017); we denote this method by **SummaRuNNer-abs**. In addition, we compare our method with the **Lead-3** summary which consists of the first three sentences from each document. We also compare our method with an abstractive summarizer that uses a similar attention-based encoder-decoder architecture (Nallapati et al., 2016), denoted by **ABS**.

6.3 Benchmark Datasets

For our experiments, we use the CNN/DailyMail corpus (Hermann et al., 2015). We use the anonymized version of this dataset, from Cheng and Lapata (2016), which has labels for important sentences, that are used for training. To obtain labels for words, we extract keywords from each gold summary using RAKE, an unsupervised keyword extraction method (Rose et al., 2010). These keywords are used to label words in the corresponding input document during training. We replace numerical values in the documents by zeros to limit the vocabulary size.

We have 193,986 training documents, 12,147 validation documents and 10,346 test documents from the DailyMail corpus and 83,568 training documents, 1,220 validation documents and 1,093 test documents from CNN subset with labels for sentences and words.

6.4 Evaluation Metrics

We use the ROUGE toolkit (Lin and Hovy, 2003) for evaluation of the generated summaries in comparison to the gold summaries. We use three variants of this metric: ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) that are computed by matching unigrams, bigrams and longest common subsequences respectively between the two summaries. To compare with (Cheng and Lapata, 2016) and (Nallapati et al., 2017) we use limited length ROUGE recall at 75 and 275 bytes for the Daily-Mail test set, and full length ROUGE-F1 score, as reported by them.

6.5 Results on Benchmark Datasets

Performance on Daily Mail Data

Models	R1	R2	RL
Lead-3	21.9	7.2	11.6
NN	22.7	8.5	12.5
SummaRuNNer-abs	23.8	9.6	13.3
SummaRuNNer	26.2	**10.8**	**14.4**
SWAP-NET	**26.4**	10.7	**14.4**

Table 1: Performance on Daily-Mail test set using the limited length recall of Rouge at 75 bytes.

Models	R1	R2	RL
Lead-3	40.5	14.9	32.6
NN	42.2	17.3	34.8
SummaRuNNer-abs	40.4	15.5	32.0
SummaRuNNer	42.0	16.9	34.1
SWAP-NET	**43.6**	**17.7**	**35.5**

Table 2: Performance on Daily-Mail test set using the limited length recall of Rouge at 275 bytes.

Table 1 shows the performance of SWAP-NET, state-of-the-art baselines NN and SummaRuNNer and other baselines, using ROUGE recall with summary length of 75 bytes, on the entire Daily Mail test set. The performance of SWAP-NET is comparable to that of SummaRuNNer and better than NN and other baselines. Table 2 compares the same algorithms using ROUGE recall with summary length of 275 bytes. SWAP-NET outperforms both state-of-the-art summarizers SummaRuNNer as well as NN.

Performance on CNN/DailyMail Data

SWAP-NET has the best performance on the combined CNN and Daily Mail corpus, outperforming the previous best reported F-score by SummaRuNNer, as seen in table 3, with a consistent improvement of over 2 ROUGE points in all three metrics.

Models	R1	R2	RL
Lead-3	39.2	15.7	35.5
ABS	35.4	13.3	32.6
SummaRuNNer-abs	37.5	14.5	33.4
SummaRuNNer	39.6	16.2	35.3
SWAP-NET	**41.6**	**18.3**	**37.7**

Table 3: Performance on CNN and Daily-Mail test set using the full length Rouge F score.

6.6 Discussion

SWAP-NET outperforms state-of-the-art extractive summarizers SummaRuNNer (Nallapati et al., 2017) and NN (Cheng and Lapata, 2016) on benchmark datasets. Our model is similar, although simpler, than that of NN and the main difference between SWAP-NET and these baselines is its explicit modeling of the interaction between key words and salient sentences.

Automatic keyword extraction has been studied extensively (Hasan and Ng, 2014). We use a popular and well tested method, RAKE (Rose et al., 2010) to obtain key words in the training documents. A disadvantage with such methods is that they do not guarantee representation, via extracted keywords, of all the topics in the text (Hasan and Ng, 2014). So, if RAKE key words are directly applied to the input test document (without using word decoder trained on RAKE words, obtained from gold summary as done in SWAP-NET), then there is a possibility of missing sentences from the missed topics. So, we train SWAP-NET to predict key words and also model their interactions with sentences.

Statistics	Lead-3	SWAP-NET
KW coverage	61.6%	73.8%
Sentences with KW	92.2%	98%

Table 4: Key word (KW) statistics per summary (average percentage) from 500 documents in Daily Mail test set. See text for definitions.

We investigate the importance of modeling this interaction and the role of key words in the final summary. Table 4 shows statistics that reflect the importance of key words in extractive summaries. *Key word coverage* measures the proportion of key

Title: @entity19 vet surprised reason license plate denial
Gold Summary: @entity9 of @entity10 , @entity1 , wanted to get ' @entity11 - 0 ' put on a license plate . that would have commemorated both @entity9 getting the @entity8 in 0 and his @entity16 . the @entity1 @entity21 denied his request , citing state regulations prohibiting the use of the number 0 because of its indecent connotations .
SWAP-NET Summary: **@entity9 of @entity10 wanted to get ' @entity11 ' put on a license plate** , the @entity14 newspaper of @entity10 reported . **that would have commemorated both @entity9 getting the @entity8 in 0 and his @entity16** , according to the newspaper . **the @entity1 @entity21 denied his request , citing state regulations prohibiting the use of the number 0 because of its indecent connotations** @entity9 had been an armored personnel carrier 's gunner during his time in the @entity29 .
SWAP-NET Key words: @entity1, @entity9, @entity8, citing, number, year, indecent, personalized, war, surprised, plate, @entity14, @entity11, @entity10, regulations, reported, wanted, connotations, license, request, according,@entity21, armored, @entity16
Lead 3 Summary: a @entity19 war veteran in @entity1 has said he 's surprised over the reason for the denial of his request for a personalized license plate commemorating the year he was wounded and awarded a @entity8 . **@entity9 of @entity10 wanted to get ' @entity11 ' put on a license plate** , the @entity14 newspaper of @entity10 reported . **that would have commemorated both @entity9 getting the @entity8 in 0 and his @entity16** , according to the newspaper .

Table 5: Sample gold summary and summaries generated by SWAP-NET and Lead-3. Key words are highlighted, bold font indicates overlap with gold summary.

words from those in the gold summary present in the generated summary. SWAP-NET obtains nearly 74% of the key words. In comparison Lead-3 has only about 62% of the key words from the gold summary.

Sentences with key words measures the proportion of sentences containing at least one key word. It is not surprising that in SWAP-NET summaries 98% of the sentences, on average, contain at least one key word: this is by design of SWAP-NET. However, note that Lead-3 which has poorer performance in all the benchmark datasets has much fewer sentences with key words. This highlights the importance of key words in finding salient sentences for extractive summaries.

Gold summary	Lead-3	SWAP-NET
0.81	0.553	0.8

Table 6: Average pairwise cosine distance between paragraph vector representations of sentences in summaries.

We also find the SWAP-NET obtains summaries that have less semantic redundancy. Table 6 shows the average distance between pairs of sentences from the gold summary, and summaries generated from SWAP-NET and Lead-3. Distances are measured using cosine distance of paragraph vectors of each sentence (Le and Mikolov, 2014) from randomly selected 500 documents of the Daily Mail test set. Paragraph vectors have been found to be effective semantic representations of sentences (Le and Mikolov, 2014) and experiments in (Dai et al., 2015) also show that paragraph vectors can be effectively used to measure semantic similarity using cosine distance. For training we use GENSIM (Řehůřek and Sojka, 2010) with embedding size 200 and initial learning rate 0.025. The model is trained on 500 documents from DailyMail dataset for 10 epochs and learning rate is decreased by 0.002 at each epoch.

The average pair-wise distance of SWAP-NET is very close to that of the gold summary, both nearly 0.8. In contrast, the average pairwise distance in Lead-3 summaries is 0.553 indicating higher redundancy. This highly desirable feature of SWAP-NET is likely due to use of of key words, that is affecting the choice of sentences in the final summary.

Table 5 shows a sample gold summary from the Daily Mail dataset and the generated summary from SWAP-NET and, for comparison, from Lead-3. We observe the presence of key words in all the overlapping segments of text with the gold summary indicating the importance of key words in finding salient sentences. Modeling this interaction, we believe, is the reason for the superior performance of SWAP-NET in our experiments.

An implementation of SWAP-NET and all the generated summaries from the test sets are available online in a github repository[1].

7 Conclusion

We present SWAP-NET, a neural sequence-to-sequence model for extractive summarization that outperforms state-of-the-art extractive summarizers SummaRuNNer (Nallapati et al., 2017) and NN (Cheng and Lapata, 2016) on large scale benchmark datasets. The architecture of SWAP-NET is simpler than that of NN but due to its effective modeling of interaction between salient sentences and key words in a document, SWAP-NET achieves superior performance. SWAP-NET models this interaction using a new two-level pointer network based architecture with a switching mechanism. Our experiments also suggest that modeling sentence-keyword interaction has the desirable property of less semantic redundancy in summaries generated by SWAP-NET.

8 Acknowledgment

The authors thank the ACL reviewers for their valuable comments. Vaibhav Rajan acknowledges the support from Singapore Ministry of Education Academic Research Fund Tier 1 towards funding this research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 406–407. ACM.

Andrew M Dai, Christopher Olah, and Quoc V Le. 2015. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, Association for Computational Linguistics, pages 10–18.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1262–1273.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[1]https://github.com/aishj10/swap-net

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*, pages 1188–1196.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78.

Rada Mihalcea. 2005. Language independent extractive summarization. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 49–52.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Nikos Papasarantopoulos, Shay B Cohen, and Mirella Lapata. 2017. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 573–580.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *Proceedings of International Joint Conferences on Artificial Intelligence*, volume 7, pages 2862–2867.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1171–1181.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 552–559.

Retrieve, Rerank and Rewrite: Soft Template Based Neural Summarization

Ziqiang Cao[1,2] **Wenjie Li**[1,2] **Furu Wei**[3] **Sujian Li**[4]
[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[2]Hong Kong Polytechnic University Shenzhen Research Institute, China
[3]Microsoft Research, Beijing, China
[4]Key Laboratory of Computational Linguistics, Peking University, MOE, China
{cszqcao, cswjli}@comp.polyu.edu.hk
fuwei@microsoft.com lisujian@pku.edu.cn

Abstract

Most previous seq2seq summarization systems purely depend on the source text to generate summaries, which tends to work unstably. Inspired by the traditional template-based summarization approaches, this paper proposes to use existing summaries as soft templates to guide the seq2seq model. To this end, we use a popular IR platform to Retrieve proper summaries as candidate templates. Then, we extend the seq2seq framework to jointly conduct template Reranking and template-aware summary generation (Rewriting). Experiments show that, in terms of informativeness, our model significantly outperforms the state-of-the-art methods, and even soft templates themselves demonstrate high competitiveness. In addition, the import of high-quality external summaries improves the stability and readability of generated summaries.

1 Introduction

The exponentially growing online information has necessitated the development of effective automatic summarization systems. In this paper, we focus on an increasingly intriguing task, i.e., abstractive sentence summarization (Rush et al., 2015a), which generates a shorter version of a given sentence while attempting to preserve its original meaning. It can be used to design or refine appealing headlines. Recently, the application of the attentional sequence-to-sequence (seq2seq) framework has attracted growing attention and achieved state-of-the-art performance on this task (Rush et al., 2015a; Chopra et al., 2016; Nallapati et al., 2016).

Most previous seq2seq models purely depend on the source text to generate summaries. However, as reported in many studies (Koehn and Knowles, 2017), the performance of a seq2seq model deteriorates quickly with the increase of the length of generation. Our experiments also show that seq2seq models tend to "lose control" sometimes. For example, 3% of summaries contain less than 3 words, while there are 4 summaries repeating a word for even 99 times. These results largely reduce the informativeness and readability of the generated summaries. In addition, we find seq2seq models usually focus on copying source words in order, without any actual "summarization". Therefore, we argue that, the free generation based on the source sentence is not enough for a seq2seq model.

Template based summarization (e.g., Zhou and Hovy (2004)) is a traditional approach to abstractive summarization. In general, a template is an incomplete sentence which can be filled with the input text using the manually defined rules. For instance, a concise template to conclude the stock market quotation is: *[REGION] shares [open/close] [NUMBER] percent [lower/higher]*, e.g., "hong kong shares close #.# percent lower". Since the templates are written by humans, the produced summaries are usually fluent and informative. However, the construction of templates is extremely time-consuming and requires a plenty of domain knowledge. Moreover, it is impossible to develop all templates for summaries in various domains.

Inspired by retrieve-based conversation systems (Ji et al., 2014), we assume the golden summaries of the similar sentences can provide a reference point to guide the input sentence summarization process. We call these existing summaries **soft templates** since no actual rules are needed to build new summaries from them. Due to the strong rewriting ability of the seq2seq framework (Cao et al., 2017a), in this paper, we propose to combine the seq2seq and template based summarization approaches. We call our summarization system Re^{3}Sum, which consists of three modules: **Re**trieve, **Re**rank and **Re**write. We utilize a widely-used Information Retrieval (IR) platform to find out candidate soft templates from the training corpus. Then, we extend the seq2seq model to jointly learn template saliency measurement (Rerank) and final summary generation (Rewrite). Specifically, a Recurrent Neural Network (RNN) encoder is applied to convert the input sentence and each candidate template into hidden states. In Rerank, we measure the informativeness of a candidate template according to its hidden state relevance to the input sentence. The candidate template with the highest predicted informativeness is regarded as the actual soft template. In Rewrite, the summary is generated according to the hidden states of both the sentence and template.

We conduct extensive experiments on the popular Gigaword dataset (Rush et al., 2015b). Experiments show that, in terms of informativeness, Re^{3}Sum significantly outperforms the state-of-the-art seq2seq models, and even soft templates themselves demonstrate high competitiveness. In addition, the import of high-quality external summaries improves the stability and readability of generated summaries.

The contributions of this work are summarized as follows:

- We propose to introduce soft templates as additional input to improve the readability and stability of seq2seq summarization systems. Code and results can be found at `http://www4.comp.polyu.edu.hk/~cszqcao/`
- We extend the seq2seq framework to conduct template reranking and template-aware summary generation simultaneously.
- We fuse the popular IR-based and seq2seq-based summarization systems, which fully utilize the supervisions from both sides.

2 Method

As shown in Fig. 1, our summarization system consists of three modules, i.e., Retrieve, Rerank and Rewrite. Given the input sentence $\mathbf{x}$, the Retrieve module filters candidate soft templates $\mathbf{C} = \{\mathbf{r}_i\}$ from the training corpus. For validation and test, we regard the candidate template with the highest predicted saliency (a.k.a informativeness) score as the actual soft template $\mathbf{r}$. For training, we choose the one with the maximal actual saliency score in $\mathbf{C}$, which speeds up convergence and shows no obvious side effect in the experiments.

Then, we jointly conduct reranking and rewriting through a shared encoder. Specifically, both the sentence $\mathbf{x}$ and the soft template $\mathbf{r}$ are converted into hidden states with a RNN encoder. In the Rerank module, we measure the saliency of $\mathbf{r}$ according to its hidden state relevance to $\mathbf{x}$. In the Rewrite module, a RNN decoder combines the hidden states of $\mathbf{x}$ and $\mathbf{r}$ to generate a summary $\mathbf{y}$. More details will be described in the rest of this section

2.1 Retrieve

The purpose of this module is to find out candidate templates from the training corpus. We assume that similar sentences should hold similar summary patterns. Therefore, given a sentence $\mathbf{x}$, we find out its analogies in the corpus and pick their summaries as the candidate templates. Since the size of our dataset is quite large (over 3M), we leverage the widely-used Information Retrieve (IR) system Lucene[1] to index and search efficiently. We keep the default settings of Lucene[2] to build the IR system. For each input sentence, we select top 30 searching results as candidate templates.

2.2 Jointly Rerank and Rewrite

To conduct template-aware seq2seq generation (rewriting), it is a necessary step to encode both the source sentence $\mathbf{x}$ and soft template $\mathbf{r}$ into hidden states. Considering that the matching networks based on hidden states have demonstrated the strong ability to measure the relevance of two pieces of texts (e.g., Chen et al. (2016)), we propose to jointly conduct reranking and rewriting through a shared encoding step. Specifically, we employ a bidirectional Recurrent Neural Network (BiRNN) encoder (Cho et al., 2014) to read $\mathbf{x}$ and $\mathbf{r}$. Take the sentence $\mathbf{x}$ as an example. Its hidden state of the forward RNN at timestamp i can be

[1]`https://lucene.apache.org/`
[2]TextField with EnglishAnalyzer

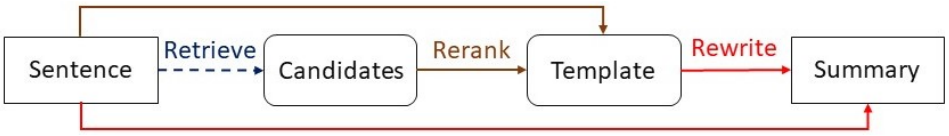


Figure 1: Flow chat of the proposed method. We use the dashed line for Retrieve since there is an IR system embedded.

represented by:

$$\overrightarrow{\mathbf{h}}_i^x = \text{RNN}(x_i, \overrightarrow{\mathbf{h}}_{i-1}^x) \quad (1)$$

The BiRNN consists of a forward RNN and a backward RNN. Suppose the corresponding outputs are $[\overrightarrow{\mathbf{h}}_1^x; \cdots; \overrightarrow{\mathbf{h}}_{-1}^x]$ and $[\overleftarrow{\mathbf{h}}_1^x; \cdots; \overleftarrow{\mathbf{h}}_{-1}^x]$, respectively, where the index "-1" stands for the last element. Then, the composite hidden state of a word is the concatenation of the two RNN representations, i.e., $\mathbf{h}_i^x = [\overrightarrow{\mathbf{h}}_i^x; \overleftarrow{\mathbf{h}}_i^x]$. The entire representation for the source sentence is $[\mathbf{h}_1^x; \cdots; \mathbf{h}_{-1}^x]$. Since a soft template $\mathbf{r}$ can also be regarded as a readable concise sentence, we use the same BiRNN encoder to convert it into hidden states $[\mathbf{h}_1^r; \cdots; \mathbf{h}_{-1}^r]$.

2.2.1 Rerank

In Retrieve, the template candidates are ranked according to the text similarity between the corresponding indexed sentences and the input sentence. However, for the summarization task, we expect the soft template $\mathbf{r}$ resembles the actual summary $\mathbf{y}^*$ as much as possible. Here we use the widely-used summarization evaluation metrics ROUGE (Lin, 2004) to measure the actual saliency $s^*(\mathbf{r}, \mathbf{y}^*)$ (see Section 3.2). We utilize the hidden states of $\mathbf{x}$ and $\mathbf{r}$ to predict the saliency s of the template. Specifically, we regard the output of the BiRNN as the representation of the sentence or template:

$$\mathbf{h}_x = [\overleftarrow{\mathbf{h}}_1^x; \overrightarrow{\mathbf{h}}_{-1}^x] \quad (2)$$

$$\mathbf{h}_r = [\overleftarrow{\mathbf{h}}_1^r; \overrightarrow{\mathbf{h}}_{-1}^r] \quad (3)$$

Next, we use Bilinear network to predict the saliency of the template for the input sentence.

$$s(\mathbf{r}, \mathbf{x}) = \text{sigmoid}(\mathbf{h}_r \mathbf{W}_s \mathbf{h}_x^{\mathbf{T}} + b_s), \quad (4)$$

where $\mathbf{W}_s$ and b_s are parameters of the Bilinear network, and we add the sigmoid activation function to make the range of s consistent with the actual saliency s^*. According to Chen et al. (2016), Bilinear outperforms multi-layer forward neural networks in relevance measurement. As shown later, the difference of s and s^* will provide additional supervisions for the seq2seq framework.

2.2.2 Rewrite

The soft template $\mathbf{r}$ selected by the Rerank module has already competed with the state-of-the-art method in terms of ROUGE evaluation (see Table 4). However, $\mathbf{r}$ usually contains a lot of named entities that does not appear in the source (see Table 5). Consequently, it is hard to ensure that the soft templates are faithful to the input sentences. Therefore, we leverage the strong rewriting ability of the seq2seq model to generate more faithful and informative summaries. Specifically, since the input of our system consists of both the sentence and soft template, we use the concatenation function[3] to combine the hidden states of the sentence and template:

$$\mathbf{H}_c = [\mathbf{h}_1^x; \cdots; \mathbf{h}_{-1}^x; \mathbf{h}_1^r; \cdots; \mathbf{h}_{-1}^r] \quad (5)$$

The combined hidden states are fed into the prevailing attentional RNN decoder (Bahdanau et al., 2014) to generate the decoding hidden state at the position t:

$$\mathbf{s}_t = \text{Att-RNN}(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{H}_c), \quad (6)$$

where y_{t-1} is the previous output summary word. Finally, a $softmax$ layer is introduced to predict the current summary word:

$$\mathbf{o_t} = softmax(\mathbf{s}_t \mathbf{W}_o), \quad (7)$$

where $\mathbf{W}_o$ is a parameter matrix.

2.3 Learning

There are two types of costs in our system. For Rerank, we expect the predicted saliency $s(\mathbf{r}, \mathbf{x})$ close to the actual saliency $s^*(\mathbf{r}, \mathbf{y}^*)$. Therefore,

[3]We also attempted complex combination approaches such as the gate network Cao et al. (2017b) but failed to achieve obvious improvement. We assume the Rerank module has partially played the role of the gate network.

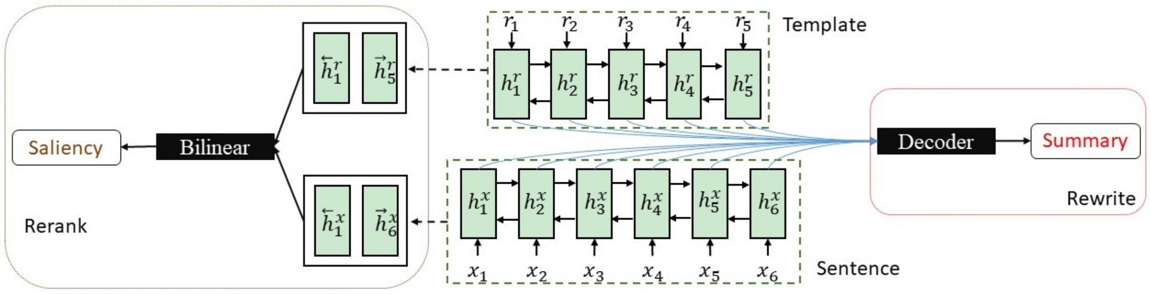


Figure 2: Jointly Rerank and Rewrite

we use the cross entropy (CE) between s and s^* as the loss function:

$$\begin{aligned} J_R(\theta) &= \text{CE}(s(\mathbf{r},\mathbf{x}), s^*(\mathbf{r},\mathbf{y}^*)) \\ &= -s^* \log s - (1-s^*)\log(1-s), \end{aligned} \tag{8}$$

where θ stands for the model parameters. For Rewrite, the learning goal is to maximize the estimated probability of the actual summary $\mathbf{y}^*$. We adopt the common negative log-likelihood (NLL) as the loss function:

$$\begin{aligned} J_G(\theta) &= -\log(p(\mathbf{y}^*|\mathbf{x},\mathbf{r})) \\ &= -\sum\nolimits_t \log(\mathbf{o}_t[y_t^*]) \end{aligned} \tag{9}$$

To make full use of supervisions from both sides, we combine the above two costs as the final loss function:

$$J(\theta) = J_R(\theta) + J_G(\theta) \tag{10}$$

We use mini-batch Stochastic Gradient Descent (SGD) to tune model parameters. The batch size is 64. To enhance generalization, we introduce dropout (Srivastava et al., 2014) with probability $p = 0.3$ for the RNN layers. The initial learning rate is 1, and it will decay by 50% if the generation loss does not decrease on the validation set.

3 Experiments

3.1 Datasets

We conduct experiments on the Annotated English Gigaword corpus, as with (Rush et al., 2015b). This parallel corpus is produced by pairing the first sentence in the news article and its headline as the summary with heuristic rules. All the training, development and test datasets can be downloaded at `https://github.com/harvardnlp/sent-summary`. The statistics of the Gigaword corpus is presented in Table 1.

Dataset	Train	Dev.	Test
Count	3.8M	189k	1951
AvgSourceLen	31.4	31.7	29.7
AvgTargetLen	8.3	8.3	8.8
COPY(%)	45	46	36

Table 1: Data statistics for English Gigaword. AvgSourceLen is the average input sentence length and AvgTargetLen is the average summary length. COPY means the copy ratio in the summaries (without stopwords).

3.2 Evaluation Metrics

We adopt ROUGE (Lin, 2004) for automatic evaluation. ROUGE has been the standard evaluation metric for DUC shared tasks since 2004. It measures the quality of summary by computing the overlapping lexical units between the candidate summary and actual summaries, such as uni-gram, bi-gram and longest common subsequence (LCS). Following the common practice, we report ROUGE-1 (uni-gram), ROUGE-2 (bi-gram) and ROUGE-L (LCS) F1 scores[4] in the following experiments. We also measure the actual saliency of a candidate template $\mathbf{r}$ with its combined ROUGE scores given the actual summary $\mathbf{y}^*$:

$$s^*(\mathbf{r},\mathbf{y}^*) = \text{RG}(\mathbf{r},\mathbf{y}^*) + \text{RG}(\mathbf{r},\mathbf{y}^*), \tag{11}$$

where "RG" stands for ROUGE for short.

ROUGE mainly evaluates informativeness. We also introduce a series of metrics to measure the summary quality from the following aspects:

LEN_DIF The absolute value of the length difference between the generated summaries and the actual summaries. We use mean value $\pm$ standard deviation to illustrate this item. The average value partially reflects the readability and informativeness, while the standard deviation links to stability.

[4]We use the ROUGE evaluation option: -m -n 2 -w 1.2

LESS_3 The number of the generated summaries, which contains less than three tokens. These extremely short summaries are usually unreadable.

COPY The proportion of the summary words (without stopwords) copied from the source sentence. A seriously large copy ratio indicates that the summarization system pays more attention to compression rather than required abstraction.

NEW_NE The number of the named entities that do not appear in the source sentence or actual summary. Intuitively, the appearance of new named entities in the summary is likely to bring unfaithfulness. We use Stanford CoreNLP (Manning et al., 2014) to recognize named entities.

3.3 Implementation Details

We use the popular seq2seq framework OpenNMT[5] as the starting point. To make our model more general, we retain the default settings of OpenNMT to build the network architecture. Specifically, the dimensions of word embeddings and RNN are both 500, and the encoder and decoder structures are two-layer bidirectional Long Short Term Memory Networks (LSTMs). The only difference is that we add the argument "-share_embeddings" to share the word embeddings between the encoder and decoder. This practice largely reduces model parameters for the monolingual task. On our computer (GPU: GTX 1080, Memory: 16G, CPU: i7-7700K), the training spends about 2 days.

During test, we use beam search of size 5 to generate summaries. We add the argument "-replace_unk" to replace the generated unknown words with the source word that holds the highest attention weight. Since the generated summaries are often shorter than the actual ones, we introduce an additional length penalty argument "-alpha 1" to encourage longer generation, like Wu et al. (2016).

3.4 Baselines

We compare our proposed model with the following state-of-the-art neural summarization systems:

ABS Rush et al. (2015a) used an attentive CNN encoder and a NNLM decoder to summarize the sentence.

ABS+ Rush et al. (2015a) further tuned the ABS model with additional hand-crafted features to balance between abstraction and extraction.

RAS-Elman As the extension of the ABS model, it used a convolutional attention-based encoder and a RNN decoder (Chopra et al., 2016).

Featseq2seq Nallapati et al. (2016) used a complete seq2seq RNN model and added the hand-crafted features such as POS tag and NER, to enhance the encoder representation.

Luong-NMT Chopra et al. (2016) implemented the neural machine translation model of Luong et al. (2015) for summarization. This model contained two-layer LSTMs with 500 hidden units in each layer.

OpenNMT We also implement the standard attentional seq2seq model with OpenNMT. All the settings are the same as our system. It is noted that OpenNMT officially examined the Gigaword dataset. We distinguish the official result[6] and our experimental result with suffixes "O" and "I" respectively.

FTSum Cao et al. (2017b) encoded the facts extracted from the source sentence to improve both the faithfulness and informativeness of generated summaries.

In addition, to evaluate the effectiveness of our joint learning framework, we develop a baseline named "PIPELINE". Its architecture is identical to Re^{3}Sum. However, it trains the Rerank module and Rewrite module in pipeline.

3.5 Informativeness Evaluation

Model	Perplexity
ABS†	27.1
RAS-Elman†	18.9
FTSum†	16.4
OpenNMT$_I$	13.2
PIPELINE	**12.5**
Re^{3}Sum	12.9

Table 2: Final perplexity on the development set. † indicates the value is cited from the corresponding paper. ABS+, Featseq2seq and Luong-NMT do not provide this value.

Let's first look at the final cost values (Eq. 9) on the development set. From Table 2, we can

[5] https://github.com/OpenNMT/OpenNMT-py

[6] http://opennmt.net/Models/

Model	RG-1	RG-2	RG-L
ABS†	29.55*	11.32*	26.42*
ABS+†	29.78*	11.89*	26.97*
Featseq2seq†	32.67*	15.59*	30.64*
RAS-Elman†	33.78*	15.97*	31.15*
Luong-NMT†	33.10*	14.45*	30.71*
FTSum†	**37.27**	17.65*	34.24
$\text{OpenNMT}_O^{\dagger}$	33.13*	16.09*	31.00*
OpenNMT_I	35.01*	16.55*	32.42*
PIPELINE	36.49	17.48*	33.90
Re^3Sum	37.04	**19.03**	**34.46**

Table 3: ROUGE F1 (%) performance. "RG" represents "ROUGE" for short. "*" indicates statistical significance of the corresponding model with respect to the baseline model on the 95% confidence interval in the official ROUGE script.

Type	RG-1	RG-2	RG-L
Random	2.81	0.00	2.72
First	24.44	9.63	22.05
Max	38.90	19.22	35.54
Optimal	52.91	31.92	48.63
Rerank	28.77	12.49	26.40

Table 4: ROUGE F1 (%) performance of different types of soft templates.

see that our model achieves much lower perplexity compared against the state-of-the-art systems. It is also noted that PIPELINE slightly outperforms Re^3Sum. One possible reason is that Re^3Sum additionally considers the cost derived from the Rerank module.

The ROUGE F1 scores of different methods are then reported in Table 3. As can be seen, our model significantly outperforms most other approaches. Note that, ABS+ and Featseq2seq have utilized a series of hand-crafted features, but our model is completely data-driven. Even though, our model surpasses Featseq2seq by 22% and ABS+ by 60% on ROUGE-2. When soft templates are ignored, our model is equivalent to the standard attentional seq2seq model OpenNMT_I. Therefore, it is safe to conclude that soft templates have great contribute to guide the generation of summaries.

Item	Template	OpenNMT	Re^3Sum
LEN_DIF	2.6±2.6	3.0±4.4	2.7±2.6
LESS_3	0	53	1
COPY(%)	31	80	74
NEW_NE	0.51	0.34	0.30

Table 5: Statistics of different types of summaries.

Type	RG-1	RG-2	RG-L
+Random	32.60	14.31	30.19
+First	36.01	17.06	33.21
+Max	41.50	21.97	38.80
+Optimal	46.21	26.71	43.19
+Rerank(Re^3Sum)	37.04	19.03	34.46

Table 6: ROUGE F1 (%) performance of Re^3Sum generated with different soft templates.

We also examine the performance of directly regarding soft templates as output summaries. We introduce five types of different soft templates:

Random An existing summary randomly selected from the training corpus.

First The top-ranked candidate template given by the Retrieve module.

Max The template with the maximal actual ROUGE scores among the 30 candidate templates.

Optimal An existing summary in the training corpus which holds the maximal ROUGE scores.

Rerank The template with the maximal predicted ROUGE scores among the 30 candidate templates. It is the actual soft template we adopt.

As shown in Table 4, the performance of Random is terrible, indicating it is impossible to use one summary template to fit various actual summaries. Rerank largely outperforms First, which verifies the effectiveness of the Rerank module. However, according to Max and Rerank, we find the Rerank performance of Re^3Sum is far from perfect. Likewise, comparing Max and First, we observe that the improving capacity of the Retrieve module is high. Notice that Optimal greatly exceeds all the state-of-the-art approaches. This finding strongly supports our practice of using existing summaries to guide the seq2seq models.

3.6 Linguistic Quality Evaluation

We also measure the linguistic quality of generated summaries from various aspects, and the results are present in Table 5. As can be seen from the rows "LEN_DIF" and "LESS_3", the performance of Re^3Sum is almost the same as that of soft templates. The soft templates indeed well guide the summary generation. Compared with

Source	grid positions after the final qualifying session in the indonesian motorcycle grand prix at the sentul circuit , west java , saturday : UNK
Target	indonesian motorcycle grand prix grid positions
Template	grid positions for **british** grand prix
OpenNMT	circuit
Re^{3}Sum	grid positions for indonesian grand prix
Source	india 's children are getting increasingly overweight and unhealthy and the government is asking schools to ban junk food , officials said thursday .
Target	indian government asks schools to ban junk food
Template	**skorean** schools to ban **soda** junk food
OpenNMT	india 's children getting fatter
Re^{3}Sum	indian schools to ban junk food

Table 7: Examples of generated summaries. We use Bold font to indicate the crucial rewriting behavior from the templates to generated summaries.

Re^{3}Sum, the standard deviation of LEN_DF is 0.7 times larger in OpenNMT, indicating that OpenNMT works quite unstably. Moreover, OpenNMT generates 53 extreme short summaries, which seriously reduces readability. Meanwhile, the copy ratio of actual summaries is 36%. Therefore, the copy mechanism is severely overweighted in OpenNMT. Our model is encouraged to generate according to human-written soft templates, which relatively diminishes copying from the source sentences. Look at the last row "NEW_NE". A number of new named entities appear in the soft templates, which makes them quite unfaithful to source sentences. By contrast, this index in Re^{3}Sum is close to the OpenNMT's. It highlights the rewriting ability of our seq2seq framework.

3.7 Effect of Templates

In this section, we investigate how soft templates affect our model. At the beginning, we feed different types of soft templates (refer to Table 4) into the Rewriting module of Re^{3}Sum. As illustrated in Table 6, the more high-quality templates are provided, the higher ROUGE scores are achieved. It is interesting to see that,while the ROUGE-2 score of Random templates is zero, our model can still generate acceptable summaries with Random templates. It seems that Re^{3}Sum can automatically judge whether the soft templates are trustworthy and ignore the seriously irrelevant ones. We believe that the joint learning with the Rerank model plays a vital role here.

Next, we manually inspect the summaries generated by different methods. We find the outputs of Re^{3}Sum are usually longer and more fluent than the outputs of OpenNMT. Some illustrative examples are shown in Table 7. In Example 1, there is no predicate in the source sentence. Since OpenNMT prefers selecting source words around the predicate to form the summary, it fails on this sentence. By contract, Re^{3}Sum rewrites the template and produces an informative summary. In Example 2, OpenNMT deems the starting part of the sentences are more important, while our model, guided by the template, focuses on the second part to generate the summary.

In the end, we test the ability of our model to generate diverse summaries. In practice, a system that can provide various candidate summaries is probably more welcome. Specifically, two candidate templates with large text dissimilarity are manually fed into the Rewriting module. The corresponding generated summaries are shown in Table 8. For the sake of comparison, we also present the 2-best results of OpenNMT with beam search. As can be seen, with different templates given, our model is likely to generate dissimilar summaries. In contrast, the 2-best results of OpenNMT is almost the same, and often a shorter summary is only a piece of the other one. To sum up, our model demonstrates promising prospect in generation diversity.

4 Related Work

Abstractive sentence summarization aims to produce a shorter version of a given sentence while preserving its meaning (Chopra et al., 2016). This task is similar to text simplification (Saggion, 2017) and facilitates headline design and refine. Early studies on sentence summariza-

Source	anny ainge said thursday he had two one-hour meetings with the new owners of the boston celtics but no deal has been completed for him to return to the franchise .
Target	ainge says no deal completed with celtics
Templates	major says no deal with spain on gibraltar
	roush racing completes deal with red sox owner
Re^{3}Sum	ainge **says no deal done** with **celtics**
	ainge **talks** with **new owners**
OpenNMT	ainge talks with **celtics** owners
	ainge talks with **new** owners
Source	european stock markets advanced strongly thursday on some bargain-hunting and gains by wall street and japanese shares ahead of an expected hike in us interest rates .
Target	european stocks bounce back UNK UNK with closing levels
Templates	european stocks bounce back strongly
	european shares sharply lower on us interest rate fears
Re^{3}Sum	european **stocks bounce back** strongly
	european **shares rise** strongly **on bargain-hunting**
OpenNMT	european stocks rise ahead of **expected** us rate hike **hike**
	european stocks rise ahead of us rate hike

Table 8: Examples of generation with diversity. We use Bold font to indicate the difference between two summaries

tion include template-based methods (Zhou and Hovy, 2004), syntactic tree pruning (Knight and Marcu, 2002; Clarke and Lapata, 2008) and statistical machine translation techniques (Banko et al., 2000). Recently, the application of the attentional seq2seq framework has attracted growing attention and achieved state-of-the-art performance on this task (Rush et al., 2015a; Chopra et al., 2016; Nallapati et al., 2016).

In addition to the direct application of the general seq2seq framework, researchers attempted to integrate various properties of summarization. For example, Nallapati et al. (2016) enriched the encoder with hand-crafted features such as named entities and POS tags. These features have played important roles in traditional feature based summarization systems. Gu et al. (2016) found that a large proportion of the words in the summary were copied from the source text. Therefore, they proposed CopyNet which considered the copying mechanism during generation. Recently, See et al. (2017) used the coverage mechanism to discourage repetition. Cao et al. (2017b) encoded facts extracted from the source sentence to enhance the summary faithfulness. There were also studies to modify the loss function to fit the evaluation metrics. For instance, Ayana et al. (2016) applied the Minimum Risk Training strategy to maximize the ROUGE scores of generated summaries. Paulus et al. (2017) used the reinforcement learning algorithm to optimize a mixed objective function of likelihood and ROUGE scores.

Guu et al. (2017) also proposed to encode human-written sentences to improvement the performance of neural text generation. However, they handled the task of Language Modeling and randomly picked an existing sentence in the training corpus. In comparison, we develop an IR system to find proper existing summaries as soft templates. Moreover, Guu et al. (2017) used a general seq2seq framework while we extend the seq2seq framework to conduct template reranking and template-aware summary generation simultaneously.

5 Conclusion and Future Work

This paper proposes to introduce soft templates as additional input to guide the seq2seq summarization. We use the popular IR platform Lucene to retrieve proper existing summaries as candidate soft templates. Then we extend the seq2seq framework to jointly conduct template reranking and template-aware summary generation. Experiments show that our model can generate informative, readable and stable summaries. In addition, our model demonstrates promising prospect in generation diversity.

We believe our work can be extended in various aspects. On the one hand, since the candidate templates are far inferior to the optimal ones, we intend to improve the Retrieve module, e.g., by indexing both the sentence and summary fields. On the other hand, we plan to test our system on the other tasks such as document-level summarization and short text conversation.

Acknowledgments

The work described in this paper was supported by Research Grants Council of Hong Kong (PolyU 152036/17E), National Natural Science Foundation of China (61672445 and 61572049) and The Hong Kong Polytechnic University (G-YBP6, 4-BCDV).

References

Shiqi Shen Ayana, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *arXiv preprint arXiv:1604.01904*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325. Association for Computational Linguistics.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017a. Joint copying and restricted generation for paraphrase. In *AAAI*, pages 3152–3158.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2017b. Faithful to the original: Fact aware neural abstractive summarization. *arXiv preprint arXiv:1711.04434*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16*, pages 93–98.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2017. Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878*.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop*, pages 74–81.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*, pages 55–60.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015a. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015b. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389.

Horacio Saggion. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Liang Zhou and Eduard Hovy. 2004. Template-filtered headline summarization. *Text Summarization Branches Out*.

Simple and Effective Text Simplification Using Semantic and Neural Methods

Elior Sulem, Omri Abend, Ari Rappoport
Department of Computer Science, The Hebrew University of Jerusalem
{eliors|oabend|arir}@cs.huji.ac.il

Abstract

Sentence splitting is a major simplification operator. Here we present a simple and efficient splitting algorithm based on an automatic semantic parser. After splitting, the text is amenable for further fine-tuned simplification operations. In particular, we show that neural Machine Translation can be effectively used in this situation. Previous application of Machine Translation for simplification suffers from a considerable disadvantage in that they are over-conservative, often failing to modify the source in any way. Splitting based on semantic parsing, as proposed here, alleviates this issue. Extensive automatic and human evaluation shows that the proposed method compares favorably to the state-of-the-art in combined lexical and structural simplification.

1 Introduction

Text Simplification (TS) is generally defined as the conversion of a sentence into one or more simpler sentences. It has been shown useful both as a preprocessing step for tasks such as Machine Translation (MT; Mishra et al., 2014; Štajner and Popović, 2016) and relation extraction (Niklaus et al., 2016), as well as for developing reading aids, e.g. for people with dyslexia (Rello et al., 2013) or non-native speakers (Siddharthan, 2002).

TS includes both structural and lexical operations. The main structural simplification operation is sentence splitting, namely rewriting a single sentence into multiple sentences while preserving its meaning. While recent improvement in TS has been achieved by the use of neural MT (NMT) approaches (Nisioi et al., 2017; Zhang et al., 2017; Zhang and Lapata, 2017), where TS is considered a case of monolingual translation, the sentence splitting operation has not been addressed by these systems, potentially due to the rareness of this operation in the training corpora (Narayan and Gardent, 2014; Xu et al., 2015).

We show that the explicit integration of sentence splitting in the simplification system could also reduce conservatism, which is a grave limitation of NMT-based TS systems (Alva-Manchego et al., 2017). Indeed, experimenting with a state-of-the-art neural system (Nisioi et al., 2017), we find that 66% of the input sentences remain unchanged, while none of the corresponding references is identical to the source. Human and automatic evaluation of the references (against other references), confirm that the references are indeed simpler than the source, indicating that the observed conservatism is excessive. Our methods for performing sentence splitting as pre-processing allows the TS system to perform other structural (e.g. deletions) and lexical (e.g. word substitutions) operations, thus increasing both structural and lexical simplicity.

For combining linguistically informed sentence splitting with data-driven TS, two main methods have been proposed. The first involves hand-crafted syntactic rules, whose compilation and validation are laborious (Shardlow, 2014). For example, Siddharthan and Angrosh (2014) used 111 rules for relative clauses, appositions, subordination and coordination. Moreover, syntactic splitting rules, which form a substantial part of the rules, are usually language specific, requiring the development of new rules when ported to other languages (Aluísio and Gasperin, 2010; Seretan, 2012; Hung et al., 2012; Barlacchi and Tonelli, 2013, for Portuguese, French, Vietnamese, and Italian respectively). The second method uses linguistic information for detecting potential splitting points, while splitting probabilities are learned using a parallel corpus. For example, in the system of Narayan and Gardent (2014) (henceforth, HYBRID), the state-of-the-art for joint structural and lexical TS, potential splitting points are determined by event boundaries.

In this work, which is the first to combine structural semantics and neural methods for TS, we propose an intermediate way for performing sentence splitting, presenting Direct Semantic Splitting (DSS), a simple and efficient algorithm based on a semantic parser which supports the direct decomposition of the sentence into its main semantic constituents. After splitting, NMT-based simplification is performed, using the NTS system. We show that the resulting system outperforms HYBRID in both automatic and human evaluation.

We use the UCCA scheme for semantic representation (Abend and Rappoport, 2013), where the semantic units are anchored in the text, which simplifies the splitting operation. We further leverage the explicit distinction in UCCA between types of Scenes (events), applying a specific rule for each of the cases. Nevertheless, the DSS approach can be adapted to other semantic schemes, like AMR (Banarescu et al., 2013).

We collect human judgments for multiple variants of our system, its sub-components, HYBRID and similar systems that use phrase-based MT. This results in a sizable human evaluation benchmark, which includes 28 systems, totaling at 1960 complex-simple sentence pairs, each annotated by three annotators using four criteria.[1] This benchmark will support the future analysis of TS systems, and evaluation practices.

Previous work is discussed in §2, the semantic and NMT components we use in §3 and §4 respectively. The experimental setup is detailed in §5. Our main results are presented in §6, while §7 presents a more detailed analysis of the system's sub-components and related settings.

2 Related Work

MT-based sentence simplification. Phrase-based Machine Translation (PBMT; Koehn et al., 2003) was first used for TS by Specia (2010), who showed good performance on lexical simplification and simple rewriting, but under-prediction of other operations. Štajner et al. (2015) took a similar approach, finding that it is beneficial to use training data where the source side is highly similar to the target. Other PBMT for TS systems include the work of Coster and Kauchak (2011b), which uses Moses (Koehn et al., 2007), the work of Coster and Kauchak (2011a), where the model is extended to include deletion, and PBMT-R (Wubben et al., 2012), where Levenshtein distance to the source is used for re-ranking to overcome conservatism.

The NTS NMT-based system (Nisioi et al., 2017) (henceforth, N17) reported superior performance over PBMT in terms of BLEU and human evaluation scores, and serves as a component in our system (see Section 4). Zhang et al. (2017) took a similar approach, adding lexical constraints to an NMT model. Zhang and Lapata (2017) combined NMT with reinforcement learning, using SARI (Xu et al., 2016), BLEU, and cosine similarity to the source as the reward. None of these models explicitly addresses sentence splitting.

Alva-Manchego et al. (2017) proposed to reduce conservatism, observed in PBMT and NMT systems, by first identifying simplification operations in a parallel corpus and then using sequence-labeling to perform the simplification. However, they did not address common structural operations, such as sentence splitting, and claimed that their method is not applicable to them.

Xu et al. (2016) used Syntax-based Machine Translation (SBMT) for sentence simplification, using a large scale paraphrase dataset (Ganitketitch et al., 2013) for training. While it does not target structural simplification, we include it in our evaluation for completeness.

Structural sentence simplification. Syntactic hand-crafted sentence splitting rules were proposed by Chandrasekar et al. (1996), Siddharthan (2002), Siddhathan (2011) in the context of rule-based TS. The rules separate relative clauses and coordinated clauses and un-embed appositives. In our method, the use of semantic distinctions instead of syntactic ones reduces the number of rules. For example, relative clauses and appositives can correspond to the same semantic category. In syntax-based splitting, a generation module is sometimes added after the split (Siddharthan, 2004), addressing issues such as reordering and determiner selection. In our model, no explicit regeneration is applied to the split sentences, which are fed directly to an NMT system.

Glavaš and Štajner (2013) used a rule-based system conditioned on event extraction and syntax

[1]The benchmark can be found in `https://github.com/eliorsulem/simplification-acl2018`.

for defining two simplification models. The event-wise simplification one, which separates events to separate output sentences, is similar to our semantic component. Differences are in that we use a single semantic representation for defining the rules (rather than a combination of semantic and syntactic criteria), and avoid the need for complex rules for retaining grammaticality by using a subsequent neural component.

Combined structural and lexical TS. Earlier TS models used syntactic information for splitting. Zhu et al. (2010) used syntactic information on the source side, based on the SBMT model of Yamada and Knight (2001). Syntactic structures were used on both sides in the model of Woodsend and Lapata (2011), based on a quasi-synchronous grammar (Smith and Eisner, 2006), which resulted in 438 learned splitting rules.

The model of Siddharthan and Angrosh (2014) is similar to ours in that it combines linguistic rules for structural simplification and statistical methods for lexical simplification. However, we use 2 semantic splitting rules instead of their 26 syntactic rules for relative clauses and appositions, and 85 syntactic rules for subordination and coordination.

Narayan and Gardent (2014) argued that syntactic structures do not always capture the semantic arguments of a frame, which may result in wrong splitting boundaries. Consequently, they proposed a supervised system (HYBRID) that uses semantic structures (Discourse Semantic Representations, (Kamp, 1981)) for sentence splitting and deletion. Splitting candidates are pairs of event variables associated with at least one core thematic role (e.g., agent or patient). Semantic annotation is used on the source side in both training and test. Lexical simplification is performed using the Moses system. HYBRID is the most similar system to ours architecturally, in that it uses a combination of a semantic structural component and an MT component. Narayan and Gardent (2016) proposed instead an unsupervised pipeline, where sentences are split based on a probabilistic model trained on the semantic structures of Simple Wikipedia as well as a language model trained on the same corpus. Lexical simplification is there performed using the unsupervised model of Biran et al. (2011). As their BLEU and adequacy scores are lower than HYBRID's, we use the latter for comparison.

Štajner and Glavaš (2017) combined rule-based simplification conditioned on event extraction, together with an unsupervised lexical simplifier. They tackle a different setting, and aim to simplify texts (rather than sentences), by allowing the deletion of entire input sentences.

Split and Rephrase. Narayan et al. (2017) recently proposed the Split and Rephrase task, focusing on sentence splitting. For this purpose they presented a specialized parallel corpus, derived from the WebNLG dataset (Gardent et al., 2017). The latter is obtained from the DBPedia knowledge base (Mendes et al., 2012) using content selection and crowdsourcing, and is annotated with semantic triplets of subject-relation-object, obtained semi-automatically. They experimented with five systems, including one similar to HYBRID, as well as sequence-to-sequence methods for generating sentences from the source text and its semantic forms.

The present paper tackles both structural and lexical simplification, and examines the effect of sentence splitting on the subsequent application of a neural system, in terms of its tendency to perform other simplification operations. For this purpose, we adopt a semantic corpus-independent approach for sentence splitting that can be easily integrated in any simplification system. Another difference is that the semantic forms in Split and Rephrase are derived semi-automatically (during corpus compilation), while we automatically extract the semantic form, using a UCCA parser.

3 Direct Semantic Splitting

3.1 Semantic Representation

UCCA (Universal Cognitive Conceptual Annotation; Abend and Rappoport, 2013) is a semantic annotation scheme rooted in typological and cognitive linguistic theory (Dixon, 2010b,a, 2012; Langacker, 2008). It aims to represent the main semantic phenomena in the text, abstracting away from syntactic forms. UCCA has been shown to be preserved remarkably well across translations (Sulem et al., 2015) and has also been successfully used for the evaluation of machine translation (Birch et al., 2016) and, recently, for the evaluation of TS (Sulem et al., 2018) and grammatical error correction (Choshen and Abend, 2018).

Formally, UCCA structures are directed acyclic graphs whose nodes (or *units*) correspond either to the leaves of the graph or to several elements viewed as a single entity according to some semantic or cognitive consideration.

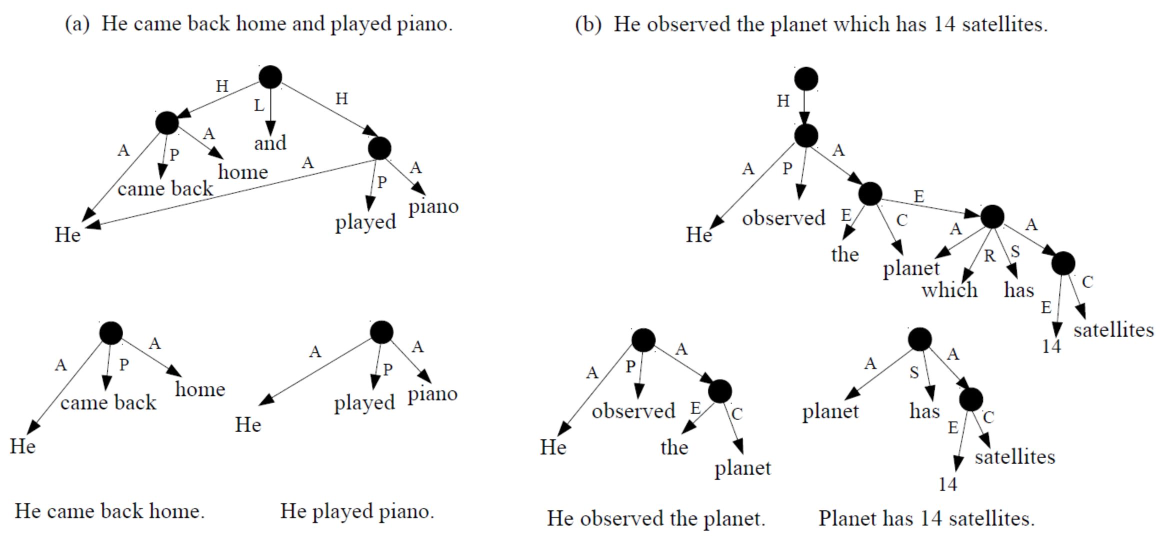


Figure 1: Example applications of rules 1 (Figure 1a) and 2 (Figure 1b). In both cases, the original sentence, the semantic parse, the extracted Scenes with the required modifications, and the output of the rules are presented top to bottom. The UCCA categories used are: Parallel Scene (H), Linker (L), Participant (A), Process/State (P/S), Center (C), Elaborator (E), Relator (R).

A *Scene* is UCCA's notion of an event or a frame, and is a unit that corresponds to a movement, an action or a state which persists in time. Every Scene contains one main relation, which can be either a Process or a State. Scenes contain one or more Participants, interpreted in a broad sense to include locations and destinations. For example, the sentence "He went to school" has a single Scene whose Process is "went". The two Participants are "He" and "to school".

Scenes can have several roles in the text. First, they can provide additional information about an established entity (Elaborator Scenes), commonly participles or relative clauses. For example, "(child) who went to school" is an Elaborator Scene in "The child who went to school is John" ("child" serves both as an argument in the Elaborator Scene and as the Center). A Scene may also be a Participant in another Scene. For example, "John went to school" in the sentence: "He said John went to school". In other cases, Scenes are annotated as Parallel Scenes (H), which are flat structures and may include a Linker (L), as in: "When$_L$ [he arrives]$_H$, [he will call them]$_H$".

With respect to units which are not Scenes, the category Center denotes the semantic head. For example, "dogs" is the Center of the expression "big brown dogs", and "box" is the center of "in the box". There could be more than one Center in a unit, for example in the case of coordination, where all conjuncts are Centers. We define the minimal center of a UCCA unit u to be the UCCA graph's leaf reached by starting from u and iteratively selecting the child tagged as Center.

For generating UCCA's structures we use TUPA, a transition-based parser (Hershcovich et al., 2017) (specifically, the TUPA$_{BiLSTM}$ model). TUPA uses an expressive set of transitions, able to support all structural properties required by the UCCA scheme. Its transition classifier is based on an MLP that receives a BiLSTM encoding of elements in the parser state (buffer, stack and intermediate graph), given word embeddings and other features.

3.2 The Semantic Rules

For performing DSS, we define two simple splitting rules, conditioned on UCCA's categories. We currently only consider Parallel Scenes and Elaborator Scenes, not separating Participant Scenes, in order to avoid splitting in cases of nominalizations or indirect speech. For example, the sentence "His arrival surprised everyone", which has, in addition to the Scene evoked by "surprised", a Participant Scene evoked by "arrival", is not split here.

Rule #1. Parallel Scenes of a given sentence are extracted, separated in different sentences, and concatenated according to the order of appearance. More formally, given a decomposition of a sentence S into parallel Scenes $Sc_1, Sc_2, \cdots Sc_n$ (indexed by the order of the first token), we obtain the following rule, where "|" is the sentence delimiter:

$$S \longrightarrow Sc_1|Sc_2|\cdots|Sc_n$$

As UCCA allows argument sharing between Scenes, the rule may duplicate the same sub-span of S across sentences. For example, the rule will convert "He came back home and played piano" into "He came back home"|"He played piano."

Rule #2. Given a sentence S, the second rule extracts Elaborator Scenes and corresponding minimal centers. Elaborator Scenes are then concatenated to the original sentence, where the Elaborator Scenes, except for the minimal center they elaborate, are removed. Pronouns such as "who", "which" and "that" are also removed.

Formally, if $\{(Sc_1, C_1)\cdots(Sc_n, C_n)\}$ are the Elaborator Scenes of S and their corresponding minimal centers, the rewrite is:

$$S \longrightarrow S - \bigcup_{i=1}^{n}(Sc_i - C_i)|Sc_1|\cdots|Sc_n$$

where $S - A$ is S without the unit A. For example, this rule converts the sentence "He observed the planet which has 14 known satellites" to "He observed the planet| Planet has 14 known satellites.". Article regeneration is not covered by the rule, as its output is directly fed into the NMT component.

After the extraction of Parallel Scenes and Elaborator Scenes, the resulting simplified Parallel Scenes are placed before the Elaborator Scenes. See Figure 1.

4 Neural Component

The split sentences are run through the NTS state-of-the-art neural TS system (Nisioi et al., 2017), built using the OpenNMT neural machine translation framework (Klein et al., 2017). The architecture includes two LSTM layers, with hidden states of 500 units in each, as well as global attention combined with input feeding (Luong et al., 2015). Training is done with a 0.3 dropout probability (Srivastava et al., 2014). This model uses alignment probabilities between the predictions and the original sentences, rather than character-based models, to retrieve the original words.

We here consider the w2v initialization for NTS (N17), where word2vec embeddings of size 300 are trained on Google News (Mikolov et al., 2013a) and local embeddings of size 200 are trained on the training simplification corpus (Řehůřek and Sojka, 2010; Mikolov et al., 2013b). Local embeddings for the encoder are trained on the source side of the training corpus, while those for the decoder are trained on the simplified side.

For sampling multiple outputs from the system, beam search is performed during decoding by generating the first 5 hypotheses at each step ordered by the log-likelihood of the target sentence given the input sentence. We here explore both the highest (h1) and fourth-ranked (h4) hypotheses, which we show to increase the SARI score and to be much less conservative.[2] We thus experiment with two variants of the neural component, denoted by NTS-h1 and NTS-h4. The pipeline application of the rules and the neural system results in two corresponding models: SENTS-h1 and SENTS-h4.

5 Experimental Setup

Corpus All systems are tested on the test corpus of Xu et al. (2016),[3] comprising 359 sentences from the PWKP corpus (Zhu et al., 2010) with 8 references collected by crowdsourcing for each of the sentences.

Semantic component. The TUPA parser[4] is trained on the UCCA-annotated *Wiki* corpus.[5]

Neural component. We use the NTS-w2v model[6] provided by N17, obtained by training on the corpus of Hwang et al. (2015) and tuning on the corpus of Xu et al. (2016). The training set is based on manual and automatic alignments between standard English Wikipedia and Simple English Wikipedia, including both good matches and partial matches whose similarity score is above the 0.45 scale threshold (Hwang et al., 2015). The total size of the training set is about 280K aligned sentences, of which 150K sentences are full matches and 130K are partial matches.[7]

Comparison systems. We compare our findings to HYBRID, which is the state of the art for joint structural and lexical simplification, imple-

[2]Similarly, N17 considered the first two hypotheses and showed that h2 has an higher SARI score and is less conservative than h1.

[3]https://github.com/cocoxu/simplification (This also includes SARI tools and the SBMT-SARI system.)

[4]https://github.com/danielhers/tupa

[5]http://www.cs.huji.ac.il/~oabend/ucca.html

[6]https://github.com/senisioi/NeuralTextSimplification

[7]We also considered the default initialization for the neural component, using the NTS model without word embeddings. Experimenting on the tuning set, the w2v approach got higher BLEU and SARI scores (for h1 and h4 respectively) than the default approach.

mented by Zhang and Lapata (2017).[8] We use the released output of HYBRID, trained on a corpus extracted from Wikipedia, which includes the aligned sentence pairs from Kauchak (2013), the aligned revision sentence pairs in Woodsend and Lapata (2011), and the PWKP corpus, totaling about 296K sentence pairs. The tuning set is the same as for the above systems.

In order to isolate the effect of NMT, we also implement SEMoses, where the neural-based component is replaced by the phrase-based MT system Moses,[9] which is also used in HYBRID. The training, tuning and test sets are the same as in the case of SENTS. MGIZA[10] is used for word alignment. The KenLM language model is trained using the target side of the training corpus.

Additional baselines. We report human and automatic evaluation scores for Identity (where the output is identical to the input), for Simple Wikipedia where the output is the corresponding aligned sentence in the PWKP corpus, and for the SBMT-SARI system, tuned against SARI (Xu et al., 2016), which maximized the SARI score on this test set in previous works (Nisioi et al., 2017; Zhang and Lapata, 2017).

Automatic evaluation. The automatic metrics used for the evaluation are: (1) BLEU (Papineni et al., 2002) (2) SARI (System output Against References and against the Input sentence; Xu et al., 2016), which compares the n-grams of the system output with those of the input and the human references, separately evaluating the quality of words that are added, deleted and kept by the systems. (3) F_{add}: the addition component of the SARI score (F-score); (4) F_{keep}: the keeping component of the SARI score (F-score); (5) P_{del}: the deletion component of the SARI score (precision).[11] Each metric is computed against the 8 available references. We also assess system conservatism, reporting the percentage of sentences copied from the input (%Same), the averaged Levenshtein distance from the source (LD_{SC}, which considers additions, deletions, and substitutions), and the number of source sentences that are split (#Split).[12]

Human evaluation. Human evaluation is carried out by 3 in-house native English annotators, who rated the different input-output pairs for the different systems according to 4 parameters: Grammaticality (G), Meaning preservation (M), Simplicity (S) and Structural Simplicity (StS). Each input-output pair is rated by all 3 annotators. Elicitation questions are given in Table 1.

As the selection process of the input-output pairs in the test corpus of Xu et al. (2016), as well as their crowdsourced references, are explicitly biased towards lexical simplification, the use of human evaluation permits us to evaluate the structural aspects of the system outputs, even where structural operations are not attested in the references. Indeed, we show that system outputs may receive considerably higher structural simplicity scores than the source, in spite of the sample selection bias.

Following previous work (e.g., Narayan and Gardent, 2014; Xu et al., 2016; Nisioi et al., 2017), Grammaticality (G) and Meaning preservation (M) are measured using a 1 to 5 scale. Note that in the first question, the input sentence is not taken into account. The grammaticality of the input is assessed by evaluating the Identity transformation (see Table 2), providing a baseline for the grammaticality scores of the other systems.

Following N17, a -2 to +2 scale is used for measuring simplicity, where a 0 score indicates that the input and the output are equally complex. This scale, compared to the standard 1 to 5 scale, permits a better differentiation between cases where simplicity is hurt (the output is more complex than the original) and between cases where the output is as simple as the original, for example in the case of the identity transformation. Structural simplicity is also evaluated with a -2 to +2 scale. The question for eliciting StS is accompanied with a negative example, showing a case of lexical simplification, where a complex word is replaced by a simple one (the other questions appear without examples). A positive example is not included so as not to bias the annotators by revealing the nature of the operations we focus on (splitting and deletion). We follow N17 in applying human evaluation on the first 70 sentences of the test corpus.[13]

The resulting corpus, totaling 1960 sentence pairs, each annotated by 3 annotators, also include

[8] https://github.com/XingxingZhang/dress

[9] http://www.statmt.org/moses/

[10] https://github.com/moses-smt/mgiza

[11] Uniform tokenization and truecasing styles for all systems are obtained using the Moses toolkit.

[12] We used the NLTK package (Loper and Bird, 2002) for these computations.

[13] We do not exclude system outputs identical to the source, as done by N17.

the additional experiments described in Section 7 as well as the outputs of the NTS and SENTS systems used with the default initialization.

The inter-annotator agreement, using Cohen's quadratic weighted κ (Cohen, 1968), is computed as the average agreement of the 3 annotator pairs. The obtained rates are 0.56, 0.75, 0.47 and 0.48 for G, M, S and StS respectively.

System scores are computed by averaging over the 3 annotators and the 70 sentences.

G	Is the output fluent and grammatical?
M	Does the output preserve the meaning of the input?
S	Is the output simpler than the input?
StS	Is the output simpler than the input, ignoring the complexity of the words?

Table 1: Questions for the human evaluation.

	G	M	S	StS
Identity	4.80	5.00	0.00	0.00
Simple Wikipedia	4.60	4.21	0.83	0.38
Only MT-Based Simplification				
SBMT-SARI	3.71	3.96	0.14	-0.15
NTS-h1	4.56	**4.48**	0.22	0.15
NTS-h4	4.29	3.90	0.31	0.19
Only Structural Simplification				
DSS	3.42	4.15	0.16	0.16
Structural+MT-based Simplification				
Hybrid	2.96	2.46	0.43	0.43
SEMoses	3.27	3.98	0.16	0.13
SENTS-h1	3.98	3.33	**0.68**	**0.63**
SENTS-h4	3.54	2.98	0.50	0.36

Table 2: Human evaluation of the different NMT-based systems. Grammaticality (G) and Meaning preservation (M) are measured using a 1 to 5 scale. A -2 to +2 scale is used for measuring simplicity (S) and structural simplicity (StS) of the output relative to the input sentence. The highest score in each column appears in bold. Structural simplification systems are those that explicitly model structural operations.

6 Results

Human evaluation. Results are presented in Table 2. First, we can see that the two SENTS systems outperform HYBRID in terms of G, M, and S. SENTS-h1 is the best scoring system, under all human measures.

In comparison to NTS, SENTS scores markedly higher on the simplicity judgments. Meaning preservation and grammaticality are lower for SENTS, which is likely due to the more conservative nature of NTS. Interestingly, the application of the splitting rules by themselves does not yield a considerably simpler sentence. This likely stems from the rules not necessarily yielding grammatical sentences (NTS often serves as a grammatical error corrector over it), and from the incorporation of deletions, which are also structural operations, and are performed by the neural system.

An example of high structural simplicity scores for SENTS resulting from deletions is presented in Table 5, together with the outputs of the other systems and the corresponding human evaluation scores. NTS here performs lexical simplification, replacing the word "incursions" by "raids" or "attacks"'. On the other hand, the high StS scores obtained by DSS and SEMoses are due to sentence splittings.

Automatic evaluation. Results are presented in Table 3. Identity obtains much higher BLEU scores than any other system, suggesting that BLEU may not be informative in this setting. SARI seems more informative, and assigns the lowest score to Identity and the second highest to the reference.

Both SENTS systems outperform HYBRID in terms of SARI and all its 3 sub-components. The h4 setting (hypothesis #4 in the beam) is generally best, both with and without the splitting rules.

Comparing SENTS to using NTS alone (without splitting), we see that SENTS obtains higher SARI scores when hypothesis #1 is used and that NTS obtains higher scores when hypothesis #4 is used. This may result from NTS being more conservative than SENTS (and HYBRID), which is rewarded by SARI (conservatism is indicated by the %Same column). Indeed for h1, %Same is reduced from around 66% for NTS, to around 7% for SENTS. Conservatism further decreases when h4 is used (for both NTS and SENTS). Examining SARI's components, we find that SENTS outperforms NTS on F_{add}, and is comparable (or even superior for $h1$ setting) to NTS on P_{del}. The superior SARI score of NTS over SENTS is thus entirely a result of a superior F_{keep}, which is easier for a conservative system to maximize.

Comparing HYBRID with SEMoses, both of which use Moses, we find that SEMoses obtains higher BLEU and SARI scores, as well as G and M human scores, and splits many more sentences. HYBRID scores higher on the human simplicity measures. We note, however, that applying NTS alone is inferior to HYBRID in terms of simplicity, and that both components are required to obtain high simplicity scores (with SENTS).

We also compare the sentence splitting component used in our systems (namely DSS) to that used in HYBRID, abstracting away from deletion-based and lexical simplification. We therefore apply DSS to the test set (554 sentences) of the

	BLEU	SARI	F_{add}	F_{keep}	P_{del}	% Same	$\mathrm{LD}_{\mathrm{SC}}$	#Split
Identity	94.93	25.44	0.00	76.31	0.00	100	0.00	0
Simple Wikipedia	69.58	39.50	8.46	61.71	48.32	0.00	33.34	0
Only MT-Based Simplification								
SBMT-SARI	74.44	**41.46**	**6.77**	69.92	**47.68**	4.18	23.31	0
NTS-h1	**88.67**	28.73	0.80	**70.95**	14.45	66.02	17.13	0
NTS-h4	79.88	36.55	2.59	65.93	41.13	2.79	24.18	1
Only Structural Simplification								
DSS	76.57	36.76	3.82	68.45	38.01	8.64	25.03	208
Structural+MT-Based Simplification								
HYBRID	52.82	27.40	2.41	43.09	36.69	1.39	61.53	3
SEMoses	74.45	36.68	3.77	67.66	38.62	7.52	27.44	208
SENTS-h1	58.94	30.27	3.01	51.52	36.28	6.69	59.18	0
SENTS-h4	57.71	31.90	3.95	51.86	39.90	0.28	54.47	17

Table 3: The left-hand side of the table presents BLEU and SARI scores for the combinations of NTS and DSS, as well as for the baselines. The highest score in each column appears in bold. The right hand side presents lexical and structural properties of the outputs. %Same: proportion of sentences copied from the input; $\mathrm{LD}_{\mathrm{SC}}$: Averaged Levenshtein distance from the source; #Split: number of split sentences. Structural simplification systems are those that explicitly model structural operations.

	BLEU	SARI	F_{add}	F_{keep}	P_{del}	% Same	$\mathrm{LD}_{\mathrm{SC}}$	#Split	G	M	S	StS
Moses	92.58	28.19	0.16	75.73	8.70	79.67	3.22	0	4.25	4.78	0	0.04
SEMoses	74.45	36.68	3.77	67.66	38.62	7.52	27.44	208	3.27	3.98	**0.16**	0.13
SETrain1-Moses	91.24	33.06	0.41	76.07	22.69	60.72	4.47	1	4.23	4.54	-0.12	-0.13
SETrain2-Moses	**94.31**	26.71	0.07	**76.20**	3.85	92.76	1.45	0	4.73	**4.99**	0.01	-0.005
Moses$_{\mathrm{LM}}$	92.66	28.19	0.18	75.68	8.71	79.39	3.43	0	4.55	4.82	-0.01	-0.04
SEMoses$_{\mathrm{LM}}$	74.49	**36.70**	**3.79**	67.67	**38.65**	7.52	27.45	208	3.32	4.08	0.15	**0.14**
SETrain1-Moses$_{\mathrm{LM}}$	85.68	36.52	2.34	72.85	34.37	27.30	6.71	33	4.03	4.63	-0.11	-0.12
SETrain2-Moses$_{\mathrm{LM}}$	94.22	26.66	0.10	76.19	3.69	92.20	1.43	0	**4.75**	**4.99**	0.01	-0.01

Table 4: Automatic and human evaluation for the different combinations of Moses and DSS. The automatic metrics as well as the lexical and structural properties reported (%Same: proportion of sentences copied from the input; $\mathrm{LD}_{\mathrm{SC}}$: Averaged Levenshtein distance from the source; #Split: number of split sentences) concern the 359 sentences of the test corpus. Human evaluation, with the G, M, S, and StS parameters, is applied to the first 70 sentences of the corpus. The highest score in each column appears in bold.

WEB-SPLIT corpus (Narayan et al., 2017) (See Section 2), which focuses on sentence splitting. We compare our results to those reported for a variant of HYBRID used without the deletion module, and trained on WEB-SPLIT (Narayan et al., 2017). DSS gets a higher BLEU score (46.45 vs. 39.97) and performs more splittings (number of output sentences per input sentence of 1.73 vs. 1.26).

7 Additional Experiments

Replacing the parser by manual annotation. In order to isolate the influence of the parser on the results, we implement a semi-automatic version of the semantic component, which uses manual UCCA annotation instead of the parser, focusing of the first 70 sentences of the test corpus. We employ a single expert UCCA annotator and use the UCCAApp annotation tool (Abend et al., 2017).

Results are presented in Table 6, for both SENTS and SEMoses. In the case of SEMoses, meaning preservation is improved when manual UCCA annotation is used. On the other hand, simplicity degrades, possibly due to the larger number of Scenes marked by the human annotator (TUPA tends to under-predict Scenes). This effect doesn't show with SENTS, where trends are similar to the automatic parses case, and high simplicity scores are obtained. This demonstrates that UCCA parsing technology is sufficiently mature to be used to carry out structural simplification.

We also directly evaluate the performance of the parser by computing F1, Recall and Precision DAG scores (Hershcovich et al., 2017), against the manual UCCA annotation.[14] We obtain for primary edges (i.e. edges that form a tree structure) scores of 68.9 %, 70.5%, and 67.4% for F1, Recall and Precision respectively. For remotes edges (i.e. additional edges, forming a DAG), the scores are 45.3%, 40.5%, and 51.5%. These results are comparable with the out-of-domain results reported by Hershcovich et al. (2017).

Experiments on Moses. We test other variants of SEMoses, where phrase-based MT is used instead of NMT. Specifically, we incorporate semantic information in a different manner by implementing two additional models: (1) SETrain1-Moses, where a new training corpus is obtained by applying the splitting rules to the target side of the

[14]We use the evaluation tools provided in `https://github.com/danielhers/ucca`, ignoring 9 sentences for which different tokenizations of proper nouns are used in the automatic and manual parsing.

		G	M	S	StS
Identity	In return, Rollo swore fealty to Charles, converted to Christianity, and undertook to defend the northern region of France against the incursions of other Viking groups.	5.00	5.00	0.00	0.00
Simple Wikipedia	In return, Rollo swore fealty to Charles, converted to Christianity, and swore to defend the northern region of France against raids by other Viking groups.	4.67	5.00	1.00	0.00
SBMT-SARI	In return, Rollo swore fealty to Charles, converted to Christianity, and set out to defend the north of France from the raids of other viking groups.	4.67	4.67	0.67	0.00
NTS-h1	In return, Rollo swore fealty to Charles, converted to Christianity, and undertook to defend the northern region of France against the raids of other Viking groups.	5.00	5.00	1.00	0.00
NTS-h4	In return, Rollo swore fealty to Charles, converted to Christianity, and undertook to defend the northern region of France against the attacks of other Viking groups.	4.67	5.00	1.00	0.00
DSS	Rollo swore fealty to Charles. Rollo converted to Christianity. Rollo undertook to defend the northern region of France against the incursions of other viking groups.	4.00	4.33	1.33	1.33
HYBRID	In return Rollo swore, and undertook to defend the region of France., Charles, converted	2.33	2.00	0.33	0.33
SEMoses	Rollo swore put his seal to Charles. Rollo converted to Christianity. Rollo undertook to defend the northern region of France against the incursions of other viking groups.	3.33	4.00	1.33	1.33
SENTS-h1	Rollo swore fealty to Charles.	5.00	2.00	2.00	2.00
SENTS-h4	Rollo swore fealty to Charles and converted to Christianity.	5.00	2.67	1.33	1.33

Table 5: System outputs for one of the test sentences with the corresponding human evaluation scores (averaged over the 3 annotators). Grammaticality (G) and Meaning preservation (M) are measured using a 1 to 5 scale. A -2 to +2 scale is used for measuring simplicity (S) and structural simplicity (StS) of the output relative to the input sentence.

	G	M	S	StS
DSS^m	3.38	3.91	-0.16	-0.16
SENTS^m-h1	4.12	3.34	0.61	0.58
SENTS^m-h4	3.60	3.24	0.26	0.12
SEMoses^m	3.32	4.27	-0.25	-0.25
SEMoses$^m_{LM}$	3.43	4.28	-0.18	-0.19

Table 6: Human evaluation using manual UCCA annotation. Grammaticality (G) and Meaning preservation (M) are measured using a 1 to 5 scale. A -2 to +2 scale is used for measuring simplicity (S) and structural simplicity (StS) of the output relative to the input sentence. X^m refers to the semi-automatic version of the system X.

training corpus; (2) SETrain2-Moses, where the rules are applied to the source side. The resulting parallel corpus is concatenated to the original training corpus. We also examine whether training a language model (LM) on split sentences has a positive effect, and train the LM on the split target side. For each system X, the version with the LM trained on split sentences is denoted by X_{LM}.

We repeat the same human and automatic evaluation protocol as in §6, presenting results in Table 4. Simplicity scores are much higher in the case of SENTS (that uses NMT), than with Moses. The two best systems according to SARI are SEMoses and SEMoses$_{LM}$ which use DSS. In fact, they resemble the performance of DSS applied alone (Tables 2 and 3), which confirms the high degree of conservatism observed by Moses in simplification (Alva-Manchego et al., 2017). Indeed, all Moses-based systems that don't apply DSS as preprocessing are conservative, obtaining high scores for BLEU, grammaticality and meaning preservation, but low scores for simplicity. Training the LM on split sentences shows little improvement.

8 Conclusion

We presented the first simplification system combining semantic structures and neural machine translation, showing that it outperforms existing lexical and structural systems. The proposed approach addresses the over-conservatism of MT-based systems for TS, which often fail to modify the source in any way. The semantic component performs sentence splitting without relying on a specialized corpus, but only an off-the-shelf semantic parser. The consideration of sentence splitting as a decomposition of a sentence into its Scenes is further supported by recent work on structural TS evaluation (Sulem et al., 2018), which proposes the SAMSA metric. The two works, which apply this assumption to different ends (TS system construction, and TS evaluation), confirm its validity. Future work will leverage UCCA's cross-linguistic applicability to support multi-lingual TS and TS pre-processing for MT.

Acknowledgments

We would like to thank Shashi Narayan for sharing his data and the annotators for participating in our evaluation and UCCA annotation experiments. We also thank Daniel Hershcovich and the anonymous reviewers for their helpful advices. This work was partially supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI) and by the Israel Science Foundation (grant No. 929/17), as well as by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL-13*, pages 228–238.

Omri Abend, Shai Yerushalmi, and Ari Rappoport. 2017. UCCAApp: Web-application for syntactic and semantic phrase-based annotation. In *Proc. of ACL'17, System Demonstrations*, pages 109–114.

Sandra Maria Aluísio and Caroline Gasperin. 2010. Foestering disgital inclusion and accessibility: The PorSimples project for simplification of Portuguese texts. In *Proc. of NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53.

Fernando Alva-Manchego, Joachim Bingel, Gustavo H. Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proc. of IJCNLP'17*, pages 295–305.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Rrepresentation for sembanking. *Proc. of Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Gianni Barlacchi and Sara Tonelli. 2013. ERNESTA: A sentence simplification tool for children's stories in Italian. In *Proc. of CICLing'13*, pages 476–487.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proc. of ACL'11*, pages 465–501.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. In *Proc. of EMNLP'16*, pages 1264–1274.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for sentence simplification. In *Proc. of COLING'96*, pages 1041–1044.

Leshem Choshen and Omri Abend. 2018. Reference-less measure of faithfulness for grammatical error correction. In *Proc. of NAACL'18 (Short papers)*. To appear.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

William Coster and David Kauchak. 2011a. Learning to simplify sentences using Wikipedia. In *Proc. of ACL, Short Papers*, pages 1–9.

William Coster and David Kauchak. 2011b. Simple English Wikipedia: A new text simplification task. In *Proc. of ACL'11*, pages 665–669.

Robert M.W. Dixon. 2010a. *Basic Linguistic Theory: Grammatical Topics*, volume 2. Oxford University Press.

Robert M.W. Dixon. 2010b. *Basic Linguistic Theory: Methodology*, volume 1. Oxford University Press.

Robert M.W. Dixon. 2012. *Basic Linguistic Theory: Further Grammatical Topics*, volume 3. Oxford University Press.

Jury Ganitketitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proc. of NAACL-HLT'13*, pages 758–764.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planning. In *Proc. of ACL'17*, pages 179–188.

Goran Glavaš and Sanja Štajner. 2013. Event-centered simplification of news stories. In *Proc. of the Student Research Workshop associated with RANLP 2013*, pages 71–78.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL'17*, pages 1127–1138.

Bui Thanh Hung, Nguyen Le Minh, and Akira Shimazu. 2012. Sentence splitting for Vietnamese-English machine translation. In *Knowledge and Systems Engineering, 2012 Fourth International Conference*, pages 156–160.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from Standard Wikipedia to Simple Wikipedia. In *Proc. of NAACL'15*, pages 211–217.

Hans Kamp. 1981. A theory of truth and semantic representation. In *Formal methods in the study of language*. Mathematisch Centrum. Number pt.1 in Mathematical Centre tracts.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proc. of ACL'13*, pages 1537–1546.

Guillam Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Open NMT: Open-source toolkit for neural machine translation. ArXiv:1701.02810 [cs:CL].

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Buch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proc. of ACL'07 on interactive poster and demonstration sessions*, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL'03*, pages 48–54.

Ronald W. Langacker. 2008. *Cognitive Grammar: A Basic Introduction*. Oxford University Press, USA.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *Proc. of EMNLP'02*, pages 63–70.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP'15*, pages 1412–1421.

Pablo N Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A multilingual cross-domain knowledge base. In *Proc. of LREC'12*, pages 1813–1817.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. of Workshop at International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kshitij Mishra, Ankush Soni, Rahul Sharma, and Dipti Misra Sharma. 2014. Exploring the effects of sentence simplification on Hindi to English Machine Translation systems. In *Proc. of the Workshop on Automatic Text Simplification: Methods and Applications in the Multilingual Society*, pages 21–29.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proc. of ACL'14*, pages 435–445.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proc. of INLG'16*, pages 111–120.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proc. of EMNLP'17*, pages 617–627.

Christina Niklaus, Bernahard Bermeitinger, Siegfried Handschuh, and André Freitas. 2016. A sentence simplification system for improving relation extraction. In *Proc. of COLING'16*.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proc. of ACL'17 (Short paper)*, pages 85–91.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL'02*, pages 311–318.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proc. of LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. Simplify or help?: text simplification strategies for people with dyslexia. In *Proc. of the 10th International Cross-Disciplinary Conference on Web Accesibility*, pages 15:1 – 15:10.

Violeta Seretan. 2012. Acquisition of syntactic simplification rules for French. In *Proc. of LREC'12*, pages 4019–4026.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Proc. of LEC*, pages 64–71.

Advaith Siddharthan. 2004. Syntactic simplification and text cohesion. Technical Report 597, University of Cambridge.

Advaith Siddharthan and M. A. Angrosh. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Proc. of EACL'14*, pages 722–731.

Advaith Siddhathan. 2011. Text simplification using typed dependencies: A comparison of the robustness of different generation strategies. In *Proc. of the 13th European Workshop on Natural Language Generation*, pages 2–11. Association of Computational Linguistics.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of the 1st Workshop in Statistical Machine Translation*, pages 23–30.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *Proc. of the 9th International Conference on Computational Processing of the Portuguese Language*, pages 30–39.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sanja Štajner, Hannah Bechara, and Horacio Saggion. 2015. A deeper exploration of the standard PB-SMT approach to text simplification and its evaluation. In *Proc. of ACL'15 (Short papers)*, pages 823–828.

Sanja Štajner and Goran Glavaš. 2017. Leveraging event-based semantics for automated text simplification. *Expert systems with applications*, 82:383–395.

Sanja Štajner and Maja Popović. 2016. Can text simplification help machine translation. *Baltic J. Modern Computing*, 4:230–242.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations. In *Proc. of 1st Workshop on Semantics-Driven Statistical Machine Translation (S2Mt 2015)*, pages 11–22.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Semantic structural evaluation for text simplification. In *Proc. of NAACL'18*. To appear.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proc. of EMNLP'11*, pages 409–420.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proc. of ACL'12*, pages 1015–1024.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: new data can help. *TACL*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL*, 4:401–415.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc.of ACL'01*, pages 523–530.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proc. of EMNLP'17*, pages 595–605.

Yaoyuan Zhang, Zhenxu Ye, Dongyan Zhao, and Rui Yan. 2017. A constrained sequence-to-sequence neural model for sentence simplification. ArXiv:1704.02312 [cs.CL].

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proc. of COLING'10*, pages 1353–1361.

Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20,000 English Words

Saif M. Mohammad
National Research Council Canada
saif.mohammad@nrc-cnrc.gc.ca

Abstract

Words play a central role in language and thought. Factor analysis studies have shown that the primary dimensions of meaning are valence, arousal, and dominance (VAD). We present the NRC VAD Lexicon, which has human ratings of valence, arousal, and dominance for more than 20,000 English words. We use Best–Worst Scaling to obtain fine-grained scores and address issues of annotation consistency that plague traditional rating scale methods of annotation. We show that the ratings obtained are vastly more reliable than those in existing lexicons. We also show that there exist statistically significant differences in the shared understanding of valence, arousal, and dominance across demographic variables such as age, gender, and personality.

1 Introduction

Words are the smallest meaningful utterances in language. They play a central role in our understanding and descriptions of the world around us. Some believe that the structure of a language even affects how we think (principle of linguistic relativity aka the SapirWhorf hypothesis). Several influential factor analysis studies have shown that the three most important, largely independent, dimensions of word meaning are valence (positiveness–negativeness/pleasure–displeasure), arousal (active–passive), and dominance (dominant–submissive) (Osgood et al., 1957; Russell, 1980, 2003).[1] Thus, when comparing the meanings of two words, we can compare their degrees of valence, arousal, or dominance. For example, the word *banquet* indicates more positiveness than the word *funeral*; *nervous* indicates more arousal than *lazy*; and *fight* indicates more dominance than *delicate*.

Access to these degrees of valence, arousal, and dominance of words is beneficial for a number of applications, including those in natural language processing (e.g., automatic sentiment and emotion analysis of text), in cognitive science (e.g., for understanding how humans represent and use language), in psychology (e.g., for understanding how people view the world around them), in social sciences (e.g., for understanding relationships between people), and even in evolutionary linguistics (e.g., for understanding how language and behaviour inter-relate to give us an advantage).

Existing VAD lexicons (Bradley and Lang, 1999; Warriner et al., 2013) were created using rating scales and thus suffer from limitations associated with the method (Presser and Schuman, 1996; Baumgartner and Steenkamp, 2001). These include: inconsistencies in annotations by different annotators, inconsistencies in annotations by the same annotator, scale region bias (annotators often have a bias towards a portion of the scale), and problems associated with a fixed granularity.

In this paper, we describe how we obtained human ratings of valence, arousal, and dominance for more than 20,000 commonly used English words by crowdsourcing. Notably, we use a comparative annotation technique called *Best-Worst Scaling (BWS)* that addresses the limitations of traditional rating scales (Louviere, 1991; Cohen, 2003; Louviere et al., 2015). The scores are fine-grained real-valued numbers in the interval from 0 (lowest V, A, or D) to 1 (highest V, A, or D). We will refer to this new lexicon as the *NRC Valence, Arousal, and Dominance (VAD) Lexicon.*[2]

[1] We will refer to the three dimensions individually as *V, A,* and *D*, and together as *VAD*.

[2] NRC refers to National Research Council Canada.

Correlations (r) between repeated annotations, through metrics such as *split-half reliability (SHR)*, are a common way to evaluate the reliabilities of ordinal and rank annotations. We show that our annotations have SHR scores of $r = 0.95$ for valence, $r = 0.90$ for arousal, and $r = 0.91$ for dominance. These scores are well above the SHR scores obtained by Warriner et al. (2013), and indicate high reliability.

Respondents who provided valence, arousal, and dominance annotations, were given the option of additionally filling out a brief demographic questionnaire to provide details of their age, gender, and personality traits. This demographic information along with the VAD annotations allows us to determine whether attributes such as age, gender, and personality impact our understanding of the valence, arousal, and dominance of words. We show that even though overall the annotations are consistent (as seen from the high SHR scores), people aged over 35 are significantly more consistent in their annotations than people aged 35 or less. We show for the first time that men have a significantly higher shared understanding of dominance and valence of words, whereas women have a higher shared understanding of the degree of arousal of words. We find that some personality traits significantly impact a person's annotations of one or more of valence, arousal, and dominance. We hope that these and other findings described in the paper foster further research into how we use language, how we represent concepts in our minds, and how certain aspects of the world are more important to certain demographic groups leading to higher degrees of shared representations of those concepts within those groups.

All of the annotation tasks described in this paper were approved by our institution's review board, which examined the methods to ensure that they were ethical. Special attention was paid to obtaining informed consent and protecting participant anonymity. The NRC VAD Lexicon is made freely available for research and non-commercial use through our project webpage.[3]

2 Related Work

Primary Dimensions of Meaning: Osgood et al. (1957) asked human participants to rate words along dimensions of opposites such as *heavy–light, good–bad, strong–weak,* etc. Factor analysis of these judgments revealed that the three most prominent dimensions of meaning are evaluation (*good–bad*), potency (*strong–weak*), and activity (*active–passive*). Russell (1980, 2003) showed through similar analyses of emotion words that the three primary independent dimensions of emotions are valence or pleasure (positiveness–negativeness/pleasure–displeasure), arousal (active–passive), and dominance (dominant–submissive). He argues that individual emotions such as joy, anger, and fear are points in a three-dimensional space of valence, arousal, and dominance. It is worth noting that even though the names given by Osgood et al. (1957) and Russell (1980) are different, they describe similar dimensions (Bakker et al., 2014).

Existing Affect Lexicons: Bradley and Lang (1999) asked annotators to rate valence, arousal, and dominance—for more than 1,000 words—on a 9-point rating scale. The ratings from multiple annotators were averaged to obtain a score between 1 (lowest V, A, or D) to 9 (highest V, A, or D). Their lexicon, called the *Affective Norms of English Words (ANEW)*, has since been widely used across many different fields of study. More than a decade later, Warriner et al. (2013) created a similar lexicon for more than 13,000 words, using a similar annotation method. There exist a small number of VAD lexicons in non-English languages as well, such as the ones created by Moors et al. (2013) for Dutch, by Võ et al. (2009) for German, and by Redondo et al. (2007) for Spanish. The NRC VAD lexicon is the largest manually created VAD lexicon (in any language), and the only one that was created via comparative annotations (instead of rating scales).

Best-Worst Scaling: Best-Worst Scaling (BWS) was developed by (Louviere, 1991), building on work in the 1960's in mathematical psychology and psychophysics. Annotators are given n items (an n-tuple, where $n > 1$ and commonly $n = 4$).[4] They are asked which item is the *best* (highest in terms of the property of interest) and which is the *worst* (least in terms of the property of interest). When working on 4-tuples, best–worst annotations are particularly efficient because each best and worst annotation will reveal the order of five of the six item pairs (e.g., for a 4-tuple with items

[3]http://saifmohammad.com/WebPages/nrc-vad.html

[4]At its limit, when $n = 2$, BWS becomes a *paired comparison* (Thurstone, 1927; David, 1963), but then a much larger set of tuples need to be annotated (closer to N^2).

A, B, C, and D, if A is the best, and D is the worst, then A > B, A > C, A > D, B > D, and C > D). Real-valued scores of association between the items and the property of interest can be determined using simple arithmetic on the number of times an item was chosen best and number of times it was chosen worst (as described in Section 3) (Orme, 2009; Flynn and Marley, 2014).

It has been empirically shown that three annotations each for $2N$ 4-tuples is sufficient for obtaining reliable scores (where N is the number of items) (Louviere, 1991; Kiritchenko and Mohammad, 2016). Kiritchenko and Mohammad (2017) showed through empirical experiments that BWS produces more reliable and more discriminating scores than those obtained using rating scales. (See Kiritchenko and Mohammad (2016, 2017) for further details on BWS.)

Within the NLP community, BWS has been used for creating datasets for relational similarity (Jurgens et al., 2012), word-sense disambiguation (Jurgens, 2013), word–sentiment intensity (Kiritchenko and Mohammad, 2016), word–emotion intensity (Mohammad, 2018), and tweet–emotion intensity (Mohammad and Bravo-Marquez, 2017; Mohammad et al., 2018; Mohammad and Kiritchenko, 2018).

Automatically Creating Affect Lexicons: There is growing work on automatically determining word–sentiment and word–emotion associations (Yang et al., 2007; Mohammad and Kiritchenko, 2015; Yu et al., 2015; Staiano and Guerini, 2014). The VAD Lexicon can be used to evaluate how accurately the automatic methods capture valence, arousal, and dominance.

3 Obtaining Human Ratings of Valence, Arousal, and Dominance

We now describe how we selected the terms to be annotated and how we crowdsourced the annotation of the terms using best–worst scaling.

3.1 Term Selection

We chose to annotate commonly used English terms. We especially wanted to include terms that denotate or connotate emotions. We also include terms common in tweets.[5] Specifically, we include terms from the following sources:

- All terms in the NRC Emotion Lexicon (Mohammad and Turney, 2013). It has about 14,000 words with labels indicating whether they are associated with any of the eight basic emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust (Plutchik, 1980).
- All 4,206 terms in the positive and negative lists of the General Inquirer (Stone et al., 1966).
- All 1,061 terms listed in ANEW (Bradley and Lang, 1999).
- All 13,915 terms listed in the Warriner et al. (2013) lexicon.
- 520 words from the Roget's Thesaurus categories corresponding to the eight basic Plutchik emotions.[6]
- About 1000 high-frequency content terms, including emoticons, from the Hashtag Emotion Corpus (HEC) (Mohammad, 2012).[7]

The union of the above sets resulted in 20,007 terms that were then annotated for valence, arousal, and dominance.

3.2 Annotating VAD via Best–Worst Scaling

We describe below how we annotated words for valence. The same approach is followed for arousal and dominance. The annotators were presented with four words at a time (4-tuples) and asked to select the word with the highest valence and the word with the lowest valence. The questionnaire uses a set of paradigm words that signify the two ends of the valence dimension. The paradigm words were taken from past literature on VAD (Bradley and Lang, 1999; Osgood et al., 1957; Russell, 1980). The questions used for valence are shown below.

Q1. Which of the four words below is associated with the MOST happiness / pleasure / positiveness / satisfaction / contentedness / hopefulness OR LEAST unhappiness / annoyance / negativeness / dissatisfaction / melancholy / despair? (Four words listed as options.)

Q2. Which of the four words below is associated with the LEAST happiness / pleasure / positiveness / satisfaction / contentedness / hopefulness OR MOST unhappiness / annoyance / negativeness / dissatisfaction / melancholy / despair? (Four words listed as options.)

[5]Tweets include non-standard language such as emoticons, emojis, creatively spelled words (*happee*), hashtags (*#takingastand, #lonely*) and conjoined words (*loveumom*).

[6]http://www.gutenberg.org/ebooks/10681

[7]All tweets in the HEC include at least one of the eight basic emotion words as a hashtag word (*#anger, #sadness*, etc.).

Dataset	#words	Location of Annotators	Annotation Item	#Items	#Annotators	MAI	#Q/Item	#Best–Worst Annotations
valence	20,007	worldwide	4-tuple of words	40,014	1,020	6	2	243,295
arousal	20,007	worldwide	4-tuple of words	40,014	1,081	6	2	258,620
dominance	20,007	worldwide	4-tuple of words	40,014	965	6	2	276,170
Total								**778,085**

Table 1: A summary of the annotations for valence, arousal, and dominance. MAI = minimum number of annotations per item. Q = questions. A total of 778,085 pairs of best–worst responses were obtained.

Questions for arousal and dominance are similar.[8]

Detailed directions and example questions (with suitable responses) were provided in advance. $2 \times N$ distinct 4-tuples were randomly generated in such a manner that each word is seen in eight different 4-tuples and no two 4-tuples have more than two items in common (where N is the number of words to be annotated).[9]

Crowdsourcing: We setup three separate crowdsourcing tasks corresponding to valence, arousal, and dominance. The 4-tuples of words were uploaded for annotation on the crowdsourcing platform, *CrowdFlower*.[10] We obtained annotations from native speakers of English residing around the world. Annotators were free to provide responses to as many 4-tuples as they wished. The annotation tasks were approved by our institution's review board.

About 2% of the data was annotated beforehand by the authors. These questions are referred to as gold questions. CrowdFlower interspersed the gold questions with the other questions. If a crowd worker answered a gold question incorrectly, then they were immediately notified, the annotation was discarded, and an additional annotation was requested from a different annotator. If an annotator's accuracy on the gold questions fell below 80%, then they were refused further annotation, and all of their annotations were discarded. This served as a mechanism to avoid malicious and random annotations. The gold questions also served as examples to guide the annotators.

[8]The two ends of the arousal dimension were described with the words: arousal, activeness, stimulation, frenzy, jitteriness, alertness AND unarousal, passiveness, relaxation, calmness, sluggishness, dullness, sleepiness. The two ends of the dominance dimension were described with the words: dominant, in control of the situation, powerful, influential, important, autonomous AND submissive, controlled by outside factors, weak, influenced, cared-for, guided.

[9]We used the script provided by Kiritchenko and Mohammad (2016) to generate the 4-tuples from the list of terms: http://saifmohammad.com/WebPages/BestWorst.html

[10]CrowdFlower later changed its name to Figure Eight: https://www.figure-eight.com

Dimension	Word	Score↑	Word	Score↓
valence	*love*	1.000	*toxic*	0.008
	happy	1.000	*nightmare*	0.005
	happily	1.000	*shit*	0.000
arousal	*abduction*	0.990	*mellow*	0.069
	exorcism	0.980	*siesta*	0.046
	homicide	0.973	*napping*	0.046
dominance	*powerful*	0.991	*empty*	0.081
	leadership	0.983	*frail*	0.069
	success	0.981	*weak*	0.045

Table 2: The terms with the highest (↑) and lowest (↓) valence (V), arousal (A), and dominance (D) scores in the VAD Lexicon.

In the task settings for CrowdFlower, we specified that we needed annotations from six people for each word.[11] However, because of the way the gold questions work in CrowdFlower, they were annotated by more than six people. Both the minimum and the median number of annotations per item was six. See Table 1 for summary statistics on the annotations.[12]

Annotation Aggregation: The final VAD scores were calculated from the BWS responses using a simple counting procedure (Orme, 2009; Flynn and Marley, 2014): For each item, the score is the proportion of times the item was chosen as the best (highest V/A/D) minus the proportion of times the item was chosen as the worst (lowest V/A/D). The scores were linearly transformed to the interval: 0 (lowest V/A/D) to 1 (the highest V/A/D). We refer to the list of words along with their scores for valence, arousal, and dominance as the *NRC Valence, Arousal, and Dominance Lexicon*, or the *NRC VAD Lexicon* for short. Table 2 shows entries from the lexicon with the highest and lowest scores for V, A, and D.

[11]Note that since each word occurs in eight different 4-tuples, it is involved in $8 \times 6 = 48$ best–worst judgments.

[12]In a post-annotation survey, the respondents gave the task high scores for clarity of instruction (an average of 4.5 out of 5) and overall satisfaction (an average of 4.3 out of 5).

Attribute	Value	%	Value	%
Gender	f	37	m	63
Age	≤35	70	>35	30
Personality	Ag	69	Di	31
	Co	52	Ea	48
	Ex	52	In	48
	Ne	40	Se	60
	Op	50	Cl	50

Table 3: Summary of the demographic information provided by the annotators.

4 Demographic Survey

Respondents who annotated our VAD questionnaires were given a special code through which they could then optionally respond to a separate CrowdFlower survey asking for their demographic information: age, gender, country they live in, and personality traits. For the latter, we asked how they viewed themselves across the big five (Barrick and Mount, 1991) personality traits:

- Agreeableness (Ag) – Disagreeableness (Di): friendly and compassionate or careful in whom to trust, argumentative
- Conscientiousness (Co) – Easygoing (Ea): efficient and organized (prefer planned and self-disciplined behaviour) or easy-going and carefree (prefer flexibility and spontaneity)
- Extrovert (Ex) – Introvert (In): outgoing, energetic, seek the company of others or solitary, reserved, meeting many people causes anxiety
- Neurotic (Ne) – Secure (Se): sensitive and nervous (often feel anger, anxiety, depression, and vulnerability) or secure and confident (rarely feel anger, anxiety, depression, and vulnerability)
- Open to experiences (Op) – Closed to experiences (Cl): inventive and curious (seek out new experiences) or consistent and cautious (anxious about new experiences)

The questionnaire described the two sides of the dimension using only the texts after the colons above.[13] The questionnaire did not ask for identifying information such as name or date of birth.

In total, 991 people (55% of the VAD annotators) chose to provide their demographic information. Table 3 shows the details.

	V	A	D
Ours–Warriner	0.814	0.615	0.326

Table 4: Pearson correlations between our V, A, and D scores and the Warriner scores.

Lexicon	V–A	A–D	V–D
Ours	-0.268	0.302	0.488
Ours (Warriner subset)	-0.287	0.322	0.463
Warriner	-0.185	-0.180	0.717

Table 5: Pearson correlations between various pair-wise combinations of V, A, and D.

5 Examining of the NRC VAD Lexicon

5.1 A Comparsion of the NRC VAD Lexicon and the Warriner et al. Lexicon Scores

We calculated the Pearson correlations r between the NRC VAD Lexicon scores and the Warriner et al. Lexicon scores. Table 4 shows the results. (These numbers were calculated for the 13,915 common terms across the two lexicons.) Observe that the especially low correlations for dominance and arousal indicate that our lexicon has substantially different scores and rankings of terms by these dimensions. Even for valence, a correlation of 0.81 indicates a marked amount of differences in scores.

5.2 Independence of Dimensions

Russell (1980) found through his factor analysis work that valence, arousal, and dominance are nearly independent dimensions. However, Warriner et al. (2013) report that their scores for valence and dominance have substantial correlation ($r = 0.717$). Given that the split-half reliability score for their dominance annotations is only 0.77, the high V–D correlations raises the suspicion whether annotators sufficiently understood the difference between dominance and valence. Table 5 shows the correlations between various pair-wise combinations of valence, arousal, and dominance for both our lexicon and the Warriner lexicon. Observe that unlike the Warriner annotations where V and D are highly correlated, our annotations show that V and D are only slightly correlated. The correlations for V–A and A–D are low in both our and Warriner annotations, albeit slightly higher in magnitude in our annotations.

[13]How people view themselves may be different from what they truly are. The conclusions in this paper apply to groups that view themselves to be a certain personality type.

Annotations	#Terms	#Annotations	V	A	D
a. Ours (on all terms)	20,007	6 per tuple	0.950	0.899	0.902
b. Ours (on only those terms also in Warriner)	13,915	6 per tuple	0.952	0.905	0.906
c. Warriner et al. (2013)	13,915	20 per term	0.914	0.689	0.770

Table 6: Split-half reliabilities (as measured by Pearson correlation) for valence, arousal, and dominance scores obtained from our annotations and the Warriner et al. annotations.

5.3 Reliability of the Annotations

A useful measure of quality is reproducibility of the end result—repeated independent manual annotations from multiple respondents should result in similar scores. To assess this reproducibility, we calculate average *split-half reliability (SHR)* over 100 trials. All annotations for an item (in our case, 4-tuples) are randomly split into two halves. Two sets of scores are produced independently from the two halves. Then the correlation between the two sets of scores is calculated. If the annotations are of good quality, then the correlation between the two halves will be high. Table 6 shows the split-half reliabilities (SHR) for valence, arousal, and dominance annotations. Row *a.* shows the SHR on the full set of terms in the VAD lexicon. Row *b.* shows the SHR on just the Warriner subset of terms in the VAD lexicon. Row *c.* shows the SHR reported by Warriner et al. (2013) on their annotations. Observe that the SHR scores for our annotations are markedly higher than those reported by Warriner et al. (2013), especially for arousal and dominance. All differences in SHR scores between rows b and c are statistically significant.

Summary of Main Results: The low correlations between the scores in our lexicon and the Warriner lexicon (especially for D and A) show that the scores in the two lexicons are substantially different. The scores for correlations across all pairs of dimensions in our lexicon are low ($r < 0.5$). SHR scores of 0.95 for valence, 0.9 for arousal, and 0.9 for dominance show for the first time that highly reliable fine-grained ratings can be obtained for valence, arousal, and dominance.

6 Shared Understanding of VAD Within and Across Demographic Groups

Human cognition and behaviour is impacted by evolutionary and socio-cultural factors. These factors are known to impact different groups of people differently (men vs. women, young vs. old, etc.). Thus it is not surprising that our understanding of the world may be slightly different depending on our demographic attributes. Consider gender—a key demographic attribute.[14] Men, women, and other genders are substantially more alike than they are different. However, they have encountered different socio-cultural influences for thousands of years. Often these disparities have been a means to exert unequal status and asymmetric power relations. Thus a crucial area in gender studies is to examine both the overt and subtle impacts of these socio-cultural influences, as well as ways to mitigate the inequity. Understanding how different genders perceive and use language is an important component of that research. Language use is also relevant to the understanding and treatment of neuropsychiatric disorders, such as sleep, mood, and anxiety disorders, which have been shown to occur more frequently in women than men (Bao and Swaab, 2011; Lewinsohn et al., 1998; McLean et al., 2011; Johnson et al., 2006; Chmielewski et al., 1995).

In addition to the VAD Lexicon (created by aggregating human judgments), we also make available the demographic information of the annotators. This demographic information along with the individual judgments on the best–worst tuples forms a significant resource in the study of how demographic attributes are correlated with our understanding of language. The data can be used to shed light on research questions such as: 'are there significant differences in the shared understanding of word meanings in men and women?', 'how is the social construct of gender reflected in language, especially in socio-political interactions?', 'does age impact our view of the valence, arousal, and dominance of concepts?', 'do people that view themselves as conscientious have slightly different judgments of valence, arousal, and dominance, than people who view themselves as easy going?', and so on.

[14]Note that the term *sex* refers to a biological attribute pertaining to the anatomy of one's reproductive system and sex chromosomes, whereas *gender* refers to a psycho-socio-cultural construct based on a person's sex or a person's self identification of levels of masculinity and femininity. One may identify their gender as female, male, agender, trans, queer, etc.

	V	A	D
f–f pairs	56.55	44.15	42.55
m–m pairs	56.88	43.80	43.55
f–m pairs	56.41	43.65	43.03

Table 7: Gender: Average agreement % on best–worst responses.

	V	A	D
f–f pairs vs. m–m pairs	y	y	y
f–f pairs vs. f–m pairs	-	y	y
m–m pairs vs. f–m pairs	y	-	y

Table 8: Gender: Significance of difference in average agreement scores (p = 0.05). 'y' = yes significant. '-' = not significant.

6.1 Experiments

We now describe experiments we conducted to determine whether demographic attributes impact how we judge words for valence, arousal, and dominance. For each demographic attribute, we partitioned the annotators into two groups: male (m) and female (f), ages 18 to 35 ($\leq$35) and ages over 35 ($>$35), and so on.[15] For each of the five personality traits, annotators are partitioned into the two groups shown in the bullet list of Section 4. We then calculated the extent to which people within the same group agreed with each other, and the extent to which people across groups agreed with each other on the VAD annotations (as described in the paragraph below). We also determined if the differences in agreement were statistically significant.

For each dimension (V, A, and D), we first collected only those 4-tuples where at least two female and at least two male responses were available. We will refer to this set as the *base set*. For each of the base set 4-tuples, we calculated three agreement percentages: 1. the percentage of all female–female best–worst responses where the two agreed with each other, 2. the percentage of all male–male responses where the two agreed with each other, and 3. the percentage of all female–male responses where the two agreed with each other. We then calculated the averages of the agreement percentages across all the 4-tuples in the base set. We conducted similar experiments for age groups and personality traits.

[15]For age, we chose 35 to create the two groups because several psychology and medical studies report changes in health and well-being at this age. Nonetheless, other partitions of age are also worth exploring.

	V	A	D
$\leq$35–$\leq$35 pairs	56.10	43.84	43.81
$>$35–$>$35 pairs	57.56	44.10	42.49
$\leq$35–$>$35 pairs	56.40	43.58	43.07

Table 9: Age: Average agreement % on best–worst responses.

	V	A	D
$\leq$35–$\leq$35 pairs vs. $>$35–$>$35 pairs	y	y	y
$\leq$35–$\leq$35 pairs vs. $\leq$35–$>$35 pairs	y	y	y
$>$35–$>$35 pairs vs. $\leq$35–$>$35 pairs	y	y	y

Table 10: Age: Significance of difference in average agreement scores (p = 0.05).

6.2 Results

Table 7 shows the results for gender. Note that the average agreement numbers are not expected to be high because often a 4-tuple may include two words that are close to each other in terms of the property of interest (V/A/D).[16] However, the relative values of the agreement percentages indicate the relative levels of agreements within groups and across groups.

Table 7 numbers indicate that women have a higher shared understanding of the degree of arousal of words (higher f–f average agreement scores on A), whereas men have a higher shared understanding of dominance and valence of words (higher m–m average agreement scores on V and D). The table also shows the cross-group (f–m) average agreements are the lowest for valence and arousal, but higher than f–f pairs for dominance. (Each of these agreements was determined from 1 to 1.5 million judgment pairs.)

Table 8 shows which of the Table 7 average agreements are statistically significantly different (shown with a 'y'). Significance values were calculated using the chi-square test for independence and significance level of 0.05. Observe that all score differences are statistically significant except for between f–f and f–m scores for V and m–m and f–m scores for A.

Tables 9 through 12 are similar to Tables 7 and 8, but for age groups and personality traits. Tables 9 and 10 show that respondents over the age of 35 obtain significantly higher agreements with each other on valence and arousal and lower agreements on dominance, than respondents aged 35 and under (with each other). Tables 11 and 12 show that

[16]Such disagreements are useful as they cause the two words to obtain scores close to each other.

	V	A	D
Agreeable (Ag) – Disagreeable (Di)			
# pairs	*1.0M*	*1.8M*	*1.7M*
Ag–Ag pairs	56.54	43.89	42.39
Di–Di pairs	55.76	43.63	43.61
Ag–Di pairs	56.28	43.57	43.01
Conscientious (Co) – Easygoing (Ea)			
# pairs	*0.9M*	*1.9M*	*1.5M*
Co–Co pairs	56.34	44.60	44.38
Ea–Ea pairs	56.39	43.15	41.36
Co–Ea pairs	56.39	43.77	42.52
Extrovert (Ex) – Introvert (In)			
# pairs	*0.9M*	*2.0M*	*1.6M*
Ex–Ex pairs	58.00	44.16	43.43
In–In pairs	56.49	43.78	42.16
Ex–In pairs	57.00	43.85	42.89
Neurotic (Ne) – Secure (Se)			
# pairs	*1.0M*	*1.8M*	*1.5M*
Ne–Ne pairs	56.33	43.78	41.98
Se–Se pairs	57.97	43.90	43.65
Ne–Se pairs	56.93	43.97	42.93
Open (Op) – Closed (Cl)			
# pairs	*0.8M*	*1.8M*	*1.3M*
Op–Op pairs	57.65	44.19	43.51
Cl–Cl pairs	56.39	43.52	43.23
Op–Cl pairs	56.90	44.03	43.36

Table 11: Personality Trait: Average agreement % on best–worst responses.

	V	A	D
Agreeable (Ag) – Disagreeable (Di)			
Ag–Ag vs. Di–Di	y	y	y
Ag–Ag vs. Ag–Di	y	y	y
Di–Di vs. Ag–Di	y	-	y
Conscientious (Co) – Easygoing (Ea)			
Co–Co vs. Ea–Ea	-	y	y
Co–Co vs. Co–Ea	-	y	y
Ea–Ea vs. Co–Ea	-	y	y
Extrovert (Ex) – Introvert (In)			
Ex–Ex vs. In–In	y	y	y
Ex–Ex vs. Ex–In	y	-	y
In–In vs. Ex–In	y	y	y
Neurotic (Ne) – Secure (Se)			
Ne–Ne vs. Se–Se	y	-	y
Ne–Ne vs. Ne–Se	y	-	y
Se–Se vs. Ne–Se	y	-	y
Open (Op) – Closed (Cl)			
Op–Op vs. Cl–Cl	y	y	y
Op–Op vs. Op–Cl	y	-	-
Cl–Cl vs. Op–Cl	y	y	-

Table 12: Personality Trait: Significance of difference in average agreement scores (p = 0.05).

some personality traits significantly impact a person's annotations of one or more of V, A, and D. Notably, those who view themselves as conscientious have a particularly higher shared understanding of the dominance of words, as compared to those who view themselves as easy going. They also have higher in-group agreement for arousal, than those who view themselves as easy going, but the difference for valence is not statistically significant. Also notable, is that those who view themselves as extroverts have a particularly higher shared understanding of the valence, arousal, and dominance of words, as compared to those who view themselves as introverts.

Finally, as a sanity check, we divided respondents into those whose CrowdFlower worker ids are odd and those whose worker ids are even. We then determined average agreements for even–even, odd-odd, and even–odd groups just as we did for the demographic variables. We found that, as expected, there were no significant differences in average agreements.

Summary of Main Results: We showed that several demographic attributes such as age, gender, and personality traits impact how we judge words for valence, arousal, and dominance. Further, people that share certain demographic attributes show a higher shared understanding of the relative rankings of words by (one or more of) V, A, or D than others. However, this raises new questions: why do certain demographic attributes impact our judgments of V, A, and D? Are there evolutionary forces that caused some groups such as women to develop a higher shared understanding or the arousal, whereas different evolutionary forces caused some groups, such as men, to have a higher shared understanding of dominance? We hope that the data collected as part of this project will spur further inquiry into these and other questions.

7 Applications and Future Work

The large number of entries in the VAD Lexicon and the high reliability of the scores make it useful for a number of research projects and applications. We list a few below:

- To provide features for sentiment or emotion detection systems. They can also be used to obtain sentiment-aware word embeddings and sentiment-aware sentence representations.
- To study the interplay between the basic emotion model and the VAD model of affect. The VAD lexicon can be used along with lists of words associated with emotions such as joy, sadness, fear, etc. to study the correlation of V, A, and D, with those emotions.

- To study the role emotion words play in high emotion intensity sentences or tweets. The Tweet Emotion Intensity Dataset has emotion intensity and valence scores for whole tweets (Mohammad and Bravo-Marquez, 2017). We will use the VAD lexicon to determine the extent to which high intensity and high valence tweets consist of high V, A, and D words, and to identify sentences that express high emotional intensity without using high V, A, and D words.
- To identify syllables that tend to occur in words with high VAD scores, which in turn can be used to generate names for literary characters and commercial products that have the desired affectual response.
- To identify high V, A, and D words in books and literature. To facilitate research in digital humanities. To facilitate work on literary analysis.
- As a source of gold (reference) scores, the entries in the VAD lexicon can be used in the evaluation of automatic methods of determining V, A, and D.
- To analyze V, A, ad D annotations for different groups of words, such as: hashtag words and emojis common in tweets, emotion denotating words, emotion associated words, neutral terms, words belonging to particular parts of speech such as nouns, verbs, and adjectives, etc.
- To analyze interactions between demographic groups and specific groups of words, for example, whether younger annotators have a higher shared understanding of tweet terms, whether a certain gender is associated with a higher shared understanding of adjectives, etc.
- To analyze the shared understanding of V, A, and D within and across geographic and language groups. We are interested in creating VAD lexicons for other languages. We can then explore characteristics of valence, arousal, and dominance that are common across cultures. We can also test whether some of the conclusions reached in this work apply only to English, or more broadly to multiple languages.
- The dataset is of use to psychologists and evolutionary linguists interested in determining how evolution shaped our representation of the world around us, and why certain personality traits are associated with higher or lower shared understanding of V, A, and D.

8 Conclusions

We obtained reliable human ratings of valence, arousal, and dominance for more than 20,000 English words. (It has about 40% more words than the largest existing manually created VAD lexicon). We used best–worst scaling to obtain fine-grained scores (and word rankings) and addressed issues of annotation consistency that plague traditional rating scale methods of annotation. We showed that the lexicon has split-half reliability scores of 0.95 for valence, 0.90 for arousal, and 0.90 for dominance. These scores are markedly higher than that of existing lexicons.

We analyzed demographic information to show that even though the annotations overall lead to consistent scores in repeated annotations, there exist statistically significant differences in agreements across demographic groups such as males and females, those above the age of 35 and those that are 35 or under, and across personality dimensions (extroverts and introverts, neurotic and secure, etc.). These results show that certain demographic attributes impact how we view the world around us in terms of the relative valence, arousal, and dominance of the concepts in it.

The NRC Valence, Arousal, and Dominance Lexicon is made available.[17] It can be used in combination with other manually created affect lexicons such as the NRC Word–Emotion Association Lexicon (Mohammad and Turney, 2013)[18] and the NRC Affect Intensity Lexicon (Mohammad, 2018).[19]

Acknowledgments

Many thanks to Svetlana Kiritchenko, Michael Wojatzki, Norm Vinson, and Tara Small for helpful discussions.

[17]**The NRC Valence, Arousal, and Dominance Lexicon** provides human ratings of valence, arousal, and dominance for more than 20,000 English words: http://saifmohammad.com/WebPages/nrc-vad.html

[18]**The NRC Emotion Lexicon** includes about 14,000 words annotated to indicate whether they are associated with any of the eight basic emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust): http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

[19]**The NRC Affect Intensity Lexicon** provides real-valued affect intensity scores for four basic emotions (anger, fear, sadness, joy): http://saifmohammad.com/WebPages/AffectIntensity.htm

References

Iris Bakker, Theo van der Voordt, Peter Vink, and Jan de Boon. 2014. Pleasure, arousal, dominance: Mehrabian and russell revisited. *Current Psychology*, 33(3):405–421.

Ai-Min Bao and Dick F Swaab. 2011. Sexual differentiation of the human brain: relation to gender identity, sexual orientation and neuropsychiatric disorders. *Frontiers in neuroendocrinology*, 32(2):214–226.

Murray R Barrick and Michael K Mount. 1991. The big five personality dimensions and job performance: a meta-analysis. *Personnel psychology*, 44(1):1–26.

Hans Baumgartner and Jan-Benedict E.M. Steenkamp. 2001. Response styles in marketing research: A cross-national investigation. *Journal of Marketing Research*, 38(2):143–156.

Margaret M Bradley and Peter J Lang. 1999. Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical report, The Center for Research in Psychophysiology, University of Florida.

Phillip M Chmielewski, Leyan OL Fernandes, Cindy M Yee, and Gregory A Miller. 1995. Ethnicity and gender in scales of psychosis proneness and mood disorders. *Journal of Abnormal Psychology*, 104(3):464.

Steven H. Cohen. 2003. Maximum difference scaling: Improved measures of importance and preference for segmentation. Sawtooth Software, Inc.

Herbert Aron David. 1963. *The method of paired comparisons*. Hafner Publishing Company, New York.

T. N. Flynn and A. A. J. Marley. 2014. Best-worst scaling: theory and methods. In Stephane Hess and Andrew Daly, editors, *Handbook of Choice Modelling*, pages 178–201. Edward Elgar Publishing.

Eric O Johnson, Thomas Roth, Lonni Schultz, and Naomi Breslau. 2006. Epidemiology of dsm-iv insomnia in adolescence: lifetime prevalence, chronicity, and an emergent gender difference. *Pediatrics*, 117(2):e247–e256.

David Jurgens. 2013. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Atlanta, GA, USA.

David Jurgens, Saif M. Mohammad, Peter Turney, and Keith Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 356–364, Montréal, Canada.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing reliable fine-grained sentiment associations by crowdsourcing and best–worst scaling. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California.

Svetlana Kiritchenko and Saif M. Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of The Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Peer M Lewinsohn, Ian H Gotlib, Mark Lewinsohn, John R Seeley, and Nicholas B Allen. 1998. Gender differences in anxiety disorders and anxiety symptoms in adolescents. *Journal of abnormal psychology*, 107(1):109.

Jordan J. Louviere. 1991. Best-worst scaling: A model for the largest difference judgments. Working Paper.

Jordan J. Louviere, Terry N. Flynn, and A. A. J. Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press.

Carmen P McLean, Anu Asnaani, Brett T Litz, and Stefan G Hofmann. 2011. Gender differences in anxiety disorders: prevalence, course of illness, comorbidity and burden of illness. *Journal of psychiatric research*, 45(8):1027–1035.

Saif Mohammad. 2012. #Emotional Tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 246–255, Montréal, Canada.

Saif M. Mohammad. 2018. Word affect intensities. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. WASSA-2017 shared task on emotion intensity. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, Copenhagen, Denmark.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Saif M. Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif M. Mohammad and Svetlana Kiritchenko. 2018. Understanding emotions: A dataset of tweets to study interactions between affect categories. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Agnes Moors, Jan De Houwer, Dirk Hermans, Sabine Wanmaker, Kevin Van Schie, Anne-Laura Van Harmelen, Maarten De Schryver, Jeffrey De Winne, and Marc Brysbaert. 2013. Norms of valence, arousal, dominance, and age of acquisition for 4,300 dutch words. *Behavior research methods*, 45(1):169–177.

Bryan Orme. 2009. Maxdiff analysis: Simple counting, individual-level logit, and HB. Sawtooth Software, Inc.

C.E. Osgood, Suci G., and P. Tannenbaum. 1957. *The measurement of meaning*. University of Illinois Press.

Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. *Emotion: Theory, research, and experience*, 1(3):3–33.

Stanley Presser and Howard Schuman. 1996. *Questions and Answers in Attitude Surveys: Experiments on Question Form, Wording, and Context*. SAGE Publications, Inc.

Jaime Redondo, Isabel Fraga, Isabel Padrón, and Montserrat Comesaña. 2007. The spanish adaptation of anew (affective norms for english words). *Behavior research methods*, 39(3):600–605.

James A Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161.

James A Russell. 2003. Core affect and the psychological construction of emotion. *Psychological review*, 110(1):145.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *arXiv preprint arXiv:1405.1605*.

Philip Stone, Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie, and associates. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Louis L. Thurstone. 1927. A law of comparative judgment. *Psychological review*, 34(4):273.

Melissa LH Võ, Markus Conrad, Lars Kuchinke, Karolina Urton, Markus J Hofmann, and Arthur M Jacobs. 2009. The berlin affective word list reloaded (bawl-r). *Behavior research methods*, 41(2):534–538.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods*, 45(4):1191–1207.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Building emotion lexicon from weblog corpora. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 133–136.

Liang-Chih Yu, Jin Wang, K Robert Lai, and Xue-jie Zhang. 2015. Predicting valence-arousal ratings of words using a weighted graph method. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 788–793.

Comprehensive Supersense Disambiguation of English Prepositions and Possessives

Nathan Schneider[*]
Georgetown University

Jena D. Hwang
IHMC

Vivek Srikumar
University of Utah

Jakob Prange
Austin Blodgett
Georgetown University

Sarah R. Moeller
University of Colorado Boulder

Aviram Stern
Adi Bitan
Omri Abend
Hebrew University of Jerusalem

Abstract

Semantic relations are often signaled with prepositional or possessive marking—but extreme polysemy bedevils their analysis and automatic interpretation. We introduce a new annotation scheme, corpus, and task for the disambiguation of prepositions and possessives in English. Unlike previous approaches, our annotations are comprehensive with respect to types and tokens of these markers; use broadly applicable supersense classes rather than fine-grained dictionary definitions; unite prepositions and possessives under the same class inventory; and distinguish between a marker's *lexical* contribution and the *role* it marks in the context of a predicate or scene. Strong interannotator agreement rates, as well as encouraging disambiguation results with established supervised methods, speak to the viability of the scheme and task.

1 Introduction

Grammar, as per a common metaphor, gives speakers of a language a shared toolbox to construct and deconstruct meaningful and fluent utterances. Being highly analytic, English relies heavily on word order and closed-class function words like prepositions, determiners, and conjunctions. Though function words bear little semantic content, they are nevertheless crucial to the meaning. Consider prepositions: they serve, for example, to convey place and time (*We met* ***at/in/outside*** *the restaurant* ***for/after*** *an hour*), to express configurational relationships like quantity, possession, part/whole, and membership (*the coats* ***of*** *dozens* ***of*** *children* ***in*** *the class*), and to indicate semantic roles in argument structure (*Grandma cooked dinner* ***for*** *the children* vs. *Grandma cooked the children* ***for*** *dinner*). Frequent prepositions like **for** are maddeningly polysemous, their interpretation depending especially on the object of the preposition—*I rode the bus* ***for*** *5 dollars/minutes*—and the governor of the prepositional phrase (PP): *I Ubered/asked* ***for*** *\$5*. Possessives are similarly ambiguous: *Whistler****'s*** *mother/painting/hat/death*. Semantic interpretation requires some form of sense disambiguation, but arriving at a linguistic representation that is flexible enough to generalize across usages and types, yet simple enough to support reliable annotation, has been a daunting challenge (§2).

(1) I was booked **for**/DURATION 2 nights **at**/LOCUS this hotel **in**/TIME Oct 2007 .

(2) I went **to**/GOAL ohm **after**/EXPLANATION⇝TIME reading some **of**/QUANTITY⇝WHOLE the reviews .

(3) It was very upsetting to see this kind **of**/SPECIES behavior especially **in_front_of**/LOCUS **my**/SOCIALREL⇝GESTALT four year_old .

Figure 1: Annotated sentences from our corpus.

This work represents a new attempt to strike that balance. Building on prior work, we argue for an approach to describing English preposition and possessive semantics with broad coverage. Given the semantic overlap between prepositions and possessives (*the hood* ***of*** *the car* vs. *the car****'s*** *hood* or ***its*** *hood*), we analyze them using the same inventory of semantic labels.[1] Our contributions include:

- a new hierarchical **inventory** ("SNACS") of 50 supersense classes, extensively documented in guidelines for English (§3);
- a gold-standard **corpus** with *comprehensive* annotations: all types and tokens of prepositions and possessives are disambiguated (§4; example sentences appear in figure 1);
- an **interannotator agreement** study that

[*] nathan.schneider@georgetown.edu

[1] Some uses of certain other closed-class markers—intransitive particles, subordinators, infinitive **to**—are also included (§3.1).

shows the scheme is reliable and generalizes across genres—and for the first time demonstrating empirically that the lexical semantics of a preposition can sometimes be detached from the PP's semantic role (§5);

- **disambiguation** experiments with two supervised classification architectures to establish the difficulty of the task (§6).

2 Background: Disambiguation of Prepositions and Possessives

Studies of preposition semantics in linguistics and cognitive science have generally focused on the domains of space and time (e.g., Herskovits, 1986; Bowerman and Choi, 2001; Regier, 1996; Khetarpal et al., 2009; Xu and Kemp, 2010; Zwarts and Winter, 2000) or on motivated polysemy structures that cover additional meanings beyond core spatial senses (Brugman, 1981; Lakoff, 1987; Tyler and Evans, 2003; Lindstromberg, 2010). Possessive constructions can likewise denote a number of semantic relations, and various factors—including semantics—influence whether attributive possession in English will be expressed with **of**, or with **'s** and possessive pronouns (the 'genitive alternation'; Taylor, 1996; Nikiforidou, 1991; Rosenbach, 2002; Heine, 2006; Wolk et al., 2013; Shih et al., 2015).

Corpus-based computational work on semantic disambiguation specifically of prepositions and possessives[2] falls into two categories: the **lexicographic/word sense disambiguation** approach (Litkowski and Hargraves, 2005, 2007; Litkowski, 2014; Ye and Baldwin, 2007; Saint-Dizier, 2006; Dahlmeier et al., 2009; Tratz and Hovy, 2009; Hovy et al., 2010, 2011; Tratz and Hovy, 2013), and the **semantic class** approach (Moldovan et al., 2004; Badulescu and Moldovan, 2009; O'Hara and Wiebe, 2009; Srikumar and Roth, 2011, 2013; Schneider et al., 2015, 2016; Hwang et al., 2017, see also Müller et al., 2012 for German). The lexicographic approach can capture finer-grained meaning distinctions, at a risk of relying upon idiosyncratic and potentially incomplete dictionary definitions. The semantic class approach, which we follow here, focuses on commonalities in meaning across multiple lexical items, and aims to generalize more easily to new types and usages.

[2]Of course, meanings marked by prepositions/possessives are to some extent captured in predicate-argument or graph-based meaning representations (e.g., Palmer et al., 2005; Fillmore and Baker, 2009; Oepen et al., 2016; Banarescu et al., 2013) and domain-centric representations like TimeML and ISO-Space (Pustejovsky et al., 2003, 2012).

The most recent class-based approach to prepositions was our initial framework of 75 **preposition supersenses** arranged in a multiple inheritance taxonomy (Schneider et al., 2015, 2016). It was based largely on relation/role inventories of Srikumar and Roth (2013) and VerbNet (Bonial et al., 2011; Palmer et al., 2017). The framework was realized in version 3.0 of our comprehensively annotated corpus, STREUSLE[3] (Schneider et al., 2016). However, several limitations of our approach became clear to us over time.

First, as pointed out by Hwang et al. (2017), the one-label-per-token assumption in STREUSLE is flawed because it in some cases puts into conflict the semantic role of the PP with respect to a predicate, and the lexical semantics of the preposition itself. Hwang et al. (2017) suggested a solution, discussed in §3.3, but did not conduct an annotation study or release a corpus to establish its feasibility empirically. We address that gap here.

Second, 75 categories is an unwieldy number for both annotators and disambiguation systems. Some are quite specialized and extremely rare in STREUSLE 3.0, which causes data sparseness issues for supervised learning. In fact, the only published disambiguation system for preposition supersenses collapsed the distinctions to just 12 labels (Gonen and Goldberg, 2016). Hwang et al. (2017) remarked that solving the aforementioned problem could remove the need for many of the specialized categories and make the taxonomy more tractable for annotators and systems. We substantiate this here, defining a new hierarchy with just 50 categories (SNACS, §3) and providing disambiguation results for the full set of distinctions.

Finally, given the semantic overlap of possessive case and the preposition **of**, we saw an opportunity to broaden the application of the scheme to include possessives. Our reannotated corpus, STREUSLE 4.0, thus has supersense annotations for over 1000 possessive tokens that were not semantically annotated in version 3.0. We include these in our annotation and disambiguation experiments alongside reannotated preposition tokens.

3 Annotation Scheme

3.1 Lexical Categories of Interest

Apart from canonical prepositions and possessives, there are many lexically and semantically overlap-

[3]`https://github.com/nert-gu/streusle/`

ping closed-class items which are sometimes classified as other parts of speech, such as adverbs, particles, and subordinating conjunctions. *The Cambridge Grammar of the English Language* (Huddleston and Pullum, 2002) argues for an expansive definition of 'preposition' that would encompass these other categories. As a practical measure, we decided to encourage annotators to focus on the semantics of these functional items rather than their syntax, so we take an inclusive stance.

Another consideration is developing annotation guidelines that can be adapted for other languages. This includes languages which have *post*positions, *circum*positions, or *in*positions rather than prepositions; the general term for such items is *ad*positions.[4] English possessive marking (via **'s** or possessive pronouns like **my**) is more generally an example of *case* marking. Note that prepositions (4a–4c) differ in word order from possessives (4d), though semantically the object of the preposition and the possessive nominal pattern together:

(4) a. eat **in** a restaurant
 b. the man **in** a blue shirt
 c. the wife **of** the ambassador
 d. the ambassador**'s** wife

Cross-linguistically, adpositions and case marking are closely related, and in general both grammatical strategies can express similar kinds of semantic relations. This motivates a common semantic inventory for adpositions and case.

We also cover multiword prepositions (e.g., **out_of**, **in_front_of**), intransitive particles (*He flew* ***away***), purpose infinitive clauses (*Open the door* ***to*** *let in some air*[5]), prepositions with clausal complements (*It rained* ***before*** *the party started*), and idiomatic prepositional phrases (***at****_large*). Our annotation guidelines give further details.

3.2 The SNACS Hierarchy

The hierarchy of preposition and possessive supersenses, which we call Semantic Network of Adposition and Case Supersenses (SNACS), is shown in figure 2. It is simpler than its predecessor—Schneider et al.'s (2016) preposition supersense hierarchy—in both size and structural complexity.

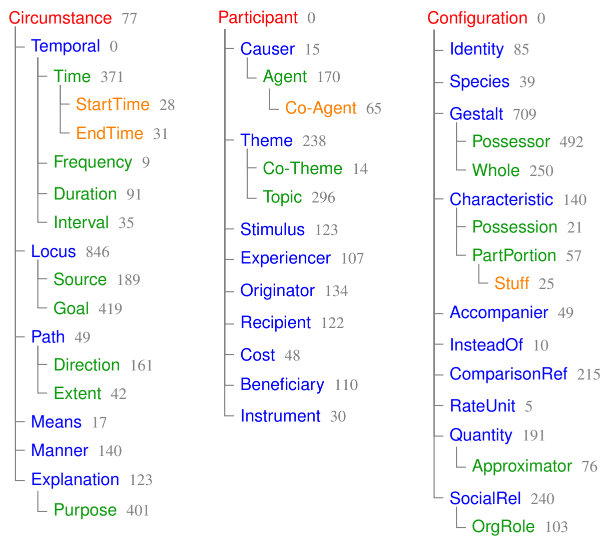


Figure 2: SNACS hierarchy of 50 supersenses and their token counts in the annotated corpus described in §4. Counts are of direct uses of labels, excluding uses of subcategories. Role and function positions are not distinguished (so if a token has different role and function labels, it will count toward two supersense frequencies).

SNACS has 50 supersenses at 4 levels of depth; the previous hierarchy had 75 supersenses at 7 levels. The top-level categories are the same:

- CIRCUMSTANCE: Circumstantial information, usually non-core properties of events (e.g., location, time, means, purpose)
- PARTICIPANT: Entity playing a role in an event
- CONFIGURATION: Thing, usually an entity or property, involved in a static relationship to some other entity

The 3 subtrees loosely parallel adverbial adjuncts, event arguments, and adnominal complements, respectively. The PARTICIPANT and CIRCUMSTANCE subtrees primarily reflect semantic relationships prototypical to verbal arguments/adjuncts and were inspired by VerbNet's thematic role hierarchy (Palmer et al., 2017; Bonial et al., 2011). Many CIRCUMSTANCE subtypes, like LOCUS (the concrete or abstract location of something), can be governed by eventive and non-eventive nominals as well as verbs: *eat* ***in*** *the restaurant*, *a party* ***in*** *the restaurant*, *a table* ***in*** *the restaurant*. CONFIGURATION mainly encompasses non-spatiotemporal relations holding between entities, such as quantity, possession, and part/whole. Unlike the previous hierarchy, SNACS does not use multiple inheritance, so there is no overlap between the 3 regions.

The supersenses can be understood as roles in fundamental types of scenes (or schemas) such as: LOCATION—THEME is located at LO-

[4]In English, **ago** is arguably a postposition because it follows rather than precedes its complement: *five minutes* ***ago***, not ****ago*** *five minutes*.

[5]**To** can be rephrased as **in_order_to** and have prepositional counterparts like in *Open the door* ***for*** *some air*.

CUS; MOTION—THEME moves from SOURCE along PATH to GOAL; TRANSITIVE ACTION—AGENT acts on THEME, perhaps using an INSTRUMENT; POSSESSION—POSSESSION belongs to POSSESSOR; TRANSFER—THEME changes possession from ORIGINATOR to RECIPIENT, perhaps with COST; PERCEPTION—EXPERIENCER is mentally affected by STIMULUS; COGNITION—EXPERIENCER contemplates TOPIC; COMMUNICATION—information (TOPIC) flows from ORIGINATOR to RECIPIENT, perhaps via an INSTRUMENT. For AGENT, CO-AGENT, EXPERIENCER, ORIGINATOR, RECIPIENT, BENEFICIARY, POSSESSOR, and SOCIALREL, the object of the preposition is prototypically animate.

Because prepositions and possessives cover a vast swath of semantic space, limiting ourselves to 50 categories means we need to address a great many nonprototypical, borderline, and special cases. We have done so in a 75-page annotation manual with over 400 example sentences (Schneider et al., 2018).

Finally, we note that the Universal Semantic Tagset (Abzianidze and Bos, 2017) defines a cross-linguistic inventory of semantic classes for content and function words. SNACS takes a similar approach to prepositions and possessives, which in Abzianidze and Bos's (2017) specification are simply tagged REL, which does not disambiguate the nature of the relational meaning. Our categories can thus be understood as refinements to REL.

3.3 Adopting the Construal Analysis

Hwang et al. (2017) have pointed out the perils of teasing apart and generalizing preposition semantics so that each use has a clear supersense label. One key challenge they identified is that the preposition itself and the situation as established by the verb may suggest different labels. For instance:

(5) a. Vernon works **at** Grunnings.
 b. Vernon works **for** Grunnings.

The semantics of the *scene* in (5a, 5b) is the same: it is an employment relationship, and the PP contains the employer. SNACS has the label ORGROLE for this purpose.[6] At the same time, **at** in (5a) strongly suggests a locational relationship, which would correspond to the label LOCUS; consistent with this hypothesis, *Where does Vernon work?* is a perfectly good way to ask a question that could be answered by the PP. In this example, then, there is overlap between locational meaning and organizational-belonging meaning. (5b) is similar except the **for** suggests a notion of BENEFICIARY: the employee is working *on behalf of* the employer. Annotators would face a conundrum if forced to pick a single label when multiple ones appear to be relevant. Schneider et al. (2016) handled overlap via multiple inheritance, but entertaining a new label for every possible case of overlap is impractical, as this would result in a proliferation of supersenses.

Instead, Hwang et al. (2017) suggest a **construal analysis** in which the lexical semantic contribution, or henceforth the **function**, of the preposition itself may be distinct from the semantic role or relation mediated by the preposition in a given sentence, called the **scene role**. The notion of scene role is a widely accepted idea that underpins the use of semantic or thematic roles: semantics licensed by the governor[7] of the prepositional phrase dictates its relationship to the prepositional phrase. The innovative claim is that, in addition to a preposition's relationship with its head, the prepositional *choice* introduces another layer of meaning or **construal** that brings additional nuance, creating the difficulty we see in the annotation of (5a, 5b). Construal is notated by ROLE⇝FUNCTION. Thus, (5a) would be annotated ORGROLE⇝LOCUS and (5b) as ORGROLE⇝BENEFICIARY to expose their common truth-semantic meaning but slightly different portrayals owing to the different prepositions.

Another useful application of the construal analysis is with the verb *put*, which can combine with any locative PP to express a destination:

(6) Put it **on/by/behind/on_top_of/**... the door.
 GOAL⇝LOCUS

I.e., the preposition signals a LOCUS, but the door serves as the GOAL with respect to the scene. This approach also allows for resolution of various se-

[6] ORGROLE is defined as "Either a party in a relation between an organization/institution and an individual who has a stable affiliation with that organization, such as membership or a business relationship."

[7] By "governor" of the preposition or prepositional phrase, we mean the head of the phrase to which the PP attaches in a constituency representation. In a dependency representation, this would be the head of the preposition itself or of the object of the preposition depending on which convention is used for PP headedness: e.g., the preposition heads the PP in CoNLL and Stanford Dependencies whereas the object is the head in Universal Dependencies. The governor is most often a verb or noun. Where the PP is a predicate complement (e.g. *Vernon is **with** Grunnings*), there is no governor to specify the nature of the scene, so annotators must rely on world knowledge and context to determine the scene.

	Train	Dev	Test	Total
Documents	347	192	184	723
Sentences	2,723	554	535	3,812
Tokens	44,804	5,394	5,381	55,579
Annotated targets	4,522	453	480	5,455
Role = function	3,101	291	310	3,702
P or PP	3,397	341	366	4,104
Multiword unit	256	25	24	305
Infinitive **to**	201	26	20	247
Genitive clitic (**'s**)	52	6	1	59
Possessive pronoun	872	80	93	1,045
Attested SNACS labels	47	46	44	47
Unique scene roles	46	43	41	47
Unique functions	41	38	37	41
Unique pairs	167	79	87	177
Role = function	41	33	34	41

Table 1: Counts for the data splits used in our experiments.

mantic phenomena including perceptual scenes (e.g., *I care* ***about*** *education*, where **about** is both the topic of cogitation and perceptual stimulus of caring: STIMULUS⇝TOPIC), and fictive motion (Talmy, 1996), where static location is described using motion verbiage (as in *The road runs* ***through*** *the forest*: LOCUS⇝PATH).

Both role and function slots are filled by supersenses from the SNACS hierarchy. Annotators have the option of using distinct supersenses for the role and function; in general it is not a requirement (though we stipulate that certain SNACS supersenses can only be used as the role). When the same label captures both role and function, we do not repeat it: *Vernon lives* ***in***/LOCUS *England*. Figure 1 shows some real examples from our corpus.

We apply the construal analysis in SNACS annotation of our corpus to test its feasibility. It has proved useful not only for prepositions, but also possessives, where the general sense of possession may overlap with other scene relations, like creator/initial-possessor (ORIGINATOR): *Da Vinci'****s***/ORIGINATOR⇝POSSESSOR *sculptures*.

4 Annotated Reviews Corpus

We applied the SNACS annotation scheme (§3) to prepositions and possessives in the STREUSLE corpus (§2), a collection of online consumer reviews taken from the English Web Treebank (Bies et al., 2012). The sentences from the English Web Treebank also comprise the primary reference treebank for English Universal Dependencies (UD; Nivre et al., 2016), and we bundle the UD version 2 syntax alongside our annotations. Table 1 shows the total number of tokens present and those that we annotated. Altogether, 5,455 tokens were annotated for scene role and function.

Rank	Role		Function	
1	LOCUS	636	LOCUS	780
2	POSSESSOR	381	GESTALT	699
⋮	⋮		⋮	
last	DIRECTION	1	POSSESSION	2

Table 2: Most and least frequent role and function labels.

The new hierarchy and annotation guidelines were developed by consensus. The original preposition supersense annotations were placed in a spreadsheet and discussed. While most tokens were unambiguously annotated, some cases required a new analysis throughout the corpus. For example, the functions of **for** were so broad that they needed to be (manually) clustered before mapping clusters onto hierarchy labels. Unusual or rare contexts also presented difficulties. Where the correct supersense remained unclear, specific instructions and examples were included in the guidelines. Possessives were not covered by the original preposition supersense annotations, and thus were annotated from scratch.[8] **Special labels** were applied to tokens deemed not to be prepositions or possessives evoking semantic relations, including uses of the infinitive marker that do not fall within the scope of SNACS (487 tokens: a majority of infinitives) and preposition-initial discourse expressions (e.g. ***after****_all*) and coordinating conjunctions (***as****_well_as*).[9] Other tokens requiring special labels are the opaque possessive slot in a multiword idiom (12 tokens), and tokens where unintelligble, incomplete, marginal, or nonnative usage made it impossible to assign a supersense (48 tokens).

Table 2 shows the most and least common labels occurring as scene role and function. Three labels never appear in the annotated corpus: TEMPORAL from the CIRCUMSTANCE hierarchy, and PARTICIPANT and CONFIGURATION which are both the highest supersense in their respective hierarchies. While all remaining supersenses are attested as scene roles, there are some that never occur as functions, such as ORIGINATOR, which is most often realized as POSSESSOR or SOURCE, and EXPERIENCER. It is interesting to note that every subtype of CIRCUMSTANCE (except TEMPORAL) appears as both scene role and function, whereas many of the subtypes of the other two hierarchies are lim-

[8] Blodgett and Schneider (2018) detail the extension of the scheme to possessives.

[9] In the corpus, lexical expression tokens appear alongside a **lexical category** indicating which inventory of supersenses, if any, applies. SNACS-annotated units are those with ADP (adposition), PP, PRON.POSS (possessive pronoun), etc., whereas DISC (discourse) and CCONJ expressions do not receive any supersense. Refer to the STREUSLE README for details.

ited to either role or function. This reflects our view that prepositions primarily capture circumstantial notions such as space and time, but have been extended to cover other semantic relations.[10]

5 Interannotator Agreement Study

Because the online reviews corpus was so central to the development of our guidelines, we sought to estimate the reliability of the annotation scheme on a new corpus in a new genre. We chose Saint-Exupéry's novella *The Little Prince*, which is readily available in many languages and has been annotated with semantic representations such as AMR (Banarescu et al., 2013). The genre is markedly different from online reviews—it is quite literary, and employs archaic or poetic figures of speech. It is also a translation from French, contributing to the markedness of the language. This text is therefore a challenge for an annotation scheme based on colloquial contemporary English. We addressed this issue by running 3 practice rounds of annotation on small passages from *The Little Prince*, both to assess whether the scheme was applicable without major guidelines changes and to prepare the annotators for this genre. For the final annotation study, we chose chapters 4 and 5, in which 242 markables of 52 types were identified heuristically (§6.2). The types *of*, *to*, *in*, *as*, *from*, and *for*, as well as possessives, occurred at least 10 times. Annotators had the option to mark units as false positives using special labels (see §4) in addition to expressing uncertainty about the unit.

For the annotation process, we adapted the open source web-based annotation tool UCCAApp (Abend et al., 2017) to our workflow, by extending it with a type-sensitive ranking module for the list of categories presented to the annotators.

Annotators. Five annotators (A, B, C, D, E), all authors of this paper, took part in this study. All are computational linguistics researchers with advanced training in linguistics. Their involvement in the development of the scheme falls on a spectrum, with annotator A being the most active figure in guidelines development, and annotator E not being involved in developing the guidelines and learning the scheme solely from reading the manual. Annotators A, B, and C are native speakers of English, while Annotators D and E are nonnative but highly fluent speakers.

	Labels	Role	Function
Exact	47	74.4%	81.3%
Depth-3	43	75.0%	81.8%
Depth-2	26	79.9%	87.4%
Depth-1	3	92.6%	93.9%

Table 3: Interannotator agreement rates (pairwise averages) on *Little Prince* sample (216 tokens) with different levels of hierarchy coarsening according to figure 2 ("Exact" means no coarsening). "Labels" refers to the number of distinct labels that annotators could have provided at that level of coarsening. Excludes tokens where at least one annotator assigned a non-semantic label.

Results. In the *Little Prince* sample, 40 out of 47 possible supersenses were applied at least once by some annotator; 36 were applied at least once by a majority of annotators; and 33 were applied at least once by all annotators. APPROXIMATOR, CO-THEME, COST, INSTEADOF, INTERVAL, RATEUNIT, and SPECIES were not used by any annotator.

To evaluate interannotator agreement, we excluded 26 tokens for which at least one annotator has assigned a non-semantic label, considering only the 216 tokens that were identified correctly as SNACS targets and were clear to all annotators. Despite varying exposure to the scheme, there is no obvious relationship between annotators' backgrounds and their agreement rates.[11]

Table 3 shows the interannotator agreement rates, averaged across all pairs of annotators. Average agreement is 74.4% on the scene role and 81.3% on the function (row 1).[12] All annotators agree on the role for 119, and on the function for 139 tokens. Agreement is higher on the function slot than on the scene role slot, which implies that the former is an easier task than the latter. This is expected considering the definition of construal: the *function* of an adposition is more lexical and less context-dependent, whereas the *role* depends on the context (the scene) and can be highly idiomatic (§3.3).

The supersense hierarchy allows us to analyze agreement at different levels of granularity (rows

[10]All told, 41 supersenses are attested as both role and function for the same token, and there are 136 unique construal combinations where the role differs from the function. Only four supersenses are never found in such a divergent construal: EXPLANATION, SPECIES, STARTTIME, RATEUNIT. Except for RATEUNIT which occurs only 5 times, their narrow use does not arise because they are rare. EXPLANATION, for example, occurs over 100 times, more than many labels which often appear in construal.

[11]See table 7 in appendix A for a more detailed description of the annotators' backgrounds and pairwise IAA results.

[12]Average of pairwise Cohen's κ is 0.733 and 0.799 on, respectively, role and function, suggesting strong agreement. However, it is worth noting that annotators selected labels from a ranked list, with the ranking determined by preposition type. The model of chance agreement underlying κ does not take the identity of the preposition into account, and thus likely underestimates the probability of chance agreement.

2–4 in table 3; see also confusion matrix in supplement). Coarser-grained analyses naturally give better agreement, with depth-1 coarsening into only 3 categories. Results show that most confusions are local with respect to the hierarchy.

6 Disambiguation Systems

We now describe systems that identify and disambiguate SNACS-annotated prepositions and possessives in two steps. Target identification heuristics (§6.2) first determine which tokens (single-word or multiword) should receive a SNACS supersense. A supervised classifier then predicts a supersense analysis for each identified target. The research objectives are (a) to study the ability of statistical models to learn roles and functions of prepositions and possessives, and (b) to compare two different modeling strategies (feature-rich and neural), and the impact of syntactic parsing.

6.1 Experimental Setup

Our experiments use the reviews corpus described in §4. We adopt the official training/development/ test splits of the Universal Dependencies (UD) project; their sizes are presented in table 1. All systems are trained on the training set only and evaluated on the test set; the development set was used for tuning hyperparameters. Gold tokenization was used throughout. Only targets with a semantic supersense analysis involving labels from figure 2 were included in training and evaluation—i.e., tokens with special labels (see §4) were excluded.

To test the impact of automatic syntactic parsing, models in the **auto syntax** condition were trained and evaluated on automatic lemmas, POS tags, and Basic Universal Dependencies (according to the v1 standard) produced by Stanford CoreNLP version 3.8.0 (Manning et al., 2014).[13] Named entity tags from the default 12-class CoreNLP model were used in all conditions.

6.2 Target Identification

§3.1 explains that the categories in our scheme apply not only to (transitive) adpositions in a very narrow definition of the term, but also to lexical items that traditionally belong to variety of syntactic classes (such as adverbs and particles), as well as possessive case markers and multiword expressions. 61.2% of the units annotated in our corpus are adpositions according to gold POS annotation, 20.2% are possessives, and 18.6% belong to other POS classes. Furthermore, 14.1% of tokens labeled as adpositions or possessives are not annotated because they are part of a multiword expression (MWE). It is therefore neither obvious nor trivial to decide which tokens and groups of tokens should be selected as targets for SNACS annotation.

To facilitate both manual annotation and automatic classification, we developed heuristics for identifying annotation targets. The algorithm first scans the sentence for known multiword expressions, using a blacklist of non-prepositional MWEs that contain preposition tokens (e.g., *take_care_of*) and a whitelist of prepositional MWEs (multiword prepositions like **out_of** and PP idioms like **in_town**). Both lists were constructed from the training data. From segments unaffected by the MWE heuristics, single-word candidates are identified by matching a high-recall set of parts of speech, then filtered through 5 different heuristics for adpositions, possessives, subordinating conjunctions, adverbs, and infinitivals. Most of these filters are based on lexical lists learned from the training portion of the STREUSLE corpus, but there are some specific rules for infinitivals that handle *for*-subjects (*I opened the door for Steve* **to** *take out the trash*—**to**, but not *for*, should receive a supersense) and comparative constructions with *too* and *enough* (*too short* **to** *ride*).

6.3 Classification

The next step of disambiguation is predicting the role and function labels. We explore two different modeling strategies.

Feature-rich Model. Our first model is based on the features for preposition relation classification developed by Srikumar and Roth (2013), which were themselves extended from the preposition sense disambiguation features of Hovy et al. (2010). We briefly describe the feature set here, and refer the reader to the original work for further details. At a high level, it consists of features extracted from selected neighboring words in the dependency tree (i.e., heuristically identified governor and object) and in the sentence (previous verb, noun and adjective, and next noun). In addition, all these features are also conjoined with the lemma of the rightmost word in the preposition token to capture

[13]The CoreNLP parser was trained on all 5 genres of the English Web Treebank—i.e., a superset of our training set. Gold syntax follows the UDv2 standard, whereas the classifiers in the auto syntax conditions are trained and tested with UDv1 parses produced by CoreNLP.

target-specific interactions with the labels. The features extracted from each neighboring word are listed in the supplementary material.

Using these features extracted from targets, we trained two multi-class SVM classifiers to predict the role and function labels using the LIBLINEAR library (Fan et al., 2008).

Neural Model. Our second classifier is a multi-layer perceptron (MLP) stacked on top of a BiLSTM. For every sentence, tokens are first embedded using a concatenation of fixed pre-trained word2vec (Mikolov et al., 2013) embeddings of the word and the lemma, and an internal embedding vector, which is updated during training.[14] Token embeddings are then fed into a 2-layer BiLSTM encoder, yielding a list of token representations.

For each identified target unit *u*, we extract its first token, and its governor and object headword. For each of these tokens, we construct a feature vector by concatenating its token representation with embeddings of its (1) language-specific POS tag, (2) UD dependency label, and (3) NER label. We additionally concatenate embeddings of *u*'s lexical category, a syntactic label indicating whether *u* is predicative/stranded/subordinating/none of these, and an indicator of whether either of the two tokens following the unit is capitalized. All these embeddings, as well as internal token embedding vectors, are considered part of the model parameters and are initialized randomly using the Xavier initialization (Glorot and Bengio, 2010). A NONE label is used when the corresponding feature is not given, both in training and at test time. The concatenated feature vector for *u* is fed into two separate 2-layered MLPs, followed by a separate softmax layer that yields the predicted probabilities for the role and function labels.

We tuned hyperparameters on the development set to maximize F-score (see supplementary material). We used the cross-entropy loss function, optimizing with simple gradient ascent for 80 epochs with minibatches of size 20. Inverted dropout was used during training. The model is implemented with the DyNet library (Neubig et al., 2017).

The model architecture is largely comparable to that of Gonen and Goldberg (2016), who experimented with a coarsened version of STREUSLE 3.0. The main difference is their use of unlabeled multilingual datasets to improve prediction by exploiting the differences in preposition ambiguities across languages.

[14] Word2vec is pre-trained on the Google News corpus. Zero vectors are used where vectors are not available.

Syntax	P	R	F
gold	88.8	89.6	89.2
auto	86.0	85.8	85.9

Table 4: Target identification results for disambiguation.

6.4 Results & Analysis

Following the two-stage disambiguation pipeline (i.e. target identification and classification), we separate the evaluation across the phases. Table 4 reports the precision, recall, and F-score (P/R/F) of the target identification heuristics. Table 5 reports the disambiguation performance of both classifiers with gold (left) and automatic target identification (right). We evaluate each classifier along three dimensions—role and function independently, and full (i.e. both role and function together). When we have the gold targets, we only report accuracy because precision and recall are equal. With automatically identified targets, we report P/R/F for each dimension. Both tables show the impact of syntactic parsing on quality. The rest of this section presents analyses of the results along various axes.

Target identification. The identification heuristics described in §6.2 achieve an F_1 score of 89.2% on the test set using gold syntax.[15] Most false positives (47/54=87%) can be ascribed to tokens that are part of a (non-adpositional or larger adpositional) multiword expression. 9 of the 50 false negatives (18%) are rare multiword expressions not occurring in the training data and there are 7 partially identified ones, which are counted as both false positives and false negatives.

Automatically generated parse trees slightly decrease quality (table 4). Target identification, being the first step in the pipeline, imposes an upper bound on disambiguation scores. We observe this degradation when we compare the Gold ID and the Auto ID blocks of table 5, where automatically identified targets decrease F-score by about 10 points in all settings.[16]

Classification. Along with the statistical classifier results in table 5, we also report performance

[15] Our evaluation script counts tokens that received special labels in the gold standard (see §4) as negative examples of SNACS targets, with the exception of the tokens labeled as unintelligible/nonnative/etc., which are not counted toward or against target ID performance.

[16] A variant of the target ID module, optimized for recall, is used as preprocessing for the agreement study discussed in §5. With this setting, the heuristic achieves an F_1 score of 90.2% (P=85.3%, R=95.6%) on the test set.

		Gold ID			Auto ID								
		Role	*Func.*	*Full*	*Role*			*Func.*			*Full*		
	Syntax	Acc.	Acc.	Acc.	P	R	F	P	R	F	P	R	F
Most frequent	N/A	40.6	53.3	37.9	37.0	37.3	37.1	49.8	50.2	50.0	34.3	34.6	34.4
Neural	gold	71.7	82.5	67.5	62.0	62.5	62.2	73.1	73.8	73.4	58.7	59.2	58.9
Feature-rich	gold	73.5	81.0	70.0	62.0	62.5	62.2	70.7	71.2	71.0	59.3	59.8	59.5
Neural	auto	67.7	78.5	64.4	56.4	56.2	56.3	66.8	66.7	66.7	53.7	53.5	53.6
Feature-rich	auto	67.9	79.4	65.2	58.2	58.1	58.2	66.8	66.7	66.7	55.7	55.6	55.7

Table 5: Overall performance of SNACS disambiguation systems on the test set. Results are reported for the role supersense (*Role*), the function supersense (*Func.*), and their conjunction (*Full*). All figures are percentages. *Left:* Accuracies with gold standard target identification (480 targets). *Right:* Precision, recall, and F_1 with automatic target identification (§6.2 and table 4).

for the most frequent baseline, which selects the most frequent role–function label pair given the (gold) lemma according to the training data. Note that all learned classifiers, across all settings, outperform the most frequent baseline for both role and function prediction. The feature-rich and the neural models perform roughly equivalently despite the significantly different modeling strategies.

Function and scene role performance. Function prediction is consistently more accurate than role prediction, with roughly a 10-point gap across all systems. This mirrors a similar effect in the interannotator agreement scores (see §5), and may be due to the reduced ambiguity of functions compared to roles (as attested by the baseline's higher accuracy for functions than roles), and by the more literal nature of function labels, as opposed to role labels that often require more context to determine.

Impact of automatic syntax. Automatic syntactic analysis decreases scores by 4 to 7 points, most likely due to parsing errors which affect the identification of the preposition's object and governor. In the auto ID/auto syntax condition, the worse target ID performance with automatic parses (noted above) contributes to lower classification scores.

6.5 Errors & Confusions

We can use the structure of the SNACS hierarchy to probe classifier performance. As with the interannotator study, we evaluate the accuracy of predicted labels when they are coarsened post hoc by moving up the hierarchy to a specific depth. Table 6 shows this for the feature-rich classifier for different depths, with depth-1 representing the coarsening of the labels into the 3 root labels. Depth-4 (Exact) represents the full results in table 5. These results show that the classifiers often mistake a label for another that is nearby in the hierarchy. Examining the most frequent confusions of both models, we observe that LOCUS is overpredicted

	Labels	Role	Function
Exact	47	67.9%	79.4%
Depth-3	43	67.9%	79.6%
Depth-2	26	76.2%	86.2%
Depth-1	3	86.0%	93.8%

Table 6: Accuracy of the feature-rich model (gold identification and syntax) on the test set (480 tokens) with different levels of hierarchy coarsening of its output. "Labels" refers to the number of labels in the training set after coarsening.

(which makes sense as it is most frequent overall), and SOCIALROLE–ORGROLE and GESTALT–POSSESSOR are often confused (they are close in the hierarchy: one inherits from the other).

7 Conclusion

This paper introduced a new approach to comprehensive analysis of the semantics of prepositions and possessives in English, backed by a thoroughly documented hierarchy and annotated corpus. We found good interannotator agreement and provided initial supervised disambiguation results. We expect that future work will develop methods to scale the annotation process beyond requiring highly trained experts; bring this scheme to bear on other languages; and investigate the relationship of our scheme to more structured semantic representations, which could lead to more robust models. Our guidelines, corpus, and software are available at `https://github.com/nert-gu/streusle/blob/master/ACL2018.md`.

Acknowledgments

We thank Oliver Richardson, whose codebase we adapted for this project; Na-Rae Han, Archna Bhatia, Tim O'Gorman, Ken Litkowski, Bill Croft, and Martha Palmer for helpful discussions and support; and anonymous reviewers for useful feedback. This research was supported in part by DTRA HDTRA1-16-1-0002/Project #1553695, by DARPA 15-18-CwC-FP-032, and by grant 2016375 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel.

References

Omri Abend, Shai Yerushalmi, and Ari Rappoport. 2017. UCCAApp: Web-application for syntactic and semantic phrase-based annotation. In *Proc. of ACL 2017, System Demonstrations*, pages 109–114, Vancouver, Canada.

Lasha Abzianidze and Johan Bos. 2017. Towards universal semantic tagging. In *Proc. of IWCS*, Montpellier, France.

Adriana Badulescu and Dan Moldovan. 2009. A Semantic Scattering model for the automatic interpretation of English genitives. *Natural Language Engineering*, 15(2):215–239.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, PA.

Austin Blodgett and Nathan Schneider. 2018. Semantic supersenses for English possessives. In *Proc. of LREC*, pages 1529–1534, Miyazaki, Japan.

Claire Bonial, William Corvey, Martha Palmer, Volha V. Petukhova, and Harry Bunt. 2011. A hierarchical unification of LIRICS and VerbNet semantic roles. In *Fifth IEEE International Conference on Semantic Computing*, pages 483–489, Palo Alto, CA, USA.

Melissa Bowerman and Soonja Choi. 2001. Shaping meanings for language: universal and language-specific in the acquisition of spatial semantic categories. In Melissa Bowerman and Stephen Levinson, editors, *Language Acquisition and Conceptual Development*, pages 475–511. Cambridge University Press, Cambridge, UK.

Claudia Brugman. 1981. *The story of 'over': polysemy, semantics and the structure of the lexicon*. MA thesis, University of California, Berkeley, Berkeley, CA. Published New York: Garland, 1981.

Daniel Dahlmeier, Hwee Tou Ng, and Tanja Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *Proc. of EMNLP*, pages 450–458, Suntec, Singapore.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9(Aug):1871–1874.

Charles J. Fillmore and Collin Baker. 2009. A frames approach to semantic analysis. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, pages 791–816. Oxford University Press, Oxford, UK.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*, pages 249–256, Chia Laguna, Sardinia, Italy.

Hila Gonen and Yoav Goldberg. 2016. Semi supervised preposition-sense disambiguation using multilingual data. In *Proc. of COLING*, pages 2718–2729, Osaka, Japan.

Bernd Heine. 2006. *Possession: Cognitive Sources, Forces, and Grammaticalization*. Cambridge University Press, Cambridge, UK.

Annette Herskovits. 1986. *Language and spatial cognition: an interdisciplinary study of the prepositions in English*. Cambridge University Press, Cambridge, UK.

Dirk Hovy, Stephen Tratz, and Eduard Hovy. 2010. What's in a preposition? Dimensions of sense disambiguation for an interesting word class. In *Coling 2010: Posters*, pages 454–462, Beijing, China.

Dirk Hovy, Ashish Vaswani, Stephen Tratz, David Chiang, and Eduard Hovy. 2011. Models and training for unsupervised preposition sense disambiguation. In *Proc. of ACL-HLT*, pages 323–328, Portland, Oregon, USA.

Rodney Huddleston and Geoffrey K. Pullum, editors. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge, UK.

Jena D. Hwang, Archna Bhatia, Na-Rae Han, Tim O'Gorman, Vivek Srikumar, and Nathan Schneider. 2017. Double trouble: the problem of construal in semantic annotation of adpositions. In *Proc. of *SEM*, pages 178–188, Vancouver, Canada.

Naveen Khetarpal, Asifa Majid, and Terry Regier. 2009. Spatial terms reflect near-optimal spatial categories. In *Proc. of the 31st Annual Conference of the Cognitive Science Society*, pages 2396–2401, Amsterdam.

George Lakoff. 1987. *Women, fire, and dangerous things: what categories reveal about the mind*. University of Chicago Press, Chicago.

Seth Lindstromberg. 2010. *English Prepositions Explained*, revised edition. John Benjamins, Amsterdam.

Ken Litkowski. 2014. Pattern Dictionary of English Prepositions. In *Proc. of ACL*, pages 1274–1283, Baltimore, Maryland, USA.

Ken Litkowski and Orin Hargraves. 2005. The Preposition Project. In *Proc. of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 171–179, Colchester, Essex, UK.

Ken Litkowski and Orin Hargraves. 2007. SemEval-2007 Task 06: Word-Sense Disambiguation of Prepositions. In *Proc. of SemEval*, pages 24–29, Prague, Czech Republic.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL: System Demonstrations*, pages 55–60, Baltimore, Maryland, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *HLT-NAACL 2004: Workshop on Computational Lexical Semantics*, pages 60–67, Boston, Massachusetts, USA.

Antje Müller, Claudia Roch, Tobias Stadtfeld, and Tibor Kiss. 2012. The annotation of preposition senses in German. In Britta Stolterfoht and Sam Featherston, editors, *Empirical Approaches to Linguistic Theory: Studies in Meaning and Structure*, pages 63–82. Walter de Gruyter, Berlin.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv:1701.03980*.

Kiki Nikiforidou. 1991. The meanings of the genitive: a case study in semantic structure and semantic change. *Cognitive Linguistics*, 2(2):149–205.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: a multilingual treebank collection. In *Proc. of LREC*, pages 1659–1666, Portorož, Slovenia.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*, pages 3991–3995, Paris, France.

Tom O'Hara and Janyce Wiebe. 2009. Exploiting semantic role resources for preposition disambiguation. *Computational Linguistics*, 35(2):151–184.

Martha Palmer, Claire Bonial, and Jena D. Hwang. 2017. VerbNet: Capturing English verb behavior, meaning and usage. In Susan E. F. Chipman, editor, *The Oxford Handbook of Cognitive Science*, pages 315–336. Oxford University Press.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003. TimeML: Robust specification of event and temporal expressions in text. In *IWCS-5, Fifth International Workshop on Computational Semantics*, Tilburg, Netherlands.

James Pustejovsky, Jessica Moszkowicz, and Marc Verhagen. 2012. A linguistically grounded annotation language for spatial information. *TAL*, 53(2):87–113.

Terry Regier. 1996. *The human semantic potential: spatial language and constrained connectionism*. MIT Press, Cambridge, MA.

Anette Rosenbach. 2002. *Genitive variation in English: conceptual factors in synchronic and diachronic studies*. Mouton de Gruyter, Berlin.

Patrick Saint-Dizier. 2006. PrepNet: a multilingual lexical description of prepositions. In *Proc. of LREC*, volume 6, pages 1021–1026, Genoa, Italy.

Nathan Schneider, Jena D. Hwang, Archna Bhatia, Na-Rae Han, Vivek Srikumar, Tim O'Gorman, Sarah R. Moeller, Omri Abend, Austin Blodgett, and Jakob Prange. 2018. Adposition and Case Supersenses v2: Guidelines for English. *arXiv:1704.02134*.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Meredith Green, Abhijit Suresh, Kathryn Conger, Tim O'Gorman, and Martha Palmer. 2016. A corpus of preposition supersenses. In *Proc. of LAW X – the 10th Linguistic Annotation Workshop*, pages 99–109, Berlin, Germany.

Nathan Schneider, Vivek Srikumar, Jena D. Hwang, and Martha Palmer. 2015. A hierarchy with, of, and for preposition supersenses. In *Proc. of The 9th Linguistic Annotation Workshop*, pages 112–123, Denver, Colorado, USA.

Stephanie Shih, Jason Grafmiller, Richard Futrell, and Joan Bresnan. 2015. Rhythm's role in genitive construction choice in spoken English. In Ralf Vogel and Ruben van de Vijver, editors, *Rhythm in cognition and grammar: a Germanic perspective*, pages 207–234. De Gruyter Mouton, Berlin.

Vivek Srikumar and Dan Roth. 2011. A joint model for extended semantic role labeling. In *Proc. of EMNLP*, pages 129–139, Edinburgh, Scotland, UK.

Vivek Srikumar and Dan Roth. 2013. Modeling semantic relations expressed by prepositions. *Transactions of the Association for Computational Linguistics*, 1:231–242.

Leonard Talmy. 1996. Fictive motion in language and "ception". In Paul Bloom, Mary A. Peterson, Nadel Lynn, and Merrill F. Garrett, editors, *Language and Space*, pages 211–276. MIT Press, Cambridge, MA.

John R. Taylor. 1996. *Possessives in English: An Exploration in Cognitive Grammar*. Clarendon Press, Oxford, UK.

Stephen Tratz and Dirk Hovy. 2009. Disambiguation of preposition sense using linguistically motivated features. In *Proc. of NAACL-HLT Student Research Workshop and Doctoral Consortium*, pages 96–100, Boulder, Colorado.

Stephen Tratz and Eduard Hovy. 2013. Automatic interpretation of the English possessive. In *Proc. of ACL*, pages 372–381, Sofia, Bulgaria.

Andrea Tyler and Vyvyan Evans. 2003. *The Semantics of English Prepositions: Spatial Scenes, Embodied Meaning and Cognition*. Cambridge University Press, Cambridge, UK.

Christoph Wolk, Joan Bresnan, Anette Rosenbach, and Benedikt Szmrecsanyi. 2013. Dative and genitive variability in Late Modern English: Exploring cross-constructional variation and change. *Diachronica*, 30(3):382–419.

Yang Xu and Charles Kemp. 2010. Constructing spatial concepts from universal primitives. In *Proc. of CogSci*, pages 346–351, Portland, Oregon.

Patrick Ye and Timothy Baldwin. 2007. MELB-YB: Preposition sense disambiguation using rich semantic features. In *Proc. of SemEval*, pages 241–244, Prague, Czech Republic.

Joost Zwarts and Yoad Winter. 2000. Vector space semantics: a model-theoretic analysis of locative prepositions. *Journal of Logic, Language and Information*, 9:169–211.

A Corpus with Multi-Level Annotations of Patients, Interventions and Outcomes to Support Language Processing for Medical Literature

Benjamin Nye
Northeastern University
nye.b@husky.neu.edu

Junyi Jessy Li
UT Austin
jessy@austin.utexas.edu

Roma Patel
Rutgers University
romapatel996@gmail.com

Yinfei Yang[*]
No affiliation
yangyin7@gmail.com

Iain J. Marshall
King's College London
iain.marshall@kcl.ac.uk

Ani Nenkova
UPenn
nenkova@seas.upenn.edu

Byron C. Wallace
Northeastern University
b.wallace@northeastern.edu

Abstract

We present a corpus of 5,000 richly annotated abstracts of medical articles describing clinical randomized controlled trials. Annotations include demarcations of text spans that describe the Patient population enrolled, the Interventions studied and to what they were Compared, and the Outcomes measured (the 'PICO' elements). These spans are further annotated at a more granular level, e.g., individual interventions within them are marked and mapped onto a structured medical vocabulary. We acquired annotations from a diverse set of workers with varying levels of expertise and cost. We describe our data collection process and the corpus itself in detail. We then outline a set of challenging NLP tasks that would aid searching of the medical literature and the practice of evidence-based medicine.

1 Introduction

In 2015 alone, about 100 manuscripts describing randomized controlled trials (RCTs) for medical interventions were published *every day*. It is thus practically impossible for physicians to know which is the best medical intervention for a given patient group and condition (Borah et al., 2017; Fraser and Dunstan, 2010; Bastian et al., 2010). This inability to easily search and organize the published literature impedes the aims of *evidence based medicine* (EBM), which aspires to inform patient care using the totality of relevant evidence. Computational methods could expedite biomedical evidence synthesis (Tsafnat et al., 2013; Wallace et al., 2013) and natural language processing (NLP) in particular can play a key role in the task.

Prior work has explored the use of NLP methods to automate biomedical evidence extraction and synthesis (Boudin et al., 2010; Marshall et al., 2017; Ferracane et al., 2016; Verbeke et al., 2012).[1] But the area has attracted less attention than it might from the NLP community, due primarily to a dearth of publicly available, annotated corpora with which to train and evaluate models.

Here we address this gap by introducing *EBM-NLP*, a new corpus to power NLP models in support of EBM. The corpus, accompanying documentation, baseline model implementations for the proposed tasks, and all code are publicly available.[2] EBM-NLP comprises ∼5,000 medical abstracts describing clinical trials, multiply annotated in detail with respect to characteristics of the underlying trial Populations (e.g., *diabetics*), Interventions (*insulin*), Comparators (*placebo*) and Outcomes (*blood glucose levels*). Collectively, these key informational pieces are referred to as PICO elements; they form the basis for well-formed clinical questions (Huang et al., 2006).

We adopt a hybrid crowdsourced labeling strategy using heterogeneous annotators with varying expertise and cost, from laypersons to MDs. Annotators were first tasked with marking text spans that described the respective PICO elements. Identified spans were subsequently anno-

[*] now at Google Inc.

[1] There is even, perhaps inevitably, a systematic review of such approaches (Jonnalagadda et al., 2015).

[2] http://www.ccs.neu.edu/home/bennye/EBM-NLP

tated in greater detail: this entailed finer-grained labeling of PICO elements and mapping these onto a normalized vocabulary, and indicating redundancy in the mentions of PICO elements.

In addition, we outline several NLP tasks that would directly support the practice of EBM and that may be explored using the introduced resource. We present baseline models and associated results for these tasks.

2 Related Work

We briefly review two lines of research relevant to the current effort: work on NLP to facilitate EBM, and research in crowdsourcing for NLP.

2.1 NLP for EBM

Prior work on NLP for EBM has been limited by the availability of only small corpora, which have typically provided on the order of a couple hundred annotated abstracts or articles for very complex information extraction tasks. For example, the ExaCT system (Kiritchenko et al., 2010) applies rules to extract 21 aspects of the reported trial. It was developed and validated on a dataset of 182 marked full-text articles. The ACRES system (Summerscales et al., 2011) produces summaries of several trial characteristic, and was trained on 263 annotated abstracts. Hinting at more challenging tasks that can build upon foundational information extraction, Alamri and Stevenson (2015) developed methods for detecting contradictory claims in biomedical papers. Their corpus of annotated claims contains 259 sentences (Alamri and Stevenson, 2016).

Larger corpora for EBM tasks have been derived using (noisy) automated annotation approaches. This approach has been used to build, e.g., datasets to facilitate work on Information Retrieval (IR) models for biomedical texts (Scells et al., 2017; Chung, 2009; Boudin et al., 2010). Similar approaches have been used to 'distantly supervise' annotation of full-text articles describing clinical trials (Wallace et al., 2016). In contrast to the corpora discussed above, these automatically derived datasets tend to be relatively large, but they include only shallow annotations.

Other work attempts to bypass basic extraction tasks and address more complex biomedical QA and (multi-document) summarization problems to support EBM (Demner-Fushman and Lin, 2007; Mollá and Santiago-Martinez, 2011; Abacha and Zweigenbaum, 2015). Such systems would directly benefit from more accurate extraction of the types codified in the corpus we present here.

2.2 Crowdsourcing

Crowdsourcing, which we here define operationally as the use of distributed lay annotators, has shown encouraging results in NLP (Novotney and Callison-Burch, 2010; Sabou et al., 2012). Such annotations are typically imperfect, but methods that aggregate redundant annotations can mitigate this problem (Dalvi et al., 2013; Hovy et al., 2014; Nguyen et al., 2017).

Medical articles contain relatively technical content, which intuitively may be difficult for persons without domain expertise to annotate. However, recent promising preliminary work has found that crowdsourced approaches can yield surprisingly high-quality annotations in the domain of EBM specifically (Mortensen et al., 2017; Thomas et al., 2017; Wallace et al., 2017).

3 Data Collection

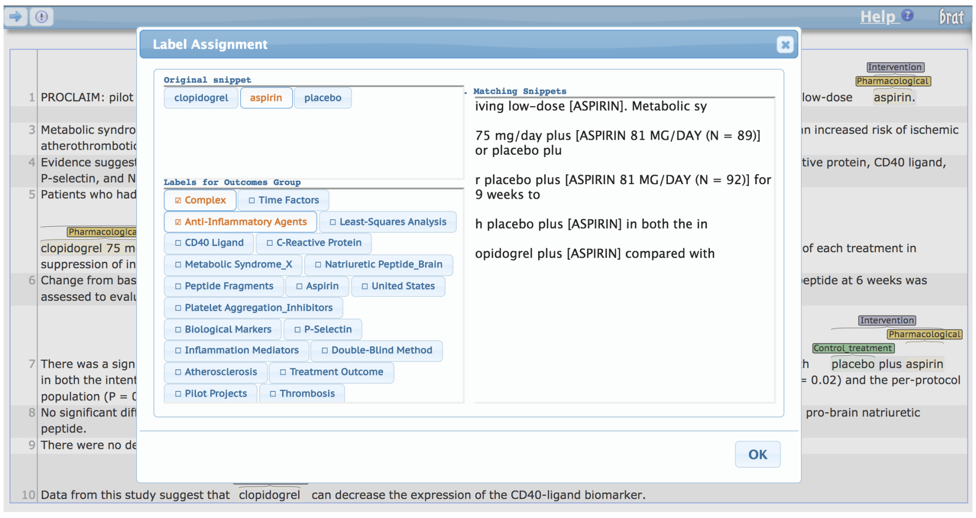
PubMed provides access to the MEDLINE database[3] which indexes titles, abstracts and metadata for articles from selected medical journals dating back to the 1970s. MEDLINE indexes over 24 million abstracts; the majority of these have been manually assigned metadata which we used to retrieved a set of 5,000 articles describing RCTs with an emphasis on cardiovascular diseases, cancer, and autism. These particular topics were selected to cover a range of common conditions.

We decomposed the annotation process into two steps, performed in sequence. First, we acquired labels demarcating spans in the text describing the clinically salient abstract elements mentioned above: the trial Population, the Interventions and Comparators studied, and the Outcomes measured. We collapse Interventions and Comparators into a single category (I). In the second annotation step, we tasked workers with providing more granular (sub-span) annotations on these spans.

For each PIO element, all abstracts were annotated with the following four types of information.

1. **Spans** exhaustive marking of text spans containing information relevant to the respective PIO categories (Stage 1 annotation).

[3]https://www.nlm.nih.gov/bsd/pmresources.html



Matching Snippets

iving low–dose [ASPIRIN]. Metabolic sy

75 mg/day plus [ASPIRIN 81 MG/DAY (N = 89)] or placebo plu

r placebo plus [ASPIRIN 81 MG/DAY (N = 92)] for 9 weeks to

h placebo plus [ASPIRIN] in both the in

opidogrel plus [ASPIRIN] compared with

1 PROCLAIM: pilot low-dose aspirin.

3 Metabolic syndro an increased risk of ischemic atherothrombotic

4 Evidence suggest tive protein, CD40 ligand, P-selectin, and N

5 Patients who had of each treatment in clopidogrel 75 m suppression of in

6 Change from bas eptide at 6 weeks was assessed to evalu

7 There was a sign h placebo plus aspirin in both the intent = 0.02) and the per-protocol population (P = 0

8 No significant dif pro-brain natriuretic peptide.

9 There were no de

10 Data from this study suggest that clopidogrel can decrease the expression of the CD40-ligand biomarker.

Figure 1: Annotation interface for assigning MeSH terms to snippets.

2. **Hierarchical labels** assignment of more specific labels to subsequences comprising the marked relevant spans (Stage 2 annotation).

3. **Repetition** grouping of labeled tokens to indicate repeated occurrences of the same information (Stage 2 annotation).

4. **MeSH terms** assignment of the metadata MeSH terms associated with the abstract to labeled subsequences (Stage 2 annotation).[4]

We collected annotations for each P, I and O element individually to avoid the cognitive load imposed by switching between label sets, and to reduce the amount of instruction required to begin the task. All annotation was performed using a modified version of the Brat Rapid Annotation Tool (BRAT) (Stenetorp et al., 2012). We include all annotation instructions provided to workers for all tasks in the Appendix.

3.1 Non-Expert (Layperson) Workers

For large scale crowdsourcing via recruitment of layperson annotators, we used Amazon Mechanical Turk (AMT). All workers were required to have an overall job approval rate of at least 90%. Each job presented to the workers required the annotation of three randomly selected abstracts from our pool of documents. As we received initial results, we blocked workers who were clearly not following instructions, and we actively recruited the best workers to continue working on our task at a higher pay rate.

We began by collecting the least technical annotations, moving on to more difficult tasks only after restricting our pool of workers to those with a demonstrated aptitude for the jobs. We obtained annotations from ≥ 3 different workers for each of the 5,000 abstracts to enable robust inference of reliable labels from noisy data. After performing filtering passes to remove non-RCT documents or those missing relevant data for the second annotation task, we are left with between 4,000 and 5,000 sets of annotations for each PIO element after the second phase of annotation.

3.2 Expert Workers

To supplement our larger-scale data collection via AMT, we collected annotations for 200 abstracts for each PIO element from workers with advanced medical training. The idea is for these to serve as reference annotations, i.e., a test set with which to evaluate developed NLP systems. We plan to enlarge this test set in the near future, at which point we will update the website accordingly.

For the initial span labeling task, two medical students from the University of Pennsylvania and Drexel University provided the reference labels. In addition, for both stages of annotation and for the detailed subspan annotation in Stage 2, we hired three medical professionals via Upwork,[5] an online platform for hiring skilled freelancers. After reviewing several dozen suggested profiles, we selected three workers that had the following characteristics: Advanced medical training (the majority of hired workers were Medical Doc-

[4]MeSH is a controlled, structured medical vocabulary maintained by the National Library of Medicine.

[5]http://www.upwork.com

tors, the one exception being a fourth-year medical student); Strong technical reading and writing skills; And an interest in medical research. In addition to providing high-quality annotations, individuals hired via Upwork also provided feedback regarding the instructions to help make the task as clear as possible for the AMT workers.

4 The Corpus

We now present corpus details, paying special attention to worker performance and agreement. We discuss and present statistics for acquired annotations on spans, tokens, repetition and MeSH terms in Sections 4.1, 4.2, 4.3, and 4.4, respectively.

4.1 Spans

For each P, I and O element, workers were asked to read the abstract and highlight all spans of text including any pertinent information. Annotations for 5,000 articles were collected from a total of 579 AMT workers across the three annotation types, and expert annotations were collected for 200 articles from two medical students.

We first evaluate the quality of the annotations by calculating token-wise label agreement between the expert annotators; this is reported in Table 2. Due to the difficulty and technicality of the material, agreement between even well-trained domain experts is imperfect. The effect is magnified by the unreliability of AMT workers, motivating our strategy of collecting several noisy annotations and aggregating over them to produce a single cleaner annotation. We tested three different aggregation strategies: a simple majority vote, the Dawid-Skene model (Dawid and Skene, 1979) which estimates worker reliability, and HMM-Crowd, a recent extension to Dawid-Skene that includes a HMM component, thus explicitly leveraging the sequential structure of contiguous spans of words (Nguyen et al., 2017).

For each aggregation strategy, we compute the token-wise precision and recall of the output labels against the unioned expert labels. As shown in Table 3, the HMMCrowd model afforded modest improvement in F-1 scores over the standard Dawid-Skene model, and was thus used to generate the inputs for the second annotation phase.

The limited overlap in the document subsets annotated by any given pair of workers, and wide variation in the number of annotations per worker make interpretation of standard agreement statistics tricky. We quantify the centrality of the AMT span annotations by calculating token-wise precision and recall for each annotation against the aggregated version of the labels (Table 4).

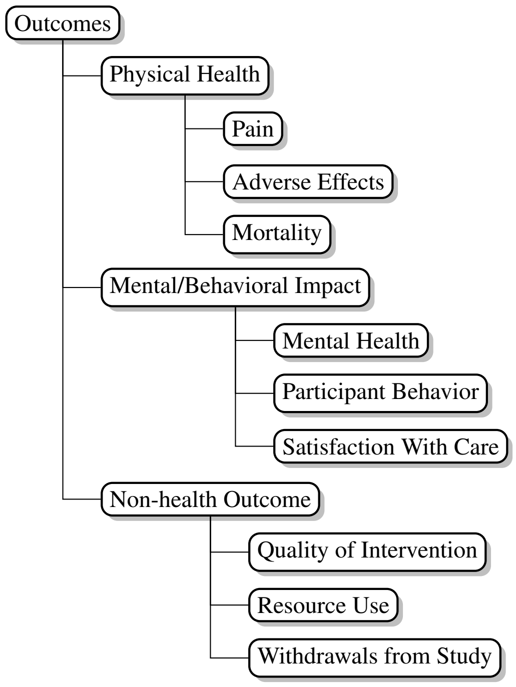


Figure 2: Outcome task label hierarchy

When comparing the average precision and recall for individual crowdworkers against the aggregated labels in Table 4, scores are poor showing very low agreement between the workers. Despite this, the aggregated labels compare favorably against the expert labels. This further supports the intuition that it is feasible to collect multiple low-quality annotations for a document and synthesize them to extract the signal from the noise.

On the dataset website, we provide a variant of the corpus that includes all individual worker span annotations (e.g., for researchers interested in crowd annotation aggregated methods), and also a version with pre-aggregated annotations for convenience.

4.2 Hierarchical Labels

For each P, I, and O category we developed a hierarchy of labels intended to capture important sub categories within these. Our labels are aligned to (and thus compatible with) the concepts codified by the Medical Subject Headings (MeSH) vocabulary of medical terms maintained by the National Library of Medicine (NLM).[6] In consulta-

[6] https://www.nlm.nih.gov/mesh/

P *Fourteen children (12 infantile autism full syndrome present, 2 atypical pervasive developmental disorder) between 5 and 13 years of age*

Text	Label	MeSH terms
– Fourteen	Sample Size (full)	
– children	Age (young)	
– 12	Sample Size (partial)	
– autism	Condition (disease)	Autistic Disorder, Child Development Disorders Pervasive
– 2	Sample Size (partial)	
– 5 and 13	Age (young)	

I *20 mg Org 2766 (synthetic analog of ACTH 4-9)/day during 4 weeks, or placebo in a randomly assigned sequence.*

Text	Label	MeSH terms
– 20 mg Org 2766	Pharmacological	Adrenocorticotropic Hormone, Double-Blind Method, Child Development Disorders Pervasive
– placebo	Control	Double-Blind Method

O *Drug effects and Aberrant Behavior Checklist ratings*

Text	Label	MeSH terms
– Drug effects	Quality of Intervention	
– Aberrant Behavior Checklist ratings	Mental (Behavior)	Attention, Stereotyped Behavior

Table 1: Partial example annotation for Participants, Interventions, and Outcomes. The full annotation includes multiple top-level spans for each PIO element as well as labels for repetition.

	Agreement
Participants	0.71
Interventions	0.69
Outcomes	0.62

Table 2: Cohen's κ between medical students for the 200 reference documents.

Participants	Precision	Recall	F-1
Majority Vote	0.903	0.507	0.604
Dawid Skene	0.840	0.641	0.686
HMMCrowd	0.719	0.761	0.698
Interventions	Precision	Recall	F-1
Majority Vote	0.843	0.432	0.519
Dawid Skene	0.755	0.623	0.650
HMMCrowd	0.644	0.800	0.683
Outcomes	Precision	Recall	F-1
Majority Vote	0.711	0.577	0.623
Dawid Skene	0.652	0.648	0.629
HMMCrowd	0.498	0.807	0.593

Table 3: Precision, recall and F-1 for aggregated AMT spans evaluated against the union of expert span labels, for all three P, I, and O elements.

	Precision	Recall	F-1
Participants	0.34	0.29	0.30
Interventions	0.20	0.16	0.18
Outcomes	0.11	0.10	0.10

Table 4: Token-wise statistics for individual AMT annotations evaluated against the aggregated versions.

	Span frequency	
	AMT	Experts
Participants	34.5	21.4
Interventions	26.5	14.3
Outcomes	33.0	26.9

Table 5: Average per-document frequency of different token labels.

tion with domain experts, we selected subsets of MeSH terms for each PIO category that captured relatively precise information without being overwhelming. For illustration, we show the outcomes label hierarchy we used in Figure 2. We reproduce the label hierarchies used for all PIO categories in the Appendix.

At this stage, workers were presented with abstracts in which relevant spans were highlighted, based on the annotations collected in the first annotation phase (and aggregated via the HMM-Crowd model). This two-step approach served dual purposes: (i) increasing the rate at which workers could complete tasks, and (ii) improving recall by directing workers to all areas in abstracts where they might find the structured information of interest. Our choice of a high recall aggregation strategy for the starting spans ensured that the large majority of relevant sections of the article were available as inputs to this task.

The three trained medical personnel hired via Upwork each annotated 200 documents and reported that spans sufficiently captured the target information. These domain experts received feedback and additional training after labeling an initial round of documents, and all annotations were reviewed for compliance. The average inter-annotator agreement is reported in Table 6.

	Agreement
Participants	0.50
Interventions	0.59
Outcomes	0.51

Table 6: Average pair-wise Cohen's κ between three medical experts for the 200 reference documents.

With respect to crowdsourcing on AMT, the task for Participants was published first, allowing us to target higher quality workers for the more technical Interventions and Outcomes annotations. We retained labels from 118 workers for Participants, the top 67 of whom were invited to continue on to the following tasks. Of these, 37 continued to contribute to the project. Several workers provided $\geq$ 1,000 annotations and continued to work on the task over a period of several months.

To produce final per-token labels, we again turned to aggregation. The subspans annotated in this second pass were by construction shorter than the starting spans, and (perhaps as a result) informal experiments revealed little benefit from HMMCrowd's sequential modeling aspect. The introduction of many label types significantly increased the complexity of the task, resulting in both lower expert inter-annotator agreement (Table 6 and decreased performance when comparing the crowdsourced labels against those of the experts (Table 7.

Participants	Precision	Recall	F-1
Majority Vote	0.46	0.58	0.51
Dawid Skene	0.66	0.60	0.63
Interventions	Precision	Recall	F-1
Majority Vote	0.56	0.49	0.52
Dawid Skene	0.56	0.52	0.54
Outcomes	Precision	Recall	F-1
Majority Vote	0.73	0.69	0.71
Dawid Skene	0.73	0.80	0.76

Table 7: Precision, recall, and F-1 for AMT labels against expert labels using different aggregation strategies.

Most observed token-level disagreements (and errors, with respect to reference annotations) involve differences in the span lengths demarcated by individuals. For example, many abstracts contain an information-dense description of the patient population, focusing on their medical condition but also including information about their sex and/or age. Workers would also sometimes fail

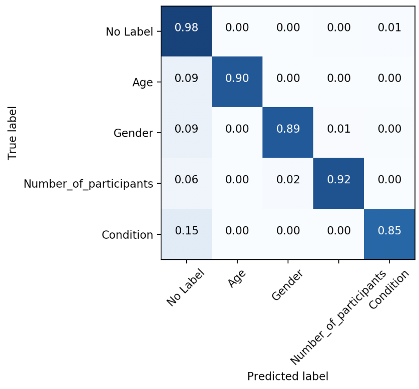


Figure 3: Confusion matrix for token-level labels provided by experts.

to capture repeated mentions of the same information, producing Type 2 errors more frequently than Type 1. This tendency can be seen in the overall token-level confusion matrix for AMT workers on the Participants task, shown in Figure 3.

In a similar though more benign category of error, workers differed in the amount of context they included surrounding each subspan. Although the instructions asked workers to highlight minimal subspans, there was variance in what workers considered relevant.

	Precision	Recall	F-1
Participants	0.39	0.71	0.50
Interventions	0.59	0.60	0.60
Outcomes	0.70	0.68	0.69

Table 8: Statistics for individual AMT annotations evaluated against the aggregated versions, macro-averaged over different labels.

For the same reasons mentioned above (little pairwise overlap in annotations, high variance with respect to annotations per worker), quantifying agreement between AMT workers is again difficult using traditional measures. We thus again take as a measure of agreement the precision, recall, and F-1 of the individual annotations against the aggregated labels and present the results in Table 8.

4.3 Repetition

Medical abstracts often mention the same information in multiple places. In particular, interventions and outcomes are typically described at the beginning of an abstract when introducing the purpose of the underlying study, and then again when discussing methods and results. It is important to

	Span frequency	
Participants	AMT	Experts
TOTAL	3.45	6.25
Age	0.49	0.66
Condition	1.77	3.69
Gender	0.36	0.34
Sample Size	0.83	1.55
Interventions	AMT	Experts
TOTAL	6.11	9.31
Behavioral	0.22	0.37
Control	0.83	0.94
Educational	0.04	0.07
No Label	0.00	0.00
Other	0.23	1.12
Pharmacological	3.37	5.19
Physical	0.87	0.88
Psychological	0.29	0.19
Surgical	0.24	0.62
Outcomes	AMT	Experts
TOTAL	6.36	10.00
Adverse effects	0.45	0.66
Mental	0.69	0.79
Mortality	0.23	0.33
Other	1.77	3.70
Pain	0.18	0.27
Physical	3.03	4.25

Table 9: Average per-document frequency of different label types.

	Precision	Recall	F-1
Participants	0.40	0.77	0.53
Interventions	0.63	0.90	0.74
Outcomes	0.47	0.73	0.57

Table 10: Comparison against the majority vote for span-level repetition labels.

be able to differentiate between novel and reiterated information, especially in cases such as complex interventions, distinct measured outcomes, or multi-armed trials. Merely identifying all occurrences of, for example, a pharmacological intervention leaves ambiguity as to how many distinct interventions were applied.

Workers identified repeated information as follows. After completing detailed labeling of abstract spans, they were asked to group together subspans that were instances of the same information (for example, redundant mentions of a particular drug evaluated as one of the interventions in the trial). This process produces labels for repetition between short spans of tokens. Due to the differences in the lengths of annotated subspans discussed in the preceding section, the labels are not naturally comparable between workers without directly modeling the entities contained in each subspan. The labels assigned by workers produce repetition labels between sets of tokens but a more sophisticated notion of co-reference is required to identify which tokens correctly represent the entity contained in the span, and which tokens are superfluous noise.

As a proxy for formally enumerating these entities, we observe that a large majority of starting spans only contain a single target relevant to the subspan labeling task, and so identifying repetition between the starting spans is sufficient. For example, consider the starting intervention span *”underwent conventional total knee arthroplasty”*; there is only one intervention in the span but some annotators assigned the SURGICAL label to all five tokens while others opted for only *”total knee arthroplasty.”* By analyzing repetition at the level of the starting spans, we can compute agreement without concern for the confounds of slight misalignments or differences in length of the subspans.

Overall agreement between AMT workers for span-level repetition, measured by computing precision and recall against the majority vote for each pair of spans, is reported in Table 10.

4.4 MeSH Terms

The National Library of Medicine maintains an extensive hierarchical ontology of medical concepts called Medical Subject Headings (MeSH terms); this is part of the overarching Metathesaurus of the Unified Medical Language System (UMLS). Personnel at the NLM manually assign citations (article titles, abstracts and meta-data) indexed in MEDLINE relevant MeSH terms. These terms have been used extensively to evaluate the content of articles, and are frequently used to facilitate document retrieval (Lu et al., 2009; Lowe and Barnett, 1994).

In the case of randomized controlled trials, MeSH terms provide structured information regarding key aspects of the underlying studies, ranging from participant demographics to methodologies to co-morbidities. A drawback to these annotations, however, is that they are applied at the document (rather than snippet or token) level. To capture where MeSH terms are instantiated within a given abstract text, we provided a list of all terms associated with said article and instructed workers to select the subset of these that applied to each set of token labels that they annotated.

MeSH terms are domain specific and many re-

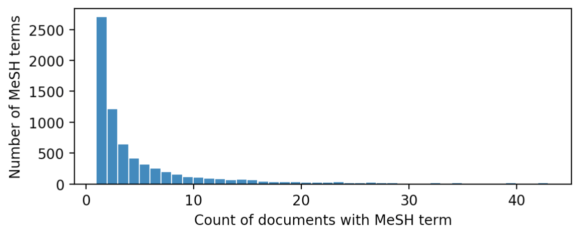


Figure 4: Histogram of the number of documents containing each MeSH term.

Inst. Freq	10%	25%	50%
Participants	65	24	7
Interventions	106	68	32
Outcomes	118	108	75

Table 11: The number of common MeSH terms (out of 135) that were assigned to a span of text in at least 10%, 25%, and 50% of the possible documents.

quire a medical background to understand, thus rendering this facet of the annotation process particularly difficult for untrained (lay) workers. Perhaps surprisingly, several AMT workers voluntarily mentioned relevant background training; our pool of workers included (self-identified) nurses and other trained medical professionals. A few workers with such training stated this background as a reason for their interest in our tasks.

The technical specificity of the more obscure MeSH terms is also exacerbated by their sparsity. Of the 6,963 unique MeSH terms occurring in our set of abstracts, 87% of them are only found in 10 documents or fewer and only 2.0% occur in at least 1% of the total documents. The full distribution of document frequency for MeSH terms is show in Figure 4.

To evaluate how often salient MeSH terms were instantiated in the text by annotators we consider only the 135 MeSH terms that occur in at least 1% of abstracts (we list these in the supplementary material). For each term, we calculate its ”instantiation frequency” as the percentage of abstracts containing the term in which at least one annotator assigned it to a span of text. The total numbers of MeSH terms with an instantiation rate above different thresholds for the respective PIO elements are shown in Table 11.

5 Tasks & Baselines

We outline a few NLP tasks that are central to the aim of processing medical literature generally and to aiding practitioners of EBM specifically. First, we consider the task of identifying spans in abstracts that describe the respective PICO elements (Section 5.1). This would, e.g., improve medical literature search and retrieval systems. Next, we outline the problem of extracting structured information from abstracts (Section 5.2). Such models would further aid search, and might eventually facilitate automated knowledge-base construction for the clinical trials literature. Furthermore, automatic extraction of structured data would enable automation of the manual evidence synthesis process (Marshall et al., 2017).

Finally, we consider the challenging task of identifying redundant mentions of the same PICO element (Section 5.3). This happens, e.g., when an intervention is mentioned by the authors repeatedly in an abstract, potentially with different terms. Achieving such disambiguation is important for systems aiming to induce structured representations of trials and their results, as this would require recognizing and normalizing the unique interventions and outcomes studied in a trial.

For each of these tasks we present baseline models and corresponding results. Note that we have pre-defined train, development and test sets across PIO elements for this corpus, comprising 4300, 500 and 200 abstracts, respectively. The latter set is annotated by domain experts (i.e., persons with medical training). These splits will, of course, be distributed along with the dataset to facilitate model comparisons.

5.1 Identifying P, I and O Spans

We consider two baseline models: a linear Conditional Random Field (CRF) (Lafferty et al., 2001) and a Long Short-Term Memory (LSTM) neural tagging model, an LSTM-CRF (Lample et al., 2016; Ma and Hovy, 2016). In both models, we treat tokens as being either Inside (I) or Outside (O) of spans.

For the CRF, features include: indicators for the current, previous and next words; part of speech tags inferred using the Stanford CoreNLP tagger (Manning et al., 2014); and character information, e.g., whether a token contains digits, uppercase letters, symbols and so on.

For the neural model, the model induces features via a bi-directional LSTM that consumes distributed vector representations of input tokens sequentially. The bi-LSTM yields a hidden vector at

CRF	Precision	Recall	F-1
Participants	0.55	0.51	0.53
Interventions	0.65	0.21	0.32
Outcomes	0.83	0.17	0.29
LSTM-CRF	Precision	Recall	F-1
Participants	0.78	0.66	0.71
Interventions	0.61	0.70	0.65
Outcomes	0.69	0.58	0.63

Table 12: Baseline models (on the test set) for the PIO span tagging task.

LogReg	Precision	Recall	F-1
Participants	0.41	0.20	0.26
Interventions	0.79	0.44	0.57
Outcomes	0.24	0.21	0.22
CRF	Precision	Recall	F-1
Participants	0.41	0.25	0.31
Interventions	0.59	0.15	0.21
Outcomes	0.60	0.51	0.55

Table 13: Baseline models for the token-level, detailed labeling task.

each token index, which is then passed to a CRF layer for prediction. We also exploit character-level information by passing a bi-LSTM over the characters comprising each word (Lample et al., 2016); these are appended to the word embedding representations before being passed through the bi-LSTM.

5.2 Extracting Structured Information

Beyond identifying the spans of text containing information pertinent to each of the PIO elements, we consider the task of predicting which of the detailed labels occur in each span, and where they are located. Specifically, we begin with the starting spans and predict a single label from the corresponding PIO hierarchy for each token, evaluating against the test set of 200 documents. Initial experiments with neural models proved unfruitful but bear further investigation.

For the CRF model we include the same features as in the previous model, supplemented with additional features encoding if the adjacent tokens include any parenthesis or mathematical operators (specifically: $\%, +, -$). For the logistic regression model, we use a one-vs-rest approach. Features include token n-grams, part of speech indicators, and the same character-level information as in the CRF model.

5.3 Detecting Repetition

To formalize repetition, we consider every pair of starting PIO spans from each abstract, and assign

	Precision	Recall	F-1
Participants	0.39	0.52	0.44
Interventions	0.41	0.50	0.45
Outcomes	0.10	0.16	0.12

Table 14: Baseline model for predicting whether pairs of spans contain redundant information.

binary labels that indicate whether they share at least one instance of the same information. Although this makes prediction easier for long and information-dense spans, a large enough majority of the spans contain only a single instance of relevant information that the task serves as a reasonable baseline. Again, the model is trained on the aggregated labels collected from AMT and evaluated against the high-quality test set.

We train a logistic regression model that operates over standard features, including bag-of-words representations and sentence-level features such as length and position in the document. All baseline model implementations are available on the corpus website.

6 Conclusions

We have presented EBM-NLP: a new, publicly available corpus comprising 5,000 richly annotated abstracts of articles describing clinical randomized controlled trials. This dataset fills a need for larger scale corpora to facilitate research on NLP methods for processing the biomedical literature, which have the potential to aid the conduct of EBM. The need for such technologies will only become more pressing as the literature continues its torrential growth.

The EBM-NLP corpus, accompanying documentation, code for working with the data, and baseline models presented in this work are all publicly available at: `http://www.ccs.neu.edu/home/bennye/EBM-NLP`.

7 Acknowledgements

This work was supported in part by the National Cancer Institute (NCI) of the National Institutes of Health (NIH), award number UH2CA203711.

References

Asma Ben Abacha and Pierre Zweigenbaum. 2015. Means: A medical question-answering system combining nlp techniques and semantic web technologies. *Information processing & management*, 51(5):570–594.

Abdulaziz Alamri and Mark Stevenson. 2015. Automatic detection of answers to research questions from medline. *Proceedings of the workshop on Biomedical Natural Language Processing (BioNLP)*, pages 141–146.

Abdulaziz Alamri and Mark Stevenson. 2016. A corpus of potentially contradictory research claims from cardiovascular research abstracts. *Journal of biomedical semantics*, 7(1):36.

Hilda Bastian, Paul Glasziou, and Iain Chalmers. 2010. Seventy-five trials and eleven systematic reviews a day: how will we ever keep up? *PLoS medicine*, 7(9):e1000326.

Rohit Borah, Andrew W Brown, Patrice L Capers, and Kathryn A Kaiser. 2017. Analysis of the time and workers needed to conduct systematic reviews of medical interventions using data from the prospero registry. *BMJ open*, 7(2):e012545.

Florian Boudin, Jian-Yun Nie, and Martin Dawes. 2010. Positional language models for clinical information retrieval. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 108–115. Association for Computational Linguistics.

Grace Y Chung. 2009. Sentence retrieval for abstracts of randomized controlled trials. *BMC medical informatics and decision making*, 9(1):10.

Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. 2013. Aggregating crowdsourced binary ratings. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 285–294. ACM.

Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28.

Dina Demner-Fushman and Jimmy Lin. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics*, 33(1):63–103.

Elisa Ferracane, Iain Marshall, Byron C Wallace, and Katrin Erk. 2016. Leveraging coreference to identify arms in medical abstracts: An experimental study. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 86–95.

Alan G Fraser and Frank D Dunstan. 2010. On the impossibility of being expert. *British Medical Journal*, 341:c6815.

Dirk Hovy, Barbara Plank, and Anders Søgaard. 2014. Experiments with crowdsourced re-annotation of a pos tagging data set. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (volume 2: Short Papers)*, volume 2, pages 377–382.

Xiaoli Huang, Jimmy Lin, and Dina Demner-Fushman. 2006. Evaluation of PICO as a knowledge representation for clinical questions. In *AMIA annual symposium proceedings*, volume 2006, page 359. American Medical Informatics Association.

Siddhartha R Jonnalagadda, Pawan Goyal, and Mark D Huffman. 2015. Automating data extraction in systematic reviews: a systematic review. *Systematic reviews*, 4(1):78.

Svetlana Kiritchenko, Berry de Bruijn, Simona Carini, Joel Martin, and Ida Sim. 2010. Exact: automatic extraction of clinical trial characteristics from journal publications. *BMC medical informatics and decision making*, 10(1):56.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Henry J Lowe and G Octo Barnett. 1994. Understanding and using the medical subject headings (mesh) vocabulary to perform literature searches. *Jama*, 271(14):1103–1108.

Zhiyong Lu, Won Kim, and W John Wilbur. 2009. Evaluation of query expansion using mesh in pubmed. *Information retrieval*, 12(1):69–80.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60.

Iain Marshall, Joël Kuiper, Edward Banner, and Byron C. Wallace. 2017. Automating Biomedical Evidence Synthesis: RobotReviewer. In *Proceedings of the Association for Computational Linguistics (ACL), System Demonstrations*, pages 7–12. Association for Computational Linguistics (ACL).

Diego Mollá and Maria Elena Santiago-Martinez. 2011. Development of a corpus for evidence based medicine summarisation.

Michael L Mortensen, Gaelen P Adam, Thomas A Trikalinos, Tim Kraska, and Byron C Wallace. 2017. An exploration of crowdsourcing citation screening for systematic reviews. *Research synthesis methods*, 8(3):366–386.

An T Nguyen, Byron C Wallace, Junyi Jessy Li, Ani Nenkova, and Matthew Lease. 2017. Aggregating and predicting sequence labels from crowd annotations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2017, page 299. NIH Public Access.

Scott Novotney and Chris Callison-Burch. 2010. Cheap, fast and good enough: Automatic speech recognition with non-expert transcription. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

Marta Sabou, Kalina Bontcheva, and Arno Scharl. 2012. Crowdsourcing research opportunities: lessons from natural language processing. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, page 17. ACM.

Harrisen Scells, Guido Zuccon, Bevan Koopman, Anthony Deacon, Leif Azzopardi, and Shlomo Geva. 2017. A test collection for evaluating retrieval of studies for inclusion in systematic reviews. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1237–1240. ACM.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Rodney L Summerscales, Shlomo Argamon, Shangda Bai, Jordan Hupert, and Alan Schwartz. 2011. Automatic summarization of results from clinical trials. In *Bioinformatics and Biomedicine (BIBM), 2011 IEEE International Conference on*, pages 372–377. IEEE.

James Thomas, Anna Noel-Storr, Iain Marshall, Byron Wallace, Steven McDonald, Chris Mavergames, Paul Glasziou, Ian Shemilt, Anneliese Synnot, Tari Turner, et al. 2017. Living systematic reviews: 2. combining human and machine effort. *Journal of clinical epidemiology*, 91:31–37.

Guy Tsafnat, Adam Dunn, Paul Glasziou, Enrico Coiera, et al. 2013. The automation of systematic reviews. *BMJ*, 346(f139):1–2.

Mathias Verbeke, Vincent Van Asch, Roser Morante, Paolo Frasconi, Walter Daelemans, and Luc De Raedt. 2012. A statistical relational learning approach to identifying evidence based medicine categories. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 579–589. Association for Computational Linguistics.

Byron C Wallace, Issa J Dahabreh, Christopher H Schmid, Joseph Lau, and Thomas A Trikalinos. 2013. Modernizing the systematic review process to inform comparative effectiveness: tools and methods. *Journal of comparative effectiveness research*, 2(3):273–282.

Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Brian Zhu, and Iain J Marshall. 2016. Extracting PICO sentences from clinical trial reports using supervised distant supervision. *Journal of Machine Learning Research*, 17(132):1–25.

Byron C Wallace, Anna Noel-Storr, Iain J Marshall, Aaron M Cohen, Neil R Smalheiser, and James Thomas. 2017. Identifying reports of randomized controlled trials (rcts) via a hybrid machine learning and crowdsourcing approach. *Journal of the American Medical Informatics Association*, 24(6):1165–1168.

Efficient Online Scalar Annotation with Bounded Support

Keisuke Sakaguchi and **Benjamin Van Durme**
Johns Hopkins University
{keisuke,vandurme}@cs.jhu.edu

Abstract

We describe a novel method for efficiently eliciting scalar annotations for dataset construction and system quality estimation by human judgments. We contrast direct assessment (annotators assign scores to items directly), online pairwise ranking aggregation (scores derive from annotator comparison of items), and a hybrid approach (EASL: **E**fficient **A**nnotation of **S**calar **L**abels) proposed here. Our proposal leads to increased correlation with ground truth, at far greater annotator efficiency, suggesting this strategy as an improved mechanism for dataset creation and manual system evaluation.

1 Introduction

We are concerned here with the construction of datasets and evaluation of systems within natural language processing (NLP). Specifically, humans providing responses that are used to derive graded values on natural language contexts, or in the ordering of systems corresponding to their perceived performance on some task.

Many NLP datasets involve eliciting from annotators some graded response. The most popular annotation scheme is the n-ary ordinal approach as illustrated in Figure 1(a). For example, text may be labeled for *sentiment* as *positive*, *neutral* or *negative* (Wiebe et al., 1999; Pang et al., 2002; Turney, 2002, inter alia); or under *political spectrum analysis* as *liberal*, *neutral*, or *conservative* (O'Connor et al., 2010; Bamman and Smith, 2015). A response may correspond to a likelihood judgment, e.g., how likely a predicate is factive (Lee et al., 2015), or that some natural language inference may hold (Zhang et al., 2017). Responses may correspond to a notion of semantic similarity, e.g., whether one word can be substituted for another in context (Pavlick et al., 2015), or whether an entire sentence is more or less similar than another (Marelli et al., 2014), and so on.

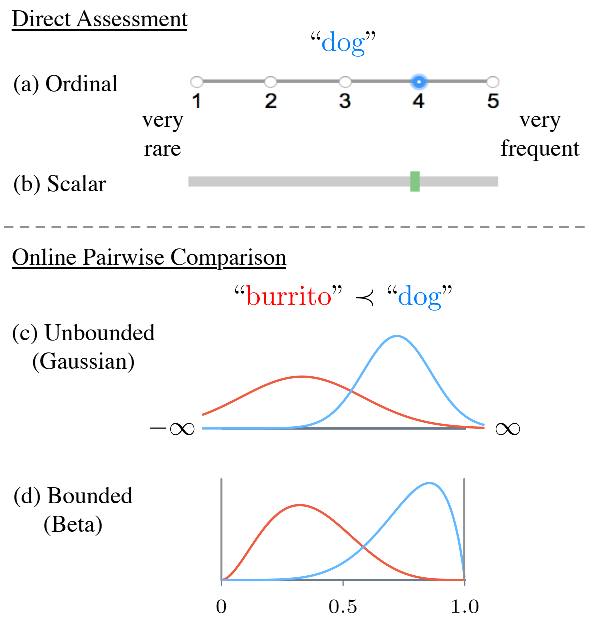


Figure 1: Elicitation strategies for graded response include direct assessment via ordinal or scalar judgments, and pairwise comparisons aggregated via an assumption of latent distributions such as Gaussians, or novel here: Beta distributions, providing bounded support. The example concerns subjective assessments of the lexical frequency of *dog*. In pairwise comparison, we assess it by comparison such as "burrito" is less frequent (≺) than "dog".

Less common in NLP are system comparisons based on direct human ratings, but an exception includes the annual shared task evaluations of the Conference on Machine Translation (WMT). There, MT practitioners submit system outputs based on a shared set of source sentences, which are then judged relative to other system outputs. Various aggregation strategies have been employed over the years to take these relative comparisons and derive competitive rankings between shared task entrants (Callison-Burch et al., 2012;

Bojar et al., 2013, 2014, 2015, 2016, 2017).

Inspired by prior work in MT system evaluation, we propose a procedure for eliciting graded responses that we demonstrate to be more efficient than prior work. While remaining applicable to system evaluation, our experimental results suggest our approach as a more general framework for a variety of future data creation tasks, allowing for higher quality data in less time and cost.

We consider three different approaches for scalar annotation: direct assessment (DA), online pairwise ranking aggregation (RA), and a hybrid method which we call EASL (**E**fficient **A**nnotation of **S**calar **L**abels).[1] DA scalar annotation, shown in Figure 1(b), directly annotates absolute judgments on some scale (e.g., 0 to 100), independently per item (§2). As an RA approach (§3), we start with conventional unbounded models, where each instance is parameterized as a Gaussian distribution, as shown in Figure 1(c). Since boundedness is essential for the scalar annotation we aim to model, we propose a bounded variant which parameterizes each instance by a beta distribution as illustrated in Figure 1(d). Finally, we propose EASL (§4) that combines benefits of DA and RA.

We illustrate the improvements enabled by our proposal on three example tasks (§5): lexical frequency inference, political spectrum inference and machine translation system ranking.[2] For example, we find that in the commonly employed condition of 3-way redundant annotation, our approach on multiple tasks gives similar quality with just 2-way redundancy: this translates to a potential 50% increase in dataset size for the same cost.

2 Direct Assessment

Direct assessment or direct annotation (DA) is a straightforward method for collecting graded response from annotators. The most popular scheme is n-ary ordinal labeling, as illustrated in Figure 1(a), where annotators are shown one instance (i.e., sample point) and asked to label one of the n-ary ordered classes.

According to the level of measurement in psychometrics (Stevens, 1946, inter alia), which classifies the numerals based on certain properties (e.g., identity, order, quantity), ordinal data do not allow for degree of difference. Namely, there is no guarantee that the distance between each label is equal, and instances in the same class are not discriminated. For example, in a typical five-level Likert scale (Likert, 1932) of likelihood – very unlikely, unlikely, unsure, likely, very likely – we cannot conclude that *very likely* instances are exactly twice as likely those marked *likely*, nor can we assume two instances with the same label have exactly the same likelihood.

The issue of distance between ordinals is perhaps obviated by using *scalar* annotations (i.e., *ratio scale* in Stevens's terminology), which directly correspond to continuous quantities (Figure 1(b)). In scalar DA,[3] each instance in the collection ($S_i \in S_1^N$) is annotated with values (e.g., on the range 0 to 100) often by several annotators. The notion of quantitative difference is enabled by the property of *absolute* zero: the scale is *bounded*. For example, distance, length, mass, size etc. are represented by this scale. In the annual shared task evaluation of the WMT, DA has been used for scoring adequacy and fluency of machine learning system outputs with human evaluation (Graham et al., 2013, 2014; Bojar et al., 2016, 2017), and has separately been used in creating datasets such as for factuality (Lee et al., 2015).

Why *perhaps* obviated? Because of two concerns: (1) annotators may not have a pre-existing, well-calibrated scale for performing DA on a particular collection according to a particular task;[4] and (2) it is known that people may be biased in their scalar estimates (Tversky and Kahneman, 1974). Regarding (1), this motivates us to consider RA on the intuition that annotators may give more calibrated responses when performed in the context of other elements. Regarding (2), our goal is not to correct for human bias, but simply to more efficiently converge to the same consensus judgments already being pursued by the community in their annotation protocols, biased or otherwise.[5]

3 Online Pairwise Ranking Aggregation

3.1 Unbounded Model

Pairwise ranking aggregation (Thurstone, 1927) is a method to obtain a total ranking on instances,

[1] Pronounced as "easel".

[2] We release the code at `http://decomp.net/`.

[3] In the rest of the paper, we take DA to mean *scalar* annotation rather than ordinals.

[4] E.g., try to imagine your level of calibration to a hypothetical task described as "On a scale of 1 to 100, label this tweet according to a conservative / liberal political spectrum."

[5] There has been a line of work on relative weighting of *annotators*, based on their agreement with others (Whitehill et al., 2009; Welinder et al., 2010; Hovy et al., 2013). In this paper, however, we do not perform such annotator weighting.

assuming that scalar value for each sample point follows a Gaussian distribution, $\mathcal{N}(\mu_i, \sigma^2)$. The parameters $\{\mu_i\}$ are interpreted as mean scalar annotation.[6]

Given the parameters, the probability that S_i is preferred ($\succ$) over S_j is defined as

$$p(S_i \succ S_j) = \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{2}\sigma}\right), \tag{1}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. The objective of pairwise ranking aggregation (including all the following models) is formulated as a maximum log-likelihood estimation:

$$\max_{\{S_1^N\}} \sum_{S_i, S_j \in \{S_1^N\}} \log p(S_i \succ S_j). \tag{2}$$

TrueSkill$^{\text{TM}}$ (Herbrich et al., 2006) extends the Thurstone model by applying a Bayesian online and active learning framework, allowing for ties. TrueSkill has been used in the Xbox Live online gaming community,[7] and has been applied for various NLP tasks, such as question difficulty estimation (Liu et al., 2013), ranking speech quality (Baumann, 2017), and ranking machine translation and grammatical error correction systems with human evaluation (Bojar et al., 2014, 2015; Sakaguchi et al., 2014, 2016)

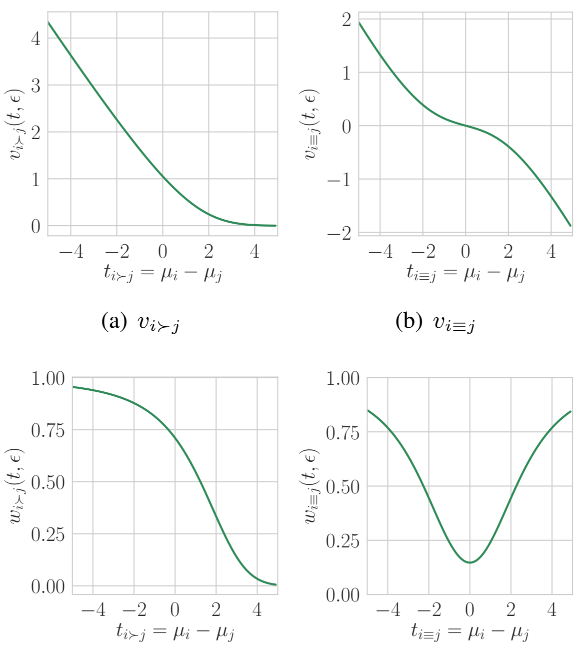
In the same way as the Thurstone model, TrueSkill assumes that scalar values for each instance S_i (i.e., skill level for each player in the context of TrueSkill) follow a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$, where σ_i is also parameterized as the *uncertainty* of the scalar value for each instance. Importantly, TrueSkill uses a Bayesian online learning scheme, and the parameters are *iteratively* updated after each observation of pairwise comparison (i.e., game result: win ($\succ$), tie ($\equiv$), or loss ($\prec$)) in proportion to how surprising the outcome is. Let $t_{i\succ j} = \mu_i - \mu_j$, the difference in scalar responses (skill levels) when we observe i wins j, and $\epsilon \geqslant 0$ be a parameter to specify the tie rate. The update functions are formulated as follows:

$$\mu_i = \mu_i + \frac{\sigma_i^2}{c} \cdot v\left(\frac{t}{c}, \frac{\epsilon}{c}\right) \tag{3}$$

$$\mu_j = \mu_j - \frac{\sigma_j^2}{c} \cdot v\left(\frac{t}{c}, \frac{\epsilon}{c}\right), \tag{4}$$

[6]Thurstone and another popular ranking method by Elo (1978) use a fixed σ for all instances.

[7]www.xbox.com/live/



(a) $v_{i\succ j}$ (b) $v_{i\equiv j}$

(c) $w_{i\succ j}$ (d) $w_{i\equiv j}$

Figure 2: Surprisal of the outcome for μ and σ^2 ($\epsilon = 0.5$).

where $c^2 = 2\gamma^2 + \sigma_i^2 + \sigma_j^2$, and v are multiplicative factors that affect the amount of change (surprisal of the outcome) in μ. In the accumulation of the variances (c^2), another free parameter called "skill chain", γ, indicates the width (or difference) of skill levels that two given players have 0.8 (80%) probability of win/lose. The multiplicative factor depends on the observation (wins or ties):

$$v_{i\succ j}(t, \epsilon) = \frac{\varphi(-\epsilon + t)}{\Phi(-\epsilon + t)}, \tag{5}$$

$$v_{i\equiv j}(t, \epsilon) = \frac{\varphi(-\epsilon - t) - \varphi(\epsilon - t)}{\Phi(\epsilon - t) - \Phi(-\epsilon - t)}, \tag{6}$$

where $\varphi(\cdot)$ is the probability density function of the standard normal distribution. As shown in Figure 2 (a) and (b), $v_{i\succ j}$ increases exponentially as t becomes smaller (i.e., the observation is unexpected), whereas $v_{i\equiv j}$ becomes close to zero when $|t|$ is close to zero. In short, v becomes larger as the outcome is more surprising.

In order to update variance (σ^2), another set of update functions is used:

$$\sigma_i^2 = \sigma_i^2 \cdot \left[1 - \frac{\sigma_i^2}{c^2} \cdot w\left(\frac{t}{c}, \frac{\epsilon}{c}\right)\right] \tag{7}$$

$$\sigma_j^2 = \sigma_j^2 \cdot \left[1 - \frac{\sigma_j^2}{c^2} \cdot w\left(\frac{t}{c}, \frac{\epsilon}{c}\right)\right], \tag{8}$$

where w serve as multiplicative factors that affect the amount of change in σ^2.

$$w_{i \succ j}(t, \epsilon) = v_{i \succ j} \cdot (v_{i \succ j} + t - \epsilon) \tag{9}$$

$$w_{i \equiv j}(t, \epsilon) = v_{i \equiv j}^2 + \frac{(\epsilon - t) \cdot \varphi(\epsilon - t) + (\epsilon + t) \cdot \varphi(\epsilon + t)}{\Phi(\epsilon - t) - \Phi(-\epsilon - t)}. \tag{10}$$

As shown in Figure 2 (c) and (d), the value of w is between 0 and 1. The underlying idea for the variance updates is that these updates always decrease the size of the variances σ^2, which means uncertainty of the instances (S_i, S_j) always decreases as we observe more pairwise comparisons. In other words, TrueSkill becomes more confident in the current estimate of μ_i and μ_j. Further details are provided by Herbrich et al. (2006).[8]

Another important property of TrueSkill is "match quality (chance to draw)". The match quality helps selecting competitive players to make games more interesting. More broadly, the match quality enables us to choose similar instances to be compared to maximize the information gain from pairwise comparisons, as in the active learning literature (Settles et al., 2008). The match quality between two instances (players) is computed as follows:

$$q(\gamma, S_i, S_j) := \sqrt{\frac{2\gamma^2}{c^2}} \exp\left(-\frac{(\mu_i - \mu_j)^2}{2c^2}\right) \tag{11}$$

Intuitively, the match quality is based on the difference $\mu_i - \mu_j$. As the difference becomes smaller, the match quality goes higher, and vice versa.

As mentioned, TrueSkill has been used for NLP tasks to infer continuous values for instances. However, it is important to note that the support of a Gaussian distribution is unbounded, namely $\mathbb{R} = (-\infty, \infty)$. This does not satisfy the property of absolute zero of scalar annotation in the level of measurement (§2). It becomes problematic when it comes to annotating a scalar (continuous) value for extremes such as extremely positive or negative sentiments. We address this issue by proposing a novel variant of TrueSkill in the next section.

3.2 Bounded Variant

TrueSkill can induce a continuous spectrum of instances (such as skill level of game players) by assuming that each instance is represented as a Gaussian distribution. However, the Gaussian distribution has unbounded support, namely $\mathbb{R} = (-\infty, \infty)$, which does not satisfy the property of *absolute* bounds for appropriate scalar annotation (i.e., ratio scale in the level of measurement).

Thus, we propose a variant of TrueSkill by changing the latent distribution from a Gaussian to a beta, using a heuristic algorithm based on TrueSkill for inference. The Beta distribution has natural $[0, 1]$ upper and lower bounds and a simple parameterization: $S_i \sim \mathcal{B}_i(\alpha_i, \beta_i)$. We choose the scalar response as the mode $\mathbb{M}[S_i]$ of the distribution and the variance as uncertainty:[9]

$$\mathbb{M}_i = \frac{\alpha_i - 1}{\alpha_i + \beta_i - 2} \tag{12}$$

$$\text{Var}_i = \sigma_i^2 = \frac{\alpha_i \beta_i}{(\alpha_i + \beta_i)^2 (\alpha_i + \beta_i + 1)} \tag{13}$$

As in TrueSkill, we iteratively update parameters of instances $\mathcal{B}(\alpha, \beta)$ according to each observation and how it is surprising. Similarly to Eqns. (3) and (4), we choose the update functions as follows;[10] first, in case that an annotator judged that S_i is preferred to S_j $(S_i \succ S_j)$,

$$\alpha_i = \alpha_i + \frac{\sigma_i^2}{c} \cdot (1 - p_{i \succ j}) \tag{14}$$

$$\beta_j = \beta_j + \frac{\sigma_j^2}{c} \cdot (1 - p_{j \prec i}) \tag{15}$$

in case of ties with $|D| > \epsilon$ and $\mathbb{M}_i > \mathbb{M}_j$,

$$\alpha_j = \alpha_j + \frac{\sigma_j^2}{c} \cdot (1 - p_{i \equiv j}) \tag{16}$$

$$\beta_i = \beta_i + \frac{\sigma_i^2}{c} \cdot (1 - p_{i \equiv j}) \tag{17}$$

and in case of ties with $|D| \leqslant \epsilon$, for both S_i, S_j,

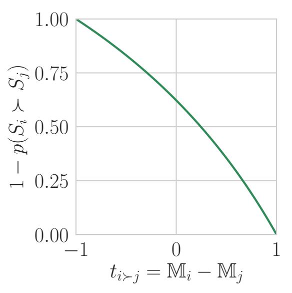
$$\alpha_{i,j} = \alpha_{i,j} + \frac{\sigma_{i,j}^2}{c} \cdot (1 - p_{i \equiv j}) \tag{18}$$

$$\beta_{i,j} = \beta_{i,j} + \frac{\sigma_{i,j}^2}{c} \cdot (1 - p_{i \equiv j}). \tag{19}$$

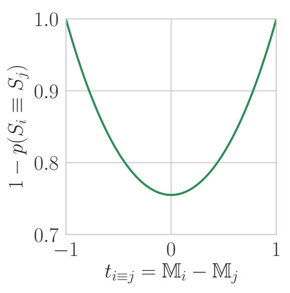
[8]The following material is also useful to understand the math behind TrueSkill (http://www.moserware.com/assets/computing-your-skill/The%20Math%20Behind%20TrueSkill.pdf).

[9]We may have instead used the mean ($\mathbb{E}[S_i] = \frac{\alpha_i}{\alpha_i + \beta_i}$) of the distribution, where in a beta ($\alpha, \beta > 1$) the mean is always closer to 0.5 than the mode, whereas mean and mode are always the same in a Gaussian distribution. The mode was selected owing to better performance in development.

[10]There may be other potential update (and surprisal) functions such as $-\log p$, instead of $1 - p$. As in our use of the mode rather than mean as scalar response, we empirically developed our update functions with respect to annotation efficiency observed through experimentation (§ 5).



(a) $1 - p_{i \succ j}$



(b) $1 - p_{i \equiv j}$

Figure 3: Surprisal of the outcome for the bounded variant ($\epsilon = 0.5$).

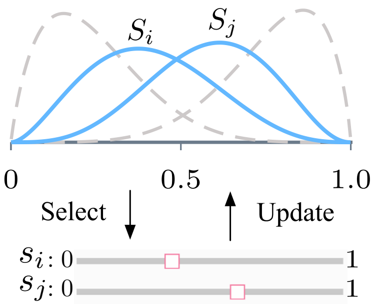


Figure 4: Illustrative example of the EASL protocol. Each instance is represented as a beta distribution. Instances are chosen to annotate according to the variance and match quality, and the parameters are updated iteratively.

Regarding the probability of pairwise comparison between instances, we follow Bradley and Terry (1952) and Rao and Kupper (1967) to describe the chance of win, tie, or loss, as follows:

$$p(S_i \succ S_j) = p(D > \epsilon) = \frac{\pi_i}{\pi_i + \theta\pi_j} \tag{20}$$

$$p(S_i \prec S_j) = p(D < -\epsilon) = \frac{\pi_j}{\theta\pi_i + \pi_j} \tag{21}$$

$$p(S_i \equiv S_j) = p(|D| \leqslant \epsilon) = \frac{(\theta^2 - 1)\pi_i\pi_j}{(\pi_i + \theta\pi_j)(\theta\pi_i + \pi_j)} \tag{22}$$

where $D = \mathbb{M}_i - \mathbb{M}_j$, $\epsilon \geqslant 0$ is a parameter to specify the tie rate, $\theta = \exp(\epsilon)$, and π is an exponential score function of S; $\pi_i = \exp(\mathbb{M}_i)$.

It is important to note that α and β never decrease (because $1 - p \geq 0$ as shown Figure 3), which satisfies the property that variance (uncertainty) always decreases as we observe more judgments, as seen in TrueSkill (§3.1). In addition, we do not need individual update functions for μ and σ^2, since the mode and variance in beta distribution depend on two shared parameters α, β (Eqns. 12 and 13).

Regarding match quality, we use the same formulation as the TrueSkill (Eqn. 11), except that the bounded model uses $\mathbb{M}$ instead of μ:

$$q(\gamma, S_i, S_j) = \sqrt{\frac{2\gamma^2}{c^2}} \exp\left(-\frac{(\mathbb{M}_i - \mathbb{M}_j)^2}{2c^2}\right) \tag{23}$$

4 Efficient Annotation of Scalar Labels

In the previous section, we propose a *bounded* online ranking aggregation model for scalar annotation. However, the amount of update by a pairwise judgment depends only on the distance between instances, not on the distance from the bounds (i.e., 0 and 1). To integrate this property into the online ranking aggregation model, we propose EASL (**E**fficient **A**nnotation of **S**calar **L**abels) that combines benefits from both direct assessment (DA) and bounded online ranking aggregation model (RA).[11]

Similarly to RA, EASL parameterizes each instance by a beta distribution (Eqns. 12 and 13), and the parameters are inferred using a computationally efficient and easy-to-implement heuristic. The difference from RA is the type of annotation. While we ask for discrete pairwise judgment ($\succ, \prec, \equiv$) between S_i and S_j in RA, here we directly ask for scalar values for them (denoted as s_i and s_j) as in DA. Thus, given an annotated score s_i which is normalized between [0,1], we change the update functions as follows:

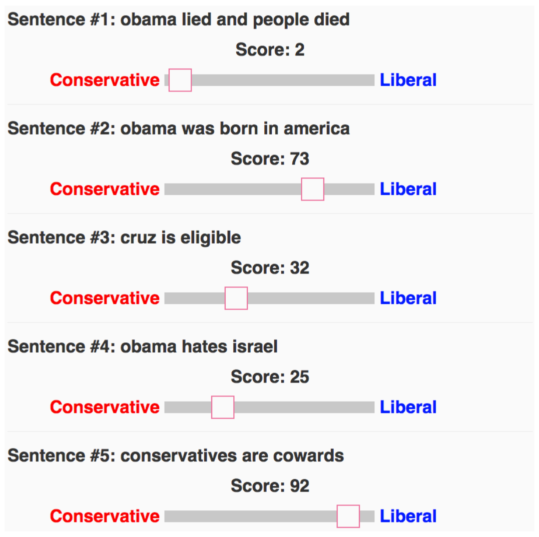
$$\alpha_i = \alpha_i + s_i \tag{24}$$

$$\beta_i = \beta_i + (1 - s_i) \tag{25}$$

This procedure may look similar to DA, where s_i is simply accumulated and averaged at the end. However, there are two differences. First, as illustrated in Figure 4, EASL parameterizes each instance as a probability distribution while DA does not. Second, DA elicits annotations independently per element, whereas EASL elicits annotations on elements in the context of other elements selected jointly according to match quality.

Further, DA generally uses a batch style annotation scheme, where the number of annotations per instance is independent from the latent scalar values. On the other hand, EASL uses online learning, which impacts the calculation of match quality. This allows us to choose instances to annotate

[11] Novikova et al. (2018) recently proposed a similar approach named RankME, which is a variant of DA with comparing multiple instances at a time. It can also be regarded as a batch-learning variant of EASL without probabilistic parameterization.


Sentence #1: obama lied and people died
Score: 2
Conservative Liberal

Sentence #2: obama was born in america
Score: 73
Conservative Liberal

Sentence #3: cruz is eligible
Score: 32
Conservative Liberal

Sentence #4: obama hates israel
Score: 25
Conservative Liberal

Sentence #5: conservatives are cowards
Score: 92
Conservative Liberal

Figure 5: Example of partial ranking with scalars (HITS)

by order of uncertainty for each instance, and as in RA, the match quality (Eqn. 23) enables us to consider similar instances in the same context.

5 Experiments

To compare different annotation methods, we conduct three experiments: (1) lexical frequency inference, (2) political spectrum inference, and (3) human evaluation for machine translation systems.

In all experiments, data collection is conducted through Amazon Mechanical Turk (AMT). We ask annotators who meet the following minimum requirements:[12] living in the US, overall approval rate $> 98\%$, and number of tasks approved > 500.

The experimental setting for DA is straightforward. We ask annotators to annotate a scalar value for each instance, one item at a time. We collect ten annotations for each instance to see the relation between the number of annotations and accuracy (i.e., correlation).

To set up the online update in RA and EASL, we use a *partial ranking* framework with scalars, where annotators are asked to rank and score n instances at one time as illustrated in Figure 5. In all three experiments, we fix $n = 5$. The partial ranking yields $\binom{n}{2}$ pairwise comparisons for RA and n scalar values for EASL.[13] It is important to note that we can simultaneously retrieve pairwise

Algorithm 1: Online pairwise ranking aggregation with bounded support.

```
   Input: Instances ${S_1^N}$
   Output: Updated instances ${S_1^N}$
   /* Initialize params                          */
1  $(\alpha_i, \beta_i)_{\in S} = (\alpha_i^{\text{init}}, \beta_i^{\text{init}})$
   /* Update $S$ over iterations                 */
2  foreach iteration do
3      HITS = SampleByMatchQuality($S, N, n$)
4      $A$ = Annotate(HITS)
5      for $obs \in A$ do                    // Update $S$
6          $i, j, d$ = parseObservation($obs$)
7          $\alpha_{i,j}, \beta_{i,j}$ = update($i, j, d$)
8  return $S$
9  Function SampleByMatchQuality($S, N, n$)
10     $k = N/n$
11     descendingSort($S$, key=Var[$S$])
12     $S'$ = top-k instances of $S$
13     HITS = []
14     foreach $S_i \in S'$ do
15         m = []
16         foreach $S_j \in S_{/S'}$ do
17             m.append([matchQuality($S_i, S_j$), $j$])
18         $p$ = normalize(m)
19         $\tilde{S}$ = sampling n-1 items by $p$
20         HITS.append([$S_i, \tilde{S}$])
21     return HITS
```

judgments ($\succ, \prec, \equiv$) as well as scalar values from this format.

In each iteration, n instances are selected by variance and match quality. We first select top k ($= N/n$) instances according to the variance, and for each selected instance we choose the other $n - 1$ instances to be compared based on match quality. This approach has been used in the NLP community in tasks such as for assessing machine translation quality (Bojar et al., 2014; Sakaguchi et al., 2014; Bojar et al., 2015, 2016) to collect pairwise judgments efficiently. The detailed procedure of iterative parameter updates in the RA and EASL is described in Algorithm 1. As mentioned in Section 4, the main difference between RA and EASL is the update functions (line 7).

Model hyper-parameters in RA and EASL are set as follows; each instance is initialized as $\alpha_i^{\text{init}} = 1.0$, $\beta_i^{\text{init}} = 1.0$. The skill chain parameter γ and tie-rate parameter ϵ are set to be 0.1.[14]

5.1 Lexical Frequency Inference

In the first experiment, we compare the three scalar annotation approaches on lexical frequency inference, in which we ask annotators to judge frequency (from very rare to very frequent) of verbs

[12]In all experiments, we set the reward of single instance to be \$0.01 (i.e., \$0.05 in RA and EASL). This is \$8/hour, assuming that annotating one instance takes five seconds. Prior to annotation, we run a pilot to make sure that the participants understand the task correctly and the instructions are clear.

[13]The partial ranking can be regarded as mini-batching.

[14]We explored the hyper-parameters γ, ϵ in a pilot task.

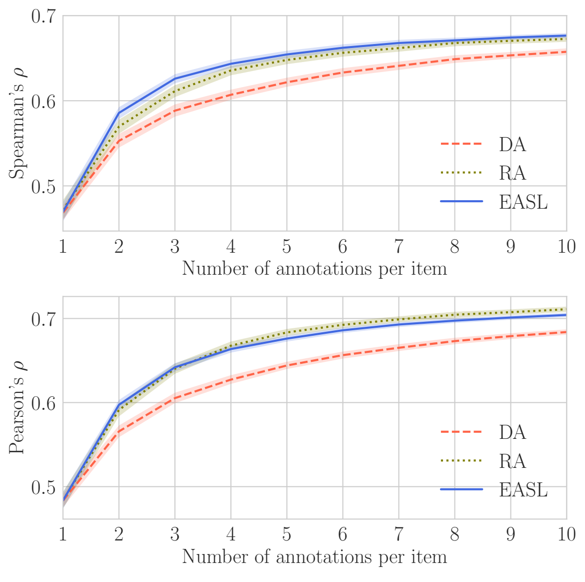


Figure 6: Spearman's (top) and Pearson's (bottom) correlations with three difference methods on lexical frequency inference annotation: direct assessment (DA), online ranking aggregation (RA), and EASL. The shade for each line indicates 95% confidence intervals by bootstrap resampling (running 100 times).

that are randomly selected from the corpus of Contemporary American English (COCA)[15]. We include this task for evaluation owing to its non-subjective ground truth (relative corpus frequency) which can be used as an oracle response we would like to maximally correlate with.[16]

We randomly select 150 verbs from COCA; the log frequency ($\log_{10}$) is regarded as the oracle. In DA, each instance is annotated by 10 different annotators.[17] In the RA and EASL, annotators are asked to rank/score five verbs for each HIT ($n = 5$). Each iteration contains 20 HITS and we run 10 iterations, which means that total number of annotations is the same in DA, RA, and EASL.[18]

Figure 6 presents Spearman's and Pearson's correlations, indicating how accurately each annotation method obtains scalar values for each instance. Overall, in all three methods, the correlations are increased as more annotations are made. The result also shows that RA and EASL approaches achieve high correlation more efficiently than DA. The gain of efficiency from DA to EASL is about 50%; two iterations in EASL achieves a close Spearman's ρ to three annotators in DA.

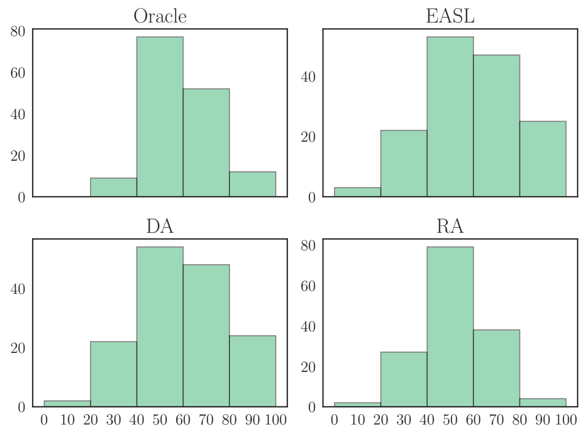


Figure 7: Histograms of scalar values on lexical frequency obtained by each annotation scheme (direct assessment (DA), online ranking aggregation (RA), and EASL), and the oracle. The scalar annotations are put into five bins to see the overall distribution. The scalar in the oracle is normalized as log_{10}(frequency(S_i)) / max log_{10}(frequency(S)).

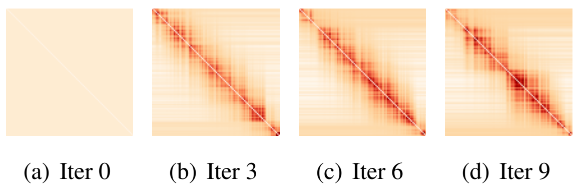


Figure 8: Heatmaps of match quality distribution across the cross-product of instances ordered by the oracle (i.e., log_{10}(frequency)).

Figure 7 presents the results of the final scalar values that each method annotated. The distribution of the histograms shows that overall three methods successfully capture the latent distribution of scalar values in the data.

Figure 8 shows a dynamic change of match quality. In the beginning (iteration 0), all the instances are equally competitive because we have no information about them and initialize them with the same parameters. As iterations go on, the instances along the diagonal have higher match quality, indicating that competitive matches are more likely to be selected for a next iteration. In other words, match-quality helps to choose informative pairs to compare at each iteration, which reduces the number of less informative annotations (e.g., a pairwise comparison between the highest and lowest instances).

[15]https://www.wordfrequency.info/

[16]Lexical frequency inference is an established experiment in (computational) psycholinguistics. E.g., human behavioral measures have been compared with predictability and bias in various corpora (Balota et al., 1999; Fine et al., 2014).

[17]The agreement rate in DA (10 annotators) is 0.37 in Spearman's ρ. Considering the difficulty of ranking 150 verbs, this rate is fair.

[18]Technically, the number of annotations per instance vary in RA and EASL, because they choose instances by match quality at each iteration.

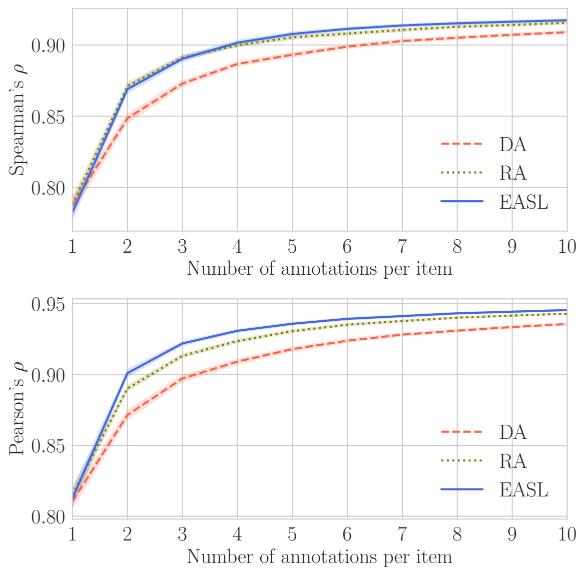


Figure 9: Spearman's (top) and Pearson's (bottom) correlations with three difference methods on political spectrum annotation: direct assessment (DA), online ranking aggregation (RA), and EASL

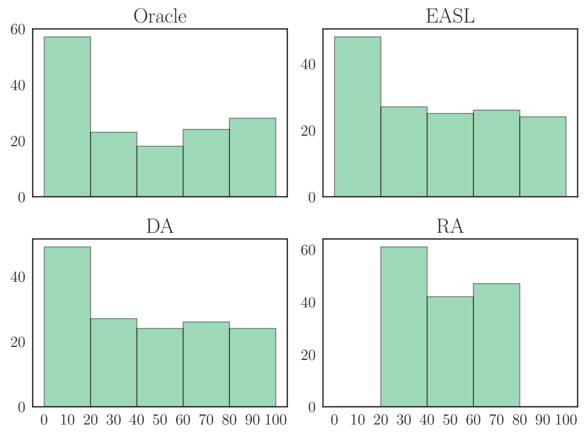


Figure 10: Histograms of scalar values on political spectrum obtained by each annotation scheme (DA, RA, EASL) and the oracle. Scalars are put into five bins to see the overall distribution.

Propositions	Gold	DA	RA	EASL
the republicans are useless	100	91.7	75.8	91.9
obama is right	92.9	90.1	74.6	90.0
hillary will win	78.6	86.3	72.9	86.4
aca is a success	75.0	78.2	68.3	77.3
harry reid is a democrat	53.6	55.5	55.8	55.9
ebola is a virus	50.0	53.0	53.8	53.5
cruz is eligible	32.2	31.0	44.0	31.4
global warming is a religion	28.6	22.4	37.3	23.0
bush kept us safe	10.7	9.6	31.5	9.6
democrats are corrupt	0.0	7.1	29.9	7.4

Table 1: Example propositions and the scalar political spectrum ranged between 0 (*very conservative*) and 100 (*very liberal*) by each approach: direct assessment, online ranking aggregation, and EASL. The dashed lines indicate a split by 5-ary ordinal scale.

5.2 Political Spectrum Inference

In the second experiment, we compare the three scalar annotation methods for political spectrum inference. We use the Fine-Grained Political Statements dataset (Bamman and Smith, 2015), which consists of 766 propositions collected from political blog comments, paired with judgments about the political belief of the statement (or the person who would say it) based on the five ordinals: *very conservative* (-2), *slightly conservative* (-1), *neutral* (0), *slightly liberal* (1), and *very liberal* (2). We normalize the ordinal scores between 0 and 1. The dataset contains the mean scores by aggregating 7 annotations for each proposition.[19]

We randomly choose 150 political propositions from the dataset (see the histogram in Figure 10 oracle).[20] The experimental setting (i.e., the number of annotations per instance, the number of iterations, and the number of HITS in each iteration) is the same as the lexical frequency inference experiment (§5.1).

Figure 9 shows Spearman's and Pearson's correlations to the oracle by each method. Overall, all the three methods achieve strong correlation above 0.9. We also find that RA and EASL reach high correlation more efficiently than DA as in the lexical frequency inference experiment (§5.1). The gain of efficiency from DA to EASL is about 50%; 4-way redundant annotation in EASL achieves a close Spearman's ρ to 6-way redundancy in DA.

Figure 10 presents the results of the annotated scalar values by each method. The distribution of the histograms shows that DA and EASL successfully fit to the distribution in the oracle, whereas RA converges to a rather narrow range. This is because of the "lack of distance from bounds" in RA that is explained in §4. We note that renormalizing the distribution in RA will not address the issue. For instance, when the dataset has only liberal propositions, RA still fails to capture the latent distribution because it looks only at relative distances between instances but not the distance from bounds. Table 1 shows the examples of scalar annotations by each method. Again, we

[19]We stress that the oracle here derives from subjective annotations: it does not necessarily reflect the true latent scalar values for each instance. However, in this experiment, we use them as a tentative oracle to compare three scalar annotation methods objectively.

[20]The agreement rate in DA (among 10 annotators) is 0.67 in Spearman's ρ. This is significantly high, considering the difficulty of ranking 150 instances in order.

see that RA approach has a narrower range than the oracle, DA, and EASL.

5.3 Ranking Machine Translation Systems

In the third experiment, we apply the scalar annotation methods for evaluating machine translation systems. This is different from two previous experiments, because the main purpose is to rank the MT systems (S_1^N) rather than the adequacy (q) of each MT output for a given source sentence (m). Namely, we want to rank S_i by observing $q_{i,m}$.

We use WMT16 German-English translation dataset (Bojar et al., 2016), which consists of 2,999 test set sentences and the translations from 10 different systems with DA annotation. Each sentence has its adequacy score annotation between 0 and 100, and the average adequacy scores are computed for each system for ranking. In this setting, annotators are asked to judge adequacy of system output(s) with the reference being given. The official scores (made by DA) and ranking in WMT16 are used as the oracle in this experiment.

In this experiment, we replicate DA and run EASL to compare the efficiency. We omit RA in this experiment, because it does not necessarily capture the distance from bounds as shown in the previous experiment (§5.2). In DA, 33,760 translation outputs (3,376 sentences per system in average) are randomly sampled without replacement to make sure that it reaches up to the same result as oracle when the entire data are used.

In EASL, we assume that adequacy (q) of an MT output by system (S_i) for a given source sentence (m) is drawn from beta distribution: $q_{i,m} \sim \mathcal{B}(\alpha_i, \beta_i)$.[21] Annotators are asked to judge adequacy of system outputs by scoring 0 and 100. Similarly to the previous experiments (§ 5.1 and § 5.2), we use the partial ranking strategy, where we show $n = 5$ system outputs (for the same source sentence l) to annotate at a time. The procedure of parameter updates is the same as previous experiments (Algorithm 1).

We compare the correlations (Spearman's ρ) of system ranking with respect to the number of annotations per system, and the result is shown in Figure 11. As seen in the previous two experiments, EASL achieves higher Spearmans correlation on ranking MT systems with smaller number of annotations than the baseline method (DA),

[21]This is the same setting as WMT14, WMT15, and WMT16 (Bojar et al., 2014, 2015), although they used TrueSkill (Gaussian) instead of EASL to rank systems.

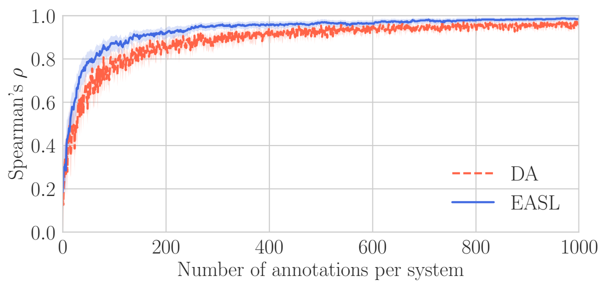


Figure 11: Spearman's correlation on ranking machine translation systems on WMT16 German-English data: direct assessment (DA), and EASL. The shade for each line indicates 95% confidence intervals by bootstrap resampling (running 100 times).

which means EASL is able to collect annotation more efficiently. The result shows that EASL can be applied for efficient system evaluation in addition to data curation.

6 Conclusions

We have presented an efficient, online model to elicit scalar annotations for computational linguistic datasets and system evaluations. The model combines two approaches for scalar annotation: direct assessment and online pairwise ranking aggregation. We conducted three illustrative experiments on lexical frequency inference, political spectrum inference, and ranking machine translation systems. We have shown that our approach, EASL (**E**fficient **A**nnotation of **S**calar **L**abels), outperforms direct assessment in terms of annotation efficiency and outperforms online ranking aggregation in terms of accurately capturing the latent distributions of scalar values. The significant gains demonstrated suggests EASL as a promising approach for future dataset curation and system evaluation in the community.

Acknowledgments

We are grateful to Rachel Rudinger, Adam Teichert, Chandler May, Tongfei Chen, Pushpendre Rastogi, and anonymous reviewers for their useful feedback. This work was supported in part by IARPA MATERIAL and DARPA LORELEI. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies of the U.S. Government.

References

David A. Balota, Cortese Michael J., and Maura Pilotti. 1999. Item-level analyses of lexical decision performance: Results from a mega-study. In *Abstracts of the 40th Annual Meeting of the Psychonomics Society*, page 44, Los Angeles, California. Psychonomic Society.

David Bamman and Noah A. Smith. 2015. Open extraction of fine-grained political statements. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 76–85, Lisbon, Portugal. Association for Computational Linguistics.

Timo Baumann. 2017. Large-scale speaker ranking from crowdsourced pairwise listener ratings.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Arpad E. Elo. 1978. *The rating of chessplayers, past and present*. Arco Pub.

Alex B. Fine, Austin F. Frank, T. Florian Jaeger, and Benjamin Van Durme. 2014. Biases in predicting the human language model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–12, Baltimore, Maryland. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 33–41, Sofia, Bulgaria. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2014. Is machine translation getting better over time? In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 443–451, Gothenburg, Sweden. Association for Computational Linguistics.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkill^{TM}: A Bayesian Skill Rating System. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 569–576, Vancouver, British Columbia, Canada.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event detection and factuality assessment with non-expert supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1648, Lisbon, Portugal. Association for Computational Linguistics.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

Jing Liu, Quan Wang, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. Question difficulty estimation in community question answering services. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 85–90, Seattle, Washington, USA. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

J. Novikova, O. Dušek, and V. Rieser. 2018. RankME: Reliable Human Ratings for Natural Language Generation. *ArXiv e-prints*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM*, 11(122-129):1–2.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 408–413, Beijing, China. Association for Computational Linguistics.

P. V. Rao and L. L. Kupper. 1967. Ties in paired-comparison experiments: A generalization of the bradley-terry model. *Journal of the American Statistical Association*, 62(317):194–204.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11, Baltimore, Maryland, USA. Association for Computational Linguistics.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, pages 1–10.

S. S. Stevens. 1946. On the theory of scales of measurement. *Science*, 103(2684):677–680.

Louis L Thurstone. 1927. The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology*, 21(4):384.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Amos Tversky and Daniel Kahneman. 1974. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. 2010. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043.

Janyce M Wiebe, Rebecca F Bruce, and Thomas P O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 246–253. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association for Computational Linguistics*, 5:379–395.

Neural Argument Generation Augmented with Externally Retrieved Evidence

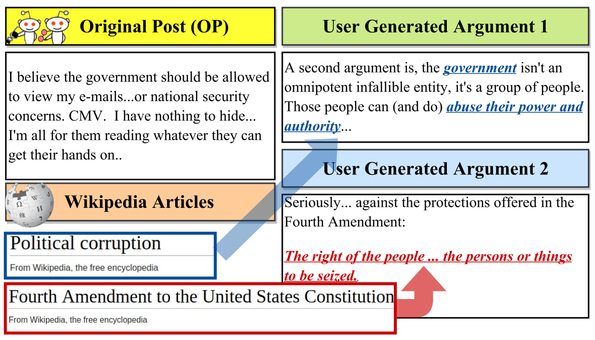
Xinyu Hua and **Lu Wang**
College of Computer and Information Science
Northeastern University
Boston, MA 02115
`hua.x@husky.neu.edu` `luwang@ccs.neu.edu`

Abstract

High quality arguments are essential elements for human reasoning and decision-making processes. However, effective argument construction is a challenging task for both human and machines. In this work, we study a novel task on *automatically generating arguments of a different stance for a given statement*. We propose an encoder-decoder style neural network-based argument generation model enriched with externally retrieved evidence from Wikipedia. Our model first generates a set of talking point phrases as intermediate representation, followed by a separate decoder producing the final argument based on both input and the keyphrases. Experiments on a large-scale dataset collected from Reddit show that our model constructs arguments with more topic-relevant content than a popular sequence-to-sequence generation model according to both automatic evaluation and human assessments.

1 Introduction

Generating high quality arguments plays a crucial role in decision-making and reasoning processes (Bonet and Geffner, 1996; Byrnes, 2013). A multitude of arguments and counter-arguments are constructed on a daily basis, both online and offline, to persuade and inform us on a wide range of issues. For instance, debates are often conducted in legislative bodies to secure enough votes for bills to pass. In another example, online deliberation has become a popular way of soliciting public opinions on new policies' pros and cons (Albrecht, 2006; Park et al., 2012). Nonetheless, constructing persuasive arguments is a daunting task, for both human and computers. We believe that developing effective argument generation models will enable a broad range of compelling applications, including debate coaching, improving students' essay writing skills, and pro-


Original Post (OP)
User Generated Argument 1
Wikipedia Articles
User Generated Argument 2
Political corruption
From Wikipedia, the free encyclopedia
Fourth Amendment to the United States Constitution
From Wikipedia, the free encyclopedia

I believe the government should be allowed to view my e-mails...or national security concerns. CMV. I have nothing to hide... I'm all for them reading whatever they can get their hands on..

A second argument is, the ***government*** isn't an omnipotent infallible entity, it's a group of people. Those people can (and do) ***abuse their power and authority***...

Seriously... against the protections offered in the Fourth Amendment:

The right of the people ... the persons or things to be seized.

Figure 1: Sample user arguments from Reddit Change My View subcommunity that argue against original post's thesis on "government should be allowed to view private emails". Both arguments leverage supporting information from Wikipedia articles.

viding context of controversial issues from different perspectives. As a consequence, there exists a pressing need for automating the argument construction process.

To date, progress made in argument generation has been limited to retrieval-based methods—arguments are ranked based on relevance to a given topic, then the top ones are selected for inclusion in the output (Rinott et al., 2015; Wachsmuth et al., 2017; Hua and Wang, 2017). Although sentence ordering algorithms are developed for information structuring (Sato et al., 2015; Reisert et al., 2015), existing methods lack the ability of synthesizing information from different resources, leading to redundancy and incoherence in the output.

In general, the task of argument generation presents numerous challenges, ranging from aggregating supporting evidence to generating text with coherent logical structure. One particular hurdle comes from the underlying natural language generation (NLG) stack, whose success has been limited to a small set of domains. Especially, most previous NLG systems rely on templates that are either constructed by rules (Hovy, 1993; Belz, 2008; Bouayad-Agha et al., 2011), or acquired from a domain-specific corpus (Angeli et al., 2010) to enhance grammaticality and coherence. This makes them unwieldy to be adapted for new domains.

In this work, we study the following novel problem: *given a statement on a controversial issue, generate an argument of an alternative stance*. To address the above challenges, we present *a neural network-based argument generation framework augmented with externally retrieved evidence*. Our model is inspired by the observation that when humans construct arguments, they often collect references from external sources, e.g., Wikipedia or research papers, and then write their own arguments by synthesizing talking points from the references. Figure 1 displays sample arguments by users from Reddit subcommunity `/r/ChangeMyView` [1] who argue against the motion that "government should be allowed to view private emails". Both replies leverage information drawn from Wikipedia, such as "political corruption" and "Fourth Amendment on protections of personal privacy".

Concretely, our neural argument generation model adopts the popular encoder-decoder-based sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014), which has achieved significant success in various text generation tasks (Bahdanau et al., 2015; Wen et al., 2015; Wang and Ling, 2016; Mei et al., 2016; Wiseman et al., 2017). Our encoder takes as input a statement on a disputed issue, and a set of relevant evidence automatically retrieved from English Wikipedia[2]. Our decoder consists of two separate parts, one of which first generates keyphrases as intermediate representation of "talking points", and the other then generates an argument based on both input and keyphrases.

Automatic evaluation based on BLEU (Papineni et al., 2002) shows that our framework generates better arguments than directly using retrieved sentences or popular seq2seq-based generation models (Bahdanau et al., 2015) that are also trained with retrieved evidence. We further design a novel evaluation procedure to measure whether the arguments are on-topic by predicting their relevance to the given statement based on a separately trained relevance estimation model. Results suggest that our model generated arguments are more likely to be predicted as on-topic, compared to other seq2seq-based generations models.

The rest of this paper is organized as follows. Section 2 highlights the roadmap of our system. The dataset used for our study is introduced in Section 3. The model formulation and retrieval methods are detailed in Sections 4 and 5. We then describe the experimental setup and results in Sections 6 and 7, followed by further analysis and future directions in Section 8. Related work is discussed in Section 9. Finally, we conclude in Section 10.

2 Framework

Our argument generation pipeline, consisting of *evidence retrieval* and *argument construction*, is depicted in Figure 2. Given a statement, a set of queries are constructed based on its topic signature words (e.g., "government" and "national security") to retrieve a list of relevant articles from Wikipedia. A reranking component further extracts sentences that may contain supporting evidence, which are used as additional input information for the neural argument generation model.

The generation model then encodes the statement and the evidence with a shared encoder in sequence. Two decoders are designed: the *keyphrase decoder* first generates an intermediate representation of talking points in the form of keyphrases (e.g., "right to privacy", "political corruption"), followed by a separate *argument decoder* which produces the final argument.

3 Data Collection and Processing

We draw data from Reddit subcommunity `/r/ChangeMyView` (henceforth CMV), which focuses on facilitating open discussions on a wide range of disputed issues. Specifically, CMV is structured as discussion threads, where the original post (OP) starts with a viewpoint on a controversial topic, followed with detailed reasons, then other users reply with counter-arguments. Importantly, when a user believes his view has been changed by an argument, a *delta* is often awarded to the reply.

In total, 26,761 threads from CMV are downloaded, dating from January 2013 to June 2017[3].

[1] `https://www.reddit.com/r/changemyview`
[2] `https://en.wikipedia.org/`

[3]Dataset used in this paper is available at `http://xinyuhua.github.io/Resources/`.

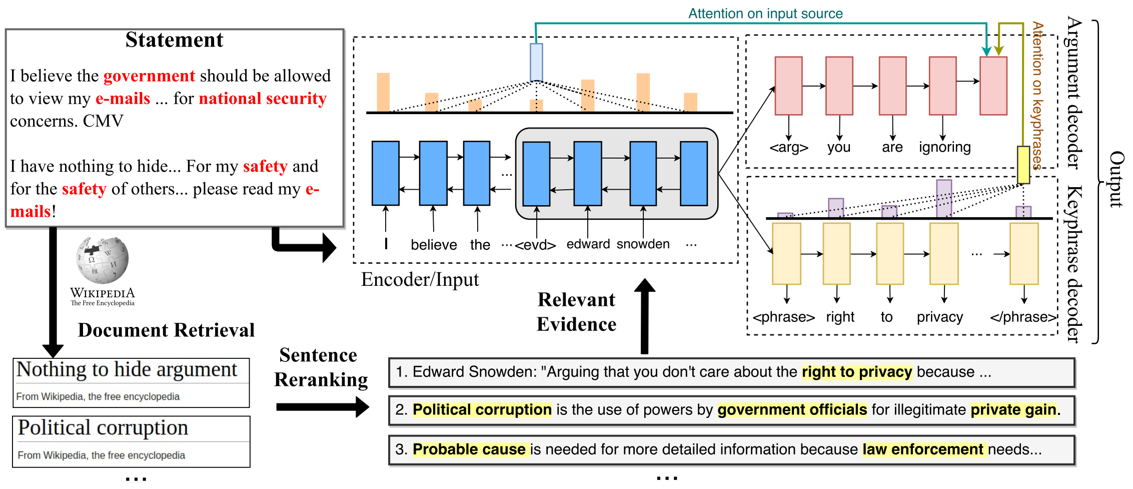

Statement
I believe the government should be allowed to view my e-mails ... for national security concerns. CMV
I have nothing to hide... For my safety and for the safety of others... please read my e-mails!
Wikipedia
Document Retrieval
Nothing to hide argument
Political corruption
Sentence Reranking
1. Edward Snowden: "Arguing that you don't care about the right to privacy because ...
2. Political corruption is the use of powers by government officials for illegitimate private gain.
3. Probable cause is needed for more detailed information because law enforcement needs...
Relevant Evidence
Encoder/Input
I believe the <evd> edward snowden
Attention on input source
Attention on keyphrases
<arg> you are ignoring
<phrase> right to privacy </phrase>
Argument decoder
Keyphrase decoder
Output

Figure 2: Overview of our system pipeline (best viewed in color). Given a statement, relevant articles are retrieved from Wikipedia with topic signatures from statement as queries (marked in red and boldface). A reranking module then outputs top sentences as evidence. The statement and the evidence (encoder states in gray panel) are concatenated and encoded as input for our argument generation model. During decoding, the keyphrase decoder first generates talking points as phrases, followed by the argument decoder which constructs the argument by attending both input and keyphrases.

Only root replies (i.e., replies directly addressing OP) that meet all of the following requirements are included: (1) longer than 5 words, (2) without offensive language[4], (3) awarded with *delta* or with more upvotes than downvotes, and (4) not generated by system moderators.

After filtering, the resultant dataset contains 26,525 OPs along with 305,475 relatively high quality root replies. We treat each OP as the input statement, and the corresponding root replies as target arguments, on which our model is trained and evaluated.

A Focused Domain Dataset. The current dataset contains diverse domains with unbalanced numbers of arguments. We therefore choose samples from the politics domain due to its large volume of discussions and good coverage of popular arguments in the domain.

However, topic labels are not available for the discussions. We thus construct a domain classifier for politics vs. non-politics posts based on a logistic regression model with unigram features, trained from our heuristically labeled Wikipedia abstracts[5]. Concretely, we manually collect two lists of keywords that are indicative of politics and non-politics. Each abstract is labeled as politics or non-politics if its title only matches keywords from one category.[6] In total, 264,670 politics abstracts and 827,437 of non-politics are labeled. Starting from this dataset, our domain classifier is trained in a bootstrapping manner by gradually adding OPs predicted as politics or non-politics.[7] Finally, 12,549 OPs are labeled as politics, each of which is paired with 9.4 high-quality target arguments on average. The average length for OPs is 16.1 sentences of 356.4 words, and 7.7 sentences of 161.1 words for arguments.

4 Model

In this section, we present our argument generation model, which jointly learns to generate talking points in the form of keyphrases and produce arguments based on the input and keyphrases. Extended from the successful seq2seq attentional model (Bahdanau et al., 2015), our proposed model is novel in the following ways. First, two separate decoders are designed, one for generating keyphrases, the other for argument construction. By sharing the encoder with keyphrase generation, our argument decoder is better aware of salient talking points in the input. Second, a novel

[4] We use offensive words collected by Google's What Do You Love project: `https://gist.github.com/jamiew/1112488`, last accessed on February 22nd, 2018.

[5] About 1.3 million English Wikipedia abstracts are downloaded from `http://dbpedia.org/page/`.

[6] Sample keywords for politics: "congress", "election", "constitution"; for non-politics: "art", "fashion","music". Full lists are provided in the supplementary material.

[7] More details about our domain classifier are provided in the supplementary material.

attention mechanism is designed for argument decoding by attending both input and the previously generated keyphrases. Finally, a reranking-based beam search decoder is introduced to promote topic-relevant generations.

4.1 Model Formulation

Our model takes as input a sequence of tokens $\boldsymbol{x} = \{\boldsymbol{x}^O; \boldsymbol{x}^E\}$, where $\boldsymbol{x}^O$ is the statement sequence and $\boldsymbol{x}^E$ contains relevant evidence that is extracted from Wikipedia based on a separate retrieval module. A special token `<evd>` is inserted between $\boldsymbol{x}^O$ and $\boldsymbol{x}^E$. Our model then first generates a set of keyphrases as a sequence $\boldsymbol{y}^p = \{y_l^p\}$, followed by an argument $\boldsymbol{y}^a = \{y_t^a\}$, by maximizing $\log P(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{y} = \{\boldsymbol{y}^p; \boldsymbol{y}^a\}$.

The objective is further decomposed into $\sum_t \log P(y_t|y_{1:t-1}, \boldsymbol{x})$, with each term estimated by a softmax function over a non-linear transformation of decoder hidden states $\boldsymbol{s}_t^a$ and $\boldsymbol{s}_t^p$, for argument decoder and keyphrase decoder, respectively. The hidden states are computed as done in Bahdanau et al. (2015) with attention:

$$\boldsymbol{s}_t = g(\boldsymbol{s}_{t-1}, \boldsymbol{c}_t, y_t) \tag{1}$$

$$\boldsymbol{c}_t = \sum_{j=1}^{T} \alpha_{tj} \boldsymbol{h}_j \tag{2}$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})} \tag{3}$$

$$e_{tj} = \boldsymbol{v}^T \tanh(\boldsymbol{W_h} \boldsymbol{h}_j + \boldsymbol{W_s} \boldsymbol{s}_t + \boldsymbol{b}_{attn}) \tag{4}$$

Notice that two sets of parameters and different state update functions $g(\cdot)$ are learned for separate decoders: $\{\boldsymbol{W}_h^a, \boldsymbol{W}_s^a, \boldsymbol{b}_{attn}^a, g^a(\cdot)\}$ for the argument decoder; $\{\boldsymbol{W}_h^p, \boldsymbol{W}_s^p, \boldsymbol{b}_{attn}^p, g^p(\cdot)\}$ for the keyphrase decoder.

Encoder. A two-layer bidirectional LSTM (biLSTM) is used to obtain the encoder hidden states $\boldsymbol{h}_i$ for each time step i. For biLSTM, the hidden state is the concatenation of forward and backward hidden states: $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}_i}]$. Word representations are initialized with 200-dimensional pre-trained GloVe embeddings (Pennington et al., 2014), and updated during training. The last hidden state of encoder is used to initialize both decoders. In our model the encoder is shared by argument and keyphrase decoders.

Decoders. Our model is equipped with two decoders: *keyphrase decoder* and *argument decoder*, each is implemented with a separate two-layer unidirectional LSTM, in a similar spirit with one-to-many multi-task sequence-to-sequence learning (Luong et al., 2015). The distinction is that our training objective is the sum of two loss functions:

$$\mathcal{L}(\theta) = -\frac{\alpha}{T_p} \sum_{(\boldsymbol{x}, \boldsymbol{y}^p) \in D} \log P(\boldsymbol{y}^p | \boldsymbol{x}; \theta) - \frac{(1-\alpha)}{T_a} \sum_{(\boldsymbol{x}, \boldsymbol{y}^a) \in D} \log P(\boldsymbol{y}^a | \boldsymbol{x}; \theta) \tag{5}$$

where T_p and T_a denote the lengths of reference keyphrase sequence and argument sequence. α is a weighting parameter, and it is set as 0.5 in our experiments.

Attention over Both Input and Keyphrases. Intuitively, the argument decoder should consider the generated keyphrases as talking points during the generation process. We therefore propose an attention mechanism that can attend both encoder hidden states and the keyphrase decoder hidden states. Additional context vector $\boldsymbol{c}'_t$ is then computed over keyphrase decoder hidden states $\boldsymbol{s}_j^p$, which is used for computing the new argument decoder state:

$$\boldsymbol{s}_t^a = g'(\boldsymbol{s}_{t-1}^a, [\boldsymbol{c}_t; \boldsymbol{c}'_t], y_t^a) \tag{6}$$

$$\boldsymbol{c}'_t = \sum_{j=1}^{T_p} \alpha'_{tj} \boldsymbol{s}_j^p \tag{7}$$

$$\alpha'_{tj} = \frac{\exp(e'_{tj})}{\sum_{k=1}^{T_p} \exp(e'_{tk})} \tag{8}$$

$$e'_{tj} = \boldsymbol{v}'^T \tanh(\boldsymbol{W'_p} \boldsymbol{s}_j^p + \boldsymbol{W'_a} \boldsymbol{s}_t^a + \boldsymbol{b}'_{attn}) \tag{9}$$

where $\boldsymbol{s}_j^p$ is the hidden state of keyphrase decoder at position j, $\boldsymbol{s}_t^a$ is the hidden state of argument decoder at timestep t, and $\boldsymbol{c}_t$ is computed in Eq. 2.

Decoder Sharing. We also experiment with a shared decoder between keyphrase generation and argument generation: the last hidden state of the keyphrase decoder is used as the initial hidden state for the argument decoder. A special token `<arg>` is inserted between the two sequences, indicating the start of argument generation.

4.2 Hybrid Beam Search Decoding

Here we describe our decoding strategy on the argument decoder. We design a hybrid beam expansion method combined with segment-based reranking to promote diversity of beams and informativeness of the generated arguments.

Hybrid Beam Expansion. In the standard beam search, the top k words of highest probability are selected deterministically based on the softmax output to expand each hypothesis. However, this may lead to suboptimal output for text generation (Wiseman and Rush, 2016), e.g., one beam often dominates and thus inhibits hypothesis diversity. Here we only pick the top n words ($n < k$), and randomly draw another $k - n$ words based on the multinomial distribution after removing the n expanded words from the candidates. This leads to a more diverse set of hypotheses.

Segment-based Reranking. We also propose to rerank the beams every p steps based on beam's coverage of content words from input. Based on our observation that likelihood-based reranking often leads to overly generic arguments (e.g., "I don't agree with you"), this operation has the potential of encouraging more informative generation. $k = 10$, $n = 3$, and $p = 10$ are used for experiments. The effect of parameter selection is studied in Section 7.

5 Relevant Evidence Retrieval

5.1 Retrieval Methodology

We take a two-step approach for retrieving *evidence sentences*: given a statement, (1) constructing one query per sentence and retrieving relevant articles from Wikipedia, and (2) reranking paragraphs and then sentences to create the final set of evidence sentences. Wikipedia is used as our evidence source mainly due to its objective perspective and broad coverage of topics. A dump of December 21, 2016 was downloaded. For training, evidence sentences are retrieved with queries constructed from target user arguments. For test, queries are constructed from OP.

Article Retrieval. We first create an inverted index lookup table for Wikipedia as done in Chen et al. (2017). For a given statement, we construct one query per sentence to broaden the diversity of retrieved articles. Therefore, multiple passes of retrieval will be conducted if more than one query is created. Specifically, we first collect topic signature words of the post. Topic signatures (Lin and Hovy, 2000) are terms strongly correlated with a given post, measured by log-likelihood ratio against a background corpus. We treat posts from other discussions in our dataset as background. For each sentence, one query is constructed based on the noun phrases and verbs containing at least one topic signature word. For instance, a query "`the government, my e-mails, national security`" is constructed for the first sentence of OP in the motivating example (Figure 2). Top five retrieved articles with highest TF-IDF similarity scores are kept per query.

	Queries Constructed from	
	OP	**Argument**
Avg # Topic Sig.	17.2	9.8
Avg # Query	6.7	1.9
Avg # Article Retrieved	26.1	8.0
Avg # Sent. Retrieved	67.3	8.5

Table 1: Statistics for evidence sentence retrieval from Wikipedia. Considering query construction from either OP or target user arguments, we show the average numbers of topic signatures collected, queries constructed, and retrieved articles and sentences.

Sentence Reranking. The retrieved articles are first segmented into paragraphs, which are reranked by TF-IDF similarity to the given statement. Up to 100 top ranked paragraphs with positive scores are retained. These paragraphs are further segmented into sentences, and reranked according to TF-IDF similarity again. We only keep up to 10 top sentences with positive scores for inclusion in the evidence set.

5.2 Gold-Standard Keyphrase Construction

To create training data for the keyphrase decoder, we use the following rules to identify keyphrases from evidence sentences that are reused by human writers for argument construction:

- Extract noun phrases and verb phrases from evidence sentences using Stanford CoreNLP (Manning et al., 2014).
- Keep phrases of length between 2 and 10 that overlap with content words in the argument.
- If there is span overlap between phrases, the longer one is kept if it has more content word coverage of the argument; otherwise the shorter one is retained.

The resultant phrases are then concatenated with a special delimiter `<phrase>` and used as gold-standard generation for training.

6 Experimental Setup

6.1 Final Dataset Statistics

Encoding the full set of evidence by our current decoder takes a huge amount of time. We there propose a sampling strategy to allow the encoder to finish encoding within reasonable time by considering only a subset of the evidence: For each sentence in the statement, up to three evidence sentences are randomly sampled from the retrieved set; then the sampled sentences are concatenated. This procedure is repeated three times per statement, where a statement is an user argument for training data and an OP for test set. In our experiments, we remove duplicates samples and the ones without any retrieved evidence sentence. Finally, we break down the augmented data into a training set of 224,553 examples (9,737 unique OPs), 13,911 for validation (640 OPs), and 30,417 retained for test (1,892 OPs).

6.2 Training Setup

For all models, we use a two-layer biLSTM as encoder and a two-layer unidirectional LSTM as decoder, with 200-dimensional hidden states in each layer. We apply dropout (Gal and Ghahramani, 2016) on RNN cells with a keep probability of 0.8. We use Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001 to optimize the cross-entropy loss. Gradient clipping is also applied with the maximum norm of 2. The input and output vocabulary sizes are both 50k.

Curriculum Training. We train the models in three stages where the truncated input and output lengths are gradually increased. Details are listed in Table 2. Importantly, this strategy allows model training to make rapid progress during early stages. Training each of our full models takes about 4 days on a Quadro P5000 GPU card with a batch size of 32. The model converges after about 10 epochs in total with pre-training initialization, which is described below.

Component	Stage 1	Stage 2	Stage 3
Encoder			
OP	50	150	400
Evidence	0	80	120
Decoder			
Keyphrases	0	80	120
Target Argument	30	80	120

Table 2: Truncation size (i.e., number of tokens including delimiters) for different stages during training. Note that in the first stage we do not include evidence and keyphrases.

Adding Pre-training. We pre-train a two-layer seq2seq model with OP as input and target argument as output from our training set. After 20 epochs (before converging), parameters for the first layer are used to initialize the first layer of all comparison models and our models (except for the keyphrase decoder). Experimental results show that pre-training boosts all methods by roughly 2 METEOR (Denkowski and Lavie, 2014) points. We describe more detailed results in the supplementary material.

6.3 Baseline and Comparisons

We first consider a RETRIEVAL-based baseline, which concatenates retrieved evidence sentences to form the argument. We further compare with three seq2seq-based generation models with different training data: (1) SEQ2SEQ: training with OP as input and the argument as output; (2) SEQ2SEQ + *encode evd*: augmenting input with evidence sentences as in our model; (3) SEQ2SEQ + *encode KP*: augmenting input with gold-standard keyphrases, which assumes some of the talking points are known. All seq2seq models use a regular beam search decoder with the same beam size as ours.

Variants of Our Models. We experiment with variants of our models based on the proposed separate decoder model (DEC-SEPARATE) or using a shared decoder (DEC-SHARED). For each, we further test whether adding keyphrase attention for argument decoding is helpful (+ *attend KP*).

System vs. Oracle Retrieval. For test time, evidence sentences are retrieved with queries constructed from OP (*System Retrieval*). We also experiment with an *Oracle Retrieval* setup, where the evidence is retrieved based on user arguments, to indicate how much gain can be expected with better retrieval results.

7 Results

7.1 Automatic Evaluation

For automatic evaluation, we use BLEU (Papineni et al., 2002), an n-gram precision-based metric (up to bigrams are considered), and METEOR (Denkowski and Lavie, 2014), measuring unigram recall and precision by considering paraphrases, synonyms, and stemming. Human arguments are used as the gold-standard. Because each OP may be paired with more than one high-quality arguments, we compute BLEU and METEOR scores for the system argument compared against all arguments, and report the best. We do not use multiple reference evaluation because

	w/ System Retrieval			*w/ Oracle Retrieval*		
	BLEU	**MTR**	**Len**	**BLEU**	**MTR**	**Len**
Baseline						
RETRIEVAL	15.32	**12.19**	151.2	10.24	**16.22**	132.7
Comparisons						
SEQ2SEQ	10.21	5.74	34.9	7.44	5.25	31.1
+ *encode evd*	18.03	7.32	67.0	13.79	10.06	68.1
+ *encode KP*	21.94	8.63	74.4	12.96	10.50	78.2
Our Models						
DEC-SHARED	21.22	8.91	69.1	15.78	11.52	68.2
+ *attend KP*	**24.71**	10.05	74.8	11.48	10.08	40.5
DEC-SEPARATE	24.24	10.63	88.6	17.48	13.15	86.9
+ *attend KP*	24.52	11.27	88.3	**17.80**	13.67	86.8

Table 3: Results on argument generation by BLEU and METEOR (MTR), with system retrieved evidence and oracle retrieval. The best performing model is highlighted in **bold** per metric. Our separate decoder models, with and without keyphrase attention, statistically significantly outperform all seq2seq-based models based on approximation randomization testing (Noreen, 1989), $p < 0.0001$.

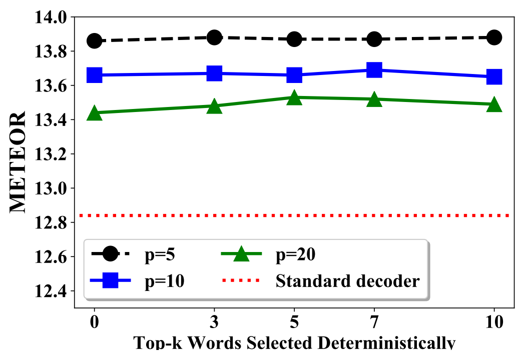


Figure 3: Effect of our reranking-based decoder. Beams are reranked at every 5, 10, and 20 steps (p). For each step size, we also show the effect of varying k, where top-k words are selected deterministically for beam expansion, with $10 - k$ randomly sampled over multinomial distribution after removing the k words. Reranking with smaller step size yields better results.

the arguments are often constructed from different angles and cover distinct aspects of the issue. For models that generate more than one arguments based on different sets of sampled evidence, the one with the highest score is considered.

As can be seen from Table 3, our models produce better BLEU scores than almost all the comparisons. Especially, our models with separate decoder yield significantly higher BLEU and METEOR scores than all seq2seq-based models (approximation randomization testing, $p < 0.0001$) do. Better METEOR scores are achieved by the RETRIEVAL baseline, mainly due to its significantly longer arguments.

Moreover, utilizing attention over both input and the generated keyphrases further boosts our models' performance. Interestingly, utilizing system retrieved evidence yields better BLEU scores than using oracle retrieval for testing. The reason could be that arguments generated based on system retrieval contain less topic-specific words and more generic argumentative phrases. Since the later is often observed in human written arguments, it may lead to higher precision and thus better BLEU scores.

Decoder Strategy Comparison. We also study the effect of our reranking-based decoder by varying the reranking step size (p) and the number of top words expanded to beam hypotheses deterministically (k). From the results in Figure 3, we find that reranking with a smaller step size, e.g., $p = 5$, can generally lead to better METEOR scores. Although varying the number of top words for beam expansion does not yield significant difference, we do observe more diverse beams from the system output if more candidate words are selected stochastically (i.e. with a smaller k).

7.2 Topic-Relevance Evaluation

During our pilot study, we observe that generic arguments, such as "I don't agree with you" or "this is not true", are prevalent among generations by seq2seq models. We believe that good arguments should include content that addresses the given topic. Therefore, we design a novel evaluation method to measure whether the generated arguments contain topic-relevant information.

To achieve the goal, we first train a topic-relevance estimation model inspired by the latent semantic model in Huang et al. (2013). A pair of OP and argument, each represented as the average of word embeddings, are separately fed into a two-layer transformation model. A dot-product is computed over the two projected low-dimensional vectors, and then a sigmoid function outputs the relevance score. For model learning, we further divide our current training data into training, developing, and test sets. For each OP and argument pair, we first randomly sample 100 arguments from other threads, and then pick the top 5 dissimilar ones, measured by Jaccard distance, as negative training samples. This model achieves a Mean Reciprocal Rank (MRR) score of 0.95 on the test set. Descriptions about model formulation and related training

	Standard Decoder		Our Decoder	
	MRR	P@1	MRR	P@1
Baseline				
RETRIEVAL	81.08	65.45	-	-
Comparisons				
SEQ2SEQ	75.29	58.85	74.46	57.06
+ *encode evd*	83.73	71.59	88.24	78.76
Our Models				
DEC-SHARED	79.80	65.57	**95.18**	**90.91**
+ *attend KP*	**94.33**	**89.76**	93.48	87.91
DEC-SEPARATE	86.85	76.74	91.70	84.72
+ *attend KP*	88.53	79.05	92.77	86.46

Table 4: Evaluation on topic relevance—models that generate arguments highly related with OP should be ranked high by a separately trained relevance estimation model, i.e., higher Mean Reciprocal Rank (MRR) and Precision at 1 (P@1) scores. All models trained with evidence significantly outperform seq2seq trained without evidence (approximation randomization testing, $p < 0.0001$).

details are included in the supplementary material.

We then take this trained model to evaluate the relevance between OP and the corresponding system arguments. Each system argument is treated as positive sample; we then select five negative samples from arguments generated for other OPs whose evidence sentences most similar to that of the positive sample. Intuitively, if an argument contains more topic relevant information, then the relevance estimation model will output a higher score for it; otherwise, the argument will receive a lower similarity score, and thus cannot be easily distinguished from negative samples. Ranking metrics of MRR and Precision at 1 (P@1) are utilized, with results reported in Table 4. The ranker yields significantly better scores over arguments generated from models trained with evidence, compared to arguments generated by SEQ2SEQ model.

Moreover, we manually pick 29 commonly used generic responses (e.g., "I don't think so") and count their frequency in system outputs. For the seq2seq model, more than 75% of its outputs contain at least one generic argument, compared to 16.2% by our separate decoder model with attention over keyphrases. This further implies that our model generates more topic-relevant content.

7.3 Human Evaluation

We also hire three trained human judges who are fluent English speakers to rate system arguments for the following three aspects on a scale of 1 to 5 (with 5 as best): *Grammaticality*—whether an argument is fluent, *informativeness*—whether the argument contains useful information and is not generic, and *relevance*—whether the argument contains information of a different stance or off-topic. 30 CMV threads are randomly selected, each of which is presented with randomly-shuffled OP statement and four system arguments.

System	Gram	Info	Rel
RETRIEVAL	**4.5** $\pm$ 0.6	**3.7** $\pm$ 0.9	**3.3** $\pm$ 1.1
SEQ2SEQ	3.3 $\pm$ 1.1	1.2 $\pm$ 0.5	1.4 $\pm$ 0.7
OUR MODEL	2.5 $\pm$ 0.8	1.6 $\pm$ 0.8	1.8 $\pm$ 0.8

Table 5: Human evaluation results on grammaticality (**Gram**), informativeness (**Info**), and relevance (**Rel**) of arguments. Our model with separate decoder and attention over keyphrases receives significantly better ratings in informativeness and relevance than seq2seq (one-way ANOVA, $p < 0.005$).

Table 5 shows that our model with separate decoder and attention over keyphrases produce significantly more informative and relevant arguments than seq2seq trained without evidence.[8] However, we also observe that human judges prefer the retrieved arguments over generation-based models, illustrating the gap between system arguments and human edited text. Sample arguments are displayed in Figure 4.

8 Further Discussion

Keyphrase Generation Analysis. Here we provide further analysis over the generated keyphrases by our separate decoder model. First, about 10% of the keyphrases output by our model also appear in the gold-standard (i.e., used by human arguments). Furthermore, 36% of generated keyphrases are reused by our system arguments. With human inspection, we find that although some keyphrases are not directly reused by the argument decoder, they represent high level talking points in the argument. For instance, in the first sample argument by our model in Figure 4, keyphrases "the motive" and "russian" are generated. Although not used, they suggest the topics that the argument should stay on.

Sample Arguments and Future Directions. As can be seen from the sample outputs in Figure 4, our model generally captures more relevant concepts, e.g., "military army" and "wars

[8]Inter-rater agreement scores for these three aspects are 0.50, 0.60, and 0.48 by Krippendorff's α.

Input: Putin is trying to re-form a “Soviet Union” with his past actions in Georgia and current actions in Ukraine
Human: So far Putin has only really targeted areas where the local population is majority russian (or historically majority russian) and sympathetic to russia. That’s not to say that he wouldn’t go further, but I don’t think there is yet any compelling reason to believe he would.
Retrieval: In 1968, the Soviet Union and Warsaw pact allies invaded Czechoslovakia to halt the Prague spring reforms. It was this significant lack of flexibility that was exploited by the US to undermine the strength of the Soviet Union and thus foster its reform. In the summer of 1991, the new union treaty, which would have turned the Soviet Union into a much looser union, was agreed upon by eight republics.
Our Model: There are two reasons you are so far beyond. There is no reason to see the military army. You can infer what they assume it, so they tend to protect up. There. If we assume it were bad, they can not be controlled. So we’re talking going to ignore the wars of the world. The fact of the matter is not bad. I would also assume it. However, the government can not be reminded of the world.
Seq2seq + evd: 1. The reason is that the government isn’t the power of the war of Russia, at the time of the war of Russia. In Russia, there is no way to continue to control the military.
Seq2seq + KP: I don’t think it’s a bit. I don’t think you’re forgetting. I don’t want to see the Syrian American dream in the US. I don’t see the Syrian American dream of American citizens. I don’t see the Syrian American dream of American citizens. I don’t see the Syrian American military in the world. I don’t think it wouldn’t be true.

Input: Voting for either Hillary or Trump is morally reprehensible.
Human: I don’t think a vote for either of them means ”I’m totally fine with...” as you say. Why is it not acceptable to pick the option whose platform most closely aligns to your values/beliefs/desires, while acknowledging neither candidate is ideal?
Retrieval: Vote trading under a democratic, majority-rule institution is sometimes considered morally reprehensible behavior.
Our Model: He’s going to happen. You’re overestimating the more premise. He’s already overestimating to this hypothetical party. He’s already killed over the supreme court positions. Looking for Bernie. This is the same reasons: Clinton is not the supreme court presidency. You’ve already sent Trump’s right to vote. The fact is that he’s going to happen. He’s already going to notice. However, it’s a long way.
Seq2seq + evd: 1. If they don’t vote the vote, they would not be able to vote for any candidate. They don’t have the same effect on their political power. They are not voting for them.
Seq2seq + KP: I agree with your view. I don’t agree with you. I don’t think it’s easy to appeal to the Clintons. If you don’t want to do this?

Figure 4: Sample arguments generated by human, our system, and seq2seq trained with evidence. Only the main thesis is shown for the input OP. System generations are manually detokenized and capitalized.

of the world”, as discussed in the first example. Meanwhile, our model also acquires argumentative style language, though there is still a noticeable gap between system arguments and human constructed arguments. As discovered by our prior work (Wang et al., 2017), both topical content and language style are essential elements for high quality arguments. For future work, generation models with a better control on linguistic style need to be designed. As for improving coherence, we believe that discourse-aware generation models (Ji et al., 2016) should also be explored in the future work to enhance text planning.

9 Related Work

There is a growing interest in argumentation mining from the natural language processing research community (Park and Cardie, 2014; Ghosh et al., 2014; Palau and Moens, 2009; Niculae et al., 2017; Eger et al., 2017). While argument understanding has received increasingly more attention, the area of automatic argument generation is much less studied. Early work on argument construction investigates the design of argumentation strategies (Reed et al., 1996; Carenini and Moore, 2000; Zukerman et al., 2000). For instance, Reed (1999) describes the first full natural language argument generation system, called Rhetorica. It however only outputs a text plan, mainly relying on heuristic rules. Due to the difficulty of text generation, none of the previous work represents a fully automated argument generation system. This work aims to close the gap by proposing an end-to-end trained argument construction framework.

Additionally, argument retrieval and extraction are investigated (Rinott et al., 2015; Hua and Wang, 2017) to deliver relevant arguments for user-specified queries. Wachsmuth et al. (2017) build a search engine from arguments collected from various online debate portals. After the retrieval step, sentence ordering algorithms are often applied to improve coherence (Sato et al., 2015; Reisert et al., 2015). Nevertheless, simply merging arguments from different resources inevitably introduces redundancy. To the best of our knowledge, this is the first automatic argument generation system that can synthesize retrieved content from different articles into fluent arguments.

10 Conclusion

We studied the novel problem of generating arguments of a different stance for a given statement. We presented a neural argument generation framework enhanced with evidence retrieved from Wikipedia. Separate decoders were designed to first produce a set of keyphrases as talking points, and then generate the final argument. Both automatic evaluation against human arguments and human assessment showed that our model produced more informative arguments than popular sequence-to-sequence-based generation models.

Acknowledgements

This work was partly supported by National Science Foundation Grant IIS-1566382, and a GPU gift from Nvidia. We thank three anonymous reviewers for their insightful suggestions on various aspects of this work.

References

Steffen Albrecht. 2006. Whose voice is heard in online deliberation?: A study of participation and representation in political debates on the internet. *Information, Community and Society* 9(1):62–82.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 502–512. http://www.aclweb.org/anthology/D10-1049.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering* 14(4):431–455.

Blai Bonet and Hector Geffner. 1996. Arguing for decisions: A qualitative model of decision making. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pages 98–105.

Nadjet Bouayad-Agha, Gerard Casamayor, and Leo Wanner. 2011. Content selection from an ontology-based knowledge base for the generation of football summaries. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Nancy, France, pages 72–81. http://www.aclweb.org/anthology/W11-2810.

James P Byrnes. 2013. *The nature and development of decision-making: A self-regulation model*. Psychology Press.

Giuseppe Carenini and Johanna Moore. 2000. A strategy for generating evaluative arguments. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*. Association for Computational Linguistics, Mitzpe Ramon, Israel, pages 47–54. https://doi.org/10.3115/1118253.1118261.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1870–1879. http://aclweb.org/anthology/P17-1171.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 376–380. http://www.aclweb.org/anthology/W14-3348.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 11–22. http://aclweb.org/anthology/P17-1002.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*. pages 1019–1027.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland, pages 39–48. http://www.aclweb.org/anthology/W14-2106.

Eduard H Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial intelligence* 63(1-2):341–385.

Xinyu Hua and Lu Wang. 2017. Understanding and detecting supporting arguments of diverse types. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 203–208. http://aclweb.org/anthology/P17-2032.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, pages 2333–2338.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 332–342. http://www.aclweb.org/anthology/N16-1037.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 495–501.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 720–730. http://www.aclweb.org/anthology/N16-1086.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured svms and rnns. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 985–995. http://aclweb.org/anthology/P17-1091.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*. ACM, pages 98–107.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland, pages 29–38. http://www.aclweb.org/anthology/W14-2105.

Joonsuk Park, Sally Klingel, Claire Cardie, Mary Newhart, Cynthia Farina, and Joan-Josep Vallbé. 2012. Facilitative moderation for online participation in erulemaking. In *Proceedings of the 13th Annual International Conference on Digital Government Research*. ACM, pages 173–182.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Chris Reed. 1999. The role of saliency in generating natural language arguments. In *IJCAI*. pages 876–883.

Chris Reed, Derek Long, and Maria Fox. 1996. An architecture for argumentative dialogue planning. In *International Conference on Formal and Applied Practical Reasoning*. Springer, pages 555–566.

Paul Reisert, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. 2015. A computational approach for generating toulmin model argumentation. In *Proceedings of the 2nd Workshop on Argumentation Mining*. Association for Computational Linguistics, Denver, CO, pages 45–55. http://www.aclweb.org/anthology/W15-0507.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 440–450. http://aclweb.org/anthology/D15-1050.

Misa Sato, Kohsuke Yanai, Toshinori Miyoshi, Toshihiko Yanase, Makoto Iwayama, Qinghua Sun, and Yoshiki Niwa. 2015. End-to-end argument generation system in debating. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*. Association for Computational Linguistics and The Asian Federation of Natural Language Processing, Beijing, China, pages 109–114. http://www.aclweb.org/anthology/P15-4019.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*. Association for Computational Linguistics, Copenhagen, Denmark, pages 49–59. http://www.aclweb.org/anthology/W17-5106.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *Transactions of the Association for Computational Linguistics* 5:219–232. https://transacl.org/ojs/index.php/tacl/article/view/1009.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 47–57. http://www.aclweb.org/anthology/N16-1007.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1711–1721. http://aclweb.org/anthology/D15-1199.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1296–1306. https://aclweb.org/anthology/D16-1137.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2253–2263. https://www.aclweb.org/anthology/D17-1239.

Ingrid Zukerman, Richard McConachy, and Sarah George. 2000. Using argumentation strategies in automated argument generation. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*. Association for Computational Linguistics, Mitzpe Ramon, Israel, pages 55–62. https://doi.org/10.3115/1118253.1118262.

A Stylometric Inquiry into Hyperpartisan and Fake News

Martin Potthast **Johannes Kiesel** **Kevin Reinartz** **Janek Bevendorff** **Benno Stein**

Leipzig University
martin.potthast@uni-leipzig.de

Bauhaus-Universität Weimar
<first>.<last>@uni-weimar.de

Abstract

We report on a comparative style analysis of hyperpartisan (extremely one-sided) news and fake news. A corpus of 1,627 articles from 9 political publishers, three each from the mainstream, the hyperpartisan left, and the hyperpartisan right, have been fact-checked by professional journalists at BuzzFeed: 97% of the 299 fake news articles identified are also hyperpartisan. We show how a style analysis can distinguish hyperpartisan news from the mainstream ($F_1 = 0.78$), and satire from both ($F_1 = 0.81$). But stylometry is no silver bullet as style-based fake news detection does not work ($F_1 = 0.46$). We further reveal that left-wing and right-wing news share significantly more stylistic similarities than either does with the mainstream. This result is robust: it has been confirmed by three different modeling approaches, one of which employs Unmasking in a novel way. Applications of our results include partisanship detection and pre-screening for semi-automatic fake news detection.

1 Introduction

The media and the public are currently discussing the recent phenomenon of “fake news” and its potential role in swaying elections, how it may affect society, and what can and should be done about it. Prone to misunderstanding and misue, the term “fake news” arose from the observation that, in social media, a certain kind of ‘news’ spreads much more successfully than others, and this kind of ‘news’ is typically extremely one-sided (hyperpartisan), inflammatory, emotional, and often riddled with untruths. Although traditional yellow press has been spreading ‘news’ of varying degrees of truthfulness long before the digital revolution, its amplification over *real* news within social media gives many people pause. The fake news hype caused a widespread disillusionment about social media, and many politicians, news publishers, IT companies, activists, and scientists concur that this is where to draw the line. For all their good intentions, however, it must be drawn very carefully (if at all), since nothing less than free speech is at stake—a fundamental right of every free society.

Many favor a two-step approach where fake news items are detected and then countermeasures are implemented to foreclose rumors and to discourage repetition. While some countermeasures are already tried in practice, such as displaying warnings and withholding ad revenue, fake news detection is still in its infancy. At any rate, a near-real time reaction is crucial: once a fake news item begins to spread virally, the damage is done and undoing it becomes arduous. Since knowledge-based and context-based approaches to fake news detection can only be applied after publication, i.e., as news events unfold and as social interactions occur, they may not be fast enough.

We have identified style-based approaches as a viable alternative, allowing for instantaneous reactions, albeit not to fake news, but to hyperpartisanship. In this regard we contribute (1) a large news corpus annotated by experts with respect to veracity and hyperpartisanship, (2) extensive experiments on discriminating fake news, hyperpartisan news, and satire based solely on writing style, and (3) validation experiments to verify our finding that the writing style of the left and the right have more in common than any of the two have with the mainstream, applying Unmasking in a novel way.

After a review of related work, Section 3 details the corpus and its construction, Section 4 introduces our methodology, and Section 5 reports the results of the aforementioned experiments.

2 Related Work

Approaches to fake news detection divide into three categories (Figure 1): they can be knowledge-based (by relating to known facts), context-based (by analyzing news spread in social media), and style-based (by analyzing writing style).

Knowledge-based fake news detection. Methods from information retrieval have been proposed early on to determine the veracity of web documents. For example, Etzioni et al. (2008) propose to identify inconsistencies by matching claims extracted from the web with those of a document in question. Similarly, Magdy and Wanas (2010) measure the frequency of documents that support a claim. Both approaches face the challenges of web data credibility, namely expertise, trustworthiness, quality, and reliability (Ginsca et al., 2015).

Other approaches rely on knowledge bases, including the semantic web and linked open data. Wu et al. (2014) "perturb" a claim in question to query knowledge bases, using the result variations as indicator of the support a knowledge base offers for the claim. Ciampaglia et al. (2015) use the shortest path between concepts in a knowledge graph, whereas Shi and Weninger (2016) use a link prediction algorithm. However, these approaches are unsuited for new claims without corresponding entries in a knowledge base, whereas knowledge bases can be manipulated (Heindorf et al., 2016).

Context-based fake news detection. Here, fake news items are identified via meta information and spread patterns. For example, Long et al. (2017) show that author information can be a useful feature for fake news detection, and Derczynski et al. (2017) attempt to determine the veracity of a claim based on the conversation it sparks on Twitter as one of the RumourEval tasks. The Facebook analysis of Mocanu et al. (2015) shows that unsubstantiated claims spread as widely as well-established ones, and that user groups predisposed to conspiracy theories are more open to sharing the former. Similarly, Acemoglu et al. (2010), Kwon et al. (2013), Ma et al. (2017), and Volkova et al. (2017) model the spread of (mis-)information, while Budak et al. (2011) and Nguyen et al. (2012) propose algorithms to limit its spread. The efficacy of countermeasures like debunking sites is studied by Tambuscio et al. (2015). While achieving good results, context-based approaches suffer from working only a posteriori, requiring large amounts of data, and disregarding the actual news content.

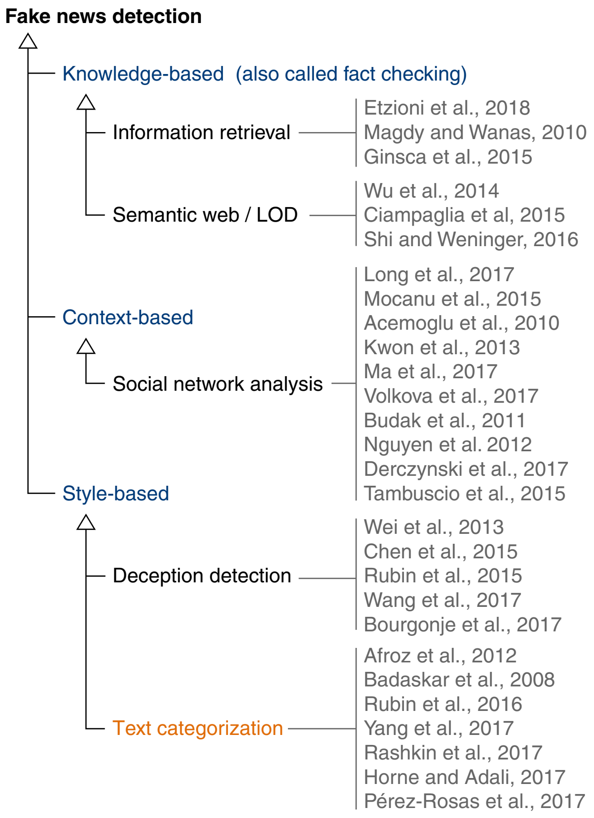


Figure 1: Taxonomy of paradigms for fake news detection alongside a selection of related work.

Style-based fake news detection. Deception detection originates from forensic linguistics and builds on the Undeutsch hypothesis—a result from forensic psychology which asserts that memories of real-life, self-experienced events differ in content and quality from imagined events (Undeutsch, 1967). The hypothesis led to the development of forensic tools to assess testimonies at the statement level. Some approaches operationalize deception detection at scale to detect uncertainty in social media posts, for example Wei et al. (2013) and Chen et al. (2015). In this regard, Rubin et al. (2015) use rhetorical structure theory as a measure of story coherence and as an indicator for fake news. Recently, Wang (2017) collected a large dataset consisting of sentence-length statements along their veracity from the fact-checking site PolitiFact.com, and then used style features to detect false statements. A related task is stance detection, where the goal is to detect the relation between a claim about an article, and the article itself (Bourgonje et al., 2017). Most prominently, stance detection was the task of the Fake News Challenge[1] which ran in 2017 and received 50 submissions, albeit hardly any participants published their approach.

[1]http://www.fakenewschallenge.org/

Where deception detection focuses on single statements, style-based text categorization as proposed by Argamon-Engelson et al. (1998) assesses entire texts. Common applications are author profiling (age, gender, etc.) and genre classification. Though susceptible to authors who can modify their writing style, such obfuscations may be detectable (e.g., Afroz et al. (2012)). As an early precursor to fake news detection, Badaskar et al. (2008) train models to identify news items that were automatically generated. Currently, text categorization methods for fake news detection focus mostly on satire detection (e.g., Rubin et al. (2016), Yang et al. (2017)). Rashkin et al. (2017) perform a statistical analysis of the stylistic differences between real, satire, hoax, and propaganda news. We make use of their results by incorporating the best-performing style features identified.

Finally, two preprint papers have been recently shared. Horne and Adali (2017) use style features for fake news detection. However, the relatively high accuracies reported must be taken with a grain of salt: their two datasets comprise only 70 news articles each, whose ground-truth is based on where an article came from, instead of resulting from a per-article expert review as in our case; their final classifier uses only 4 features (number of nouns, type-token ratio, word count, number of quotes), which can be easily manipulated; and based on their experimental setup, it cannot be ruled out that the classifier simply differentiates news portals rather than fake and real articles. We avoid this problem by testing our classifiers on articles from portals which were not represented in the training data. Similarly, Pérez-Rosas et al. (2017) also report on constructing two datasets comprising around 240 and 200 news article *excerpts* (i.e., the 5-sentence lead) with a balanced distribution of fake vs. real. The former was collected via crowdsourcing, asking workers to write a fake news item based on a real news item, the latter was collected from the web. For style analysis, the former dataset may not be suitable, since the authors note themselves that "workers succeeded in mimicking the reporting style from the original news". The latter dataset encompasses only celebrity news (i.e., yellow press), which introduces a bias. Their feature selection follows that of Rubin et al. (2016), which is covered by our experiments, but also incorporates topic features, rendering the resulting classifier not generalizable.

3 The BuzzFeed-Webis Fake News Corpus

This section introduces the BuzzFeed-Webis Fake News Corpus 2016, detailing its construction and annotation by professional journalists employed at BuzzFeed, as well as key figures and statistics.[2]

3.1 Corpus Construction

The corpus encompasses the output of 9 publishers on 7 workdays close to the US presidential elections 2016, namely September 19 to 23, 26, and 27. Table 1 gives an overview. Among the selected publishers are six prolific hyperpartisan ones (three left-wing and three right-wing), and three mainstream ones. All publishers earned Facebook's blue checkmark ✓, indicating authenticity and an elevated status within the network. Every post and linked news article has been fact-checked by 4 BuzzFeed journalists, including about 19% of posts forwarded from third parties. Having checked a total of 2,282 posts, 1,145 mainstream, 471 left-wing, and 666 right-wing, Silverman et al. (2016) reported key insights as a data journalism article. The annotations were published alongside the article.[3] However, this data only comprises URLs to the original Facebook posts. To construct our corpus, we archived the posts, the linked articles, and attached media as well as relevant meta data to ensure long-term availability. Due to the rapid pace at which the publishers change their websites, we were able to recover only 1,627 articles, 826 mainstream, 256 left-wing, and 545 right-wing.

Manual fact-checking. A binary distinction between fake and real news turned out to be infeasible, since hardly any piece of fake news is entirely false, and pieces of real news may not be flawless. Therefore, posts were rated "mostly true," "mixture of true and false," "mostly false," or, if the post was opinion-driven or otherwise lacked a factual claim, "no factual content." Four BuzzFeed journalists worked on the manual fact-checks of the news articles: to minimize costs, each article was reviewed only once and articles were assigned round robin. The ratings "mixture of true and false" and "mostly false" had to be justified, and, when in doubt about a rating, a second opinion was collected, whereas disagreements were resolved by a third one. Finally, all news rated "mostly false" underwent a final check to ensure the rating was justified, lest the respective publishers would contest it.

[2]Corpus download: https://doi.org/10.5281/zenodo.1239675
[3]http://github.com/BuzzFeedNews/2016-10-facebook-fact-check

The journalists were given the following guidance:

Mostly true: The post and any related link or image are based on factual information and portray it accurately. The authors may interpret the event/info in their own way, so long as they do not misrepresent events, numbers, quotes, reactions, etc., or make information up. This rating does not allow for unsupported speculation or claims.

Mixture of true and false (mix, for short): Some elements of the information are factually accurate, but some elements or claims are not. This rating should be used when speculation or unfounded claims are mixed with real events, numbers, quotes, etc., or when the headline of the link being shared makes a false claim but the text of the story is largely accurate. It should also only be used when the unsupported or false information is roughly equal to the accurate information in the post or link. Finally, use this rating for news articles that are based on unconfirmed information.

Mostly false: Most or all of the information in the post or in the link being shared is inaccurate. This should also be used when the central claim being made is false.

No factual content (n/a, for short): This rating is used for posts that are pure opinion, comics, satire, or any other posts that do not make a factual claim. This is also the category to use for posts that are of the "Like this if you think..." variety.

3.2 Limitations

Given the significant workload (i.e., costs) required to carry out the aforementioned annotations, the corpus is restricted to the given temporal period and biased toward the US culture and political landscape, comprising only English news articles from a limited number of publishers. Annotations were recorded at the article level, not at statement level. For text categorization, this is sufficient. At the time of writing, our corpus is the largest of its kind that has been annotated by professional journalists.

3.3 Corpus Statistics

Table 1 shows the fact-checking results and some key statistics per article. Unsurprisingly, none of the mainstream articles are mostly false, whereas 8 across all three publishers are a mixture of true and false. Disregarding non-factual articles, a little more than a quarter of all hyperpartisan left-wing articles were found faulty: 15 articles mostly false, and 51 a mixture of true and false. Publisher "The Other 98%" sticks out by achieving an almost perfect score. By contrast, almost 45% of the right-wing articles are a mixture of true and false (153) or mostly false (72). Here, publisher "Right Wing News" sticks out by supplying more than half of mixtures of true and false alone, whereas mostly false articles are equally distributed.

Orientation	**Fact-checking results**					**Key statistics per article**				
Publisher	true	mix	false	n/a	Σ	Paras.	Links		Words	
							extern	all	quoted	all
Mainstream	806	8	0	12	826	20.1	2.2	3.7	18.1	692.0
ABC News	90	2	0	3	95	21.1	1.0	4.8	21.0	551.9
CNN	295	4	0	8	307	19.3	2.4	2.5	15.3	588.3
Politico	421	2	0	1	424	20.5	2.3	4.3	19.9	798.5
Left-wing	182	51	15	8	256	14.6	4.5	4.9	28.6	423.2
Addicting Info	95	25	8	7	135	15.9	4.4	4.5	30.5	430.5
Occupy Democrats	55	23	6	0	91	10.9	4.1	4.7	29.0	421.7
The Other 98%	32	3	1	1	30	20.2	6.4	7.2	21.2	394.5
Right-wing	276	153	72	44	545	14.1	2.5	3.1	24.6	397.4
Eagle Rising	107	47	25	36	214	12.9	2.6	2.8	17.3	388.3
Freedom Daily	48	24	22	4	99	14.6	2.2	2.3	23.5	419.3
Right Wing News	121	82	25	4	232	15.0	2.5	3.6	33.6	396.6
Σ	1264	212	87	64	1627	17.2	2.7	3.7	20.6	551.0

Table 1: The BuzzFeed-Webis Fake News Corpus 2016 at a glance ("Paras." short for "paragraphs").

Regarding key statistics per article, it is interesting that the articles from all mainstream publishers are on average about 20 paragraphs long with word counts ranging from 550 words on average at ABC News to 800 at Politico. Except for one publisher, left-wing articles and right-wing articles are shorter on average in terms of paragraphs as well as word count, averaging at about 420 words and 400 words, respectively. Left-wing articles quote on average about 10 words more than the mainstream, and right-wing articles 6 words more. When articles comprise links, they are usually external ones, whereas ABC News rather uses internal links, and only half of the links found at Politico articles are external. Left-wing news articles stick out by containing almost double the amount of links across publishers than mainstream and right-wing ones.

3.4 Operationalizing Fake News

In our experiments, we operationalize the category of fake news by joining the articles that were rated mostly false with those rated a mixture of true and false. Arguably, the latter may not be exactly what is deemed "fake news" (as in: a complete fabrication), however, practice shows fake news are hardly ever devoid of truth. More often, true facts are misconstrued or framed badly. In our experiments, we hence call mostly true articles real news, mostly false plus mixtures of true and false—except for satire—fake news, and disregard all articles rated non-factual.

4 Methodology

This section covers our methodology, including our feature set to capture writing style, and a brief recap of Unmasking by Koppel et al. (2007), which we employ for the first time to distinguish genre styles as opposed to author styles. For sake of reproducibility, all our code has been published.[4]

4.1 Style Features and Feature Selection

Our writing style model incorporates common features as well as ones specific to the news domain. The former are n-grams, n in $[1, 3]$, of characters, stop words, and parts-of-speech. Further, we employ 10 readability scores[5] and dictionary features, each indicating the frequency of words from a tailor-made dictionary in a document, using the General Inquirer Dictionaries as a basis (Stone et al., 1966). The domain-specific features include ratios of quoted words and external links, the number of paragraphs, and their average length.

In each of our experiments, we carefully select from the aforementioned features the ones worthwhile using: all features are discarded that are hardly represented in our corpus, namely word tokens that occur in less than 2.5% of the documents, and n-gram features that occur in less than 10% of the documents. Discarding these features prevents overfitting and improves the chances that our model will generalize.

If not stated otherwise, our experiments share a common setup. In order to avoid biases from the respective training sets, we balance them using oversampling. Furthermore, we perform 3-fold cross-validation where each fold comprises one publisher from each orientation, so that the classifier does not learn a publisher's style. For non-Unmasking experiments we use WEKA's random forest implementation with default settings.

4.2 Unmasking Genre Styles

Unmasking, as proposed by Koppel et al. (2007), is a meta learning approach for authorship verification. We study for the first time whether it can be used to assess the similarity of more broadly defined style categories, such as left-wing vs. right-wing vs. mainstream news. This way, we uncover relations between the writing styles that people may involuntarily adopt as per their political orientation.

[4]Code download: http://www.github.com/webis-de/ACL-18

[5]Automated Readability Index, Coleman Liau Index, Flesh Kincaid Grade Level and Reading Ease, Gunning Fog Index, LIX, McAlpine EFLAW Score, RIX, SMOG Grade, Strain Index

Originally, Unmasking takes two documents as input and outputs its confidence whether they have been written by the same author. Three steps are taken to accomplish this: first, each document is chunked into a set of at least 500-word long chunks; second, classification errors are measured while iteratively removing the most discriminative features of a style model consisting of the 250 most frequent words, separating the two chunk sets with a linear classifier; and third, the resulting classification accuracy curves are analyzed with regard to their slope. A steep decrease is more likely than a shallow decrease if the two documents have been written by the same author, since there are presumably less discriminating features between documents written by the same author than between documents written by different authors. Training a classifier on many examples of error curves obtained from same-author document pairs and different-author document pairs yields an effective authorship verifier—at least for long documents that can be split up into a sufficient number of chunks.

It turns out that what applies to the style of authors also applies to genre styles. We adapt Unmasking by skipping its first step and using two sets of documents (e.g., left-wing articles and right-wing articles) as input. When plotting classification error curves for visual inspection, steeper decreases in these plots, too, indicate higher style similarity of the two input document sets, just as with chunk sets of two documents written by the same author.

4.3 Baselines

We employ four baseline models: a topic-based bag of words model, often used in the literature, but less practical since news topics change frequently and drastically; a model using only the domain-specific news style features to check whether the differences between categories measured as corpus statistics play a significant role; and naive baselines that classify all items into one of the categories in question, relating our results to the class distributions.

4.4 Performance Measures

Classification performance is measured as accuracy, and class-wise precision, recall, and F_1. We favor these measures over, e.g., areas under the ROC curve or the precision recall curve for simplicity sake. Also, the tasks we are tackling are new, so that little is known to date about user preferences. This is also why we chose the evenly-balanced F_1.

5 Experiments

We report on the results of two series of experiments that investigate style differences and similarities between hyperpartisan and mainstream news, and between fake, real, and satire news, shedding light on the following questions:

1. Can (left/right) hyperpartisanship be distinguished from the mainstream?
2. Is style-based fake news detection feasible?
3. Can fake news be distinguished from satire?

Our first experiment addressing the first question uncovered an odd behavior of our classifier: it would often misjudge left-wing for right-wing news, while being much better at distinguishing both combined from the mainstream. To explain this behavior, we hypothesized that *maybe* the writing style of the hyperpartisan left and right are more similar to one another than to the mainstream. To investigate this hypothesis, we devised two additional validation experiments, yielding three sources of evidence instead of just one.

Features	**Accuracy**	**Precision**			**Recall**			**F_1**		
	all	left	right	main.	left	right	main.	left	right	main.
Style	0.60	0.21	0.56	0.75	0.20	0.59	0.74	0.20	0.57	0.75
Topic	0.64	0.24	0.62	0.72	0.15	0.54	0.86	0.19	0.58	0.79
News style	0.39	0.09	0.35	0.59	0.14	0.36	0.49	0.11	0.36	0.53
All-left	0.16	0.16	-	-	1.00	0.0	0.0	0.27	-	-
All-right	0.33	-	0.33	-	0.0	1.00	0.0	-	0.50	-
All-main.	0.51	-	-	0.51	0.0	0.0	1.00	-	-	0.68

Table 2: Performance of predicting orientation.

Features	**Accuracy**	**Precision**		**Recall**		**F_1**	
	all	hyp.	main.	hyp.	main.	hyp.	main.
Style	0.75	0.69	0.86	0.89	0.62	0.78	0.72
Topic	0.71	0.66	0.79	0.83	0.60	0.74	0.68
News style	0.56	0.54	0.58	0.65	0.47	0.59	0.52
All-hyp.	0.49	0.49	-	1.00	0.0	0.66	-
All-main.	0.51	-	0.51	0.0	1.00	-	0.68

Table 3: Performance of predicting hyperpartisanship.

Features	**Left**		**Right**	
Trained on:	right+main.	all	left+main.	all
Style	0.74	0.90	0.66	0.89
Topic	0.68	0.79	0.48	0.85
News style	0.52	0.61	0.47	0.66

Table 4: Ratio of left articles misclassified right when omitting left articles from training, and vice versa.

5.1 Hyperpartisanship vs. Mainstream

A. Predicting orientation. Table 2 shows the classification performance of a ternary classifier trained to discriminate left, right, and mainstream—an obvious first experiment for our dataset. Separating the left and right orientation from the mainstream does not work too well: the topic baseline outperforms the style-based models with regard to accuracy, whereas the results for class-wise precision and recall are a mixed bag. The left-wing articles are apparently significantly more difficult to be identified compared to articles from the other two orientations. When we inspected the confusion matrix (not shown), it turned out that 66% of misclassifications of left-wing articles are falsely classified as right-wing articles, whereas 60% of all misclassified right-wing articles are classified as mainstream articles. Misclassified mainstream articles spread almost evenly across the other classes.

The poor performance of the domain-specific news style features by themselves demonstrate that orientation cannot be discriminated based on the basic corpus characteristics observed with respect to paragraphs, quotations, and hyperlinks. This holds for all subsequent experiments.

B. Predicting hyperpartisanship. Given the apparent difficulty of telling apart individual orientations, we did not frantically add features or switch classifiers to make it work. Rather, we trained a binary classifier to discriminate hyperpartisanship in general from the mainstream. Table 3 shows the performance values. This time, the best classification accuracy of 0.75 at a remarkable 0.89 recall for the hyperpartisan class is achieved by the style-based classifier, outperforming the topic baseline.

Comparing Table 2 and Table 3, we were left with a riddle: all other things being equal, how could it be that hyperpartisanship in general can be much better discriminated from the mainstream than individual orientation? Attempts to answer this question gave rise to our aforementioned hypothesis that, perhaps, the writing style of hyperpartisan left and right are not altogether different, despite their opposing agendas. Or put another way, if style and topic are orthogonal concepts, then being an extremist should not exert a different style dependent on political orientation. Excited, we sought ways to *independently* disprove the hypothesis, and found two: Experiments C and D.

C. Validation using leave-out classification. If left-wing and right-wing articles have a more similar style than either of them compared to mainstream articles, then what class would a binary classifier assign to a left-wing article, if it were trained to distinguish only the right-wing from the mainstream, and vice versa? Table 4 shows the results of this experiment. As indicated by proportions well above 0.50, full style-based classifiers have a tendency of clas-

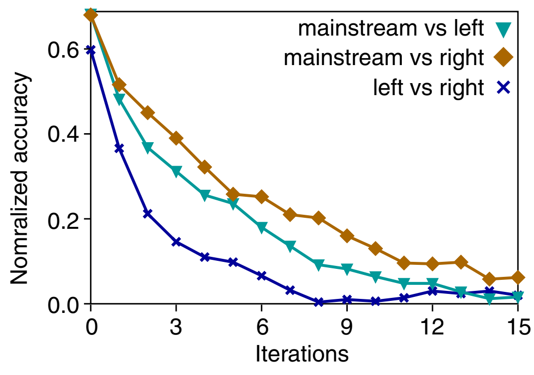


Figure 2: Unmasking applied to pairs of political orientations. The steeper a curve, the more similar the respective styles.

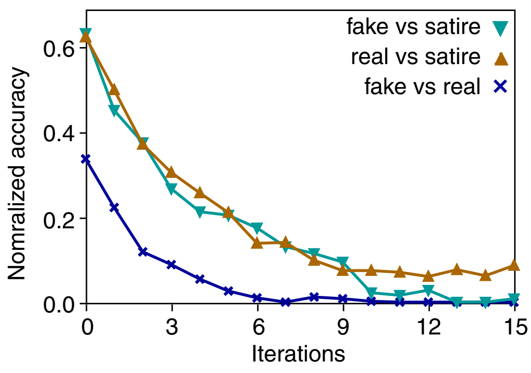


Figure 3: Unmasking applied to pairs of sets of news that are fake, real, and satire.

sifying left as right and right as left. The topic baseline, though, gets confused especially when omitting right articles from the training set with performance close to random. The fact that the topic baseline works better when omitting left from the training set may be explainable: leading up to the elections, the hyperpartisan left was often merely reacting to topics prompted by the hyperpartisan right, instead of bringing up their own.

D. Validation using Unmasking. Based on Koppel et al.'s original approach in the context of authorship verification, for the first time, we generalize Unmasking to assess genre styles: just like author style similarity, genre style similarity will be characterized by the slope of a given Unmasking curve, where a steeper decrease indicates higher similarity. We apply Unmasking as described in Section 4.2 onto pairs of sets of left, right, and mainstream articles. Figure 2 shows the resulting Unmasking curves (Unmasking is symmetrical, hence three curves). The curves are averaged over 5 runs, where each run comprised sets of 100 articles from each orientation. In case of the left-wing orientation, where less than 500 articles are available in our corpus, once all of them had been used, they were shuffled again to select articles for the remainder of the runs. As can be seen, the curve comparing left vs. right has a distinctly steeper slope than either of the others. This result hence matches the findings of the previous experiments.

With caution, we conclude that the evidence gained from our three independent experimental setups supports our hypothesis that the hyperpartisan left and the hyperpartisan right have more in common in terms of writing style than any of the two have with the mainstream. Another more tangible (e.g., practical) outcome of Experiment B is the finding that hyperpartisan news can apparently be discriminated well from the mainstream: in particular the high recall of 0.89 at a reasonable precision of 0.69 gives us confidence that, with some further effort, a practical classifier can be built that detects hyperpartisan news at scale and in real time, since an article's style can be assessed immediately without referring to external information.

5.2 Fake vs. Real (vs. Satire)

This series of experiments targets research questions (2) and (3). Again, we conduct three experiments, where the first is about predicting veracity, and the last two about discriminating satire.

A. Predicting veracity. When taking into account that the mainstream news publishers in our corpus did not publish any news items that are mostly false, and only very few instances that are mixtures of true and false, we may safely disregard them for the task of fake news detection. A reliable classifier for hyperpartisan news can act as a prefilter for a subsequent, more in-depth fake news detection approach, which may in turn be tailored to a much more narrowly defined classification task. We hence use only the left-wing articles and the right-wing articles of our corpus for our attempt at a style-based fake news classifier.

Table 5 shows the performance values for a generic classifier that predicts fake news across orientations, and orientation-specific classifiers that have been individually trained on articles from either orientation. Although all classifiers outperform the naive baselines of classifying everything into one of the classes in terms of precision, the slight increase comes at the cost of a large decrease in recall. While the orientation-specific classifiers are slightly better for most metrics, none of them outperform the naive baselines regarding the F-Measure. We conclude that style-based fake news classification simply does not work in general.

Features	Accuracy	Precision		Recall		F$_1$	
	all	fake	real	fake	real	fake	real
Generic classifier							
Style	0.55	0.42	0.62	0.41	0.64	0.41	0.63
Topic	0.52	0.41	0.62	0.48	0.55	0.44	0.58
Orientation-specific classifier							
Style	0.55	0.43	0.64	0.49	0.59	0.46	0.61
Topic	0.58	0.46	0.65	0.45	0.66	0.46	0.66
All-fake	0.39	0.39	-	1.00	0.0	0.56	-
All-real	0.61	-	0.61	0.0	1.00	-	0.76

Table 5: Performance of predicting veracity.

Features	Accuracy	Precision		Recall		F$_1$	
	all	sat.	real	sat.	real	sat.	real
Style	0.82	0.84	0.80	0.78	0.85	0.81	0.82
Topic	0.77	0.78	0.75	0.74	0.79	0.76	0.77
All-sat.	0.50	0.50	-	1.00	0.0	0.67	-
All-real	0.50	-	0.50	0.00	1.00	-	0.67
Rubin et al.	n/a	0.90	n/a	0.84	n/a	0.87	n/a

Table 6: Performance of predicting satire (sat.).

B. Predicting satire. Yet, not all fake news are the same. One should distinguish satire from the rest, which takes the form of news but lies more or less obviously to amuse its readers. Regardless the problems that spreading fake news may cause, satire should never be filtered, but be discriminated from other fakes. Table 6 shows the performance values of our classifier in the satire-detection setting used by Rubin et al. (2016) (the S-n-L News DB corpus), distinguishing satire from real news. This setting uses a balanced 3:1 training-to-test set split over 360 articles (180 per class). As can be seen, our style-based model significantly outperforms all baselines across the board, achieving an accuracy of 0.82, and an F score of 0.81. It clearly improves over topic classification, but does not outperform Rubin et al.'s classifier, which includes features based on topic, absurdity, grammar, and punctuation. We argue that incorporating topic into satire detection is not appropriate, since the topics of satire change along the topics of news. A classifier with topic features therefore does not generalize. Apparently, a style-based model is competitive, and we believe that satire can be detected at scale this way, so as to prevent other fake news detection technology from falsely filtering it.

C. Unmasking satire. Given the above results on stylistic similarities between left and right news, the question remains how satire fits into the picture. We assess the style similarity of satire from Rubin et al.'s corpus compared to fake news and real news from ours, again applying Unmasking to compare pairs of the three categories of news as described above. Figure 3 shows the resulting Unmasking curves. The curve for the pair of fake vs. real news drops faster compared to the other two pairs. Apparently, the style of fake news has more in common with that of real news than either of the two have with satire. These results are encouraging: satire is distinct enough from fake and real news, so that, just like with hyperpartisan news compared to mainstream news, it can be discriminated with reasonable accuracy.

6 Conclusion

Fact-checking for fake news detection poses an interdisciplinary challenge: technology is required to extract factual statements from text, to match facts with a knowledge base, to dynamically retrieve and maintain knowledge bases from the web, to reliably assess the overall veracity of an entire article rather than individual statements, to do so in real time as news events unfold, to monitor the spread of fake news within and across social media, to measure the reputation of information sources, and to raise awareness in readers. These are only the most salient things that need be done to tackle the problem, and as our cross-section of related work shows, a large body of work must be covered. Notwithstanding the many attacks on fake news by developing one way or another of fact-checking, we believe it worthwhile to mount our attack from another angle: writing style.

We show that news articles conveying a hyperpartisan world view can be distinguished from more balanced news by writing style alone. Moreover, for the first time, we found quantifiable evidence that the writing styles of news of the two opposing orientations are in fact very similar: there appears to be a common writing style of left and right extremism. We further show that satire can be distinguished well from other news, ensuring that humor will not be outcast by fake news detection technology. All of these results offer new, tangible, short-term avenues of development, lest large-scale fact-checking is still far out of reach. Employed as pre-filtering technologies to separate hyperpartisan news from mainstream news, our approach allows for directing the attention of human fact checkers to the most likely sources of fake news.

Acknowledgements

We thank Craig Silverman, Lauren Strapagiel, Hamza Shaban, Ellie Hall, and Jeremy Singer-Vine from BuzzFeed for making their data available, enabling our research.

References

Daron Acemoglu, Asuman Ozdaglar, and Ali ParandehGheibi. 2010. Spread of (Mis)Information in Social Networks. *Games and Economic Behavior*, 70(2):194–227.

Sadia Afroz, Michael Brennan, and Rachel Greenstadt. 2012. Detecting Hoaxes, Frauds, and Deception in Writing Style Online. In *2012 IEEE Symposium on Security and Privacy*, pages 461–475.

Shlomo Argamon-Engelson, Moshe Koppel, and Galit Avneri. 1998. Style-based text categorization: What newspaper am i reading. In *Proc. of the AAAI Workshop on Text Categorization*, pages 1–4.

Sameer Badaskar, Sachin Agarwal, and Shilpa Arora. 2008. Identifying real or fake articles: Towards better language modeling. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008, Hyderabad, India, January 7-12, 2008*, pages 817–822. The Association for Computer Linguistics.

Peter Bourgonje, Julián Moreno Schneider, and Georg Rehm. 2017. From clickbait to fake news detection: An approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 Workshop: Natural Language Processing meets Journalism, NLPmJ@EMNLP, Copenhagen, Denmark, September 7, 2017*, pages 84–89.

Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 665–674, New York, NY, USA. ACM.

Yimin Chen, Niall J. Conroy, and Victoria L. Rubin. 2015. News in an Online World: The Need for an "Automatic Crap Detector". In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, ASIST '15, pages 81:1–81:4, Silver Springs, MD, USA. American Society for Information Science.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational Fact Checking from Knowledge Networks. *PloS one*, 10(6):e0128193.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 69–76.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open Information Extraction from the Web. *Commun. ACM*, 51(12):68–74.

Alexandru L. Ginsca, Adrian Popescu, and Mihai Lupu. 2015. Credibility in Information Retrieval. *Found. Trends Inf. Retr.*, 9(5):355–475.

Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism Detection in Wikidata. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management (CIKM 16)*, pages 327–336. ACM.

Benjamin D. Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR*, abs/1703.09398.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *J. Mach. Learn. Res.*, 8:1261–1276.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent Features of Rumor Propagation in Online Social Media. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1103–1108. IEEE.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 252–256.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 708–717.

Amr Magdy and Nayer Wanas. 2010. Web-based Statistical Fact Checking of Textual Documents. In *Proceedings of the 2Nd International Workshop on Search and Mining User-generated Contents*, SMUC '10, pages 103–110, New York, NY, USA. ACM.

Delia Mocanu, Luca Rossi, Qian Zhang, Marton Karsai, and Walter Quattrociocchi. 2015. Collective Attention in the Age of (Mis)Information. *Comput. Hum. Behav.*, 51(PB):1198–1204.

Nam P. Nguyen, Guanhua Yan, My T. Thai, and Stephan Eidenbenz. 2012. Containment of Misinformation Spread in Online Social Networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, pages 213–222, New York, NY, USA. ACM.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *CoRR*, abs/1708.07104.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2931–2937.

Victoria Rubin, Niall Conroy, and Yimin Chen. 2015. Towards News Verification: Deception Detection Methods for News Discourse. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS48) Symposium on Rapid Screening Technologies, Deception Detection and Credibility Assessment Symposium*, Kauai, Hawaii, USA.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, San Diego, California. Association for Computational Linguistics.

Baoxu Shi and Tim Weninger. 2016. Fact Checking in Heterogeneous Information Networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 101–102, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Craig Silverman, Lauren Strapagiel, Hamza Shaban, Ellie Hall, and Jeremy Singer-Vine. 2016. Hyperpartisan Facebook Pages are Publishing False and Misleading Information at an Alarming Rate. https://www.buzzfeed.com/craigsilverman/partisan-fb-pages-analysis. BuzzFeed.

Philip J. Stone, Dexter C. Dunphy, and Marshall S. Smith. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT press.

Marcella Tambuscio, Giancarlo Ruffo, Alessandro Flammini, and Filippo Menczer. 2015. Fact-checking Effect on Viral Hoaxes: A Model of Misinformation Spread in Social Networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 977–982, New York, NY, USA. ACM.

Udo Undeutsch. 1967. Beurteilung der glaubhaftigkeit von aussagen. *Handbuch der Psychologie*, 11:26–181.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Oken Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 647–653.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 422–426.

Zhongyu Wei, Junwen Chen, Wei Gao, Binyang Li, Lanjun Zhou, Yulan He, and Kam-Fai Wong. 2013. An empirical study on uncertainty identification in social media context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 58–62, Sofia, Bulgaria. Association for Computational Linguistics.

You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward Computational Fact-checking. *Proc. VLDB Endow.*, 7(7):589–600.

Fan Yang, Arjun Mukherjee, and Eduard Constantin Dragut. 2017. Satirical news detection and analysis using attention mechanism and linguistic features. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1979–1989.

Retrieval of the Best Counterargument without Prior Topic Knowledge

Henning Wachsmuth
Paderborn University
Computational Social Science Group
henningw@upb.de

Shahbaz Syed and **Benno Stein**
Bauhaus-Universität Weimar
Faculty of Media, Webis Group
<first>.<last>@uni-weimar.de

Abstract

Given any argument on any controversial topic, how to counter it? This question implies the challenging retrieval task of finding the best counterargument. Since prior knowledge of a topic cannot be expected in general, we hypothesize the best counterargument to invoke the same aspects as the argument while having the opposite stance. To operationalize our hypothesis, we simultaneously model the similarity and dissimilarity of pairs of arguments, based on the words and embeddings of the arguments' premises and conclusions. A salient property of our model is its independence from the topic at hand, i.e., it applies to arbitrary arguments. We evaluate different model variations on millions of argument pairs derived from the web portal *idebate.org*. Systematic ranking experiments suggest that our hypothesis is true for many arguments: For 7.6 candidates with opposing stance on average, we rank the best counterargument highest with 60% accuracy. Even among all 2801 test set pairs as candidates, we still find the best one about every third time.

1 Introduction

Many controversial topics in real life divide us into opposing camps, such as whether to ban guns, who should become president, or what phone to buy. When being confronted with arguments against our stance, but also when forming own arguments, we need to think about how they could best be countered. Argumentation theory tells us that — aside from ad-hominem attacks — a counterargument denies either an argument's premises, its conclusion, or the reasoning between them (Walton, 2009). Take the following argument in favor of the right to bear arms from the web portal idebate.org:

Argument *"Gun ownership is an integral aspect of the right to self defence.* (conclusion)
Law-abiding citizens deserve the right to protect their families in their own homes, especially if the police are judged incapable of dealing with the threat of attack. [...]" (premise)

While the conclusion seems well-reasoned, the web portal directly provides a counter to the argument:

Counterargument *"Burglary should not be punished by vigilante killings of the offender. No amount of property is worth a human life. Perversely, the danger of attack by homeowners may make it more likely that criminals will carry their own weapons. If a right to self-defence is granted in this way, many accidental deaths are bound to result. [...]"*

As in this example, we observe that a counterargument often takes on the aspects of the topic invoked by the argument, while adding a new perspective to its conclusion and/or premises, conveying the opposite stance. Research has tackled the stance of argument units (Bar-Haim et al., 2017) as well as the attack relations between arguments (Cabrio and Villata, 2012). However, existing approaches learn the interplay of aspects and topics on training data or infer it from external knowledge bases (details in Section 2). This does not work for topics unseen before. Moreover, to our knowledge, no work so far aims at actual *counter*arguments.

This paper studies the task of automatically finding the best counterargument to any argument. In the general case, we cannot expect prior knowledge of an argument's topic. Following the observation above, we thus just hypothesize the best counterargument to invoke the same aspects as the argument while having the opposite stance. Figure 1 sketches how we operationalize the hypothesis. In particular, we simultaneously model the topic similarity and stance *dis*similarity of a candidate counterargument

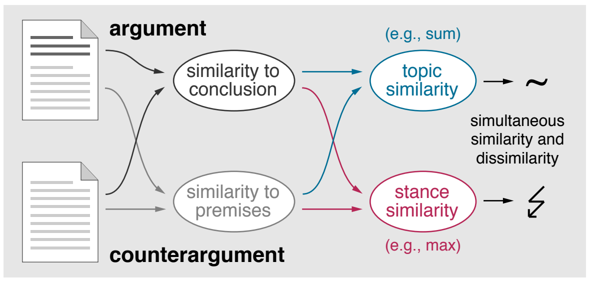


Figure 1: Modeling the simultaneous similarity and dissimilarity of a counterargument to an argument.

to the argument. Both are inferred — in different ways — from the similarities to the argument's conclusion and premises, since it is unclear in advance, whether either of these units or the reasoning between them is countered. Thereby, we find the most dissimilar among the most similar arguments.

To study counteraguments, we provide a new corpus with 6753 argument-counterargument pairs, taken from 1069 debates on idebate.org, as well as millions of false pairs derived from them. Given the corpus, we define eight retrieval tasks that differ in the types of candidate counterarguments. Based on the words and embeddings of the arguments, we develop similarity functions that realize the outlined model as a ranking approach. In systematic experiments, we evaluate the different building blocks of our model on all defined tasks.

The results suggest that our hypothesis is true for many arguments. The best model configuration improves common word and embedding similarity measures by eight to ten points accuracy in all tasks. Inter alia, we rank 60.3% of the best counterarguments highest when given all arguments with opposite stance (7.6 on average). Even with all 2801 test arguments as candidates, we still achieve 32.4% (and a mean rank of 15), fitting the intuition that off-topic arguments are easier to discard. Our analysis reveals notable gaps across topical themes though.

Contributions We believe that our findings will be important for applications such as automatic debating technologies (Rinott et al., 2015) and argument search (Wachsmuth et al., 2017b). To summarize, our main contributions are:

- A large corpus for studying multiple counterargument retrieval tasks (Sections 3 and 4).
- A topic-independent approach to find the best counterargument to any argument (Section 5).
- Evidence that many counterarguments can be found without topic knowledge (Section 6).

The corpus as well as the Java source code for reproducing the experiments are available at `http://www.arguana.com`.

2 Related Work

Counterarguments rebut arguments. In the theoretical model of Toulmin (1958), a rebuttal in fact does not attack the argument, but it merely shows exceptions to the argument's reasoning. Govier (2010) suggests to rather speak of counterconsiderations in such cases. Unlike Damer (2009), who investigates *how to attack* several kinds of fallacies, we are interested in *how to identify attacks*. We focus on those that target arguments, excluding personal (ad-hominem) attacks (Habernal et al., 2018).

Following Walton (2006), an argument can be attacked in two ways: one is to question its validity — not meaning that its conclusion must be wrong. The other is to rebut it with a *counterargument* that entails the opposite conclusion, often by revisiting aspects or introducing new ones. This is the type of attack we study. As Walton (2009) details, rebuttals may target an argument's premises or conclusion, or they may undercut the reasoning between them.

Recently, the computational analysis of natural language argumentation is receiving much attention. Most research focuses on argument mining, ranging from segmenting a text into argument units (Ajjour et al., 2017), over identifying unit types (Rinott et al., 2015) and roles (Niculae et al., 2017), to classifying argument schemes (Feng and Hirst, 2011) and relations (Lawrence and Reed, 2017). Some works detect counterconsiderations in a text (Peldszus and Stede, 2015) or their absence (Stab and Gurevych, 2016). Such considerations make arguments more balanced (see above). In contrast, we seek for arguments that defeat others.

Many approaches mine attack relations between arguments. Some use deep learning to find attacks in discussions (Cocarascu and Toni, 2017). Closer to this paper, others determine them in a given set of arguments, using textual entailment (Cabrio and Villata, 2012) or a combination of markov logic and stance classification (Hou and Jochim, 2017). In principle, any attacking argument denotes a counterargument. Unlike previous work, however, we aim for the *best* counterargument to an argument.

Classifying the stance of a text towards a topic (pro or con) generally defines an alternative way of addressing counterarguments. Sobhani et al. (2015) specifically classify health-related arguments using supervised learning, while we do not expect to have prior topic knowledge. Bar-Haim et al. (2017) approach the stance of claims towards open-domain topics. Their approach combines aspect-based sentiment with external relations between aspects and topics from Wikipedia. As such, it is in fact limited to the topics covered there. Our model applies to arbitrary arguments and counterarguments.

We need to identify only whether arguments oppose each other, not their actual stance. Similarly, Menini et al. (2017) classify only the *disagreement* of political texts. Part of their approach is to detect topical key aspects in an unsupervised manner, which seems useful for our purposes. Analogously, Beigman Klebanov et al. (2010) study differences in vocabulary choice for the related task of perspective classification, and Tan et al. (2016) find that the best way to persuade opinion holders in the *Change my view* forum on reddit.com is to use dissimilar words. As we report later, however, our experiments did not show such results for the argument-counterargument pairs we deal with.

The goal of persuasion reveals the association of counterarguments to argumentation quality. Many quality criteria have been assessed for arguments, surveyed in (Wachsmuth et al., 2017a). In the study of Habernal and Gurevych (2016), one reason annotators gave for why an argument was more convincing than another was that it tackled flaws in the opposing view. Zhang et al. (2016) even found that debate winners tend to counter opposing arguments rather than focusing on their own arguments.

Argument quality assessment is particularly important in retrieval scenarios. Existing approaches aim to retrieve documents that contain many claims (Roitman et al., 2016) or that provide most support for their claims (Braunstain et al., 2016). In Wachsmuth et al. (2017c), we adapt PageRank to argumentative relations, in order to assess argument relevance objectively. While our search engine *args* for arguments on the web still uses content-based relevance measures in its first version (Wachsmuth et al., 2017b), its long-term idea is to rank the best arguments highest.[1] The model present in this work finds the best *counter*arguments, but it is meant to be integrated into *args* at some point.

Like here, *args* uses idebate.org arguments. Others take data from that portal for studying support (Boltužić and Šnajder, 2014) or for the distant supervision of argument mining (Al-Khatib et al., 2016). Our corpus is not only larger, though, but it is the first to utilize a unique feature of idebate.org: the explicit specification of counterarguments.

[1]Argument search engine *args*: `http://args.me`

3 The ArguAna Counterargs Corpus

This section introduces our *ArguAna Counterargs corpus* with argument-counterargument pairs, created automatically from the structure of idebate.org. The corpus is freely available at `http://www.arguana.com/data`. We also provide the code to replicate the construction process.

3.1 The Web Portal idebate.org

On the *portal* idebate.org, diverse controversial topics of usually rather general interest are discussed in *debates*, subsumed under 15 *themes*, such as “economy” and “health”. Each debate has a title capturing a thesis on a topic, such as “This House would limit the right to bear arms”, followed by an introductory text, a set of mostly elaborated and well-written *points* that have a pro or a con stance towards the thesis, and a bibliography.

A specific feature of idebate.org is that virtually every point comes along with a *counter* that immediately attacks the point and its stance. Both points and counters can be seen as *arguments*. While a point consists of a one-sentence claim (the argument’s *conclusion*) and a few sentences justifying the claim (the *premise(s)*), the counter’s (opposite) conclusion remains implicit.

All arguments on the portal are established by a community with the goal of showing both sides of a topic in a balanced manner. We therefore assume each counter to be the best counterargument available for the respective point, and we use all resulting *true argument pairs* as the basis of our corpus. Figure 2 illustrates the italicized concepts, showing the structure of idebate.org. An example argument pair has been discussed in Section 1.

3.2 Corpus Construction

We crawled all debates from idebate.org that follow the portal’s theme-guided folder structure (last access: January 30, 2018). From each debate, we extracted the thesis, the introductory text, all points and counters, the bibliography, and some metadata. Each was stored separately in one plain text file, and we also created a file with the entire debate in its original order. Only points and counters are used in our experiments in Section 6. The underlying experiment settings are described in Section 4.

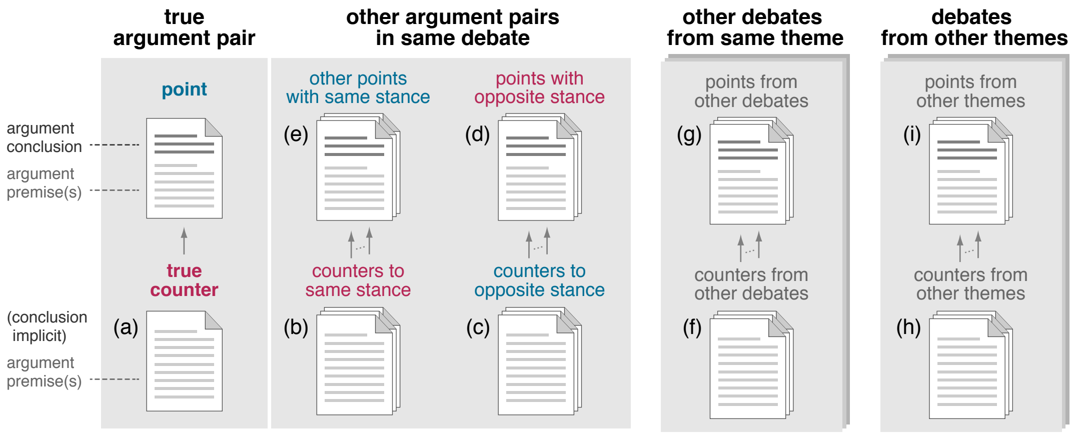

true argument pair
other argument pairs in same debate
other debates from same theme
debates from other themes
point
other points with same stance
points with opposite stance
points from other debates
points from other themes
argument conclusion
argument premise(s)
(e)
(d)
(g)
(i)
true counter
counters to same stance
counters to opposite stance
counters from other debates
counters from other themes
(conclusion implicit)
argument premise(s)
(a)
(b)
(c)
(f)
(h)

Figure 2: Structure of idebate.org for one true argument pair in our corpus. Colors denote matching stance; we assume arguments from other debates to have no stance towards a point. Points have a conclusion and premises, counters only premises. (a)–(i) are used in Section 4 to specify the candidates in different tasks.

Theme	**Debates**	**Points**	**Counters**
Culture	46	278	278
Digital freedoms	48	341	341
Economy	95	590	588
Education	58	382	381
Environment	36	215	215
Free speech debate	43	274	273
Health	57	334	333
International	196	1315	1307
Law	116	732	730
Philosophy	50	320	320
Politics	155	982	978
Religion	30	179	179
Science	41	271	269
Society	75	436	431
Sport	23	130	130
Training set	644	4083	4065
Validation set	211	1290	1287
Test set	214	1406	1401
counterargs-18	**1069**	**6779**	**6753**

Table 1: Distribution of debates, points, and counters over the themes in the *counterargs-18* corpus. The bottom rows show the size of the datasets.

3.3 Corpus Statistics

Table 1 lists the number of debates crawled for each theme, along with the numbers of points and counters in the debates. The 26 found points without a counter are included in the corpus, but we do not use them in our experiments.

In total, the ArguAna Counterargs corpus consists of 1069 debates with 6753 points that have a counter. The mean length of points is 196.3 words, whereas counters span only 129.6 words, largely due to the missing explicit conclusion. To avoid exploiting this corpus bias, no approach in our experiments captures length differences.

3.4 Datasets

We split the corpus into a training set, consisting of the first 60% of all debates of each theme (ordered by alphabet), as well as a validation set and a test set, each covering 20%. The dataset sizes are found at the bottom of Table 1. By putting all arguments from a debate into a single dataset, no specific topic knowledge can be gained from the training or validation set. We include all themes in all datasets, because we expect the set of themes to be stable.

We checked for duplicates. Among the 13 532 point and counters, 3407 appear twice, 723 three times, 36 four times, and 1 five times. We ensure that no true pair is used as a false pair in our tasks.

4 Counterargument Retrieval Tasks

Based on the new corpus, we define the following eight counterargument retrieval tasks of different complexity. All tasks consider all true argument-counterargument pairs, while differing in terms of what arguments (points and/or counters) from which context (same debate, same theme, or entire portal) are candidates for a given argument.

Same Debate: Opposing Counters All counters in the same debate with stance opposite to the given argument are candidates (Figure 2: a, b). The task is to find the best counterargument among all counters to the argument's stance.

Same Debate: Counters All counters in the same debate irrespective of their stance are candidates (Figure 2: a–c). The task is to find the best counterargument among all on-topic arguments phrased as counters.

		Training Set			Validation Set			Test Set		
Context	**Candidate Counterarg's**	**True**	**False**	**Ratio**	**True**	**False**	**Ratio**	**True**	**False**	**Ratio**
Same debate	Opposing counters	4 065	11 672	1:2.9	1 287	3 590	1:2.8	1 401	4 052	1:2.9
	Counters	4 065	27 024	1:6.6	1 287	8 348	1:6.5	1 401	9 312	1:6.6
	Opposing arguments	4 065	27 026	1:6.6	1 287	8 350	1:6.5	1 401	9 312	1:6.6
	Arguments	4 065	54 070	1:13.3	1 287	16 700	1:13.0	1 401	18 630	1:13.3
Same theme	Counters	4 065	1 616 000	1:398	1 287	176 266	1:137	1 401	189 870	1:136
	Arguments	4 065	3 232 038	1:795	1 287	352 536	1:274	1 401	379 746	1:271
Entire portal	Counters	4 065	16 517 994	1:4063	1 287	1 654 878	1:1286	1 401	1 961 182	1:1400
	Arguments	4 065	33 038 154	1:8127	1 287	3 309 760	1:2572	1 401	3 922 582	1:2800

Table 2: Number of true and false argument-counterargument pairs as well as their ratio for each evaluated context and type of candidate counterarguments in the three datasets. Each line defines one retrieval task.

Same Debate: Opposing Arguments All arguments in the same debate with opposite stance are candidates (Figure 2: a, b, d). The task is to find the best among all on-topic counterarguments.

Same Debate: Arguments All arguments in the same debate irrespective of their stance are candidates (Figure 2: a–e). The task is to find the best counterargument among all on-topic arguments.

Same Theme: Counters All counters from the same theme are candidates (Figure 2: a–c, f). The task is to find the best counterargument among all on-theme arguments phrased as counters.

Same Theme: Arguments All arguments from the same theme are candidates (Figure 2: a–g). The task is to find the best counterargument among all on-theme arguments.

Entire Portal: Counters All counters are candidates (Figure 2: a–c, f, h). The task is to find the best counterargument among all arguments phrased as counters.

Entire Portal: Arguments All arguments are candidates (Figure 2: a–i). The task is to find the best counterargument among all arguments.

Table 2 lists the numbers of true and false pairs for each task. Experiment files containing the file paths of all candidate pairs are provided in our corpus.

5 Retrieval of the Best Counterargument without Prior Topic Knowledge

The eight defined tasks indicate the subproblems of retrieving the best counterargument to a given argument: Finding all arguments that address the *same topic*, filtering those arguments with an *opposite stance* towards the topic, and identifying the *best counter* among these arguments. This section presents our approach to solving these problems computationally without prior knowledge of the argument's topic, based on the simultaneous similarity and dissimilarity of arguments.[2]

5.1 Topic as Word and Embedding Similarity

We do not reinvent the wheel to assess topical relevance, but rather follow common practice. Concretely, we hypothesize a candidate counterargument to be on-topic if it is similar to the argument in terms of its words and its embedding. We capture these two types of similarity as follows.

Word Argument Similarity To best represent the words in arguments, we did initial counterargument retrieval tests with token, stem, and lemma n-grams, $n \in \{1, 2, 3\}$. While the differences were not large, stems worked best and stem 1-grams sufficed. Both might be a consequence of the limited data size. In our experiments in Section 6, we determine the *stem 1-grams* to be considered on the training set of each task.

For word similarity computation, we tested four inverse vector-based distance measures: Cosine, Euclidean, Manhattan, and, Jaccard similarity (Cha, 2007). On the validation sets, the Manhattan similarity performed best, closely followed by the Jaccard similarity. Both clearly outperformed Euclidean and especially Cosine similarity. This suggests that the presence and absence of words are equally important and that outliers should not be punished more. For brevity, we report only results for the *Manhattan similarity* below.

[2]As indicated above, counters on idebate.org (including all true counterarguments) may also differ linguistically from points (all of which are false). However, we assume this to be a specific corpus bias and hence do not explicitly account for it. Section 6 will show whether having both points and counters as candidates makes counterargument retrieval harder.

Embedding Argument Similarity We evaluated five pretrained word embedding models for representing arguments in first tests: GoogleNews-vectors (Mikolov et al., 2013), ConceptNet Numberbatch (Speer et al., 2017), wiki-news-300d-1M, wiki-news-300d-1M-subword, and crawl-300d-2M (Mikolov et al., 2017). The former two were competitive, the others performed notably worse. Since *ConceptNet Numberbatch* is smaller and supposed to have less bias, we used it in all experiments.

To capture argument-level embedding similarity, we compared the four inverse vector-based distance measures above on average word embeddings against the inverse Word Mover's distance, which quantifies the optimum alignment of two word embedding sequences (Kusner et al., 2015). This *Word Mover's similarity* consistently beat the others, so we decided to restrict our view to it.

5.2 Stance as Topic Dissimilarity

Stance classification without prior topic knowledge is challenging: While we can compare the topics of any two arguments, it is impossible in general to infer the stance of the specific aspects invoked by one argument to those of the other. As sketched in Section 2, related work employs external knowledge to infer stance relations of aspects and topics (Bar-Haim et al., 2017) or trains classifying attack relations (Cabrio and Villata, 2012). Unfortunately, both does not apply to topics unseen before.

For argument pairs invoking similar aspects, a way to go is in principle to assess sentiment polarity; intuitively, two arguments with the same topic but opposite sentiment have opposing stance. However, we tested topic-agnostic sentiment lexicons (Baccianella et al., 2010) and state-of-the-art sentiment classifiers, trained on large-scale multiple-domain review data (Prettenhofer and Stein, 2010; Joulin et al., 2017). The correlation between sentiment and stance differences of training arguments was close to zero. A possible explanation is the limited explicitness of sentiment on idebate.org, making the lexicons and classifiers fail there.

Other related work suggests that the vocabulary of opposing sides differs (Beigman Klebanov et al., 2010). We thus checked on the training set whether counterarguments are similar in their embeddings but dissimilar in their words. The measures above did not support this hypothesis, i.e., both embedding and word similarity increased the likelihood of a candidate counterargument being the best. Still, there must be a difference between an argument and its counterargument by concept. As a solution, we capture dissimilarity with the same similarity functions as above, but we change the granularity level on which we measure similarity.

5.3 Simultaneous Similarity and Dissimilarity

The arising question is how to assess similarity and dissimilarity at the same time. We hypothesize the best counterargument to be very similar in overall terms, but very dissimilar in certain respects. To capture this intuition, we rely on expert knowledge from argumentation theory (see Section 2).

Word and Embedding Unit Similarities In particular, we follow the notion that a counterargument attacks either the conclusion of an argument, the argument's premises, or both. As a consequence, we compute two word and two embedding similarities as specified above for each candidate counterargument; once to the argument's conclusion (called w_c and e_c for words and embeddings respectively) and once to the argument's premises (w_p and e_p).

Now, to capture similarity and dissimilarity simultaneously, we need multiple ways to aggregate conclusion and premise similarities. As we do not generally know which argument unit is attacked, we resort to four standard aggregation functions that generalize over the unit similarities. For words, these are the following *word unit similarities*:

$$\begin{array}{llll} w_{\downarrow} := & \min\{w_c, w_p\} \qquad & w_{\times} := & w_c \cdot w_p \\ w_{\uparrow} := & \max\{w_c, w_p\} \qquad & w_{+} := & w_c + w_p \end{array}$$

Accordingly, we define four respective *embedding unit similarities*, $e_{\downarrow}$, $e_{\uparrow}$, $e_{\times}$, and e_{+}.

As mentioned above, both word similarity and embedding similarity positively affect the likelihood that a candidate is the best counterargument. Therefore, we combine each pair of similarities as $w_{\downarrow} + e_{\downarrow}$, $w_{\uparrow} + e_{\uparrow}$, $w_{\times} + e_{\times}$, and $w_{+} + e_{+}$, but we also evaluate their impact in isolation below.[3]

Counterargument Scoring Model Based on the unit similarities, we finally define a scoring model for a given pair of argument and candidate counterargument. The model includes two unit similarity values, sim and $dissim$, but $dissim$ is subtracted from sim, such that it actually favors dissimilarity. Thereby, we realize the topic and

[3]In principle, other unit similarities could be used for words than for embeddings. However, we decided to couple them to maintain interpretability of our experiment results.

stance similarity sketched in Figure 1. We weight the two values with a damping factor α:

$$\alpha \cdot sim \;-\; (1-\alpha) \cdot dissim$$

where $sim, dissim \in \{w_{\downarrow}+e_{\downarrow}, w_{\uparrow}+e_{\uparrow}, w_{\times}+e_{\times}, w_{+}+e_{+}\}$ and $sim \neq dissim$.

The general idea of the scoring model is that sim rewards one type of similarity, whereas subtracting $dissim$ punishes another type. We seek to thereby find the most dissimilar candidate among the similar candidates. The model is meant to give a higher score to a pair the more likely the candidate is the best counterargument to the argument, so the scores can be used for ranking.

What combination of sim and $dissim$ turns out best, is hard to foresee and may depend on the retrieval task at hand. We hence evaluate different combinations empirically below. The same holds for the damping factor $\alpha \in [0, 1]$. If our hypothesis on similarity and dissimilarity is true, then the best α should be close to but lower than 1. Conversely, if $\alpha = 1.0$ achieves the best performance, then only similarity would be captured by our model.

6 Evaluation

We now report on systematic ranking experiments with our counterargument scoring model. The goal is to evaluate on all eight retrieval tasks from Section 4 to what extent our hypothesis holds that the best counterargument to an argument invokes the same aspects while having opposing stance. The Java source code of the experiments is available at: http://www.arguana.com/software

6.1 Experimental Set-up

We evaluated the following set-up of tasks, data, measures, baselines, and approaches.

Tasks We tackled each of the eight retrieval tasks as a ranking problem, i.e., we aimed to rank the best counterargument to each argument highest, given all candidates. Accordingly, only one candidate counterargument per argument is correct.[4]

Data Table 2 has shown the true and false argument pairs in all datasets. We undersampled each training set, resulting in 4065 true and 4065 false training pairs in all tasks.[5] Our model does not do any learning-to-rank on these pairs, but we derived lexicons for the word similarities from them (all stems included in at least 1% of all pairs). As detailed below, we then determined the best model configurations on the validation sets and evaluated these configurations on the test sets.

Measures As only one candidate is true per argument, we report the *accuracy@1* of each approach, i.e., the percentage of arguments for which the true counterargument was ranked highest. Besides, we compute the rounded *mean rank* of the best counterargument in all rankings, reflecting the average performance of an approach. Exemplarily, we also mention the *mean reciprocal rank (MRR)*, which is more sensitive to outliers.

Baselines A trivial way to address the given tasks is to pick any candidate by chance for each argument. This *random baseline* allows quantifying the impact of other approaches. As counterargument retrieval has not been tackled yet, we do not use any existing baseline.[6] Instead, we evaluate the effects of the different building blocks of our scoring model. On one hand, we check the need for distinguishing conclusions and premises by comparing to the word argument similarity (w) and the embedding argument similarity (e). On the other hand, we consider all eight word and embedding unit similarities ($w_{\downarrow}$, $w_{\uparrow}$, ..., e_{+}) as baselines, in order to see whether and how to best aggregate them.

Approaches After initial tests, we reduced the set of tested values of the damping factor α in our scoring model to $\{0.8, 0.9, 1.0\}$. On the validation sets of the first six tasks,[7] we then analyzed all possible combinations of $w_{\downarrow}+e_{\downarrow}$, $w_{\uparrow}+e_{\uparrow}$, $w_{\times}+e_{\times}$, $w_{+}+e_{+}$, as well as $w+e$ for sim and $dissim$. Three configurations of the model turned out best:

$$\begin{aligned} we &:= 1.0 \cdot (w_{\times} + e_{\times}) \\ we_{\downarrow} &:= 0.9 \cdot (w_{\times} + e_{\times}) - 0.1 \cdot (w_{\downarrow} + e_{\downarrow}) \\ we_{\uparrow} &:= 0.9 \cdot (w_{+} + e_{+}) - 0.1 \cdot (w_{\uparrow} + e_{\uparrow}) \end{aligned}$$

[4]One alternative would be to see each argument pair as one instance of a classification problem. However, our preliminary tests confirmed the intuition that identifying the best counterargument is hard without knowing the other candidates, i.e., there is no general (dis)similarity threshold that makes an argument the best counterargument. Rather, how similar or dissimilar a counterargument needs to be depends on the topic and on the other candidates. Another alternative would be to treat all candidates for an argument as one instance, but this makes the experimental set-up very intricated.

[5]Undersampling was done stratified, such that the same number of false counterarguments was taken from each type, b–i, in Figure 2 that is relevant in the respective task.

[6]Notice, though, that we tested a number of approaches to identify opposing stance, as discussed in Section 5.

[7]We did not expect "game-changing" validation set results for the last two tasks and, so, left them out for time reasons.

		Same Debate								**Same Theme**				**Entire Portal**			
		Opp. Ctr.'s		*Counters*		*Opposing*		*Arguments*		*Counters*		*Arguments*		*Counters*		*Arguments*	
#	**Baseline / Approach**	**@1**	**R**	**@1**	**R**	**@1**	**R**	**@1**	**R**	**@1**	**R**	**@1**	**R**	**@1**	**R**	**@1**	**R**
w	Word argument similarity	65.9	2	48.5	**2**	42.5	3	30.0	4	44.1	5	28.3	10	39.7	22	21.8	49
e	Embedding argument similarity	62.9	2	44.6	**2**	51.6	**2**	36.8	4	38.8	7	32.9	10	34.2	39	23.9	55
$w_\downarrow$	Word unit similarity minimum	53.8	2	38.4	3	45.9	3	33.7	5	28.5	22	24.8	42	21.4	206	18.5	403
$w_\uparrow$	Word unit similarity maximum	66.1	2	48.0	**2**	44.0	3	30.2	4	44.0	5	28.3	9	38.0	21	21.2	44
$w_\times$	Word unit similarity product	64.9	2	49.5	3	56.1	**2**	40.7	4	44.3	18	36.8	35	37.8	177	26.8	354
w_+	Word unit similarity sum	71.5	**1**	53.7	**2**	54.1	**2**	39.1	4	49.0	4	36.8	7	44.7	17	28.6	33
$e_\downarrow$	Embedding unit sim. minimum	61.6	2	44.9	3	43.4	3	32.1	4	37.8	7	27.4	13	32.5	42	20.7	74
$e_\uparrow$	Embedding unit sim. maximum	63.4	2	44.5	**2**	47.5	**2**	33.2	4	39.8	5	29.8	8	32.1	20	20.1	33
$e_\times$	Embedding unit sim. product	69.7	**1**	52.0	**2**	55.4	**2**	41.0	**3**	44.3	4	37.1	6	43.2	14	27.8	21
e_+	Embedding unit sim. sum	69.7	**1**	51.8	**2**	55.4	**2**	40.5	**3**	47.5	4	36.8	6	43.0	13	27.6	21
we	$1.0\cdot(w_\times+e_\times)$	72.1	**1**	55.2	**2**	‡**60.3**	**2**	†**44.9**	**3**	50.4	4	40.9	7	46.0	19	32.2	34
$we_\downarrow$	$0.9\cdot(w_\times+e_\times)-0.1\cdot(w_\downarrow+e_\downarrow)$	72.0	**1**	55.5	**2**	59.5	**2**	44.1	**3**	51.3	4	†**41.0**	7	46.3	19	31.7	35
$we_\uparrow$	$0.9\cdot(w_++e_+)-0.1\cdot(w_\uparrow+e_\uparrow)$	†**74.5**	**1**	†**57.7**	**2**	59.6	**2**	44.1	**3**	‡**54.2**	**3**	40.8	**5**	‡**50.0**	**9**	‡**32.4**	**15**
r	Random baseline	25.7	2	13.1	4	13.1	4	7.0	7	0.7	69	0.4	137	0.1	701	0.0	1401

Table 3: Test set accuracy of ranking the best counterargument highest (**@1**) and mean rank (**R**) for 14 baselines and approaches (w, e, $w_\downarrow$, ..., r) in all eight tasks (given by **Context** and *Candidates*). Each best accuracy value (bold) significantly outperforms the best baseline with 99% (†) or 99.9% (‡) confidence.

we was best on the validation set of *Same Debate: Opposing Arguments* (accuracy@1: 62.1) and $we_\downarrow$ on the one of *Same Debate: Arguments* (49.0). All other tasks were dominated by $we_\uparrow$. Especially, $we_\uparrow$ was better than $1.0 \cdot (w_+ + e_+)$ in all of them with clear leads of up to 2.2 points. This underlines the importance of modeling dissimilarity for counterargument retrieval. We took we, $we_\downarrow$, and $we_\uparrow$ as our approaches for the test set.[8]

6.2 Results

Table 3 shows the accuracy@1 and the mean rank of all baselines and approaches on each of the eight given retrieval tasks.

Overall, the counter-only tasks seem slightly harder within the *same debate* (comparing *Counters* to *Opposing*), i.e., stance is harder to assess than topical relevance. Conversely, the other *Counters* tasks seem easier, suggesting that topically close but false candidate counterarguments with the same stance as the argument (which are not included in any *Counters* task) are classified wrongly most often. Besides, these results support that potential differences in the phrasing of counters are not exploited, as desired.

The accuracy of the standard similarity measures, w and e, goes from 65.9 and 62.9 respectively in the smallest task down to 21.8 and 23.9 in the largest. w is stronger when only counters are candidates, e otherwise. This implies that words capture differences between the best and other counters, whereas embeddings rather help discard false candidates with the same stance as the argument.

From the eight unit similarity baselines, w_+ performs best on five tasks ($e_\times$ twice, $w_\times$ once). w_+ finds 71.5% true counterarguments among all opposing counters in a debate, and 28.6% among all test arguments from the entire portal. In that task, however, the mean ranks of w_+ (33) and particularly of $w_\times$ (354) are much worse than for $e_\times$ (21), meaning that words are insufficient to robustly find counterarguments.

we, $we_\downarrow$, and $we_\uparrow$ outperform all baselines in all tasks, improving the accuracy by 8.1 (*Same Theme: Arguments*) to 10.3 points (*Entire Portal: Counters*) over w and e, and at least 3.0 over the best baseline in each task. Among all opposing arguments from the same debate (true-to-false ratio 1:6.6), we finds 60.3% of the best counterarguments, 44.9% when all arguments are given (1:13.3).

The winner in our evaluation is $we_\uparrow$, though, being best in five of the eight tasks. It found the true among all opposing counters in 74.5% of all cases, and about every third time (32.4) among all 2801 test set arguments; a setting where the random baseline has virtually no chance. Given all arguments from the same theme, $we_\uparrow$ puts the best counterargument at a mean rank of 5 (MRR 0.58), and for the entire portal still at 15 (MRR 0.5).

[8] All validation set results are found in the supplementary material, which we provide at `http://www.arguana.com/publications`

Entire Portal: Arguments		Accuracy@1		Mean Rank	
Theme	Arguments	w_+	$we_\uparrow$	w_+	$we_\uparrow$
Culture	69	31.9	36.2	12	9
Digital freedoms	61	37.7	44.3	58	20
Economy	125	27.2	25.6	21	10
Education	81	38.3	39.5	36	17
Environment	46	17.4	21.7	22	7
Free speech debate	58	10.3	12.1	130	55
Health	77	28.6	36.4	26	14
International	271	25.8	31.4	31	19
Law	134	38.8	38.1	16	8
Philosophy	85	34.1	38.8	29	14
Politics	202	28.7	33.2	28	11
Religion	45	24.4	33.3	58	8
Science	57	19.3	28.1	6	5
Society	60	16.7	20.0	45	22
Sport	30	43.3	46.7	35	9
All themes	**1401**	**28.6**	**32.4**	**33**	**15**

Table 4: Accuracy@1 and mean rank of the best baseline (w_+) and approach ($we_\uparrow$) on each theme when all 2801 test set arguments are candidates.

Although our scoring model thus does not *solve* the retrieval tasks, we conclude that it serves as a robust approach to rank the best counterargument high.

To test significance, we separately computed the accuracy@1 for the arguments from each theme. The differences between the 15 values of the best approach on each task and those of the best baseline (w_+, $w_\times$, or $e_\times$) were normally distributed. Since the baselines and approaches are dependent, we used a one-tailed dependent t-test with paired samples. As Table 3 specifies, our approaches are consistently better, partly with at least 99% confidence, partly even with 99.9% confidence.

In Table 4, we exemplarily detail the comparison of the best approach ($we_\uparrow$) to the best baseline (w_+) on *Entire Portal: Arguments*. The mean ranks across themes underline the robustness of $we_\uparrow$, being in the top 10 for 7 and in the top 20 even for 13 themes. Still, the accuracy@1 of both w_+ and $we_\uparrow$ varies notably, in case of $we_\uparrow$ from 12.1 for *free speech debate* to 46.7 for *sport*. For free speech debates (e.g., "This House would criminalise blasphemy"), we observed that their arguments tend to be overproportionally long, which might lead to deviating similarities. In case of sports, the topical specificity (e.g., "This House would ban boxing") reduces the probability of mistakenly choosing candidates from other themes.

Free speech debate turned out the hardest theme in seven tasks, *health* in the remaining one. Besides sports, in some tasks the best results were obtained for *religion* and *science*, both of which share the characteristic of dealing with very specific topics.[9]

7 Conclusion

This paper has asked how to find the best counterargument to any argument without prior knowledge of the argument's topic. We did *not* aim to engineer the best approach to this retrieval task, but to study whether we can model the simultaneous similarity and dissimilarity of a counterargument to an argument computationally. For the restricted domain of debate portal arguments, our main result is quite intriguing: The best model ($we_\uparrow$) rewards a high overall similarity to the argument's conclusion and premises while punishing a too high similarity to either of them. Despite its simplicity, $we_\uparrow$ found the best counterargument among 2801 candidates in almost a third of all cases, and ranked it into the top 15 on average. This speaks for our hypothesis that the best counterargument often just addresses the same topical aspects with opposite stance.

Of course, our hypothesis is simplifying, i.e., there are counterarguments that will not be found based on aspect and stance similarity only. Apart from some hyperparameters, however, our model is unsupervised and it does not make any assumption about an argument's topic. Hence, it applies to any argument, given a pool of candidate counterarguments. While the model can be considered *open-topic*, a next step will be to study counterargument retrieval *open-source*.

We are confident that the modeled intuition generalizes beyond idebate.org. To obtain further insights into the nature of counterarguments, deeper linguistic analysis along with supervised learning may be needed, though. We provide a corpus to train respective approaches, but leave the according research to future work.

The intended practical application of our model is to retrieve counterarguments in automatic debating technologies (Rinott et al., 2015) and argument search (Wachsmuth et al., 2017b). While debate portal arguments are often suitable in this regard, in general not always a real counterargument exists to an argument. Still, returning one that addresses similar aspects with opposite stance makes sense then. An alternative would be to *generate* counterarguments, but we believe that humans are better than machines in writing them — currently.

[9]The individual results of the best approach and baseline on each theme are also found in the supplementary material.

References

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit segmentation of argumentative texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128. Association for Computational Linguistics.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404. Association for Computational Linguistics.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261. Association for Computational Linguistics.

Beata Beigman Klebanov, Eyal Beigman, and Daniel Diermeier. 2010. Vocabulary choice as an indicator of perspective. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 253–257. Association for Computational Linguistics.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58. Association for Computational Linguistics.

Liora Braunstain, Oren Kurland, David Carmel, Idan Szpektor, and Anna Shtok. 2016. Supporting human answers for advice-seeking questions in CQA sites. In *Proceedings of the 38th European Conference on IR Research*, pages 129–141.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 208–212. Association for Computational Linguistics.

Sung-Hyuk Cha. 2007. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307.

Oana Cocarascu and Francesca Toni. 2017. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1374–1379. Association for Computational Linguistics.

T. Edward Damer. 2009. *Attacking Faulty Reasoning: A Practical Guide to Fallacy-Free Arguments*, 6th edition. Wadsworth, Cengage Learning, Belmont, CA.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996. Association for Computational Linguistics.

Trudy Govier. 2010. *A Practical Study of Argument*, 7th edition. Wadsworth, Cengage Learning, Belmont, CA.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223. Association for Computational Linguistics.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, to appear.

Yufang Hou and Charles Jochim. 2017. Argument relation classification using a joint inference model. In *Proceedings of the 4th Workshop on Argument Mining*, pages 60–66. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, pages 957–966.

John Lawrence and Chris Reed. 2017. Mining argumentative structure from natural language text using automatically generated premise-conclusion topic models. In *Proceedings of the 4th Workshop on Argument Mining*, pages 39–48. Association for Computational Linguistics.

Stefano Menini, Federico Nanni, Simone Paolo Ponzetto, and Sara Tonelli. 2017. Topic-based agreement and disagreement in us electoral manifestos. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2938–2944. Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *CoRR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015. Towards detecting counter-considerations in text. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 104–109. Association for Computational Linguistics.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127. Association for Computational Linguistics.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, M. Mitesh Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence — An automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450. Association for Computational Linguistics.

Haggai Roitman, Shay Hummel, Ella Rabinovich, Benjamin Sznajder, Noam Slonim, and Ehud Aharoni. 2016. On the retrieval of wikipedia articles containing claims on controversial topics. In *Proceedings of the 25th International Conference on World Wide Web, Companion Volume*, pages 991–996.

Parinaz Sobhani, Diana Inkpen, and Stan Matwin. 2015. From argumentation mining to stance classification. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 67–77. Association for Computational Linguistics.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451.

Christian Stab and Iryna Gurevych. 2016. Recognizing the absence of opposing arguments in persuasive essays. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 113–118. Association for Computational Linguistics.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International World Wide Web Conference*, pages 613–624.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017a. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187. Association for Computational Linguistics.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017b. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59. Association for Computational Linguistics.

Henning Wachsmuth, Benno Stein, and Yamen Ajjour. 2017c. "PageRank" for Argument Relevance. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1117–1127. Association for Computational Linguistics.

Douglas Walton. 2006. *Fundamentals of Critical Argumentation*. Cambridge University Press.

Douglas Walton. 2009. Objections, rebuttals and refutations. pages 1–10.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in Oxford-style debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 136–141. Association for Computational Linguistics.

LinkNBed: Multi-Graph Representation Learning with Entity Linkage

Rakshit Trivedi [*]
College of Computing
Georgia Tech

Bunyamin Sisman
Amazon.com

Jun Ma
Amazon.com

Christos Faloutsos
SCS, CMU
and Amazon.com

Hongyuan Zha
College of Computing
Georgia Tech

Xin Luna Dong
Amazon.com

Abstract

Knowledge graphs have emerged as an important model for studying complex multi-relational data. This has given rise to the construction of numerous large scale but incomplete knowledge graphs encoding information extracted from various resources. An effective and scalable approach to jointly learn over multiple graphs and eventually construct a unified graph is a crucial next step for the success of knowledge-based inference for many downstream applications. To this end, we propose **LinkNBed**, a deep relational learning framework that learns entity and relationship representations across multiple graphs. We identify **entity linkage** across graphs as a vital component to achieve our goal. We design a novel objective that leverage entity linkage and build an efficient multi-task training procedure. Experiments on link prediction and entity linkage demonstrate substantial improvements over the state-of-the-art relational learning approaches.

1 Introduction

Reasoning over multi-relational data is a key concept in Artificial Intelligence and knowledge graphs have appeared at the forefront as an effective tool to model such multi-relational data. Knowledge graphs have found increasing importance due to its wider range of important applications such as information retrieval (Dalton et al., 2014), natural language processing (Gabrilovich and Markovitch, 2009), recommender systems (Catherine and Cohen, 2016), question-answering (Cui et al., 2017) and many more. This has led to the increased efforts in constructing numerous large-scale Knowledge Bases (e.g. Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), Google's Knowledge graph (Dong et al., 2014), Yago (Suchanek et al., 2007) and NELL (Carlson et al., 2010)), that can cater to these applications, by representing information available on the web in relational format.

All knowledge graphs share common drawback of *incompleteness* and *sparsity* and hence most existing relational learning techniques focus on using observed triplets in an incomplete graph to infer unobserved triplets for that graph (Nickel et al., 2016a). Neural embedding techniques that learn vector space representations of entities and relationships have achieved remarkable success in this task. However, these techniques only focus on learning from a single graph. In addition to incompleteness property, these knowledge graphs also share a set of overlapping entities and relationships with varying information about them. This makes a compelling case to design a technique that can learn over multiple graphs and eventually aid in constructing a unified giant graph out of them. While research on learning representations over single graph has progressed rapidly in recent years (Nickel et al., 2011; Dong et al., 2014; Trouillon et al., 2016; Bordes et al., 2013; Xiao et al., 2016; Yang et al., 2015), there is a conspicuous lack of principled approach to tackle the unique challenges involved in learning across multiple graphs.

One approach to multi-graph representation learning could be to first solve graph alignment problem to merge the graphs and then use existing relational learning methods on merged graph. Unfortunately, graph alignment is an important but still unsolved problem and there exist several techniques addressing its challenges (Liu and Yang, 2016; Pershina et al., 2015; Koutra et al., 2013; Buneman and Staworko, 2016) in limited settings.

[*] Correspondence: `rstrivedi@gatech.edu` and `bunyamis@amazon.com`. Work done when Rakshit Trivedi interned at Amazon.

The key challenges for the graph alignment problem emanate from the fact that the real world data are noisy and intricate in nature. The noisy or sparse data make it difficult to learn robust alignment features, and data abundance leads to computational challenges due to the combinatorial permutations needed for alignment. These challenges are compounded in multi-relational settings due to heterogeneous nodes and edges in such graphs.

Recently, deep learning has shown significant impact in learning useful information over noisy, large-scale and heterogeneous graph data (Rossi et al., 2017). We, therefore, posit that combining graph alignment task with deep representation learning across multi-relational graphs has potential to induce a synergistic effect on both tasks. Specifically, we identify that a key component of graph alignment process—entity linkage—also plays a vital role in learning across graphs. For instance, the embeddings learned over two knowledge graphs for an actor should be closer to one another compared to the embeddings of all the other entities. Similarly, the entities that are already aligned together across the two graphs should produce better embeddings due to the shared context and data. To model this phenomenon, we propose **LinkNBed**, a novel deep learning framework that jointly performs representation learning and graph linkage task. To achieve this, we identify key challenges involved in the learning process and make the following contributions to address them:

- We propose novel and principled approach towards jointly learning entity representations and entity linkage. The novelty of our framework stems from its ability to support linkage task across heterogeneous types of entities.

- We devise a graph-independent inductive framework that learns functions to capture contextual information for entities and relations. It combines the structural and semantic information in individual graphs for joint inference in a principled manner.

- Labeled instances (specifically positive instances for linkage task) are typically very sparse and hence we design a novel multi-task loss function where entity linkage task is tackled in robust manner across various learning scenarios such as learning only with unlabeled instances or only with negative instances.

- We design an efficient training procedure to perform joint training in linear time in the number of triples. We demonstrate superior performance of our method on two datasets curated from Freebase and IMDB against state-of-the-art neural embedding methods.

2 Preliminaries

2.1 Knowledge Graph Representation

A knowledge graph $\mathcal{G}$ comprises of set of facts represented as triplets (e^s, r, e^o) denoting the relationship r between subject entity e^s and object entity e^o. Associated to this knowledge graph, we have a set of attributes that describe observed characteristics of an entity. Attributes are represented as set of key-value pairs for each entity and an attribute can have null (missing) value for an entity. We follow *Open World Assumption - triplets not observed in knowledge graph are considered to be missing but not false.* We assume that there are no duplicate triplets or self-loops.

2.2 Multi-Graph Relational Learning

Definition. Given a collection of knowledge graphs $\mathcal{G}$, Multi-Graph Relational Learning refers to the the task of learning information rich representations of entities and relationships across graphs. The learned embeddings can further be used to infer new knowledge in the form of link prediction or learn new labels in the form of entity linkage. We motivate our work with the setting of two knowledge graphs where given two graphs $G_1, G_2 \in \mathcal{G}$, the task is to match an entity $e_{G_1} \in G_1$ to an entity $e_{G_2} \in G_2$ if they represent the same real-world entity. We discuss a straightforward extension of this setting to more than two graphs in Section 7.

Notations. Let X and Y represent realization of two such knowledge graphs extracted from two different sources. Let n_e^X and n_e^Y represent number of entities in X and Y respectively. Similarly, n_r^X and n_r^Y represent number of relations in X and Y. We combine triplets from both X and Y to obtain set of all observed triplets $\mathcal{D} = \{(e^s, r, e^o)_p\}_{p=1}^{P}$ where P is total number of available records across from both graphs. Let $\mathcal{E}$ and $\mathcal{R}$ be the set of all entities and all relations in $\mathcal{D}$ respectively. Let $|\mathcal{E}| = n$ and $|\mathcal{R}| = m$. In addition to $\mathcal{D}$, we also have set of linkage labels $\mathcal{L}$ for entities between X and Y. Each record in $\mathcal{L}$ is represented as triplet ($e^X \in X$, $e^Y \in Y$, $l \in \{0, 1\}$) where $l = 1$ when the entities are matched and $l = 0$ otherwise.

3 Proposed Method: LinkNBed

We present a novel *inductive multi-graph* relational learning framework that learns a set of aggregator functions capable of ingesting various contextual information for both entities and relationships in multi-relational graph. These functions encode the ingested structural and semantic information into low-dimensional entity and relation embeddings. Further, we use these representations to learn a relational score function that computes how two entities are likely to be connected in a particular relationship. The key idea behind this formulation is that when a triplet is observed, the relationship between the two entities can be explained using various contextual information such as local *neighborhood* features of both entities, *attribute* features of both entities and *type* information of the entities which participate in that relationship.

We outline two key insights for establishing the relationships between embeddings of the entities over multiple graphs in our framework:

Insight 1 (Embedding Similarity): If the two entities $e^X \in X$ and $e^Y \in Y$ represent the same real-world entity then their embeddings $\mathbf{e^X}$ and $\mathbf{e^Y}$ will be close to each other.

Insight 2 (Semantic Replacement): For a given triplet $t = (e^s, r, e^o) \in X$, denote $g(t)$ as the function that computes a relational score for t using entity and relation embeddings. If there exists a matching entity $e^{s'} \in Y$ for $e^s \in X$, denote $t' = (e^{s'}, r, e^o)$ obtained after replacing e^s with $e^{s'}$. In this case, $g(t) \sim g(t')$ i.e. score of triplets t and t' will be similar.

For a triplet $(e^s, r, e^o) \in \mathcal{D}$, we describe encoding mechanism of LinkNBed as three-layered architecture that computes the final output representations of $\mathbf{z}^r, \mathbf{z}^{e^s}, \mathbf{z}^{e^o}$ for the given triplet. Figure 1 provides an overview of LinkNBed architecture and we describe the three steps below:

3.1 Atomic Layer

Entities, Relations, Types and Attributes are first encoded in its basic vector representations. We use these basic representations to derive more complex contextual embeddings further.

Entities, Relations and Types. The embedding vectors corresponding to these three components are learned as follows:

$$\mathbf{v^{e^s}} = f(\mathbf{W^E e^s}) \qquad \mathbf{v^{e^o}} = f(\mathbf{W^E e^o}) \qquad (1)$$

$$\mathbf{v^r} = f(\mathbf{W^R r}) \qquad \mathbf{v^t} = f(\mathbf{W^T t}) \qquad (2)$$

where $\mathbf{v^{e^s}}, \mathbf{v^{e^o}} \in \mathbb{R}^d$. $\mathbf{e^s}$, $\mathbf{e^o} \in \mathbb{R}^n$ are “one-hot” representations of e^s and e^o respectively. $\mathbf{v^r} \in \mathbb{R}^k$ and $\mathbf{r} \in \mathbb{R}^m$ is “one-hot” representation of r. $\mathbf{v^t} \in \mathbb{R}^q$ and $\mathbf{t} \in \mathbb{R}^z$ is ”one-hot” representation of t . $\mathbf{W^E} \in \mathbb{R}^{d\times n}$, $\mathbf{W^R} \in \mathbb{R}^{k\times m}$ and $\mathbf{W^T} \in \mathbb{R}^{q\times z}$ are the entity, relation and type embedding matrices respectively. f is a nonlinear activation function (Relu in our case). $\mathbf{W^E}$, $\mathbf{W^R}$ and $\mathbf{W^T}$ can be initialized randomly or using pre-trained word embeddings or vector compositions based on name phrases of components (Socher et al., 2013).

Attributes. For a given attribute a represented as key-value pair, we use paragraph2vec (Le and Mikolov, 2014) type of embedding network to learn attribute embedding. Specifically, we represent attribute embedding vector as:

$$\mathbf{a} = f(\mathbf{W^{key} a_{key}} + \mathbf{W^{val} a_{val}}) \qquad (3)$$

where $\mathbf{a} \in \mathbb{R}^y$, $\mathbf{a_{key}} \in \mathbb{R}^u$ and $\mathbf{a_{val}} \in \mathbb{R}^v$. $\mathbf{W^{key}} \in \mathbb{R}^{y\times u}$ and $\mathbf{W^{val}} \in \mathbb{R}^{y\times v}$. $\mathbf{a_{key}}$ will be “one-hot” vector and $\mathbf{a_{val}}$ will be feature vector. Note that the dimensions of the embedding vectors do not necessarily need to be the same.

3.2 Contextual Layer

While the entity and relationship embeddings described above help to capture very generic latent features, embeddings can be further enriched to capture structural information, attribute information and type information to better explain the existence of a fact. Such information can be modeled as context of nodes and edges in the graph. To this end, we design the following canonical aggregator function that learns various contextual information by aggregating over relevant embedding vectors:

$$\mathbf{c}(z) = \text{AGG}(\{\mathbf{z}', \forall z' \in C(z)\}) \qquad (4)$$

where $\mathbf{c}(z)$ is the vector representation of the aggregated contextual information for component z. Here, component z can be either an entity or a relation. $C(z)$ is the set of components in the context of z and $\mathbf{z}'$ correspond to the vector embeddings of those components. AGG is the aggregator function which can take many forms such Mean, Max, Pooling or more complex LSTM based aggregators. It is plausible that different components in a context may have varied impact on the component for which the embedding is being learned. To account for this, we employ a soft attention mechanism where we learn attention coefficients

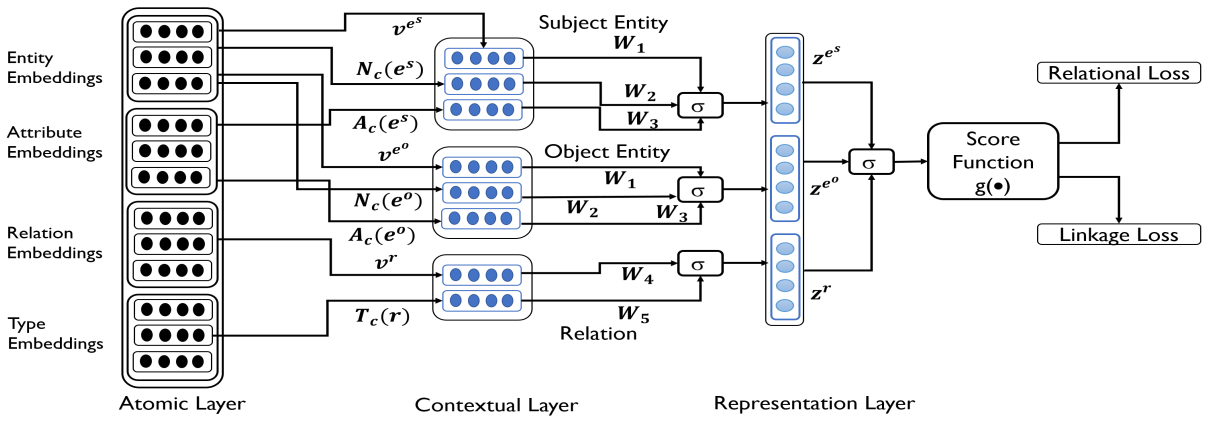


Figure 1: LinkNBed Architecture Overview - one step score computation for a given triplet (e^s, r, e^o). The Attribute embeddings are not simple lookups but they are learned as shown in Eq 3

to weight components based on their impact before aggregating them. We modify Eq. 4 as:

$$\mathbf{c}(z) = \text{AGG}(\mathbf{q}(z) * \{\mathbf{z}', \forall z' \in C(z)\}) \qquad (5)$$

where

$$\mathbf{q}(z) = \frac{\exp(\theta_z)}{\sum\limits_{z' \in C(z)} \exp(\theta_{z'})} \qquad (6)$$

and θ_z's are the parameters of attention model. Following contextual information is modeled in our framework:

Entity Neighborhood Context $\mathbf{N_c}(e) \in \mathbb{R}^d$. Given a triplet (e^s, r, e^o), the neighborhood context for an entity e^s will be the nodes located near e^s other than the node e^o. This will capture the effect of local neighborhood in the graph surrounding e^s that drives e^s to participate in fact (e^s, r, e^o). We use Mean as aggregator function. As there can be large number of neighbors, we collect the neighborhood set for each entity as a pre-processing step using a random walk method. Specifically, given a node e, we run k rounds of random-walks of length l following (Hamilton et al., 2017) and create set $\mathcal{N}(e)$ by adding all unique nodes visited across these walks. This context can be similarly computed for object entity.

Entity Attribute Context $\mathbf{A_c}(e) \in \mathbb{R}^y$. For an entity e, we collect all attribute embeddings for e obtained from Atomic Layer and learn aggregated information over them using Max operator given in Eq. 4.

Relation Type Context $\mathbf{T_c}(r) \in \mathbb{R}^q$. We use type context for relation embedding i.e. for a given relationship r, this context aims at capturing the effect of type of entities that have participated in this relationship. For a given triplet (e^s, r, e^o), type context for relationship r is computed by aggregation with mean over type embeddings corresponding to the context of r. Appendix C provides specific forms of contextual information.

3.3 Representation Layer

Having computed the atomic and contextual embeddings for a triplet (e^s, r, e^o), we obtain the final embedded representations of entities and relation in the triplet using the following formulation:

$$\mathbf{z^{e^s}} = \sigma(\underbrace{\mathbf{W_1 v^{e^s}}}_{\text{Subject Entity Embedding}} + \underbrace{\mathbf{W_2 N_c}(e^s)}_{\text{Neighborhood Context}} + \underbrace{\mathbf{W_3 A_c}(e^s))}_{\text{Subject Entity Attributes}} \qquad (7)$$

$$\mathbf{z^{e^o}} = \sigma(\underbrace{\mathbf{W_1 v^{e^o}}}_{\text{Object Entity Embedding}} + \underbrace{\mathbf{W_2 N_c}(e^o)}_{\text{Neighborhood Context}} + \underbrace{\mathbf{W_3 A_c}(e^o))}_{\text{Object Entity Attributes}} \qquad (8)$$

$$\mathbf{z^r} = \sigma(\underbrace{\mathbf{W_4 v^r}}_{\text{Relation Embedding}} + \underbrace{\mathbf{W_5 T_c}(r))}_{\text{Entity Type Context}} \qquad (9)$$

where $\mathbf{W_1}, \mathbf{W_2} \in \mathbb{R}^{d \times d}$, $\mathbf{W_3} \in \mathbb{R}^{d \times y}$, $\mathbf{W_4} \in \mathbb{R}^{d \times k}$ and $\mathbf{W_5} \in \mathbb{R}^{d \times q}$. σ is nonlinear activation function – generally Tanh or Relu.

Following is the rationale for our formulation: An entity's representation can be enriched by encoding information about the local neighborhood features and attribute information associated with the entity in addition to its own latent features. Parameters $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}$ learn to capture these different aspects and map them into the entity embedding space. Similarly, a relation's representation can be enriched by encoding information about entity types that participate in that relationship in addition to its own latent features. Parameters $\mathbf{W_4}, \mathbf{W_5}$ learn to capture these aspects and map them into the relation embedding space. Further, as the ultimate goal is to jointly learn over multiple graphs, shared parameterization in our model facilitate the propagation of information across graphs thereby making it a graph-independent inductive model. The flexibility of the model stems from the ability to shrink it (to a very simple model considering atomic entity and relation embeddings only) or expand it (to a complex model by adding different contextual information) without affecting any other step in the learning procedure.

3.4 Relational Score Function

Having observed a triplet (e^s, r, e^o), we first use Eq. 7, 8 and 9 to compute entity and relation representations. We then use these embeddings to capture relational interaction between two entities using the following score function $g(\cdot)$:

$$g(e^s, r, e^o) = \sigma(\mathbf{z}^{r^T} \cdot (\mathbf{z}^{e^s} \odot \mathbf{z}^{e^o})) \tag{10}$$

where $\mathbf{z}^r, \mathbf{z}^{e^s}, \mathbf{z}^{e^o} \in \mathbb{R}^d$ are d-dimensional representations of entity and relationships as described below. σ is the nonlinear activation function and $\odot$ represent element-wise product.

4 Efficient Learning Procedure

4.1 Objective Function

The complete parameter space of the model can be given by: $\mathbf{\Omega} = \{\{\mathbf{W_i}\}_{\mathbf{i=1}}^{\mathbf{5}}, \mathbf{W^E}, \mathbf{W^R}, \mathbf{W^{key}}, \mathbf{W^{val}}, \mathbf{W^t}, \mathbf{\Theta}\}$. To learn these parameters, we design a novel multi-task objective function that jointly trains over two graphs. As identified earlier, the goal of our model is to leverage the available linkage information across graphs for optimizing the entity and relation embeddings such that they can explain the observed triplets across the graphs. Further, we want to leverage these optimized embeddings to match entities across graphs and expand the available linkage information. To achieve this goal, we define following two different loss functions catering to each learning task and jointly optimize over them as a multi-task objective to learn model parameters:

Relational Learning Loss. This is conventional loss function used to learn knowledge graph embeddings. Specifically, given a p-th triplet $(e^s, r, e^o)_p$ from training set $\mathcal{D}$, we sample C negative samples by replacing either head or tail entity and define a contrastive max margin function as shown in (Socher et al., 2013):

$$\begin{aligned} L_{rel} = \sum_{c=1}^{C} \max(0, \gamma - g(e_p^s, r_p, e_p^o) \\ + g'(e_c^s, r_p, e_p^o)) \end{aligned} \tag{11}$$

where, γ is margin, e_c^s represent corrupted entity and $g'(e_c^s, r_p, e_p^o)$ represent corrupted triplet score.

Linkage Learning Loss: We design a novel loss function to leverage pairwise label set $\mathcal{L}$. Given a triplet (e_X^s, r_X, e_X^o) from knowledge graph X, we first find the entity e_Y^+ from graph Y that represent the same real-world entity as e_X^s. We then replace e_X^s with e_Y^+ and compute score $g(e_Y^+, r_X, e_X^o)$. Next, we find set of all entities E_Y^- from graph Y that has a negative label with entity e_X^s. We consider them analogous to the negative samples we generated for Eq. 11. We then propose the label learning loss function as:

$$\begin{aligned} L_{lab} = \sum_{z=1}^{Z} \max(0, \gamma - g(e_Y^+, r_X, e_X^o) \\ + (g'(e_Y^-, r_X, e_X^o)_z)) \end{aligned} \tag{12}$$

where, Z is the total number of negative labels for e_X. γ is margin which is usually set to 1 and $e_Y^- \in E_Y^-$ represent entity from graph Y with which entity e_X^s had a negative label. Please note that this applies symmetrically for the triplets that originate from graph Y in the overall dataset. Note that if both entities of a triplet have labels, we will include both cases when computing the loss. Eq. 12 is inspired by *Insight 1* and *Insight 2* defined earlier in Section 2. Given a set $\mathcal{D}$ of N observed triplets across two graphs, we define complete multi-task objective as:

$$\mathbf{L}(\mathbf{\Omega}) = \sum_{i=1}^{N} [b \cdot L_{rel} + (1-b) \cdot L_{lab}] + \lambda \left\| \mathbf{\Omega} \right\|_2^2 \tag{13}$$

Algorithm 1 LinkNBed mini-batch Training

```
Input: Mini-batch $\mathcal{M}$, Negative Sample Size $C$, Negative Label Size $Z$, Attribute data $att\_data$, Neighborhood data $nhbr\_data$, Type data $type\_data$, Positive Label Dict $pos\_dict$, Negative Label Dict $neg\_dict$
Output: Mini-batch Loss $\mathcal{L}_{\mathcal{M}}$.
$\mathcal{L}_{\mathcal{M}} = 0$
score_pos = []; score_neg = []; score_pos_lab = []; score_neg_lab = []
for $i = 0$ to size($\mathcal{M}$) do
  input_tuple = $\mathcal{M}[i]$ = $(e^s, r, e^o)$
  sc = compute_triplet_score($e^s, r, e^o$) (Eq. 10)
  score_pos.append(sc)
  for $j = 0$ to $C$ do
    Select $e^s_c$ from entity list such that $e^s_c \neq e^s$ and $e^s_c \neq e^o$ and $(e^s_c, r, e^o) \notin \mathcal{D}$
    sc_neg = compute_triplet_score($e^s_c, r, e^o$)
    score_neg.append(sc_neg)
  end for
  if $e^s$ in $pos\_dict$ then
    $e^{s+}$ = positive label for $e^s$
    sc_pos_l = compute_triplet_score($e^{s+}, r, e^o$)
    score_pos_lab.append(sc_pos_l)
  end if
  for $k = 0$ to $Z$ do
    Select $e^{s-}$ from $neg\_dict$
    sc_neg_l = compute_triplet_score($e^{s-}, r, e^o$)
    score_neg_lab.append(sc_neg_l)
  end for
end for
$\mathcal{L}_{\mathcal{M}}$ += compute_minibatch_loss(score_pos, score_neg, score_pos_lab, score_neg_lab) (Eq. 13)
Back-propagate errors and update parameters $\mathbf{\Omega}$
return $\mathcal{L}_{\mathcal{M}}$
```

where $\mathbf{\Omega}$ is set of all model parameters and λ is regularization hyper-parameter. b is weight hyper-parameter used to attribute importance to each task. We train with mini-batch SGD procedure (Algorithm 1) using Adam Optimizer.

Missing Positive Labels. It is expensive to obtain positive labels across multiple graphs and hence it is highly likely that many entities will not have positive labels available. For those entities, we will modify Eq. 12 to use the original triplet (e^s_X, r_X, e^o_X) in place of perturbed triplet $g(e^+_Y, r_X, e^o_X)$ for the positive label. The rationale here again arises from *Insight 2* wherein embeddings of two duplicate entities should be able to replace each other without affecting the score.

Training Time Complexity. Most contextual information is pre-computed and available to all training steps which leads to constant time embedding lookup for those context. But for attribute network, embedding needs to be computed for each attribute separately and hence the complexity to compute score for one triplet is $\mathcal{O}(2a)$ where a is number of attributes. Also for training, we generate C negative samples for relational loss function and use Z negative labels for label loss function. Let $k = C + Z$. Hence, the training time complexity for a set of n triplets will be $\mathcal{O}(2ak * n)$ which is linear in number of triplets with a constant factor as $ak << n$ for real world knowledge graphs. This is desirable as the number of triplets tend to be very large per graph in multi-relational settings.

Memory Complexity. We borrow notations from (Nickel et al., 2016a) and describe the parameter complexity of our model in terms of the number of each component and corresponding embedding dimension requirements. Let $H_a = 2 * N_e H_e + N_r H_r + N_t H_t + N_k H_k + N_v H_v$. The parameter complexity of our model is: $H_a * (H_b + 1)$. Here, N_e, N_r, N_t, N_k, N_v signify number of entities, relations, types, attribute keys and vocab size of attribute values across both datasets. Here H_b is the output dimension of the hidden layer.

5 Experiments

5.1 Datasets

We evaluate LinkNBed and baselines on two real world knowledge graphs: D-IMDB (derived from large scale IMDB data snapshot) and D-FB (derived from large scale Freebase data snapshot). Table 5.1 provides statistics for our final dataset used in the experiments. Appendix B.1 provides complete details about dataset processing.

Dataset Name	# Entities	# Relations	# Attributes	# Entity Types	# Available Triples
D-IMDB	378207	38	23	41	143928582
D-FB	39667	146	69	324	22140475

Table 1: Statistics for Datasets: D-IMDB and D-FB

5.2 Baselines

We compare the performance of our method against state-of-the-art representation learning baselines that use neural embedding techniques to learn entity and relation representation. Specifically, we consider compositional methods of

RESCAL (Nickel et al., 2011) as basic matrix factorization method, DISTMULT (Yang et al., 2015) as simple multiplicative model good for capturing symmetric relationships, and Complex (Trouillon et al., 2016), an upgrade over DISTMULT that can capture asymmetric relationships using complex valued embeddings. We also compare against translational model of STransE that combined original structured embedding with TransE and has shown state-of-art performance in benchmark testing (Kadlec et al., 2017). Finally, we compare with GAKE (Feng et al., 2016), a model that captures context in entity and relationship representations.

In addition to the above state-of-art models, we analyze the effectiveness of different components of our model by comparing with various versions that use partial information. Specifically, we report results on following variants:
LinkNBed - Embed Only. Only use entity embeddings, **LinkNBed - Attr Only.** Only use Attribute Context, **LinkNBed - Nhbr Only.** Only use Neighborhood Context, **LinkNBed - Embed + Attr.** Use both Entity embeddings and Attribute Context, **LinkNBed - Embed + Nhbr.** Use both Entity embeddings and Neighbor Context and **LinkNBed - Embed All.** Use all three Contexts.

5.3 Evaluation Scheme

We evaluate our model using two inference tasks:
Link Prediction. Given a test triplet (e^s, r, e^o), we first score this triplet using Eq. 10. We then replace e^o with all other entities in the dataset and filter the resulting set of triplets as shown in (Bordes et al., 2013). We score the remaining set of perturbed triplets using Eq. 10. All the scored triplets are sorted based on the scores and then the rank of the ground truth triplet is used for the evaluation. We use this ranking mechanism to compute HITS@10 (predicted rank $\leq$ 10) and reciprocal rank ($\frac{1}{rank}$) of each test triplet. We report the mean over all test samples.

Entity Linkage. In alignment with *Insight 2*, we pose a novel evaluation scheme to perform entity linkage. Let there be two ground truth test sample triplets: $(e_X, e_Y^+, 1)$ representing a positive duplicate label and $(e_X, e_Y^-, 0)$ representing a negative duplicate label. Algorithm 2 outlines the procedure to compute linkage probability or score q ($\in [0, 1]$) for the pair (e_X, e_Y). We use $L1$ distance between the two vectors analogous to Mean Absolute Error (MAE). In lieu of hard-labeling test pairs, we use score q to compute Area Under the Precision-Recall Curve (AUPRC).

For the baselines and the unsupervised version (with no labels for entity linkage) of our model, we use second stage multilayer Neural Network as classifier for evaluating entity linkage. Appendix B.2 provides training configuration details.

Algorithm 2 Entity Linkage Score Computation

```
Input: Test pair – $(e_X \in X, e_Y \in Y)$.
Output: Linkage Score – $q$.

1. Collect all triplets involving $e_X$ from graph $X$ and all triplets involving $e_Y$ from graph $Y$ into a combined set $\mathcal{O}$. Let $|\mathcal{O}| = k$.
2. Construct $S_{orig} \in \mathbb{R}^k$.
For each triplet $o \in \mathcal{O}$, compute score $g(o)$ using Eq. 10 and store the score in $S_{orig}$.
3. Create triplet set $\mathcal{O}'$ as following:
if $o \in \mathcal{O}$ contain $e^X \in X$ then
    Replace $e^X$ with $e^Y$ to create perturbed triplet $o'$ and store it in $\mathcal{O}'$
end if
if $o \in \mathcal{O}$ contain $e^Y \in Y$ then
    Replace $e^Y$ with $e^X$ to create perturbed triplet $o'$ and store it in $\mathcal{O}'$
end if
4. Construct $S_{repl} \in \mathbb{R}^k$.
For each triplet $o' \in \mathcal{O}'$, compute score $g(o')$ using Eq. 10 and store the score in $S_{repl}$.
5. Compute $q$.
Elements in $S_{orig}$ and $S_{repl}$ have one-one correspondence so take the mean absolute difference:
$q$ = $|S_{orig}$ - $S_{repl}|_1$
return $q$
```

5.4 Predictive Analysis

Link Prediction Results. We train LinkNBed model jointly across two knowledge graphs and then perform inference over individual graphs to report link prediction reports. For baselines, we train each baseline on individual graphs and use parameters specific to the graph to perform link prediction inference over each individual graph. Table 5.4 shows link prediction performance for all methods. Our model variant with attention mechanism outperforms all the baselines with 4.15% improvement over single graph state-of-the-art Complex model on D-IMDB and 8.23% improvement on D-FB dataset. D-FB is more challenging dataset to

Method	D-IMDB-HITS10	D-IMDB-MRR	D-FB-HITS10	D-FB-MRR
RESCAL	75.3	0.592	69.99	0.147
DISTMULT	79.5	0.691	72.34	0.556
Complex	83.2	0.752	75.67	0.629
STransE	80.7	0.421	69.87	0.397
GAKE	69.5	0.114	63.22	0.093
LinkNBed-Embed Only	79.9	0.612	73.2	0.519
LinkNBed-Attr Only	82.2	0.676	74.7	0.588
LinkNBed-Nhbr Only	80.1	0.577	73.4	0.572
LinkNBed-Embed + Attr	84.2	0.673	78.39	0.606
LinkNBed-Embed + Nhbr	81.7	0.544	73.45	0.563
LinkNBed-Embed All	84.3	0.725	80.2	0.632
LinkNBed-Embed All (Attention)	**86.8**	**0.733**	**81.9**	**0.677**
Improvement (%)	**4.15**	**1.10**	**8.23**	**7.09**

Table 2: Link Prediction Results on both datasets

Method	AUPRC (Supervised)	AUPRC (Unsupervised)
RESCAL	-	0.327
DISTMULT	-	0.292
Complex	-	0.359
STransE	-	0.231
GAKE	-	**0.457**
LinkNBed-Embed Only	0.376	0.304
LinkNBed-Attr Only	0.451	0.397
LinkNBed-Nhbr Only	0.388	0.322
LinkNBed-Embed + Attr	0.512	0.414
LinkNBed-Embed + Nhbr	0.429	0.356
LinkNBed-Embed All	0.686	0.512
LinkNBed-Embed All (Attention)	**0.691**	**0.553**
Improvement (%)	**33.86**	**17.35**

Table 3: Entity Linkage Results - Unsupervised case uses classifier at second step

learn as it has a large set of sparse relationships, types and attributes and it has an order of magnitude lesser relational evidence (number of triplets) compared to D-IMDB. Hence, LinkNBed's pronounced improvement on D-FB demonstrates the effectiveness of the model. The simplest version of LinkNBed with only entity embeddings resembles DISTMULT model with different objective function. Hence closer performance of those two models aligns with expected outcome. We observed that the Neighborhood context alone provides only marginal improvements while the model benefits more from the use of attributes. Despite being marginal, attention mechanism also improves accuracy for both datasets. Compared to the baselines which are obtained by trained and evaluated on individual graphs, our superior performance demonstrates the effectiveness of multi-graph learning.

Entity Linkage Results. We report entity linkage results for our method in two settings: a.) Supervised case where we train using both the objective functions. b.) Unsupervised case where we learn with only the relational loss function. The latter case resembles the baseline training where each model is trained separately on two graphs in an unsupervised manner. For performing the entity linkage in unsupervised case for all models, we first train a second stage of simple neural network classifier and then perform inference. In the supervised case, we use Algorithm 2 for performing the inference. Table 5.4 demonstrates the performance of all methods on this task. Our method significantly outperforms all the baselines with 33.86% over second best baseline in supervised case and 17.35% better performance in unsupervised case. The difference in the performance of our method in two cases demonstrate that the two training objectives are helping one another by learning across the graphs. GAKE's superior performance on this task compared to the other state-of-the-art relational baselines shows the importance of using contextual information for entity linkage. Performance of other variants of our model again demonstrate that attribute information is more helpful than neighborhood context and attention provides marginal improvements. We provide further insights with examples and detailed discussion on entity linkage task in Appendix A.

6 Related Work

6.1 Neural Embedding Methods for Relational Learning

Compositional Models learn representations by various composition operators on entity and relational embeddings. These models are multiplicative in nature and highly expressive but often suffer from scalability issues. Initial models include RESCAL (Nickel et al., 2011) that uses a relation specific weight matrix to explain triplets via pairwise interactions of latent features, Neural Tensor Network (Socher et al., 2013), more expressive model that combines a standard NN layer with a bilinear tensor layer and (Dong et al., 2014) that employs a concatenation-projection method to project entities and relations to lower dimensional space. Later, many sophisticated models (Neural Association Model (Liu et al., 2016), HoLE (Nickel et al., 2016b)) have been proposed. Path based composition models (Toutanova et al., 2016) and contextual models GAKE (Feng et al., 2016) have been recently studied to capture more information from graphs. Recently, model like Complex (Trouillon et al., 2016) and Analogy (Liu et al., 2017) have demonstrated state-of-the art performance on relational learning tasks. **Translational Models** ((Bordes et al., 2014), (Bordes et al., 2011), (Bordes et al., 2013), (Wang et al., 2014), (Lin et al., 2015), (Xiao et al., 2016)) learn representation by employing translational operators on the embeddings and optimizing based on their score. They offer an additive and efficient alternative to expensive multiplicative models. Due to their simplicity, they often loose expressive power. For a comprehensive survey of relational learning methods and empirical comparisons, we refer the readers to (Nickel et al., 2016a), (Kadlec et al., 2017), (Toutanova and Chen, 2015) and (Yang et al., 2015). None of these methods address multi-graph relational learning and cannot be adapted to tasks like entity linkage in straightforward manner.

6.2 Entity Resolution in Relational Data

Entity Resolution refers to resolving entities available in knowledge graphs with entity mentions in text. (Dredze et al., 2010) proposed entity disambiguation method for KB population, (He et al., 2013) learns entity embeddings for resolution, (Huang et al., 2015) propose a sophisticated DNN architecture for resolution, (Campbell et al., 2016) proposes entity resolution across multiple social domains, (Fang et al., 2016) jointly embeds text and knowledge graph to perform resolution while (Globerson et al., 2016) proposes Attention Mechanism for Collective Entity Resolution.

6.3 Learning across multiple graphs

Recently, learning over multiple graphs have gained traction. (Liu and Yang, 2016) divides a multi-relational graph into multiple homogeneous graphs and learns associations across them by employing product operator. Unlike our work, they do not learn across multiple multi-relational graphs. (Pujara and Getoor, 2016) provides logic based insights for cross learning, (Pershina et al., 2015) does pairwise entity matching across multi-relational graphs and is very expensive, (Chen et al., 2017) learns embeddings to support multi-lingual learning and Big-Align (Koutra et al., 2013) tackles graph alignment problem efficiently for bipartite graphs. None of these methods learn latent representations or jointly train graph alignment and learning which is the goal of our work.

7 Concluding Remarks and Future Work

We present a novel relational learning framework that learns entity and relationship embeddings across multiple graphs. The proposed representation learning framework leverage an efficient learning and inference procedure which takes into account the duplicate entities representing the same real-world entity in a multi-graph setting. We demonstrate superior accuracies on link prediction and entity linkage tasks compared to the existing approaches that are trained only on individual graphs. We believe that this work opens a new research direction in joint representation learning over multiple knowledge graphs.

Many data driven organizations such as Google and Microsoft take the approach of constructing a unified super-graph by integrating data from multiple sources. Such unification has shown to significantly help in various applications, such as search, question answering, and personal assistance. To this end, there exists a rich body of work on linking entities and relations, and conflict resolution (e.g., knowledge fusion (Dong et al., 2014). Still, the problem remains challenging for large scale knowledge graphs and this paper proposes a deep learning solution that can play a vital role in this construction process. In real-world setting, we envision our method to be integrated in a large scale system that would include various other components for tasks like conflict resolution, active learning and human-in-loop learning to ensure quality of constructed super-graph. However, we point out that our method is not restricted to such use cases—one can readily apply our method to directly make inference over multiple graphs to support applications like question answering and conversations.

For future work, we would like to extend the current evaluation of our work from a two-graph setting to multiple graphs. A straightforward approach is to create a unified dataset out of more than two graphs by combining set of triplets as described in Section 2, and apply learning and inference on the unified graph without any major change in the methodology. Our inductive framework learns functions to encode contextual information and hence is graph independent. Alternatively, one can develop sophisticated approaches with iterative merging and learning over pairs of graphs until exhausting all graphs in an input collection.

Acknowledgments

We would like to give special thanks to Ben London, Tong Zhao, Arash Einolghozati, Andrew Borthwick and many others at Amazon for helpful comments and discussions. We thank the reviewers for their valuable comments and efforts towards improving our manuscript. This project was supported in part by NSF(IIS-1639792, IIS-1717916).

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Peter Buneman and Slawek Staworko. 2016. Rdf graph alignment with bisimulation. *Proc. VLDB Endow.*

W. M. Campbell, Lin Li, C. Dagli, J. Acevedo-Aviles, K. Geyer, J. P. Campbell, and C. Priebe. 2016. Cross-domain entity resolution in social media. *arXiv:1608.01386v1*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

Rose Catherine and William Cohen. 2016. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: Learning question answering over qa corpora and knowledge bases. *Proc. VLDB Endow.*

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *CoNLL*.

Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. 2016. Gake: Graph aware knowledge embedding. In *COLING*.

Evgeniy Gabrilovich and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation for natural language processing. *J. Artif. Int. Res.*

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv:1709.05584*.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv:1504.07678v1*.

Rudolph Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

Danai Koutra, HangHang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. *AAAI Conference on Artificial Intelligence*.

Hanxiao Liu, Yuexin Wu, and Yimin Yang. 2017. Analogical inference for multi-relatinal embeddings. In *Proceedings of the 34th International Conference on Machine Learning*.

Hanxiao Liu and Yimin Yang. 2016. Cross-graph learning of multi-relational associations. In *Proceedings of the 33rd International Conference on Machine Learning*.

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv:1603.07704v2*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic embeddings of knowledge graphs.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 809–816.

Maria Pershina, Mohamed Yakout, and Kaushik Chakrabarti. 2015. Holistic entity matching across knowledge graphs. In *2015 IEEE International Conference on Big Data (Big Data)*.

Jay Pujara and Lise Getoor. 2016. Generic statistical relational entity resolution in knowledge graphs. In *Sixth International Workshop on Statistical Relational AI*.

Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Deep feature learning for graphs. *arXiv:1704.08829*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *ACL*.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1434–1444.

Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Transg: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *arXiv:1412.6575*.

Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures

Luke Vilnis* **Xiang Li*** **Shikhar Murty** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{luke,xiangl,smurty,mccallum}@cs.umass.edu

Abstract

Embedding methods which enforce a partial order or lattice structure over the concept space, such as Order Embeddings (OE) (Vendrov et al., 2016), are a natural way to model transitive relational data (e.g. entailment graphs). However, OE learns a deterministic knowledge base, limiting expressiveness of queries and the ability to use uncertainty for both prediction and learning (e.g. learning from expectations). Probabilistic extensions of OE (Lai and Hockenmaier, 2017) have provided the ability to somewhat calibrate these *denotational probabilities* while retaining the consistency and inductive bias of ordered models, but lack the ability to model the negative correlations found in real-world knowledge. In this work we show that a broad class of models that assign probability measures to OE can never capture negative correlation, which motivates our construction of a novel *box lattice* and accompanying probability measure to capture anticorrelation and even disjoint concepts, while still providing the benefits of probabilistic modeling, such as the ability to perform rich joint and conditional queries over arbitrary sets of concepts, and both learning from and predicting calibrated uncertainty. We show improvements over previous approaches in modeling the Flickr and WordNet entailment graphs, and investigate the power of the model.

* Equal contribution.

1 Introduction

Structured embeddings based on regions, densities, and orderings have gained popularity in recent years for their inductive bias towards the essential asymmetries inherent in problems such as image captioning (Vendrov et al., 2016), lexical and textual entailment (Erk, 2009; Vilnis and McCallum, 2015; Lai and Hockenmaier, 2017; Athiwaratkun and Wilson, 2018), and knowledge graph completion and reasoning (He et al., 2015; Nickel and Kiela, 2017; Li et al., 2017).

Models that easily encode asymmetry, and related properties such as transitivity (the two components of commonplace relations such as partially ordered sets and lattices), have great utility in these applications, leaving less to be learned from the data than arbitrary relational models. At their best, they resemble a hybrid between embedding models and structured prediction. As noted by Vendrov et al. (2016) and Li et al. (2017), while the models learn sets of embeddings, these parameters obey rich structural constraints. The entire set can be thought of as one, sometimes provably consistent, structured prediction, such as an ontology in the form of a single directed acyclic graph.

While the structured prediction analogy applies best to Order Embeddings (OE), which embeds consistent partial orders, other region- and density-based representations have been proposed for the express purpose of inducing a bias towards asymmetric relationships. For example, the Gaussian Embedding (GE) model (Vilnis and McCallum, 2015) aims to represent the asymmetry and uncertainty in an object's relations and attributes by means of uncertainty in the representation. However, while the space of representations is a manifold of probability distributions, the model is not truly probabilistic in that it does not model asymmetries and relations in terms of probabilities, but in terms of asymmetric comparison functions such as the originally proposed KL divergence and the recently proposed *thresholded divergences* (Athiwaratkun and Wilson, 2018).

Probabilistic models are especially compelling for modeling ontologies, entailment graphs, and knowledge graphs. Their desirable properties include an ability to remain consistent in the presence of noisy data, suitability towards semi-supervised training using the expectations and uncertain labels present in these large-scale applications, the naturality of representing the inherent uncertainty of knowledge they store, and the ability to answer complex queries involving more than 2 variables. Note that the final one requires a true joint probabilistic model with a tractable inference procedure, not something provided by e.g. matrix factorization.

We take the dual approach to density-based embeddings and model uncertainty about relationships and attributes as explicitly probabilistic, while basing the probability on a latent space of geometric objects that obey natural structural biases for modeling transitive, asymmetric relations. The most similar work are the probabilistic order embeddings (POE) of Lai (Lai and Hockenmaier, 2017), which apply a probability measure to each order embedding's forward cone (the set of points greater than the embedding in each dimension), assigning a finite and normalized volume to the unbounded space. However, POE suffers severe limitations as a probabilistic model, including an inability to model negative correlations between concepts, which motivates the construction of our box lattice model.

Our model represents objects, concepts, and events as high-dimensional products-of-intervals (*hyperrectangles* or *boxes*), with an event's unary probability coming from the box volume and joint probabilities coming from overlaps. This contrasts with POE's approach of defining events as the forward cones of vectors, extending to infinity, integrated under a probability measure that assigns them finite volume.

One desirable property of a structured representation for ordered data, originally noted in (Vendrov et al., 2016) is a "slackness" shared by OE, POE, and our model: when the model predicts an "edge" or lack thereof (i.e. $P(a|b) = 0$ or 1, or a zero constraint violation in the case of OE), being exposed to that fact again will not update the model. Moreover, there are large degrees of freedom in parameter space that exhibit this slackness, giving the model the ability to embed complex structure with 0 loss when compared to models based on symmetric inner products or distances between embeddings, e.g. bilinear GLMs (Collins et al., 2002), Trans-E (Bordes et al., 2013), and other embedding models which must always be pushing and pulling parameters towards and away from each other.

Our experiments demonstrate the power of our approach to probabilistic ordering-biased relational modeling. First, we investigate an instructive 2-dimensional toy dataset that both demonstrates the way the model self organizes its box event space, and enables sensible answers to queries involving arbitrary numbers of variables, despite being trained on only pairwise data. We achieve a new state of the art in denotational probability modeling on the Flickr entailment dataset (Lai and Hockenmaier, 2017), and a matching state-of-the-art on WordNet hypernymy (Vendrov et al., 2016; Miller, 1995) with the concurrent work on thresholded Gaussian embedding of Athiwaratkun and Wilson (2018), achieving our best results by training on additional co-occurrence expectations aggregated from leaf types.

We find that the strong empirical performance of probabilistic ordering models, and our box lattice model in particular, and their endowment of new forms of training and querying, make them a promising avenue for future research in representing structured knowledge.

2 Related Work

In addition to the related work in structured embeddings mentioned in the introduction, our focus on directed, transitive relational modeling and ontology induction shares much with the rich field of directed graphical models and causal modeling (Pearl, 1988), as well as learning the structure of those models (Heckerman et al., 1995). Work in undirected structure learning such the Graphical Lasso (Friedman et al., 2008) is also relevant due to our desire to learn from pairwise joint/conditional probabilities and moment matrices, which are closely related in the setting of discrete variables.

Especially relevant research in Bayesian networks are applications towards learning taxonomic structure of relational data (Bansal et al., 2014), although this work is often restricted towards tree-shaped ontologies, which allow efficient inference by Chu-Liu-Edmonds' algorithm (Chu and Liu, 1995), while we focus on arbitrary DAGs.

As our model is based on populating a latent "event space" into boxes (products of intervals), it is especially reminiscent of the Mondrian process (Roy and Teh, 2009). However, the Mondrian process partitions the space as a high dimensional tree (a non-parametric kd-tree), while our model allows the arbitrary box placement required for DAG structure, and is much more tractable in high dimensions compared to the Mondrian's Bayesian non-parametric inference.

Embedding applications to relational learning constitute a huge field to which it is impossible to do justice, but one general difference between our approaches is that e.g. a matrix factorization model treats the embeddings as objects to score relation links with, as opposed to POE or our model in which embeddings represent subsets of probabilistic event space which are directly integrated. They are full probabilistic models of the joint set of variables, rather than embedding-based approximations of only low-order joint and conditional probabilities. That is, any set of our parameters can answer any arbitrary probabilistic question (possibly requiring intractable computation), rather than being fixed to modeling only certain subsets of the joint.

Embedding-based learning's large advantage over the combinatorial structure learning presented by classical PGM approaches is its applicability to large-scale probability distributions containing hundreds of thousands of events or more, as in both our WordNet and Flickr experiments.

3 Background

3.1 Partial Orders and Lattices

A non-strict *partial ordered set* (*poset*) is a set P equipped with a binary relation $\preceq$ such that for all $a, b, c \in P$,

- $a \preceq a$ (reflexivity)
- $a \preceq b \preceq a$ implies $a = b$ (antisymmetry)
- $a \preceq b \preceq c$ implies $a \preceq c$ (transitivity)

This is simply a generalization of a totally ordered set that allows some elements to be incomparable, and is a good model for the kind of acyclic directed graph data found in knowledge bases.

A *lattice* is a poset where any subset has a a unique least upper and greatest lower bound, which will be true of all posets (lattices) considered in this paper. The least upper bound of two elements $a, b \in P$ is called the *join*, denoted $a \vee b$, and the greatest lower bound is called the *meet*, denoted $a \wedge b$.

Additionally, in a *bounded lattice* we have two extra elements, called *top*, denoted $\top$ and *bottom*, denoted $\perp$, which are respectively the least upper bound and greatest lower bound of the entire space. Using the extended real number line (adding points at infinity), all lattices considered in this paper are bounded lattices.

3.2 Order Embeddings (OE)

Vendrov et al. (2016) introduced a method for embedding partially ordered sets and a task, *partial order completion*, an abstract term for things like hypernym or entailment prediction (learning transitive relations). The goal is to learn a mapping from the partially-ordered data domain to some other partially-ordered space that will enable generalization.

Definition 1. *Vendrov et al. (2016)*
A function $f : (X, \preceq_X) \to (Y, \preceq_Y)$ is an order-embedding *if for all $u, v \in X$*

$$u \preceq_X v \iff f(u) \preceq_Y f(v)$$

They choose Y to be a vector space, and the order $\preceq_Y$ to be based on the *reverse product order* on $\mathbb{R}^n_+$, which specifies

$$x \preceq y \iff \forall i \in \{1..n\}, \ x_i \geq y_i$$

so an embedding is below another in the hierarchy if all of the coordinates are larger, and 0 provides a top element.

Although Vendrov et al. (2016) do not explicitly discuss it, their model does not just capture partial orderings, but is a standard construction of a vector (Hilbert) lattice, in which the operations of meet and join can be defined as taking the pointwise maximum and minimum of two vectors, respectively (Zaanen, 1997). This observation is also used in (Li et al., 2017) to generate extra constraints for training order embeddings.

As noted in the original work, these single point embeddings can be thought of as regions, i.e. the cone extending out from the vector towards infinity. All concepts "entailed" by a given concept must lie in this cone.

This ordering is optimized from examples of ordered elements and negative samples via a max-margin loss.

3.3 Probabilistic Order Embeddings (POE)

Lai and Hockenmaier (2017) built on the "region" idea to derive a probabilistic formulation (which we will refer to as POE) to model entailment probabilities in a consistent, hierarchical way.

Noting that all of OE's regions obviously have the same infinite area under the standard (Lebesgue) measure of $\mathbb{R}^n_+$, they propose a probabilistic interpretation where the Bernoulli probability of each concept a or joint set of concepts $\{a, b\}$ with corresponding vectors $\{x, y\}$ is given by its volume under the exponential measure:

$$p(a) = \exp(-\sum_i x_i) = \int_{z \preceq x} \exp(-\|z\|_1) dz$$
$$p(a, b) = p(x \wedge y) = \exp(-\|\max(x_i, y_i)\|_1)$$

since the meet of two vectors is simply the intersection of their area cones, and replacing sums with ℓ_1 norms for brevity since all coordinates are positive. While having the intuition of measuring the areas of cones, this also automatically gives a valid probability distribution over concepts since this is just the product likelihood under a coordinatewise exponential distribution.

However, they note a deficiency of their model — it can only model positive (Pearson) correlations between concepts (Bernoulli variables).

Consider two Bernoulli variables a and b, whose probabilities correspond to the areas of cones x and y. Recall the Bernoulli covariance formula (we will deal with covariances instead of correlations when convenient, since they always have the same sign):

$$\text{cov}(a, b) = p(a, b) - p(a)p(b) =$$
$$\exp(-\|\max(x_i, y_i)\|_1) - \exp(-\|x_i + y_i\|_1)$$

Since the sum of two positive vectors can only be greater than the sum of their pointwise maximum, this quantity will always be nonnegative. This has real consequences for probabilistic modeling in KBs: conditioning on more concepts will only make probabilities higher (or unchanged), e.g. $p(\text{dog}|\text{plant}) \geq p(\text{dog})$.

3.4 Probabilistic Asymmetric Transitive Relations

Probabilistic models have pleasing consistency properties for modeling asymmetric transitive relations, in particular compared to density-based embeddings — a pairwise conditional probability table can almost always (in the technical sense) be asymmetrized to produce a DAG by simply taking an edge if $P(a|b) > P(b|a)$. A matrix of pairwise Gaussian KL divergences cannot be consistently asymmetrized in this manner. These claims are proven in Appendix C. While a high $P(a|b)$ does not always indicate an edge in an ontology due to confounding variables, existing graphical model structure learning methods can be used to further prune on the base graph without adding a cycle, such as Graphical Lasso or simple thresholding (Fattahi and Sojoudi, 2017).

4 Method

We develop a probabilistic model for lattices based on hypercube embeddings that can model both positive and negative correlations. Before describing this, we first motivate our choice to abandon OE/POE type cone-based models for this purpose.

4.1 Correlations from Cone Measures

Claim. *For a pair of Bernoulli variables $p(a)$ and $p(b)$, $cov(a, b) \geq 0$ if the Bernoulli probabilities come from the volume of a cone as measured under any product (coordinate-wise) probability measure $p(x) = \prod_i^n p_i(x_i)$ on $\mathbb{R}^n$, where F_i, the associated CDF for p_i, is monotone increasing.*

Proof. For any product measure we have

$$\int_{z \preceq x} p(z) dz = \prod_i^n \int_{x_i \leq z_i} p_i(z_i) dz_i = \prod_i^n 1 - F_i(x_i)$$

This is just the area of the unique box corresponding to $\prod_i^n [F_i(x_i), 1] \in [0, 1]^n$, under the uniform measure. This box is unique as a monotone increasing univariate CDF is bijective with $(0, 1)$ — cones in $\mathbb{R}^n$ can be invertibly mapped to boxes of equivalent measure inside the unit hypercube $[0, 1]^n$. These boxes have only half their degrees of freedom, as they have the form $[F_i(x_i), 1]$ per dimension, (intuitively, they have one end "stuck at infinity" since the cone integrates to infinity.

So W.L.O.G. we can consider two transformed cones x and y corresponding to our Bernoulli variables a and b, and letting $F_i(x_i) = u_i$ and $F_i(y_i) = v_i$, their intersection in the unit hypercube is $\prod_i^n [\max(u_i, v_i), 1]$.

Pairing terms in the right-hand product, we have

$$p(a,b) - p(a)p(b) =$$
$$\prod_i^n (1 - \max(u_i, v_i)) - \prod_i^n (1 - u_i)(1 - v_i) \geq 0$$

since the right contains all the terms of the left and can only grow smaller. This argument is easily modified to the case of the nonnegative orthant, *mutatis mutandis*. □

An open question for future work is what non-product measures this claim also applies to. Note that some non-product measures, such as multivariate Gaussian, can be transformed into product measures easily (*whitening*) and the above proof would still apply. It seems probable that some measures, nonlinearly entangled across dimensions, could encode negative correlations in cone volumes. However, it is not generally tractable to integrate high-dimensional cones under arbitrary non-product measures.

4.2 Box Lattices

The above proof gives us intuition about the possible form of a better representation. Cones can be mapped into boxes within the unit hypercube while preserving their measure, and the lack of negative correlation seems to come from the fact that they always have an overly-large intersection due to "pinning" the maximum in each dimension to 1. To remedy this, we propose to learn representations in the space of *all* boxes (axis-aligned hyperrectangles), gaining back an extra degree of freedom. These representations can be learned with a suitable probability measure in $\mathbb{R}^n$, the nonnegative orthant $\mathbb{R}^n_+$, or directly in the unit hypercube with the uniform measure, which we elect.

We associate each concept with 2 vectors, the minimum and maximum value of the box at each dimension. Practically for numerical reasons these are stored as a minimum, a positive offset plus an ϵ term to prevent boxes from becoming too small and underflowing.

Let us define our box embeddings as a pair of vectors in $[0,1]^n$, (x_m, x_M), representing the maximum and minimum at each coordinate.

Then we can define a partial ordering by inclusion of boxes, and a lattice structure as

$$x \wedge y = \bot \text{ if } x \text{ and } y \text{ disjoint, else}$$
$$x \wedge y = \prod_i [\max(x_{m,i}, y_{m,i}), \min(x_{M,i}, y_{M,i})]$$
$$x \vee y = \prod_i [\min(x_{m,i}, y_{m,i}), \max(x_{M,i}, y_{M,i})]$$

where the meet is the intersecting box, or bottom (the empty set) where no intersection exists, and join is the smallest enclosing box. This lattice, considered on its own terms as a non-probabilistic object, is strictly more general than the order embedding lattice in any dimension, which is proven in Appendix B.

However, the finite sizes of all the lattice elements lead to a natural probabilistic interpretation under the uniform measure. Joint and marginal probabilities are given by the volume of the (intersection) box. For concept a with associated box (x_m, x_M), probability is simply $p(a) = \prod_i^n (x_{M,i} - x_{m,i})$ (under the uniform measure). $p(\bot)$ is of course zero since no probability mass is assigned to the empty set.

It remains to show that this representation can represent both positive and negative correlations.

Claim. *For a pair of Bernoulli variables $p(a)$ and $p(b)$, corr(a,b) can take on any value in $[-1,1]$ if the probabilities come from the volume of associated boxes in $[0,1]^n$.*

Proof. Boxes can clearly model disjointness (exactly -1 correlation if the total volume of the boxes equals 1). Two identical boxes give their concepts exactly correlation 1. The area of the meet is continuous with respect to translations of intersecting boxes, and all other terms in correlation stay constant, so by continuity of the correlation function our model can achieve all possible correlations for a pair of variables. □

This proof can be extended to boxes in $\mathbb{R}^n$ with product measures by the previous reduction.

Limitations: Note that this model cannot perfectly describe all possible probability distributions or concepts as embedded objects. For example, the complement of a box is not a box. However, queries about complemented variables can be calculated by the Inclusion-Exclusion principle, made more efficient by the fact that all non-negated terms can be grouped and calculated exactly. We show some toy exact calculations with negated variables in Appendix A. Also, note that in a knowledge graph often true complements are not required — for example *mortal* and *immortal* are not actually complements, because the concept *color* is neither.

Additionally, requiring the total probability mass covered by boxes to equal 1, or exactly matching marginal box probabilities while modeling all correlations is a difficult box-packing-type problem and not generally possible. Modeling limitations aside, the union of boxes having mass < 1 can be seen as an open-world assumption on our KB (not all points in space have corresponding concepts, yet).

4.3 Learning

While inference (calculation of pairwise joint, unary marginal, and pairwise conditional probabilities) is quite straightforward by taking intersections of boxes and computing volumes (and their ratios), learning does not appear easy at first glance. While the (sub)gradient of the joint probability is well defined when boxes intersect, it is non-differentiable otherwise. Instead we optimize a lower bound.

Clearly $p(a \vee b) \geq p(a \cup b)$, with equality only when $a = b$, so this can give us a lower bound:

$$\begin{aligned} p(a \wedge b) &= p(a) + p(b) - p(a \cup b) \\ &\geq p(a) + p(b) - p(a \vee b) \end{aligned}$$

Where probabilities are always given by the volume of the associated box. This lower bound always exists and is differentiable, even when the joint is not. It is guaranteed to be nonpositive except when a and b intersect, in which case the true joint likelihood should be used.

While a negative bound on a probability is odd, inspecting the bound we see that its gradient will push the enclosing box to be smaller, while increasing areas of the individual boxes, until they intersect, which is a sensible learning strategy.

Since we are working with small probabilities it is advisable to negate this term and maximize the negative logarithm:

$$-\log(p(a \vee b) - p(a) - p(b))$$

This still has an unbounded gradient as the lower bound approaches 0, so it is also useful to add a constant within the logarithm function to avoid numerical problems.

Since the likelihood of the full data is usually intractable to compute as a conjunction of many negations, we optimize binary conditional and unary marginal terms separately by maximum likelihood.

In this work, we parametrize the boxes as $(\min, \Delta = \max - \min)$, with Euclidean projections after gradient steps to keep our parameters in the unit hypercube and maintain the minimum/delta constraints.

Now that we have the ability to compute probabilities and (surrogate) gradients for arbitrary marginals in the model, and by extension conditionals, we will see specific examples in the experiments.

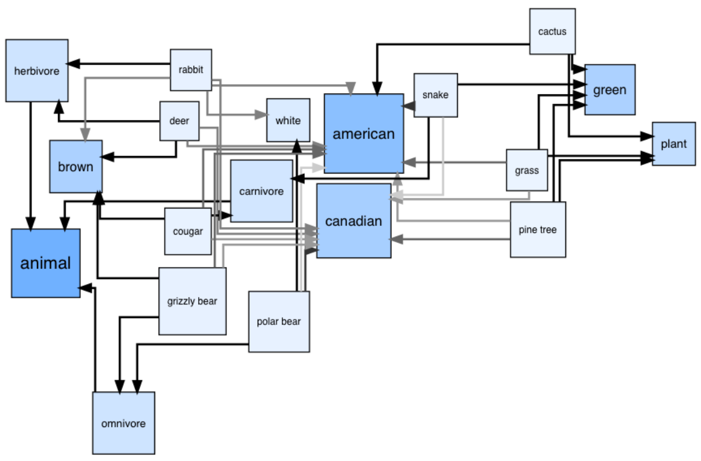
5 Experiments

5.1 Warmup: 2D Embedding of a Toy Lattice

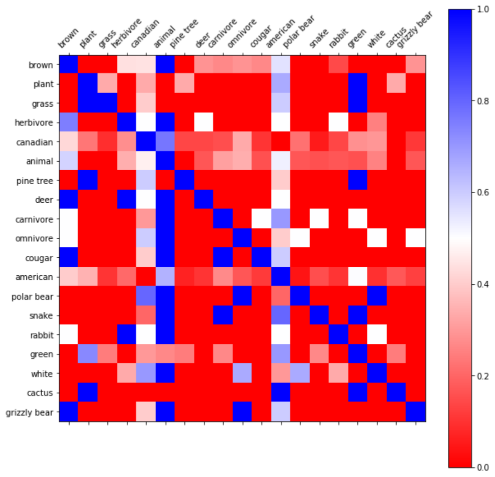
We begin by investigating properties of our model in modeling a small toy problem, consisting of a small hand constructed ontology over 19 concepts, aggregated from atomic synthetic examples first into a probabilistic lattice (e.g. some rabbits are brown, some are white), and then a full CPD. We model it using only 2 dimensions to enable visualization of the way the model self-organizes its "event space", training the model by minimize weighted cross-entropy with both the unary marginals and pairwise conditional probabilities. We also conduct a parallel experiment with POE as embedded in the unit cube, where each representation is constrained to touch the faces $x = 1, y = 1$. In Figure 2, we show the representation of lattice structures by POE and the box lattice model as compared to the abstract probabilistic lattice used to construct the data, shown in Figure 1, and compare the conditional probabilities produced by our model to the ground truth, demonstrating the richer capacity of the box model in capturing strong positive and negative correlations. In Table 1, we perform a series of multivariable conditional queries and demonstrate intuitive results on high-order queries containing up to 4 variables, despite the model being trained on only 2-way information.

5.2 WordNet

We experiment on WordNet hypernym prediction, using the same train, development and test split as Vendrov et al. (2016), created by randomly taking 4,000 hypernym pairs from the 837,888-

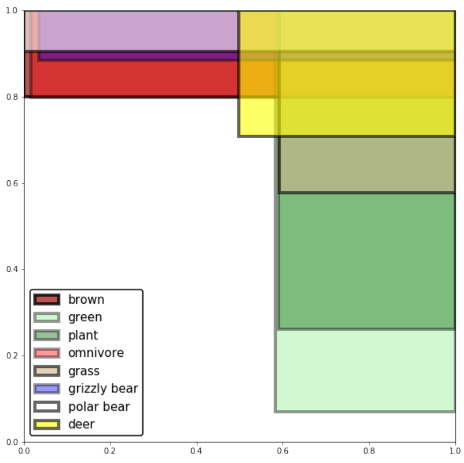


(a) Original lattice

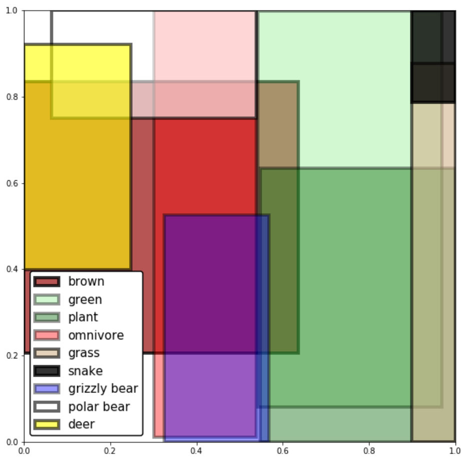


(b) Ground truth CPD

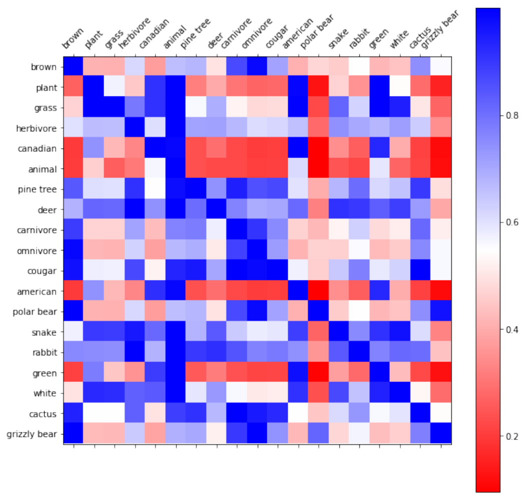
Figure 1: Representation of the toy probabilistic lattice used in Section 5.1. Darker color corresponds to more unary marginal probability. The associated CPD is obtained by a weighted aggregation of leaf elements.



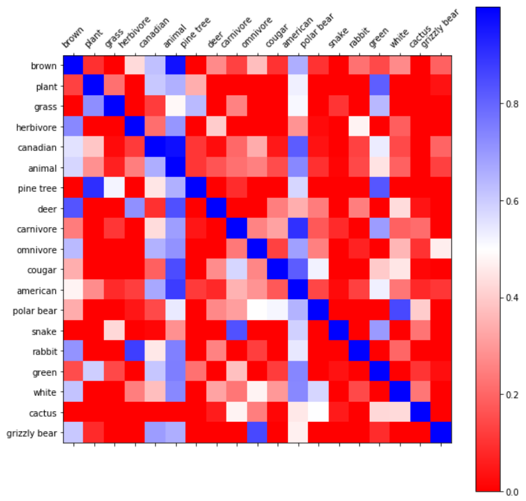
(a) POE lattice



(b) Box lattice



(c) POE CPD



(d) Box CPD

Figure 2: Lattice representations and conditional probabilities from POE vs. box lattice. Note how the box lattice model's lack of "anchoring" to a corner allows it vastly more expressivity in matching the ground truth CPD seen in Figure 1.

P(grizzly bear | ...)		P(cactus | ...)		P(plant | ...)	
P(grizzly bear)	0.12	P(cactus)	0.10	P(plant)	0.20
omnivore	0.29	green	0.16	green	0.37
white	0.00	plant	0.39	snake	0.00
brown	0.30	american, green	0.19	carnivore	0.00
omnivore, white	0.00	plant, green, american	0.40	cactus	0.78
omnivore, brown	0.38	american, carnivore	0.00	american, cactus	0.85

Table 1: Multi-way queries: conditional probabilities adjust when adding additional evidence or contradiction. In constrast, POE can only raise or preserve probability when conditioning.

term1	term2
craftsman.n.02	shark.n.03
homogenized_milk.n.01	apple_juice.n.01
tongue_depresser.n.01	paintbrush.n.01
deerstalker.n.01	bathing_cap.n.01
skywriting.n.01	transcript.n.01

Table 2: Negatively correlated variables produced by the model.

Method	Test Accuracy %
transitive	88.2
word2gauss	86.6
OE	90.6
Li et al. (2017)	91.3
DOE (KL)	**92.3**
POE	91.6
POE (100 dim)	91.7
Box	92.2
Box + CPD	**92.3**

Table 3: Classification accuracy on WordNet test set.

edge transitive closure of the WordNet hypernym hierarchy as positive training examples for the development set, 4,000 for the test set, and using the rest as training data. Negative training examples are created by randomly corrupting a train/development/test edge (u, v) by replacing either u or v with a randomly chosen negative node. We use their specific train/dev/test split, while Athiwaratkun and Wilson (2018) use a different train/dev split with the same test set (personal communication) to examine the effect of different negative sampling techniques. We cite their best performing model, called DOE (KL).

Since our model is probabilistic, we would like a sensible value for $P(n)$, where n is a node. We assign these marginal probabilities by looking at the number of descendants in the hierarchy under a node, and normalizing over all nodes, taking $P(n) = \frac{|\textit{ descendants(n) }|}{|\textit{ nodes }|}$.

Furthermore, we use the graph structure (only of the subset of edges in the training set to avoid leaking data) to augment the data with approximate conditional probabilities $P(x|y)$. For each leaf, we consider all of its ancestors as pairwise co-occurences, then aggregate and divide by the number of leaves to get an approximate joint probability distribution, $P(x, y) = \frac{|\textit{ x, y co-occur in ancestor set }|}{|\textit{ leaves }|}$. With this and the unary marginals, we can create a conditional probability table, which we prune based on the difference of $P(x|y)$ and $P(y|x)$ and add cross entropy with these conditional "soft edges" to the training data. We refer to experiments using this additional data as Box + CPD in Table 3.

We use 50 dimensions in our experiments. Since our model has 2 parameters per dimension, we also perform an apples-to-apples comparison with a 100D POE model. As seen in Table 3, we outperform POE significantly even with this added representational power. We also observe sensible negatively correlated examples, shown in 2, in the trained box model, while POE cannot represent such relationships. We tune our models on the development set, with parameters documented in Appendix D.1. We observe that not only does our model outperform POE, it beats all previous results on WordNet, aside from the concurrent work of Athiwaratkun and Wilson (2018) (using different train/dev negative examples), the baseline POE model does as well. This indicates that probabilistic embeddings for transitive relations are a promising avenue for future work. Additionally, the ability of the model to learn from the expected "soft edges" improves it to state-of-the-art level. We expect that co-occurrence counts gathered from real textual corpora, rather than merely aggregating up the WordNet lattice, would further strengthen this effect.

5.3 Flickr Entailment Graph

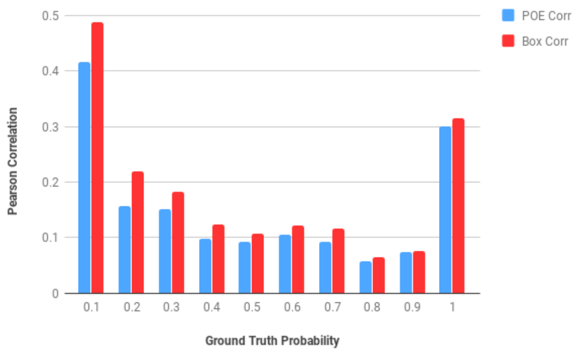


Figure 3: R between model and gold probabilities.

	$P(x\|y)$	
Full test data	KL	Pearson R
POE	0.031	0.949
POE*	0.031	0.949
Box	**0.020**	**0.967**
Unseen pairs		
POE	0.048	0.920
POE*	0.046	0.925
Box	**0.025**	**0.957**
Unseen words		
POE	0.127	0.696
POE*	0.084	0.854
Box	**0.050**	**0.900**

Table 4: KL and Pearson correlation between model and gold probability.

We conduct experiments on the large-scale Flickr entailment dataset of 45 million image caption pairs. We use the exactly same train/dev/test from Lai and Hockenmaier (2017). We use a slightly different unseen word pairs and unseen words test data, obtained from the author. We include their published results and also use their published code, marked $*$, for comparison.

For these experiments, we relax our boxes from the unit hypercube to the nonnegative orthant and obtain probabilities under the exponential measure, $p(x) = \exp(-x)$. We enforce the nonnegativity constraints by clipping the LSTM-generated embedding (Hochreiter and Schmidhuber, 1997) for the box minimum with a ReLU, and parametrize our Δ embeddings using a softplus activation to prevent dead units. As in Lai and Hockenmaier (2017), we use 512 hidden units in our LSTM to compose sentence vectors. We then apply two single-layer feed-forward networks with 512 units applied to the final LSTM state to produce the embeddings.

As we can see from Table 4, we note large improvements in KL and Pearson correlation to the ground truth entailment probabilities. In further analysis, Figure 3 demonstrates that while the box model outperforms POE in nearly every regime, the highest gains come from the comparatively difficult to calibrate small entailment probabilities, indicating the greater capability of our model to produce fine-grained distinctions.

6 Conclusion and Future Work

We have only scratched the surface of possible applications. An exciting direction is the incorporation of multi-relational data for general knowledge representation and inference. Secondly, more complex representations, such as $2n$-dimensional products of 2-dimensional convex polyhedra, would offer greater flexibility in tiling event space. Improved inference of the latent boxes, either through better optimization or through Bayesian approaches is another natural extension. Our greatest interest is in the application of this powerful new tool to the many areas where other structured embeddings have shown promise.

7 Acknowledgments

We thank Alice Lai for making the code from her original paper public, and for providing the additional unseen pairs and unseen words data. We also thank Haw-Shiuan Chang, Laurent Dinh, and Ben Poole for helpful discussions. We also thank the anonymous reviewers for their constructive feedback.

This work was supported in part by the Center for Intelligent Information Retrieval and the Center for Data Science, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction., and in part by the National Science Foundation under Grant No. IIS-1514053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Ben Athiwaratkun and Andrew Gordon Wilson. 2018. On modeling hierarchical data via probabilistic order embeddings. In *International Conference on Learning Representations*.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1041–1051.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Y. J. Chu and T. H. Liu. 1995. On the shortest arborescence of a directed graph. *Science Sinica*, 20.

Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. 2002. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pages 617–624.

Katrin Erk. 2009. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 57–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Salar Fattahi and Somayeh Sojoudi. 2017. Graphical lasso and thresholding: Equivalence and closed-form solutions. *arXiv preprint arXiv:1708.09479*.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 623–632, New York, NY, USA. ACM.

David Heckerman, Dan Geiger, and David M Chickering. 1995. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243.

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

Alice Lai and Julia Hockenmaier. 2017. Learning to predict denotational probabilities for modeling entailment. In *EACL*.

Xiang Li, Luke Vilnis, and Andrew McCallum. 2017. Improved representation learning for predicting commonsense ontologies. *NIPS Workshop on Structured Prediction*.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc.

Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems*.

Daniel M Roy and Yee W Teh. 2009. The mondrian process. In *Advances in neural information processing systems*, pages 1377–1384.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *ICLR*.

Adriaan C. Zaanen. 1997. *Introduction to Operator Theory in Riesz Spaces*. Springer Berlin Heidelberg.

Graph-to-Sequence Learning using Gated Graph Neural Networks

Daniel Beck[†] **Gholamreza Haffari**[‡] **Trevor Cohn**[†]
[†]School of Computing and Information Systems
University of Melbourne, Australia
{d.beck,t.cohn}@unimelb.edu.au
[‡]Faculty of Information Technology
Monash University, Australia
gholamreza.haffari@monash.edu

Abstract

Many NLP applications can be framed as a graph-to-sequence learning problem. Previous work proposing neural architectures on this setting obtained promising results compared to grammar-based approaches but still rely on linearisation heuristics and/or standard recurrent networks to achieve the best performance. In this work, we propose a new model that encodes the full structural information contained in the graph. Our architecture couples the recently proposed Gated Graph Neural Networks with an input transformation that allows nodes and edges to have their own hidden representations, while tackling the parameter explosion problem present in previous work. Experimental results show that our model outperforms strong baselines in generation from AMR graphs and syntax-based neural machine translation.

1 Introduction

Graph structures are ubiquitous in representations of natural language. In particular, many whole-sentence semantic frameworks employ directed acyclic graphs as the underlying formalism, while most tree-based syntactic representations can also be seen as graphs. A range of NLP applications can be framed as the process of transducing a graph structure into a sequence. For instance, language generation may involve realising a semantic graph into a surface form and syntactic machine translation involves transforming a tree-annotated source sentence to its translation.

Previous work in this setting rely on grammar-based approaches such as tree transducers (Flanigan et al., 2016) and hyperedge replacement grammars (Jones et al., 2012). A key limitation of these approaches is that alignments between graph nodes and surface tokens are required. These alignments are usually automatically generated so they can propagate errors when building the grammar. More recent approaches transform the graph into a linearised form and use off-the-shelf methods such as phrase-based machine translation (Pourdamghani et al., 2016) or neural sequence-to-sequence (henceforth, `s2s`) models (Konstas et al., 2017). Such approaches ignore the full graph structure, discarding key information.

In this work we propose a model for graph-to-sequence (henceforth, `g2s`) learning that leverages recent advances in neural encoder-decoder architectures. Specifically, we employ an encoder based on Gated Graph Neural Networks (Li et al., 2016, GGNNs), which can incorporate the full graph structure without loss of information. Such networks represent edge information as label-wise parameters, which can be problematic even for small sized label vocabularies (in the order of hundreds). To address this limitation, we also introduce a graph transformation that changes edges to additional nodes, solving the parameter explosion problem. This also ensures that edges have graph-specific hidden vectors, which gives more information to the attention and decoding modules in the network.

We benchmark our model in two graph-to-sequence problems, generation from Abstract Meaning Representations (AMRs) and Neural Machine Translation (NMT) with source dependency information. Our approach outperforms strong `s2s` baselines in both tasks *without relying on standard RNN encoders*, in contrast with previous work. In particular, for NMT we show that we avoid the need for RNNs by adding sequential edges between contiguous words in the dependency tree. This illustrates the generality of our

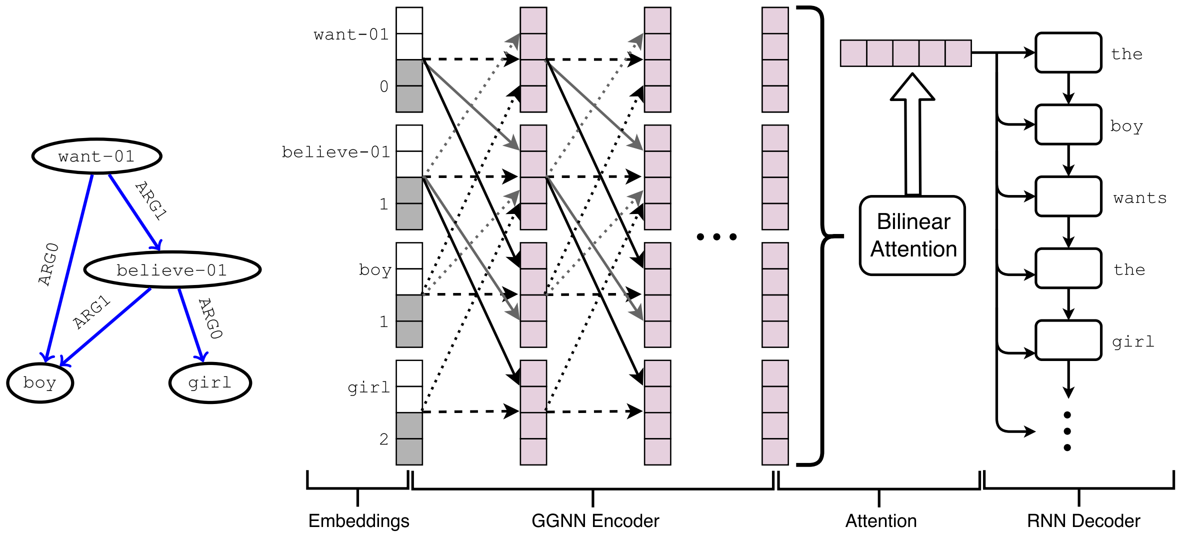


Figure 1: Left: the AMR graph representing the sentence "The boy wants the girl to believe him.". Right: Our proposed architecture using the same AMR graph as input and the surface form as output. The first layer is a concatenation of node and positional embeddings, using distance from the root node as the position. The GGNN encoder updates the embeddings using edge-wise parameters, represented by different colors (in this example, `ARG0` and `ARG1`). The encoder also add corresponding reverse edges (dotted arrows) and self edges for each node (dashed arrows). All parameters are shared between layers. Attention and decoder components are similar to standard `s2s` models. This is a pictorial representation: in our experiments the graphs are transformed before being used as inputs (see §3).

approach: linguistic biases can be added to the inputs by simple graph transformations, without the need for changes to the model architecture.

2 Neural Graph-to-Sequence Model

Our proposed architecture is shown in Figure 1, with an example AMR graph and its transformation into its surface form. Compared to standard `s2s` models, the main difference is in the encoder, where we employ a GGNN to build a graph representation. In the following we explain the components of this architecture in detail.[1]

2.1 Gated Graph Neural Networks

Early approaches for recurrent networks on graphs (Gori et al., 2005; Scarselli et al., 2009) assume a fixed point representation of the parameters and learn using contraction maps. Li et al. (2016) argues that this restricts the capacity of the model and makes it harder to learn long distance relations between nodes. To tackle these issues, they propose Gated Graph Neural Networks, which extend these architectures with gating mechanisms in a similar fashion to Gated Recurrent Units (Cho et al., 2014). This allows the network to be learnt via modern backpropagation procedures.

In following, we formally define the version of GGNNs we employ in this study. Assume a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, L_\mathcal{V}, L_\mathcal{E}\}$, where $\mathcal{V}$ is a set of nodes (v, ℓ_v), $\mathcal{E}$ is a set of edges (v_i, v_j, ℓ_e) and $L_\mathcal{V}$ and $L_\mathcal{E}$ are respectively vocabularies for nodes and edges, from which node and edge labels (ℓ_v and ℓ_e) are defined. Given an input graph with nodes mapped to embeddings $\mathbf{X}$, a GGNN is defined as

$$\begin{aligned}
\mathbf{h}_v^0 &= \mathbf{x}_v \\
\mathbf{r}_v^t &= \sigma\left(c_v^r \sum_{u \in \mathcal{N}_v} \mathbf{W}_{\ell_e}^r \mathbf{h}_u^{(t-1)} + \mathbf{b}_{\ell_e}^r\right) \\
\mathbf{z}_v^t &= \sigma\left(c_v^z \sum_{u \in \mathcal{N}_v} \mathbf{W}_{\ell_e}^z \mathbf{h}_u^{(t-1)} + \mathbf{b}_{\ell_e}^z\right) \\
\widetilde{\mathbf{h}}_v^t &= \rho\left(c_v \sum_{u \in \mathcal{N}_v} \mathbf{W}_{\ell_e}\left(\mathbf{r}_u^t \odot \mathbf{h}_u^{(t-1)}\right) + \mathbf{b}_{\ell_e}\right) \\
\mathbf{h}_v^t &= (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{(i-1)} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}}_v^t
\end{aligned}$$

where $e = (u, v, \ell_e)$ is the edge between nodes u and v, $\mathcal{N}(v)$ is the set of neighbour nodes for v, ρ is a non-linear function, σ is the sigmoid function

[1]Our implementation uses MXNet (Chen et al., 2015) and is based on the Sockeye toolkit (Hieber et al., 2017). Code is available at `github.com/beckdaniel/acl2018_graph2seq`.

and $c_v = c_v^z = c_v^r = |\mathcal{N}_v|^{-1}$ are normalisation constants.

Our formulation differs from the original GGNNs from Li et al. (2016) in some aspects: 1) we add bias vectors for the hidden state, reset gate and update gate computations; 2) label-specific matrices do not share any components; 3) reset gates are applied to all hidden states before any computation and 4) we add normalisation constants. These modifications were applied based on preliminary experiments and ease of implementation.

An alternative to GGNNs is the model from Marcheggiani and Titov (2017), which add edge label information to Graph Convolutional Networks (GCNs). According to Li et al. (2016), the main difference between GCNs and GGNNs is analogous to the difference between convolutional and recurrent networks. More specifically, GGNNs can be seen as multi-layered GCNs where layer-wise parameters are tied and gating mechanisms are added. A large number of layers can propagate node information between longer distances in the graph and, unlike GCNs, GGNNs can have an arbitrary number of layers without increasing the number of parameters. Nevertheless, our architecture borrows ideas from GCNs as well, such as normalising factors.

2.2 Using GGNNs in attentional encoder-decoder models

In s2s models, inputs are sequences of tokens where each token is represented by an embedding vector. The encoder then transforms these vectors into hidden states by incorporating context, usually through a recurrent or a convolutional network. These are fed into an attention mechanism, generating a single context vector that informs decisions in the decoder.

Our model follows a similar structure, where the encoder is a GGNN that receives *node* embeddings as inputs and generates *node* hidden states as outputs, using the graph structure as context. This is shown in the example of Figure 1, where we have 4 hidden vectors, one per node in the AMR graph. The attention and decoder components follow similar standard s2s models, where we use a bilinear attention mechanism (Luong et al., 2015) and a 2-layered LSTM (Hochreiter and Schmidhuber, 1997) as the decoder. Note, however, that other decoders and attention mechanisms can be easily employed instead. Bastings et al. (2017) employs a similar idea for syntax-based NMT, but using GCNs instead.

2.3 Bidirectionality and positional embeddings

While our architecture can in theory be used with general graphs, rooted directed acyclic graphs (DAGs) are arguably the most common kind in the problems we are addressing. This means that node embedding information is propagated in a top down manner. However, it is desirable to have information flow from the reverse direction as well, in the same way RNN-based encoders benefit from right-to-left propagation (as in bidirectional RNNs). Marcheggiani and Titov (2017) and Bastings et al. (2017) achieve this by adding reverse edges to the graph, as well as self-loops edges for each node. These extra edges have specific labels, hence their own parameters in the network.

In this work, we also follow this procedure to ensure information is evenly propagated in the graph. However, this raises another limitation: because the graph becomes essentially undirected, the encoder is now unaware of any intrinsic hierarchy present in the input. Inspired by Gehring et al. (2017) and Vaswani et al. (2017), we tackle this problem by adding positional embeddings to every node. These embeddings are indexed by integer values representing the minimum distance from the root node and are learned as model parameters.[2] This kind of positional embedding is restricted to rooted DAGs: for general graphs, different notions of distance could be employed.

3 Levi Graph Transformation

The g2s model proposed in §2 has two key deficiencies. First, GGNNs have three linear transformations *per edge type*. This means that the number of parameters can explode: AMR, for instance, has around 100 different predicates, which correspond to edge labels. Previous work deal with this problem by explicitly grouping edge labels into a single one (Marcheggiani and Titov, 2017; Bastings et al., 2017) but this is not an ideal solution since it incurs in loss of information.

[2]Vaswani et al. (2017) also proposed fixed positional embeddings based on sine and cosine wavelengths. Preliminary experiments showed that this approach did not work in our case: we speculate this is because wavelengths are more suitable to sequential inputs.

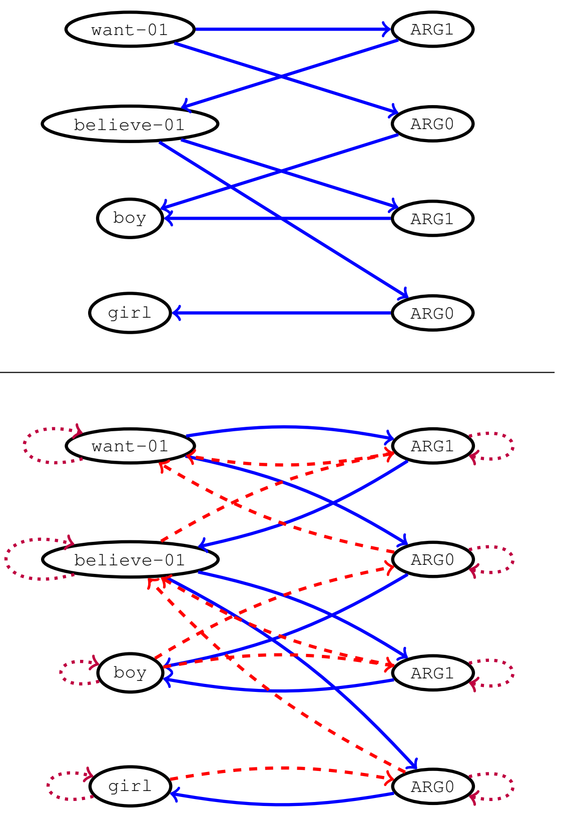


Figure 2: Top: the AMR graph from Figure 1 transformed into its corresponding Levi graph. Bottom: Levi graph with added reverse and self edges (colors represent different edge labels).

The second deficiency is that edge label information is encoded in the form of GGNN parameters in the network. This means that each label will have the same "representation" across all graphs. However, the latent information in edges can depend on the content in which they appear in a graph. Ideally, edges should have instance-specific *hidden states*, in the same way as nodes, and these should also inform decisions made in the decoder through the attention module. For instance, in the AMR graph shown in Figure 1, the `ARG1` predicate between `want-01` and `believe-01` can be interpreted as the preposition "to" in the surface form, while the `ARG1` predicate connecting `believe-01` and `boy` is realised as a pronoun. Notice that edge hidden vectors are already present in `s2s` networks that use linearised graphs: we would like our architecture to also have this benefit.

Instead of modifying the architecture, we propose to transform the input graph into its equivalent Levi graph (Levi, 1942; Gross and Yellen, 2004, p. 765). Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, L_{\mathcal{V}}, L_{\mathcal{E}}\}$, a Levi graph[3] is defined as $\mathcal{G} = \{\mathcal{V}', \mathcal{E}', L_{\mathcal{V}'}, L_{\mathcal{E}'}\}$, where $\mathcal{V}' = \mathcal{V} \cup \mathcal{E}$, $L_{\mathcal{V}'} = L_{\mathcal{V}} \cup L_{\mathcal{E}}$ and $L_{\mathcal{E}'} = \varnothing$. The new edge set $\mathcal{E}'$ contains a edge for every (node, edge) pair that is present in the original graph. By definition, the Levi graph is bipartite.

Intuitively, transforming a graph into its Levi graph equivalent turns edges into additional nodes. While simple in theory, this transformation addresses both modelling deficiencies mentioned above in an elegant way. Since the Levi graph has no labelled edges there is no risk of parameter explosion: original edge labels are represented as embeddings, in the same way as nodes. Furthermore, the encoder now naturally generates hidden states for original edges as well.

In practice, we follow the procedure in §2.3 and add reverse and self-loop edges to the Levi graph, so the practical edge label vocabulary is $L_{\mathcal{E}'} = \{\text{default}, \text{reverse}, \text{self}\}$. This still keeps the parameter space modest since we have only three labels. Figure 2 shows the transformation steps in detail, applied to the AMR graph shown in Figure 1. Notice that the transformed graphs are the ones fed into our architecture: we show the original graph in Figure 1 for simplicity.

It is important to note that this transformation can be applied to any graph and therefore is independent of the model architecture. We speculate this can be beneficial in other kinds of graph-based encoder such as GCNs and leave further investigation to future work.

4 Generation from AMR Graphs

Our first `g2s` benchmark is language generation from AMR, a semantic formalism that represents sentences as rooted DAGs (Banarescu et al., 2013). Because AMR abstracts away from syntax, graphs do not have gold-standard alignment information, so generation is not a trivial task. Therefore, we hypothesize that our proposed model is ideal for this problem.

4.1 Experimental setup

Data and preprocessing We use the latest AMR corpus release (LDC2017T10) with the default split of 36521/1368/1371 instances for training,

[3]Formally, a Levi graph is defined over any *incidence structure*, which is a general concept usually considered in a geometrical context. Graphs are an example of incidence structures but so are points and lines in the Euclidean space, for instance.

development and test sets. Each graph is preprocessed using a procedure similar to what is performed by Konstas et al. (2017), which includes entity simplification and anonymisation. This preprocessing is done before transforming the graph into its Levi graph equivalent. For the `s2s` baselines, we also add scope markers as in Konstas et al. (2017). We detail these procedures in the Supplementary Material.

Models Our baselines are attentional `s2s` models which take linearised graphs as inputs. The architecture is similar to the one used in Konstas et al. (2017) for AMR generation, where the encoder is a BiLSTM followed by a unidirectional LSTM. All dimensionalities are fixed to 512.

For the `g2s` models, we fix the number of layers in the GGNN encoder to 8, as this gave the best results on the development set. Dimensionalities are also fixed at 512 except for the GGNN encoder which uses 576. This is to ensure all models have a comparable number of parameters and therefore similar capacity.

Training for all models uses Adam (Kingma and Ba, 2015) with 0.0003 initial learning rate and 16 as the batch size.[4] To regularise our models we perform early stopping on the dev set based on perplexity and apply 0.5 dropout (Srivastava et al., 2014) on the source embeddings. We detail additional model and training hyperparameters in the Supplementary Material.

Evaluation Following previous work, we evaluate our models using BLEU (Papineni et al., 2001) and perform bootstrap resampling to check statistical significance. However, since recent work has questioned the effectiveness of BLEU with bootstrap resampling (Graham et al., 2014), we also report results using sentence-level CHRF++ (Popović, 2017), using the Wilcoxon signed-rank test to check significance. Evaluation is case-insensitive for both metrics.

Recent work has shown that evaluation in neural models can lead to wrong conclusions by just changing the random seed (Reimers and Gurevych, 2017). In an effort to make our conclusions more robust, we run each model 5 times using different seeds. From each pool, we report results using the median model according to performance on the dev set (simulating what is expected from a single run) and using an ensemble of the 5 models.

[4]Larger batch sizes hurt dev performance in our preliminary experiments. There is evidence that small batches can lead to better generalisation performance (Keskar et al., 2017). While this can make training time slower, it was doable in our case since the dataset is small.

	BLEU	CHRF++	#params
Single models			
`s2s`	21.7	49.1	28.4M
`s2s` (-s)	18.4	46.3	28.4M
`g2s`	23.3	50.4	28.3M
Ensembles			
`s2s`	26.6	52.5	142M
`s2s` (-s)	22.0	48.9	142M
`g2s`	**27.5**	**53.5**	141M
Previous work (early AMR treebank versions)			
KIYCZ17	22.0	–	–
Previous work (as above + unlabelled data)			
KIYCZ17	33.8	–	–
PKH16	26.9	–	–
SPZWG17	25.6	–	–
FDSC16	22.0	–	–

Table 1: Results for AMR generation on the test set. All score differences between our models and the corresponding baselines are significantly different ($p<0.05$). "(-s)" means input without scope marking. KIYCZ17, PKH16, SPZWG17 and FDSC16 are respectively the results reported in Konstas et al. (2017), Pourdamghani et al. (2016), Song et al. (2017) and Flanigan et al. (2016).

Finally, we also report the number of parameters used in each model. Since our encoder architectures are quite different, we try to match the number of parameters between them by changing the dimensionality of the hidden layers (as explained above). We do this to minimise the effects of model capacity as a confounder.

4.2 Results and analysis

Table 1 shows the results on the test set. For the `s2s` models, we also report results without the scope marking procedure of Konstas et al. (2017). Our approach significantly outperforms the `s2s` baselines both with individual models and ensembles, while using a comparable number of parameters. In particular, we obtain these results without relying on scoping heuristics.

On Figure 3 we show an example where our model outperforms the baseline. The AMR graph contains four reentrancies, predicates that refer-

Original AMR graph

```
(p / propose-01
  :ARG0 (c / country
   :wiki "Russia"
   :name (n / name
    :op1 "Russia"))
  :ARG1 (c5 / cooperate-01
   :ARG0 c
   :ARG1 (a / and
    :op1 (c2 / country
     :wiki "India"
     :name (n2 / name
      :op1 "India"))
    :op2 (c3 / country
     :wiki "China"
     :name (n3 / name
      :op1 "China"))))
  :purpose (i / increase-01
   :ARG0 c5
   :ARG1 (s / security)
   :location (a2 / around
    :op1 (c4 / country
     :wiki "Afghanistan"
     :name (n4 / name
      :op1 "Afghanistan")))
   :purpose (b / block-01
    :ARG0 (a3 / and
     :op1 c :op2 c2 :op3 c3
    :ARG1 (s2 / supply-01
     :ARG1 (d / drug)))))
```

Reference surface form

Russia proposes cooperation with **India and China** to increase security around Afghanistan to block drug supplies.

s2s output (CHRF++ 61.8)

Russia proposed cooperation with **India and China** to increase security around the Afghanistan to block security around the Afghanistan , **India and China**.

g2s output (CHRF++ 78.2)

Russia proposed cooperation with **India and China** to increase security around Afghanistan to block drug supplies.

Figure 3: Example showing overgeneration due to reentrancies. Top: original AMR graph with key reentrancies highlighted. Bottom: reference and outputs generated by the s2s and g2s models, highlighting the overgeneration phenomena.

ence previously defined concepts in the graph. In the s2s models including Konstas et al. (2017), reentrant nodes are copied in the linearised form, while this is not necessary for our g2s models. We can see that the s2s prediction overgenerates the "India and China" phrase. The g2s prediction avoids overgeneration, and almost perfectly matches the reference. While this is only a single example, it provides evidence that retaining the full graphical structure is beneficial for this task, which is corroborated by our quantitative results.

Table 1 also show BLEU scores reported in previous work. These results are not strictly comparable because they used different training set versions and/or employ additional unlabelled corpora; nonetheless some insights can be made. In particular, our g2s ensemble performs better than many previous models that combine a smaller training set with a large unlabelled corpus. It is also most informative to compare our s2s model with Konstas et al. (2017), since this baseline is very similar to theirs. We expected our single model baseline to outperform theirs since we use a larger training set but we obtained similar performance. We speculate that better results could be obtained by more careful tuning, but nevertheless we believe such tuning would also benefit our proposed g2s architecture.

The best results with unlabelled data are obtained by Konstas et al. (2017) using Gigaword sentences as additional data and a paired trained procedure with an AMR parser. It is important to note that this procedure is orthogonal to the individual models used for generation and parsing. Therefore, we hypothesise that our model can also benefit from such techniques, an avenue that we leave for future work.

5 Syntax-based Neural Machine Translation

Our second evaluation is NMT, using as graphs source language dependency syntax trees. We focus on a medium resource scenario where additional linguistic information tends to be more beneficial. Our experiments comprise two language pairs: English-German and English-Czech.

5.1 Experimental setup

Data and preprocessing We employ the same data and settings from Bastings et al. (2017),[5] which use the News Commentary V11 corpora from the WMT16 translation task.[6] English text is tokenised and parsed using SyntaxNet[7] while German and Czech texts are tokenised and split into subwords using byte-pair encodings (Sennrich et al., 2016, BPE) (8000 merge operations).

[5]We obtained the data from the original authors to ensure results are comparable without any influence from preprocessing steps.

[6]http://www.statmt.org/wmt16/translation-task.html

[7]https://github.com/tensorflow/models/tree/master/syntaxnet

We refer to Bastings et al. (2017) for further information on the preprocessing steps.

Labelled dependency trees in the source side are transformed into Levi graphs as a preprocessing step. However, unlike AMR generation, in NMT the inputs are originally surface forms that contain important sequential information. This information is lost when treating the input as dependency trees, which might explain why Bastings et al. (2017) obtain the best performance when using an initial RNN layer in their encoder. To investigate this phenomenon, we also perform experiments adding sequential connections to each word in the dependency tree, corresponding to their order in the original surface form (henceforth, `g2s+`). These connections are represented as edges with specific left and right labels, which are added after the Levi graph transformation. Figure 4 shows an example of an input graph for `g2s+`, with the additional sequential edges connecting the words (reverse and self edges are omitted for simplicity).

Models Our `s2s` and `g2s` models are almost the same as in the AMR generation experiments (§4.1). The only exception is the GGNN encoder dimensionality, where we use 512 for the experiments with dependency trees only and 448 when the inputs have additional sequential connections. As in the AMR generation setting, we do this to ensure model capacity are comparable in the number of parameters. Another key difference is that the `s2s` baselines do not use dependency trees: they are trained on the sentences only.

In addition to neural models, we also report results for Phrase-Based Statistical MT (PB-SMT), using Moses (Koehn et al., 2007). The PB-SMT models are trained using the same data conditions as `s2s` (no dependency trees) and use the standard setup in Moses, except for the language model, where we use a 5-gram LM trained on the target side of the respective parallel corpus.[8]

Evaluation We report results in terms of BLEU and CHRF++, using case-sensitive versions of both metrics. Other settings are kept the same as in the AMR generation experiments (§4.1). For PB-SMT, we also report the median result of 5 runs, obtained by tuning the model using MERT (Och and Ney, 2002) 5 times.

[8]Note that target data is segmented using BPE, which is not the usual setting for PB-SMT. We decided to keep the segmentation to ensure data conditions are the same.

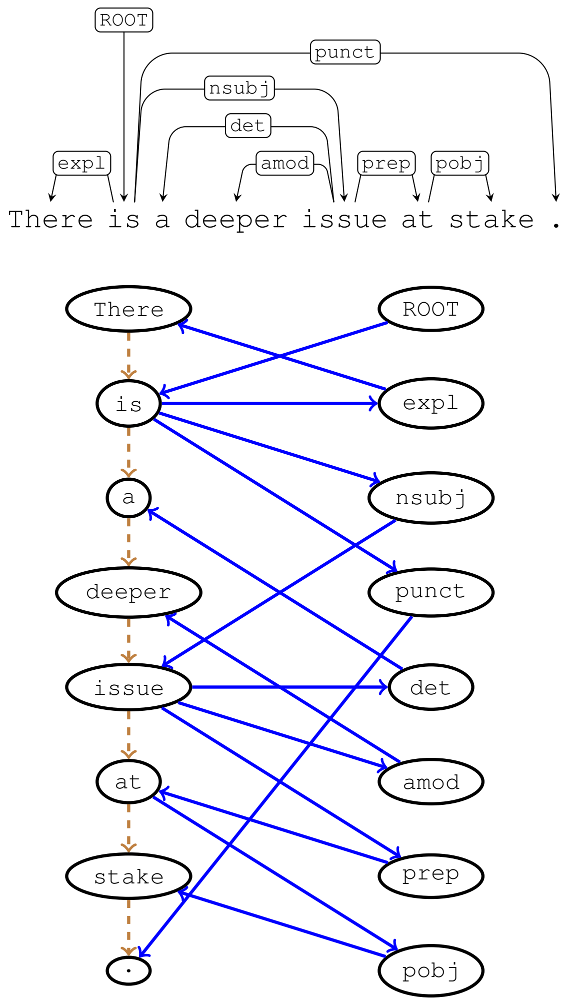


Figure 4: Top: a sentence with its corresponding dependency tree. Bottom: the transformed tree into a Levi graph with additional sequential connections between words (dashed lines). The full graph also contains reverse and self edges, which are omitted in the figure.

5.2 Results and analysis

Table 2 shows the results on the respective test set for both language pairs. The `g2s` models, which do not account for sequential information, lag behind our baselines. This is in line with the findings of Bastings et al. (2017), who found that having a BiRNN layer was key to obtain the best results. However, the `g2s+` models outperform the baselines in terms of BLEU scores under the same parameter budget, in both single model and ensemble scenarios. This result show that it is possible to incorporate sequential biases in our model without relying on RNNs or any other modification in the architecture.

English-German			
	BLEU	CHRF++	#params
Single models			
PB-SMT	12.8	43.2	–
s2s	15.5	40.8	41.4M
g2s	15.2	41.4	40.8M
g2s+	16.7	42.4	41.2M
Ensembles			
s2s	19.0	44.1	207M
g2s	17.7	43.5	204M
g2s+	**19.6**	**45.1**	206M
Results from (Bastings et al., 2017)			
BoW+GCN	12.2	–	–
BiRNN	14.9	–	–
BiRNN+GCN	16.1	–	–
English-Czech			
	BLEU	CHRF++	#params
Single models			
PB-SMT	8.6	**36.4**	–
s2s	8.9	33.8	39.1M
g2s	8.7	32.3	38.4M
g2s+	9.8	33.3	38.8M
Ensembles			
s2s	11.3	**36.4**	195M
g2s	10.4	34.7	192M
g2s+	**11.7**	35.9	194M
Results from (Bastings et al., 2017)			
BoW+GCN	7.5	–	–
BiRNN	8.9	–	–
BiRNN+GCN	9.6	–	–

Table 2: Results for syntax-based NMT on the test sets. All score differences between our models and the corresponding baselines are significantly different ($p<0.05$), including the negative CHRF++ result for En-Cs.

Interestingly, we found different trends when analysing the CHRF++ numbers. In particular, this metric favours the PB-SMT models for both language pairs, while also showing improved performance for s2s in En-Cs. CHRF++ has been shown to better correlate with human judgments compared to BLEU, both at system and sentence level for both language pairs (Bojar et al., 2017), which motivated our choice as an additional metric. We leave further investigation of this phenomena for future work.

We also show some of the results reported by Bastings et al. (2017) in Table 2. Note that their results were based on a different implementation, which may explain some variation in performance. Their BoW+GCN model is the most similar to ours, as it uses only an embedding layer and a GCN encoder. We can see that even our simpler g2s model outperforms their results. A key difference between their approach and ours is the Levi graph transformation and the resulting hidden vectors for edges. We believe their architecture would also benefit from our proposed transformation. In terms of baselines, s2s performs better than their BiRNN model for En-De and comparably for En-Cs, which corroborates that our baselines are strong ones. Finally, our g2s+ single models outperform their BiRNN+GCN results, in particular for En-De, which is further evidence that RNNs are not necessary for obtaining the best performance in this setting.

An important point about these experiments is that we did not tune the architecture: we simply employed the same model we used in the AMR generation experiments, only adjusting the dimensionality of the encoder to match the capacity of the baselines. We speculate that even better results would be obtained by tuning the architecture to this task. Nevertheless, we still obtained improved performance over our baselines and previous work, underlining the generality of our architecture.

6 Related work

Graph-to-sequence modelling Early NLP approaches for this problem were based on Hyperedge Replacement Grammars (Drewes et al., 1997, HRGs). These grammars assume the transduction problem can be split into rules that map portions of a graph to a set of tokens in the output sequence. In particular, Chiang et al. (2013) defines a parsing algorithm, followed by a complexity analysis, while Jones et al. (2012) report experiments on semantic-based machine translation using HRGs. HRGs were also used in previous work on AMR parsing (Peng et al., 2015). The main drawback of these grammar-based approaches though is the need for alignments between graph nodes and surface tokens, which are usually not available in gold-standard form.

Neural networks for graphs Recurrent networks on general graphs were first proposed under the name Graph Neural Networks (Gori et al., 2005; Scarselli et al., 2009). Our work is based on the architecture proposed by Li et al. (2016), which add gating mechanisms. The main difference between their work and ours is that they focus on problems that concern the input graph itself such as node classification or path finding while we focus on generating strings. The main alternative for neural-based graph representations is Graph Convolutional Networks (Bruna et al., 2014; Duvenaud et al., 2015; Kipf and Welling, 2017), which have been applied in a range of problems. In NLP, Marcheggiani and Titov (2017) use a similar architecture for Semantic Role Labelling. They use heuristics to mitigate the parameter explosion by grouping edge labels, while we keep the original labels through our Levi graph transformation. An interesting alternative is proposed by Schlichtkrull et al. (2017), which uses tensor factorisation to reduce the number of parameters.

Applications Early work on AMR generation employs grammars and transducers (Flanigan et al., 2016; Song et al., 2017). Linearisation approaches include (Pourdamghani et al., 2016) and (Konstas et al., 2017), which showed that graph simplification and anonymisation are key to good performance, a procedure we also employ in our work. However, compared to our approach, linearisation incurs in loss of information. MT has a long history of previous work that aims at incorporating syntax (Wu, 1997; Yamada and Knight, 2001; Galley et al., 2004; Liu et al., 2006, inter alia). This idea has also been investigated in the context of NMT. Bastings et al. (2017) is the most similar work to ours, and we benchmark against their approach in our NMT experiments. Eriguchi et al. (2016) also employs source syntax, but using constituency trees instead. Other approaches have investigated the use of syntax in the target language (Aharoni and Goldberg, 2017; Eriguchi et al., 2017). Finally, Hashimoto and Tsuruoka (2017) treats source syntax as a latent variable, which can be pretrained using annotated data.

7 Discussion and Conclusion

We proposed a novel encoder-decoder architecture for graph-to-sequence learning, outperforming baselines in two NLP tasks: generation from AMR graphs and syntax-based NMT. Our approach addresses shortcomings from previous work, including loss of information from linearisation and parameter explosion. In particular, we showed how graph transformations can solve issues with graph-based networks without changing the underlying architecture. This is the case of the proposed Levi graph transformation, which ensures the decoder can attend to edges as well as nodes, but also to the sequential connections added to the dependency trees in the case of NMT. Overall, because our architecture can work with general graphs, it is straightforward to add linguistic biases in the form of extra node and/or edge information. We believe this is an interesting research direction in terms of applications.

Our architecture nevertheless has two major limitations. The first one is that GGNNs have a fixed number of layers, even though graphs can vary in size in terms of number of nodes and edges. A better approach would be to allow the encoder to have a dynamic number of layers, possibly based on the diameter (longest path) in the input graph. The second limitation comes from the Levi graph transformation: because edge labels are represented as nodes they end up sharing the vocabulary and therefore, the same semantic space. This is not ideal, as nodes and edges are different entities. An interesting alternative is Weave Module Networks (Kearnes et al., 2016), which explicitly decouples node and edge representations without incurring in parameter explosion. Incorporating both ideas to our architecture is an research direction we plan for future work.

Acknowledgements

This work was supported by the Australian Research Council (DP160102686). The research reported in this paper was partly conducted at the 2017 Frederick Jelinek Memorial Summer Workshop on Speech and Language Technologies, hosted at Carnegie Mellon University and sponsored by Johns Hopkins University with unrestricted gifts from Amazon, Apple, Facebook, Google, and Microsoft. The authors would also like to thank Joost Bastings for sharing the data from his paper's experiments.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards String-to-Tree Neural Machine Translation. In *Proceedings of ACL*. pages 132–140.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin

Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. pages 178–186.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Proceedings of EMNLP*. pages 1947–1957.

Ondej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the WMT17 Metrics Shared Task. In *Proceedings of WMT*. volume 2, pages 293–301.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proceedings of ICLR*. page 14.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. In *Proceedings of the Workshop on Machine Learning Systems*. pages 1–6.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing Graphs with Hyperedge Replacement Grammars. In *Proceedings of ACL*. pages 924–932.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*. pages 1724–1734.

Frank Drewes, Hans Jörg Kreowski, and Annegret Habel. 1997. Hyperedge Replacement Graph Grammars. *Handbook of Graph Grammars and Computing by Graph Transformation* .

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Proceedings of NIPS*. pages 2215–2223.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of ACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of ACL*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using Tree Transducers. In *Proceedings of NAACL*. pages 731–739.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL*. pages 273–280.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. *arXiv preprint* .

Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A New Model for Learning in Graph Domains. In *Proceedings of IJCNN*. volume 2, pages 729–734.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized Significance Tests in Machine Translation. In *Proceedings of WMT*. pages 266–274.

Jonathan Gross and Jay Yellen, editors. 2004. *Handbook of Graph Theory*. CRC Press.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural Machine Translation with Source-Side Latent Graph Parsing. In *Proceedings of EMNLP*. pages 125–135.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint* pages 1–18.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of COLING*. pages 1359–1376.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular Graph Convolutions: Moving Beyond Fingerprints. *Journal of Computer-Aided Molecular Design* 30(8):595–608.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *Proceedings of ICLR*. pages 1–16.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*. pages 1–15.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL Demo Session*. pages 177–180.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of ACL*. pages 146–157.

Friedrich Wilhelm Levi. 1942. Finite Geometrical Systems.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated Graph Sequence Neural Networks. In *Proceedings of ICLR*. 1, pages 1–20.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*. pages 609–616.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*. pages 1412–1421.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of EMNLP*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. page 295. https://doi.org/10.3115/1073083.1073133.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. pages 311–318.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing. In *Proceedings of CoNLL*. pages 32–41.

Maja Popović. 2017. chrF ++: words helping character n-grams. In *Proceedings of WMT*. pages 612–618.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of INLG*. volume 0, pages 21–25.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of EMNLP*. pages 338–348.

Franco Scarselli, Marco Gori, Ah Ching Tsoi, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20(1):61–80.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks pages 1–12.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL*. pages 1715–1725.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. AMR-to-text Generation with Synchronous Node Replacement Grammar. In *Proceedings of ACL*. pages 7–13.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics* 23(3):377–403.

Kenji Yamada and Kevin Knight. 2001. A Syntax-based Statistical Translation Model. In *Proceedings of ACL*. pages 523–530.

Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context

Urvashi Khandelwal, He He, Peng Qi, Dan Jurafsky
Computer Science Department
Stanford University
{urvashik,hehe,pengqi,jurafsky}@stanford.edu

Abstract

We know very little about how neural language models (LM) use prior linguistic context. In this paper, we investigate the role of context in an LSTM LM, through ablation studies. Specifically, we analyze the increase in perplexity when prior context words are shuffled, replaced, or dropped. On two standard datasets, Penn Treebank and WikiText-2, we find that the model is capable of using about 200 tokens of context on average, but sharply distinguishes nearby context (recent 50 tokens) from the distant history. The model is highly sensitive to the order of words within the most recent sentence, but ignores word order in the long-range context (beyond 50 tokens), suggesting the distant past is modeled only as a rough semantic field or topic. We further find that the neural caching model (Grave et al., 2017b) especially helps the LSTM to copy words from within this distant context. Overall, our analysis not only provides a better understanding of how neural LMs use their context, but also sheds light on recent success from cache-based models.

1 Introduction

Language models are an important component of natural language generation tasks, such as machine translation and summarization. They use context (a sequence of words) to estimate a probability distribution of the upcoming word. For several years now, neural language models (NLMs) (Graves, 2013; Jozefowicz et al., 2016; Grave et al., 2017a; Dauphin et al., 2017; Melis et al., 2018; Yang et al., 2018) have consistently outperformed classical n-gram models, an improvement often attributed to their ability to model long-range dependencies in faraway context. Yet, how these NLMs use the context is largely unexplained.

Recent studies have begun to shed light on the information encoded by Long Short-Term Memory (LSTM) networks. They can remember sentence lengths, word identity, and word order (Adi et al., 2017), can capture some syntactic structures such as subject-verb agreement (Linzen et al., 2016), and can model certain kinds of semantic compositionality such as negation and intensification (Li et al., 2016).

However, all of the previous work studies LSTMs at the sentence level, even though they can potentially encode longer context. Our goal is to complement the prior work to provide a richer understanding of the role of context, in particular, long-range context beyond a sentence. We aim to answer the following questions: (i) How much context is used by NLMs, in terms of the number of tokens? (ii) Within this range, are nearby and long-range contexts represented differently? (iii) How do copy mechanisms help the model use different regions of context?

We investigate these questions via ablation studies on a standard LSTM language model (Merity et al., 2018) on two benchmark language modeling datasets: Penn Treebank and WikiText-2. Given a pretrained language model, we perturb the prior context in various ways *at test time*, to study how much the perturbed information affects model performance. Specifically, we alter the context length to study how many tokens are used, permute tokens to see if LSTMs care about word order in both local and global contexts, and drop and replace target words to test the copying abilities of LSTMs with and without an external copy mechanism, such as the neural cache (Grave et al., 2017b). The cache operates by first recording target words and their context representations seen in the history, and then encouraging the model to copy a word from the past when the current context representation matches that word's recorded context vector.

We find that the LSTM is capable of using about 200 tokens of context on average, with no observable differences from changing the hyperparameter settings. Within this context range, word order is only relevant within the 20 most recent tokens or about a sentence. In the long-range context, order has almost no effect on performance, suggesting that the model maintains a high-level, rough semantic representation of faraway words. Finally, we find that LSTMs can regenerate some words seen in the nearby context, but heavily rely on the cache to help them copy words from the long-range context.

2 Language Modeling

Language models assign probabilities to sequences of words. In practice, the probability can be factorized using the chain rule

$$P(w_1, \ldots, w_t) = \prod_{i=1}^{t} P(w_i | w_{i-1}, \ldots, w_1),$$

and language models compute the conditional probability of a *target word* w_t given its preceding context, $w_1, \ldots, w_{t-1}$.

Language models are trained to minimize the negative log likelihood of the training corpus:

$$\text{NLL} = -\frac{1}{T} \sum_{t=1}^{T} \log P(w_t | w_{t-1}, \ldots, w_1),$$

and the model's performance is usually evaluated by perplexity (PP) on a held-out set:

$$\text{PP} = \exp(\text{NLL}).$$

When testing the effect of ablations, we focus on comparing differences in the language model's losses (NLL) on the dev set, which is equivalent to relative improvements in perplexity.

3 Approach

Our goal is to investigate the effect of contextual features such as the length of context, word order and more, on LSTM performance. Thus, we use ablation analysis, during evaluation, to measure changes in model performance in the absence of certain contextual information.

	PTB		**Wiki**	
	Dev	Test	Dev	Test
# Tokens	73,760	82,430	217,646	245,569
Perplexity (no cache)	59.07	56.89	67.29	64.51
Avg. Sent. Len.	20.9	20.9	23.7	22.6

Table 1: Dataset statistics and performance relevant to our experiments.

Typically, when testing the language model on a held-out sequence of words, all tokens prior to the target word are fed to the model; we call this the *infinite-context* setting. In this study, we observe the change in perplexity or NLL when the model is fed a perturbed context $\delta(w_{t-1}, \ldots, w_1)$, at test time. δ refers to the perturbation function, and we experiment with perturbations such as dropping tokens, shuffling/reversing tokens, and replacing tokens with other words from the vocabulary.[1] It is important to note that we do not train the model with these perturbations. This is because the aim is to start with an LSTM that has been trained in the standard fashion, and discover how much context it uses and which features in nearby vs. long-range context are important. Hence, the mismatch in training and test is a necessary part of experiment design, and all measured losses are upper bounds which would likely be lower, were the model also trained to handle such perturbations.

We use a standard LSTM language model, trained and finetuned using the Averaging SGD optimizer (Merity et al., 2018).[2] We also augment the model with a cache *only* for Section 6.2, in order to investigate why an external copy mechanism is helpful. A short description of the architecture and a detailed list of hyperparameters is listed in Appendix A, and we refer the reader to the original paper for additional details.

We analyze two datasets commonly used for language modeling, Penn Treebank (PTB) (Marcus et al., 1993; Mikolov et al., 2010) and Wikitext-2 (Wiki) (Merity et al., 2017). PTB consists of Wall Street Journal news articles with 0.9M tokens for training and a 10K vocabulary. Wiki is a larger and more diverse dataset, containing Wikipedia articles across many topics with 2.1M tokens for training and a 33K vocabulary. Additional dataset statistics are provided in Ta-

[1]Code for our experiments available at `https://github.com/urvashik/lm-context-analysis`

[2]Public release of their code at `https://github.com/salesforce/awd-lstm-lm`

ble 1.

In this paper, we present results only on the dev sets, in order to avoid revealing details about the test sets. However, we have confirmed that all results are consistent with those on the test sets. In addition, for all experiments we report averaged results from three models trained with different random seeds. Some of the figures provided contain trends from only one of the two datasets and the corresponding figures for the other dataset are provided in Appendix B.

4 How much context is used?

LSTMs are designed to capture long-range dependencies in sequences (Hochreiter and Schmidhuber, 1997). In practice, LSTM language models are provided an infinite amount of prior context, which is as long as the test sequence goes. However, it is unclear how much of this history has a direct impact on model performance. In this section, we investigate how many tokens of context achieve a similar loss (or 1-2% difference in model perplexity) to providing the model infinite context. We consider this the *effective context size*.

LSTM language models have an effective context size of about 200 tokens on average. We determine the effective context size by varying the number of tokens fed to the model. In particular, at test time, we feed the model the most recent n tokens:

$$\delta_{\text{truncate}}(w_{t-1}, \ldots, w_1) = (w_{t-1}, \ldots, w_{t-n}), \quad (1)$$

where $n > 0$ and all tokens farther away from the target w_t are dropped.[3] We compare the dev loss (NLL) from truncated context, to that of the infinite-context setting where all previous words are fed to the model. The resulting increase in loss indicates how important the dropped tokens are for the model.

Figure 1a shows that the difference in dev loss, between truncated- and infinite-context variants of the test setting, gradually diminishes as we increase n from 5 tokens to 1000 tokens. In particular, we only see a 1% increase in perplexity as we move beyond a context of 150 tokens on PTB and 250 tokens on Wiki. Hence, we provide empirical evidence to show that LSTM language models do, in fact, model long-range dependencies, without help from extra context vectors or caches.

[3]Words at the beginning of the test sequence with fewer than n tokens in the context are ignored for loss computation.

Changing hyperparameters does not change the effective context size. NLM performance has been shown to be sensitive to hyperparameters such as the dropout rate and model size (Melis et al., 2018). To investigate if these hyperparameters affect the effective context size as well, we train separate models by varying the following hyperparameters one at a time: (1) number of timesteps for truncated back-propogation (2) dropout rate, (3) model size (hidden state size, number of layers, and word embedding size). In Figure 1b, we show that while different hyperparameter settings result in different perplexities in the infinite-context setting, the trend of how perplexity changes as we reduce the context size remains the same.

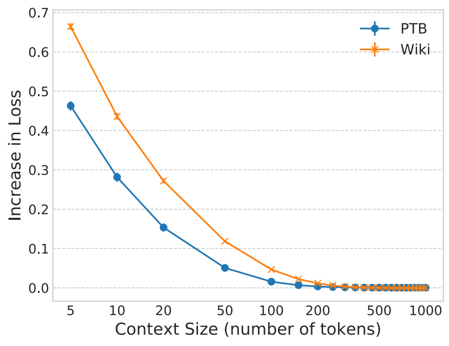
4.1 Do different types of words need different amounts of context?

The effective context size determined in the previous section is aggregated over the entire corpus, which ignores the type of the upcoming word. Boyd-Graber and Blei (2009) have previously investigated the differences in context used by different types of words and found that function words rely on less context than content words. We investigate whether the effective context size varies across different types of words, by categorizing them based on either frequency or parts-of-speech. Specifically, we vary the number of context tokens in the same way as the previous section, and aggregate loss over words within each class separately.

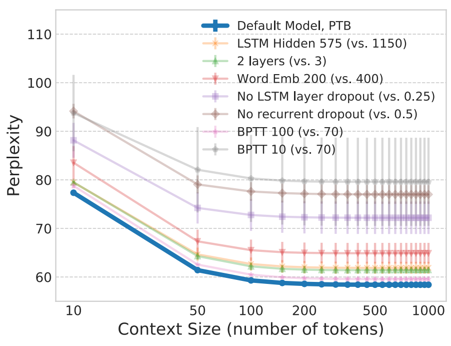
Infrequent words need more context than frequent words. We categorize words that appear at least 800 times in the training set as *frequent*, and the rest as *infrequent*. Figure 1c shows that the loss of frequent words is insensitive to missing context beyond the 50 most recent tokens, which holds across the two datasets. Infrequent words, on the other hand, require more than 200 tokens.

Content words need more context than function words. Given the parts-of-speech of each word, we define *content words* as nouns, verbs and adjectives, and *function words* as prepositions and determiners.[4] Figure 1d shows that the loss of nouns and verbs is affected by distant context, whereas when the target word is a determiner, the model only relies on words within the last 10 tokens.

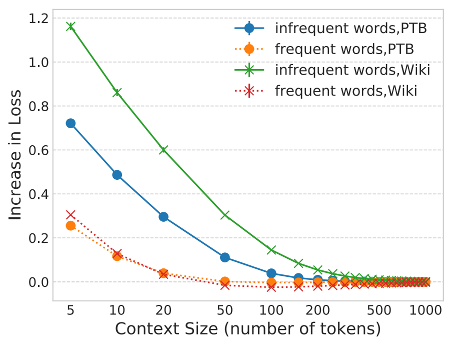
[4]We obtain part-of-speech tags using Stanford CoreNLP (Manning et al., 2014).



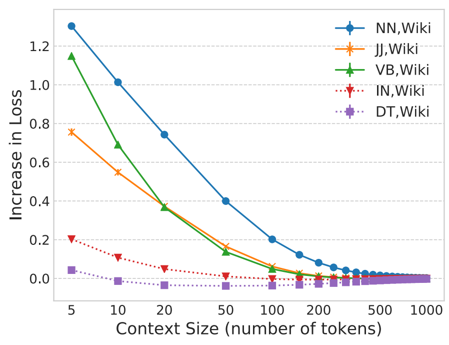
(a) Varying context size.



(b) Changing model hyperparameters.



(c) Frequent vs. infrequent words.



(d) Different parts-of-speech.

Figure 1: Effects of varying the number of tokens provided in the context, as compared to the same model provided with infinite context. Increase in loss represents an absolute increase in NLL over the entire corpus, due to restricted context. All curves are averaged over three random seeds, and error bars represent the standard deviation. **(a)** The model has an effective context size of 150 on PTB and 250 on Wiki. **(b)** Changing model hyperparameters does not change the context usage trend, but does change model performance. We report perplexities to highlight the consistent trend. **(c)** Infrequent words need more context than frequent words. **(d)** Content words need more context than function words.

Discussion. Overall, we find that the model's effective context size is dynamic. It depends on the target word, which is consistent with what we know about language, e.g., determiners require less context than nouns (Boyd-Graber and Blei, 2009). In addition, these findings are consistent with those previously reported for different language models and datasets (Hill et al., 2016; Wang and Cho, 2016).

5 Nearby vs. long-range context

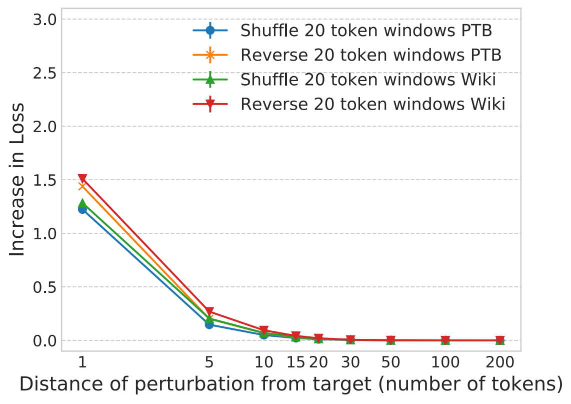
An effective context size of 200 tokens allows for representing linguistic information at many levels of abstraction, such as words, sentences, topics, etc. In this section, we investigate the importance of contextual information such as word order and word identity. Unlike prior work that studies LSTM embeddings at the sentence level, we look at both nearby and faraway context, and analyze how the language model treats contextual information presented in different regions of the context.

5.1 Does word order matter?

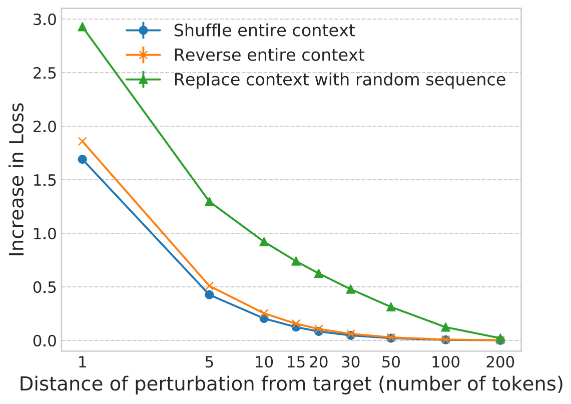
Adi et al. (2017) have shown that LSTMs are aware of word order within a sentence. We investigate whether LSTM language models are sensitive to word order within a larger context window. To determine the range in which word order affects model performance, we permute substrings in the context to observe their effect on dev loss compared to the unperturbed baseline. In particular, we perturb the context as follows,

$$\delta_{\text{permute}}(w_{t-1}, \ldots, w_{t-n}) = (w_{t-1}, .., \rho(w_{t-s_1-1}, .., w_{t-s_2}), .., w_{t-n}) \quad (2)$$

where $\rho \in \{\text{shuffle}, \text{reverse}\}$ and $(s_1, s_2]$ denotes the range of the substring to be permuted. We refer to this substring as the *permutable span*. For



(a) Perturb order locally, within 20 tokens of each point.



(b) Perturb global order, i.e. all tokens in the context before a given point, in Wiki.

Figure 2: Effects of shuffling and reversing the order of words in 300 tokens of context, relative to an unperturbed baseline. All curves are averages from three random seeds, where error bars represent the standard deviation. **(a)** Changing the order of words within a 20-token window has negligible effect on the loss after the first 20 tokens. **(b)** Changing the global order of words within the context does not affect loss beyond 50 tokens.

the following analysis, we distinguish *local word order*, within 20-token permutable spans which are the length of an average sentence, from *global word order*, which extends beyond local spans to include all the farthest tokens in the history. We consider selecting permutable spans within a context of $n = 300$ tokens, which is greater than the effective context size.

Local word order only matters for the most recent 20 tokens. We can locate the region of context beyond which the local word order has no relevance, by permuting word order locally at various points within the context. We accomplish this by varying s_1 and setting $s_2 = s_1 + 20$. Figure 2a shows that local word order matters very much within the most recent 20 tokens, and far less beyond that.

Global order of words only matters for the most recent 50 tokens. Similar to the local word order experiment, we locate the point beyond which the general location of words within the context is irrelevant, by permuting global word order. We achieve this by varying s_1 and fixing $s_2 = n$. Figure 2b demonstrates that after 50 tokens, shuffling or reversing the remaining words in the context has no effect on the model performance.

In order to determine whether this is due to insensitivity to word order or whether the language model is simply not sensitive to any changes in the long-range context, we further replace words in the permutable span with a randomly sampled sequence of the same length from the training set. The gap between the permutation and replacement curves in Figure 2b illustrates that the identity of words in the far away context is still relevant, and only the order of the words is not.

Discussion. These results suggest that word order matters only within the most recent sentence, beyond which the order of sentences matters for 2-3 sentences (determined by our experiments on global word order). After 50 tokens, word order has almost no effect, but the identity of those words is still relevant, suggesting a high-level, rough semantic representation for these faraway words. In light of these observations, we define 50 tokens as the boundary between nearby and long-range context, for the rest of this study. Next, we investigate the importance of different word types in the different regions of context.

5.2 Types of words and the region of context

Open-class or *content words* such as nouns, verbs, adjectives and adverbs, contribute more to the semantic context of natural language than *function words* such as determiners and prepositions. Given our observation that the language model represents long-range context as a rough semantic representation, a natural question to ask is how important are function words in the long-range

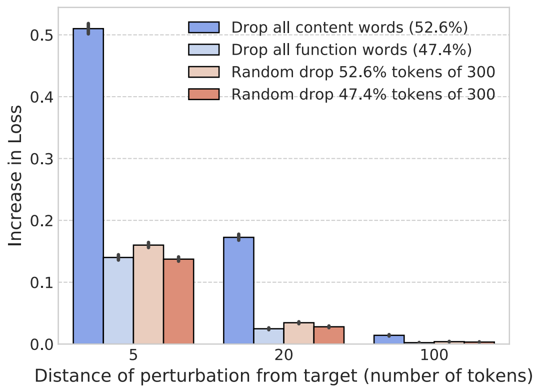


Figure 3: Effect of dropping content and function words from 300 tokens of context relative to an unperturbed baseline, on PTB. Error bars represent 95% confidence intervals. Dropping both content and function words 5 tokens away from the target results in a nontrivial increase in loss, whereas beyond 20 tokens, only content words are relevant.

context? Below, we study the effect of these two classes of words on the model's performance. Function words are defined as all words that are not nouns, verbs, adjectives or adverbs.

Content words matter more than function words. To study the effect of content and function words on model perplexity, we drop them from different regions of the context and compare the resulting change in loss. Specifically, we perturb the context as follows,

$$\delta_{\text{drop}}(w_{t-1}, \ldots, w_{t-n}) = (w_{t-1}, .., w_{t-s_1}, f_{\text{pos}}(y, (w_{t-s_1-1}, .., w_{t-n}))) \tag{3}$$

where $f_{\text{pos}}(y, \text{span})$ is a function that drops all words with POS tag y in a given span. s_1 denotes the starting offset of the perturbed subsequence. For these experiments, we set $s_1 \in \{5, 20, 100\}$. On average, there are slightly more content words than function words in any given text. As shown in Section 4, dropping more words results in higher loss. To eliminate the effect of dropping different fractions of words, for each experiment where we drop a specific word type, we add a control experiment where the same number of tokens are sampled randomly from the context, and dropped.

Figure 3 shows that dropping content words as close as 5 tokens from the target word increases model perplexity by about 65%, whereas dropping the same proportion of tokens at random, results in a much smaller 17% increase. Dropping all function words, on the other hand, is not very different from dropping the same proportion of words at random, but still increases loss by about 15%. This suggests that within the most recent sentence, content words are extremely important but function words are also relevant since they help maintain grammaticality and syntactic structure. On the other hand, beyond a sentence, only content words have a sizeable influence on model performance.

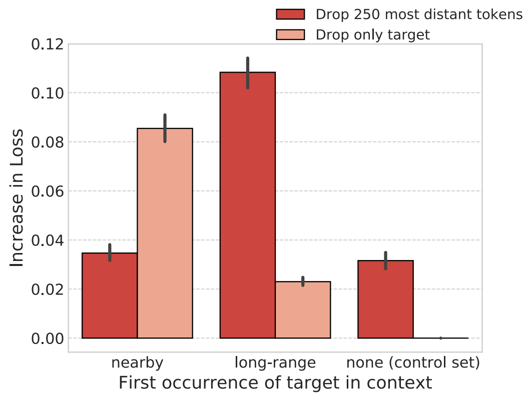
6 To cache or not to cache?

As shown in Section 5.1, LSTM language models use a high-level, rough semantic representation for long-range context, suggesting that they might not be using information from any specific words located far away. Adi et al. (2017) have also shown that while LSTMs are aware of which words appear in their context, this awareness degrades with increasing length of the sequence. However, the success of copy mechanisms such as attention and caching (Bahdanau et al., 2015; Hill et al., 2016; Merity et al., 2017; Grave et al., 2017a,b) suggests that information in the distant context is very useful. Given this fact, can LSTMs copy any words from context without relying on external copy mechanisms? Do they copy words from nearby and long-range context equally? How does the caching model help? In this section, we investigate these questions by studying how LSTMs copy words from different regions of context. More specifically, we look at two regions of context, nearby (within 50 most recent tokens) and long-range (beyond 50 tokens), and study three categories of target words: those that can be copied from nearby context (C_{near}), those that can *only* be copied from long-range context (C_{far}), and those that cannot be copied at all given a limited context (C_{none}).

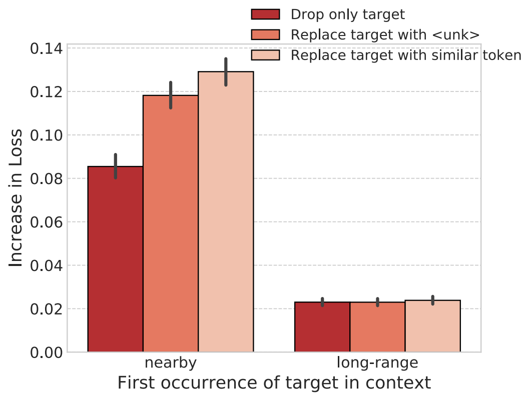
6.1 Can LSTMs copy words without caches?

Even without a cache, LSTMs often regenerate words that have already appeared in prior context. We investigate how much the model relies on the previous occurrences of the upcoming target word, by analyzing the change in loss after dropping and replacing this target word in the context.

LSTMs can regenerate words seen in nearby context. In order to demonstrate the usefulness



(a) Dropping tokens



(b) Perturbing occurrences of target word in context.

Figure 4: Effects of perturbing the target word in the context compared to dropping long-range context altogether, on PTB. Error bars represent 95% confidence intervals. **(a)** Words that can only be copied from long-range context are more sensitive to dropping all the distant words than to dropping the target. For words that can be copied from nearby context, dropping only the target has a much larger effect on loss compared to dropping the long-range context. **(b)** Replacing the target word with other tokens from vocabulary hurts more than dropping it from the context, for words that can be copied from nearby context, but has no effect on words that can only be copied from far away.

of target word occurrences in context, we experiment with dropping all the distant context versus dropping only occurrences of the target word from the context. In particular, we compare removing all tokens after the 50 most recent tokens, (Equation 1 with $n = 50$), versus removing only the target word, in context of size $n = 300$:

$$\delta_{\text{drop}}(w_{t-1}, \ldots, w_{t-n}) = f_{\text{word}}(w_t, (w_{t-1}, \ldots, w_{t-n})), \tag{4}$$

where $f_{\text{word}}(w, \text{span})$ drops words equal to w in a given span. We compare applying both perturbations to a baseline model with unperturbed context restricted to $n = 300$. We also include the target words that never appear in the context (C_{none}) as a control set for this experiment.

The results show that LSTMs rely on the rough semantic representation of the faraway context to generate C_{far}, but direclty copy C_{near} from the nearby context. In Figure 4a, the long-range context bars show that for words that can only be copied from long-range context (C_{far}), removing all distant context is far more disruptive than removing only occurrences of the target word (12% and 2% increase in perplexity, respectively). This suggests that the model relies more on the rough semantic representation of faraway context to predict these C_{far} tokens, rather than directly copying them from the distant context. On the other hand, for words that can be copied from nearby context (C_{near}), removing all long-range context has a smaller effect (about 3.5% increase in perplexity) as seen in Figure 4a, compared to removing the target word which increases perplexity by almost 9%. This suggests that these C_{near} tokens are more often copied from nearby context, than inferred from information found in the rough semantic representation of long-range context.

However, is it possible that dropping the target tokens altogether, hurts the model too much by adversely affecting grammaticality of the context? We test this theory by replacing target words in the context with other words from the vocabulary. This perturbation is similar to Equation 4, except instead of dropping the token, we replace it with a different one. In particular, we experiment with replacing the target with `<unk>`, to see if having the generic word is better than not having any word. We also replace it with a word that has the same part-of-speech tag and a similar frequency in the dataset, to observe how much this change confuses the model. Figure 4b shows that replacing the target with other words results in up to a 14% increase in perplexity for C_{near}, which suggests that the replacement token seems to confuse the model far more than when the token is simply dropped. However, the words that rely on the long-range context, C_{far}, are largely unaffected by these changes, which confirms our conclusion from dropping the target tokens: C_{far}

</s> snack-food UNK increased a strong NUM NUM in the third quarter while domestic profit increased in double UNK mr. calloway said </s> excluding the british snack-food business acquired in july snack-food international UNK jumped NUM NUM with sales strong in spain mexico and brazil </s> total snack-food profit rose NUM NUM </s> led by pizza hut and UNK bell restaurant earnings increased about NUM NUM in the third quarter on a NUM NUM sales increase </s> UNK sales for pizza hut rose about NUM NUM while UNK bell 's increased NUM NUM as the chain continues to benefit from its UNK strategy </s> UNK bell has turned around declining customer counts by permanently lowering the price of its UNK </s> same UNK for kentucky fried chicken which has struggled with increased competition in the fast-food chicken market and a lack of new products rose only NUM NUM </s> the operation which has been slow to respond to consumers ' shifting UNK away from fried foods has been developing a UNK product that may be introduced nationally at the end of next year </s> the new product has performed well in a market test in las vegas nev. mr. **calloway**

Figure 5: Success of neural cache on PTB. Brightly shaded region shows peaky distribution.

offerings outside the u.s. </s> goldman sachs & co. will manage the offering </s> macmillan said berlitz intends to pay quarterly dividends on the stock </s> the company said it expects to pay the first dividend of NUM cents a share in the NUM first quarter </s> berlitz will borrow an amount equal to its expected net proceeds from the offerings plus $ NUM million in connection with a credit agreement with lenders </s> the total borrowing will be about $ NUM million the company said </s> proceeds from the borrowings under the credit agreement will be used to pay an $ NUM million cash dividend to macmillan and to lend the remainder of about $ NUM million to maxwell communications in connection with a UNK note </s> proceeds from the offering will be used to repay borrowings under the short-term parts of a credit agreement </s> berlitz which is based in princeton n.j. provides language instruction and translation services through more than NUM language centers in NUM countries </s> in the past five years more than NUM NUM of its sales have been outside the u.s. </s> macmillan has owned berlitz since NUM </s> in the first six **months**

Figure 6: Failure of neural cache on PTB. Lightly shaded regions show flat distribution.

words are predicted from the rough representation of faraway context instead of specific occurrences of certain words.

6.2 How does the cache help?

If LSTMs can already regenerate words from nearby context, how are copy mechanisms helping the model? We answer this question by analyzing how the neural cache model (Grave et al., 2017b) helps with improving model performance. The cache records the hidden state h_t at each timestep t, and computes a cache distribution over the words in the history as follows:

$$P_{\text{cache}}(w_t|w_{t-1},\dots,w_1;h_t,\dots,h_1) \propto \sum_{i=1}^{t-1} \mathbb{1}[w_i = w_t] \exp(\theta h_i^T h_t), \tag{5}$$

where θ controls the flatness of the distribution. This cache distribution is then interpolated with the model's output distribution over the vocabulary. Consequently, certain words from the history are upweighted, encouraging the model to copy them.

Caches help words that can be copied from long-range context the most. In order to study the effectiveness of the cache for the three classes of words ($C_{\text{near}}, C_{\text{far}}, C_{\text{none}}$), we evaluate an LSTM language model with and without a cache, and measure the difference in perplexity for these words. In both settings, the model is provided all prior context (not just 300 tokens) in order to replicate the Grave et al. (2017b) setup. The amount of history recorded, known as the cache size, is a hyperparameter set to 500 past timesteps for PTB and 3,875 for Wiki, both values very similar to the average document lengths in the respective datasets.

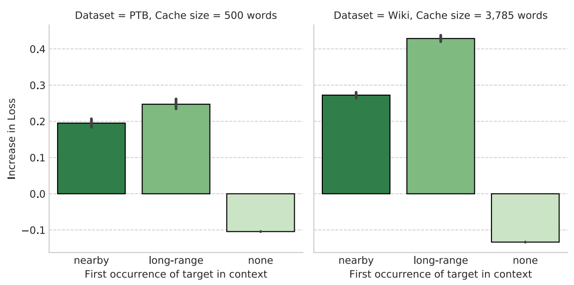


Figure 7: Model performance relative to using a cache. Error bars represent 95% confidence intervals. Words that can only be copied from the distant context benefit the most from using a cache.

We find that the cache helps words that can only be copied from long-range context (C_{far}) more than words that can be copied from nearby (C_{near}). This is illustrated by Figure 7 where without caching, C_{near} words see a 22% increase in perplexity for PTB, and a 32% increase for Wiki, whereas C_{far} see a 28% increase in perplexity for PTB, and a whopping 53% increase for Wiki. Thus, the cache is, in a sense, complementary to the standard model, since it especially helps regenerate words from the long-range context where the latter falls short.

However, the cache also hurts about 36% of the words in PTB and 20% in Wiki, which are words that cannot be copied from context (C_{none}), as illustrated by bars for "none" in Figure 7. We also provide some case studies showing success (Fig. 5) and failure (Fig. 6) modes for the cache. We find that for the successful case, the cache distribution is concentrated on a single word that it wants to copy. However, when the target is not present in the history, the cache distribution is more flat, illustrating the model's confusion, shown in Figure 6. This suggests that the neural cache model might benefit from having the option to ignore the cache when it cannot make a confident choice.

7 Discussion

The findings presented in this paper provide a great deal of insight into how LSTMs model context. This information can prove extremely useful for improving language models. For instance, the discovery that some word types are more important than others can help refine word dropout strategies by making them adaptive to the different word types. Results on the cache also show that we can further improve performance by allowing the model to ignore the cache distribution when it is extremely uncertain, such as in Figure 6. Differences in nearby vs. long-range context suggest that memory models, which feed explicit context representations to the LSTM (Ghosh et al., 2016; Lau et al., 2017), could benefit from representations that specifically capture information orthogonal to that modeled by the LSTM.

In addition, the empirical methods used in this study are model-agnostic and can generalize to models other than the standard LSTM. This opens the path to generating a stronger understanding of model classes beyond test set perplexities, by comparing them across additional axes of information such as how much context they use on average, or how robust they are to shuffled contexts.

Given the empirical nature of this study and the fact that the model and data are tightly coupled, separating model behavior from language characteristics, has proved challenging. More specifically, a number of confounding factors such as vocabulary size, dataset size etc. make this separation difficult. In an attempt to address this, we have chosen PTB and Wiki - two standard language modeling datasets which are diverse in content (news vs. factual articles) and writing style, and are structured differently (eg: Wiki articles are 4-6x longer on average and contain extra information such as titles and paragraph/section markers). Making the data sources diverse in nature, has provided the opportunity to somewhat isolate effects of the model, while ensuring consistency in results. An interesting extension to further study this separation would lie in experimenting with different model classes and even different languages.

Recently, Chelba et al. (2017), in proposing a new model, showed that on PTB, an LSTM language model with 13 tokens of context is similar to the infinite-context LSTM performance, with close to an 8% [5] increase in perplexity. This is compared to a 25% increase at 13 tokens of context in our setup. We believe this difference is attributed to the fact that their model was trained with restricted context and a different error propagation scheme, while ours is not. Further investigation would be an interesting direction for future work.

8 Conclusion

In this analytic study, we have empirically shown that a standard LSTM language model can effectively use about 200 tokens of context on two benchmark datasets, regardless of hyperparameter settings such as model size. It is sensitive to word order in the nearby context, but less so in the long-range context. In addition, the model is able to regenerate words from nearby context, but heavily relies on caches to copy words from far away. These findings not only help us better understand these models but also suggest ways for improving them, as discussed in Section 7. While observations in this paper are reported at the token level, deeper understanding of sentence-level interactions warrants further investigation, which we leave to future work.

Acknowledgments

We thank Arun Chaganty, Kevin Clark, Reid Pryzant, Yuhao Zhang and our anonymous reviewers for their thoughtful comments and suggestions. We gratefully acknowledge support of the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF15-1-0462 and the NSF via grant IIS-1514268.

[5] Table 3, 91 perplexity for the 13-gram vs. 84 for the infinite context model.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=BJh6Ztuxl.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)* https://arxiv.org/pdf/1409.0473.pdf.

Jordan Boyd-Graber and David Blei. 2009. Syntactic topic models. In *Advances in neural information processing systems*. pages 185–192. https://papers.nips.cc/paper/3398-syntactic-topic-models.pdf.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram language modeling using recurrent neural network estimation. *arXiv preprint arXiv:1703.10724* https://arxiv.org/pdf/1703.10724.pdf.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. *International Conference on Machine Learning (ICML)* https://arxiv.org/pdf/1612.08083.pdf.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems (NIPS)*. pages 1019–1027. https://arxiv.org/pdf/1512.05287.pdf.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *Workshop on Large-scale Deep Learning for Data Mining, KDD* https://arxiv.org/pdf/1602.06291.pdf.

Edouard Grave, Moustapha M Cisse, and Armand Joulin. 2017a. Unbounded cache model for online language modeling with open vocabulary. In *Advances in Neural Information Processing Systems (NIPS)*. pages 6044–6054. https://papers.nips.cc/paper/7185-unbounded-cache-model-for-online-language-modeling-with-open-vocabulary.pdf.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017b. Improving Neural Language Models with a Continuous Cache. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=B184E5qee.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* https://arxiv.org/pdf/1308.0850.pdf.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *International Conference on Learning Representations (ICLR)* https://arxiv.org/pdf/1511.02301.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=r1aPbsFle.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* https://arxiv.org/pdf/1602.02410.pdf.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically Driven Neural Language Model. *Association for Computational Linguistics (ACL)* https://doi.org/10.18653/v1/P17-1033.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. *North American Association of Computational Linguistics (NAACL)* http://www.aclweb.org/anthology/N16-1082.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics (TACL)* http://aclweb.org/anthology/Q16-1037.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330. http://aclweb.org/anthology/J93-2004.

Gabor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=ByJHuTgA-.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=SyyGPP0TZ.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=Byj72udxe.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. *European Chapter of the Association for Computational Linguistics* http://aclweb.org/anthology/E17-2025.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*. pages 1058–1066.

Tian Wang and Kyunghyun Cho. 2016. Larger-Context Language Modelling with Recurrent Neural Network. *Association for Computational Linguistics (ACL)* https://doi.org/10.18653/v1/P16-1125.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2018. Breaking the softmax bottleneck: a high-rank rnn language model. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=HkwZSG-CZ.

SoPa: Bridging CNNs, RNNs, and Weighted Finite-State Machines

Roy Schwartz[*] [◇♡] **Sam Thomson**[*] [♣] **Noah A. Smith**[◇]
[◇]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]Language Technologies Institute, Carnegie Mellon University
[♡]Allen Institute for Artificial Intelligence
{roysch,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

Abstract

Recurrent and convolutional neural networks comprise two distinct families of models that have proven to be useful for encoding natural language utterances. In this paper we present SoPa, a new model that aims to bridge these two approaches. SoPa combines neural representation learning with weighted finite-state automata (WFSAs) to learn a soft version of traditional surface patterns. We show that SoPa is an extension of a one-layer CNN, and that such CNNs are equivalent to a restricted version of SoPa, and accordingly, to a restricted form of WFSA. Empirically, on three text classification tasks, SoPa is comparable or better than both a BiLSTM (RNN) baseline and a CNN baseline, and is particularly useful in small data settings.

1 Introduction

Recurrent neural networks (RNNs; Elman, 1990) and convolutional neural networks (CNNs; LeCun, 1998) are two of the most useful text representation learners in NLP (Goldberg, 2016). These methods are generally considered to be quite different: the former encodes an arbitrarily long sequence of text, and is highly expressive (Siegelmann and Sontag, 1995). The latter is more local, encoding fixed length windows, and accordingly less expressive. In this paper, we seek to bridge the gap between RNNs and CNNs, presenting **SoPa** (for **So**ft **Pa**tterns), a model that lies in between them.

SoPa is a neural version of a weighted finite-state automaton (WFSA), with a restricted set of transitions. Linguistically, SoPa is appealing as it

[*]The first two authors contributed equally.

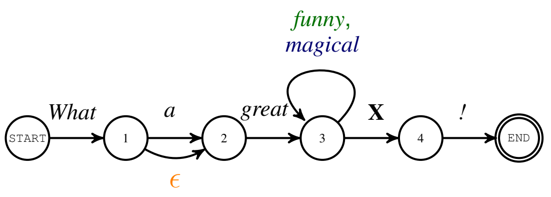


Figure 1: A representation of a surface pattern as a six-state automaton. **Self-loops** allow for repeatedly inserting words (e.g., "*funny*"). **ϵ-transitions** allow for dropping words (e.g., "*a*").

is able to capture a soft notion of surface patterns (e.g., *"what a great X !"*; Hearst, 1992), where some words may be dropped, inserted, or replaced with similar words (see Figure 1). From a modeling perspective, SoPa is interesting because WFSAs are well-studied and come with efficient and flexible inference algorithms (Mohri, 1997; Eisner, 2002) that SoPa can take advantage of.

SoPa defines a set of soft patterns of different lengths, with each pattern represented as a WFSA (Section 3). While the number and lengths of the patterns are hyperparameters, the patterns themselves are learned end-to-end. SoPa then represents a document with a vector that is the aggregate of the scores computed by matching each of the patterns with each span in the document. Because SoPa defines a hidden state that depends on the input token and the previous state, it can be thought of as a simple type of RNN.

We show that SoPa is an extension of a one-layer CNN (Section 4). Accordingly, one-layer CNNs can be viewed as a collection of linear-chain WFSAs, each of which can only match fixed-length spans, while our extension allows matches of flexible-length. As a simple type of RNN that is more expressive than a CNN, SoPa helps to link CNNs and RNNs.

To test the utility of SoPa, we experiment with three text classification tasks (Section 5). We compare against four baselines, including both a bidirectional LSTM and a CNN. Our model performs on par with or better than all baselines on all tasks (Section 6). Moreover, when training with smaller datasets, SoPa is particularly useful, outperforming all models by substantial margins. Finally, building on the connections discovered in this paper, we offer a new, simple method to interpret SoPa (Section 7). This method applies equally well to CNNs. We release our code at `https://github.com/Noahs-ARK/soft_patterns`.

2 Background

Surface patterns. Patterns (Hearst, 1992) are particularly useful tool in NLP (Lin et al., 2003; Etzioni et al., 2005; Schwartz et al., 2015). The most basic definition of a pattern is a sequence of words and wildcards (e.g., "**X** is a **Y**"), which can either be manually defined or extracted from a corpus using cooccurrence statistics. Patterns can then be matched against a specific text span by replacing wildcards with concrete words.

Davidov et al. (2010) introduced a *flexible* notion of patterns, which supports partial matching of the pattern with a given text by skipping some of the words in the pattern, or introducing new words. In their framework, when a sequence of text partially matches a pattern, hard-coded partial scores are assigned to the pattern match. Here, we represent patterns as WFSAs with neural weights, and support these partial matches in a soft manner.

WFSAs. We review weighted finite-state automata with ϵ-transitions before we move on to our special case in Section 3. A WFSA-ϵ with d states over a vocabulary V is formally defined as a tuple $F = \langle \pi, \mathbf{T}, \eta \rangle$, where $\pi \in \mathbb{R}^d$ is an initial weight vector, $\mathbf{T} : (V \cup \{\epsilon\}) \to \mathbb{R}^{d \times d}$ is a transition weight function, and $\eta \in \mathbb{R}^d$ is a final weight vector. Given a sequence of words in the vocabulary $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$, the Forward algorithm (Baum and Petrie, 1966) scores $\boldsymbol{x}$ with respect to F. Without ϵ-transitions, Forward can be written as a series of matrix multiplications:

$$p'_{\text{span}}(\boldsymbol{x}) = \pi^\top \left(\prod_{i=1}^{n} \mathbf{T}(x_i) \right) \eta \qquad (1)$$

ϵ-transitions are followed without consuming a word, so Equation 1 must be updated to reflect the possibility of following any number (zero or more) of ϵ-transitions in between consuming each word:

$$p_{\text{span}}(\boldsymbol{x}) = \pi^\top \mathbf{T}(\epsilon)^* \left(\prod_{i=1}^{n} \mathbf{T}(x_i) \mathbf{T}(\epsilon)^* \right) \eta \qquad (2)$$

where * is matrix asteration: $\mathbf{A}^* := \sum_{j=0}^{\infty} \mathbf{A}^j$. In our experiments we use a first-order approximation, $\mathbf{A}^* \approx \mathbf{I} + \mathbf{A}$, which corresponds to allowing zero or one ϵ-transition at a time. When the FSA F is probabilistic, the result of the Forward algorithm can be interpreted as the marginal probability of all paths through F while consuming $\boldsymbol{x}$ (hence the symbol "p").

The Forward algorithm can be generalized to any semiring (Eisner, 2002), a fact that we make use of in our experiments and analysis.[1] The vanilla version of Forward uses the sum-product semiring: $\oplus$ is addition, $\otimes$ is multiplication. A special case of Forward is the Viterbi algorithm (Viterbi, 1967), which sets $\oplus$ to the $\max$ operator. Viterbi finds the *highest scoring* path through F while consuming $\boldsymbol{x}$. Both Forward and Viterbi have runtime $O(d^3 + d^2 n)$, requiring just a single linear pass through the phrase. Using first-order approximate asteration, this runtime drops to $O(d^2 n)$.[2]

Finally, we note that Forward scores are for *exact matches*—the entire phrase must be consumed. We show in Section 3.2 how phrase-level scores can be summarized into a document-level score.

3 SoPa: A Weighted Finite-State Automaton RNN

We introduce SoPa, a WFSA-based RNN, which is designed to represent text as collection of surface pattern occurrences. We start by showing how a single pattern can be represented as a WFSA-ϵ (Section 3.1). Then we describe how to score a complete document using a pattern (Section 3.2), and how multiple patterns can be used to encode a document (Section 3.3). Finally, we show that SoPa can be seen as a simple variant of an RNN (Section 3.4).

[1]The semiring parsing view (Goodman, 1999) has produced unexpected connections in the past (Eisner, 2016). We experiment with max-product and max-sum semirings, but note that our model could be easily updated to use any semiring.

[2]In our case, we also use a sparse transition matrix (Section 3.1), which further reduces our runtime to $O(dn)$.

3.1 Patterns as WFSAs

We describe how a pattern can be represented as a WFSA-ϵ. We first assume a single pattern. A pattern *is* a WFSA-ϵ, but we impose hard constraints on its shape, and its transition weights are given by differentiable functions that have the power to capture concrete words, wildcards, and everything in between. Our model is designed to behave similarly to flexible hard patterns (see Section 2), but to be learnable directly and "end-to-end" through backpropagation. Importantly, it will still be interpretable as simple, almost linear-chain, WFSA-ϵ.

Each pattern has a sequence of d states (in our experiments we use patterns of varying lengths between 2 and 7). Each state i has exactly three possible outgoing transitions: a **self-loop**, which allows the pattern to consume a word without moving states, a **main path** transition to state $i+1$ which allows the pattern to consume one token and move forward one state, and an **ϵ-transition** to state $i+1$, which allows the pattern to move forward one state without consuming a token. All other transitions are given score 0. When processing a sequence of text with a pattern p, we start with a special `START` state, and only move forward (or stay put), until we reach the special `END` state.[3] A pattern with d states will tend to match token spans of length $d-1$ (but possibly shorter spans due to ϵ-transitions, or longer spans due to self-loops). See Figure 1 for an illustration.

Our transition function, $\mathbf{T}$, is a parameterized function that returns a $d \times d$ matrix. For a word x:

$$[\mathbf{T}(x)]_{i,j} = \begin{cases} E(\mathbf{u}_i \cdot \mathbf{v}_x + a_i), & \text{if } j = i \text{ \textit{(self-loop)}} \\ E(\mathbf{w}_i \cdot \mathbf{v}_x + b_i), & \text{if } j = i+1 \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathbf{u}_i$ and $\mathbf{w}_i$ are vectors of parameters, a_i and b_i are scalar parameters, $\mathbf{v}_x$ is a fixed pre-trained word vector for x,[4] and E is an encoding function, typically the identity function or sigmoid. ϵ-transitions are also parameterized, but don't consume a token and depend only on the current state:

$$[\mathbf{T}(\epsilon)]_{i,j} = \begin{cases} E(c_i), & \text{if } j = i+1 \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where c_i is a scalar parameter.[5] As we have only three non-zero diagonals in total, the matrix multiplications in Equation 2 can be implemented using vector operations, and the overall runtimes of Forward and Viterbi are reduced to $O(dn)$.[6]

Words vs. wildcards. Traditional *hard* patterns distinguish between words and wildcards. Our model does not explicitly capture the notion of either, but the transition weight function can be interpreted in those terms. Each transition is a logistic regression over the next word vector $\mathbf{v}_x$. For example, for a main path out of state i, $\mathbf{T}$ has two parameters, $\mathbf{w}_i$ and b_i. If $\mathbf{w}_i$ has large magnitude and is close to the word vector for some word y (e.g., $\mathbf{w}_i \approx 100\mathbf{v}_y$), and b_i is a large negative bias (e.g., $b_i \approx -100$), then the transition is essentially matching the specific word y. Whereas if $\mathbf{w}_i$ has small magnitude ($\mathbf{w}_i \approx \mathbf{0}$) and b_i is a large positive bias (e.g., $b_i \approx 100$), then the transition is ignoring the current token and matching a wildcard.[7] The transition could also be something in between, for instance by focusing on specific dimensions of a word's meaning encoded in the vector, such as POS or semantic features like animacy or concreteness (Rubinstein et al., 2015; Tsvetkov et al., 2015).

3.2 Scoring Documents

So far we described how to calculate how well a pattern matches a token span exactly (consuming the whole span). To score a complete document, we prefer a score that aggregates over all matches on subspans of the document (similar to "search" instead of "match" in regular expression parlance). We still assume a single pattern.

Either the Forward algorithm can be used to calculate the expected count of the pattern in the document, $\sum_{1 \le i \le j \le n} p_{\text{span}}(\boldsymbol{x}_{i:j})$, or Viterbi to calculate $s_{\text{doc}}(\boldsymbol{x}) = \max_{1 \le i \le j \le n} s_{\text{span}}(\boldsymbol{x}_{i:j})$, the score of the highest-scoring match. In short documents, we expect patterns to typically occur at most once, so in our experiments we choose the Viterbi algorithm, i.e., the max-product semiring.

Implementation details. We give the specific recurrences we use to score documents in a single

[3] To ensure that we start in the `START` state and end in the `END` state, we fix $\pi = [1, 0, \ldots, 0]$ and $\eta = [0, \ldots, 0, 1]$.

[4] We use GloVe 300d 840B (Pennington et al., 2014).

[5] Adding ϵ-transitions to WFSAs does not increase their expressive power, and in fact slightly complicates the Forward equations. We use them as they require fewer parameters, and make the modeling connection between (hard) flexible patterns and our (soft) patterns more direct and intuitive.

[6] Our implementation is optimized to run on GPUs, so the observed runtime is even *sublinear* in d.

[7] A large bias increases the eagerness to match *any* word.

pass with this model. We define:

$$[\mathrm{maxmul}(\mathbf{A}, \mathbf{B})]_{i,j} = \max_k A_{i,k} B_{k,j}. \quad (5)$$

We also define the following for taking zero or one ϵ-transitions:

$$\mathrm{eps}\,(\mathbf{h}) = \mathrm{maxmul}\,(\mathbf{h}, \max(\mathbf{I}, \mathbf{T}(\epsilon))) \quad (6)$$

where max is element-wise max. We maintain a row vector $\mathbf{h}_t$ at each token:[8]

$$\mathbf{h}_0 = \mathrm{eps}(\pi^\top), \quad (7a)$$
$$\mathbf{h}_{t+1} = \max\left(\mathrm{eps}(\mathrm{maxmul}\,(\mathbf{h}_t, \mathbf{T}(x_{t+1}))), \mathbf{h}_0\right), \quad (7b)$$

and then extract and aggregate END state values:

$$s_t = \mathrm{maxmul}\,(\mathbf{h}_t, \eta), \quad (8a)$$
$$s_{\mathrm{doc}} = \max_{1 \leq t \leq n} s_t. \quad (8b)$$

$[\mathbf{h}_t]_i$ represents the score of the best path through the pattern that ends in state i after consuming t tokens. By including $\mathbf{h}_0$ in Equation 7b, we are accounting for spans that start at time $t+1$. s_t is the maximum of the exact match scores for all spans ending at token t. And s_{doc} is the maximum score of any subspan in the document.

3.3 Aggregating Multiple Patterns

We describe how k patterns are aggregated to score a document. These k patterns give k different s_{doc} scores for the document, which are stacked into a vector $\mathbf{z} \in \mathbb{R}^k$ and constitute the final document representation of SoPa. This vector representation can be viewed as a feature vector. In this paper, we feed it into a multilayer perceptron (MLP), culminating in a softmax to give a probability distribution over document labels. We minimize cross-entropy, allowing the SoPa and MLP parameters to be learned end-to-end.

SoPa uses a total of $(2e+3)dk$ parameters, where e is the word embedding dimension, d is the number of states and k is the number of patterns. For comparison, an LSTM with a hidden dimension of h has $4((e+1)h+h^2)$. In Section 6 we show that SoPa consistently uses fewer parameters than a BiLSTM baseline to achieve its best result.

3.4 SoPa as an RNN

SoPa can be considered an RNN. As shown in Section 3.2, a single pattern with d states has a hidden state vector of size d. Stacking the k hidden state vectors of k patterns into one vector of size $k \times d$ can be thought of as the hidden state of our model. This hidden state is, like in any other RNN, dependent of the input and the previous state. Using self-loops, the hidden state at time point i can in theory depend on the entire history of tokens up to x_i (see Figure 2 for illustration). We do want to discourage the model from following too many self-loops, only doing so if it results in a better fit with the remainder of the pattern. To do this we use the sigmoid function as our encoding function E (see Equation 3), which means that all transitions have scores strictly less than 1. This works to keep pattern matches close to their intended length. Using other encoders, such as the identity function, can result in different dynamics, potentially encouraging rather than discouraging self-loops.

Although even single-layer RNNs are Turing complete (Siegelmann and Sontag, 1995), SoPa's expressive power depends on the semiring. When a WFSA is thought of as a function from finite sequences of tokens to semiring values, it is restricted to the class of functions known as **rational series** (Schützenberger, 1961; Droste and Gastin, 1999; Sakarovitch, 2009).[9] It is unclear how limiting this theoretical restriction is in practice, especially when SoPa is used as a component in a larger network. We defer the investigation of the exact computational properties of SoPa to future work. In the next section, we show that SoPa is an extension of a one-layer CNN, and hence more expressive.

4 SoPa as a CNN Extension

A convolutional neural network (**CNN**; LeCun, 1998) moves a fixed-size sliding window over the document, producing a vector representation for each window. These representations are then often summed, averaged, or max-pooled to produce a document-level representation (Kim, 2014; Yin and Schütze, 2015). In this section, we show that SoPa is an extension of one-layer, max-pooled CNNs.

To recover a CNN from a soft pattern with $d+1$ states, we first remove self-loops and ϵ-transitions,

[8]Here a row vector $\mathbf{h}$ of size n can also be viewed as a $1 \times n$ matrix.

[9]Rational series generalize recognizers of *regular languages*, which are the special case of the Boolean semiring.

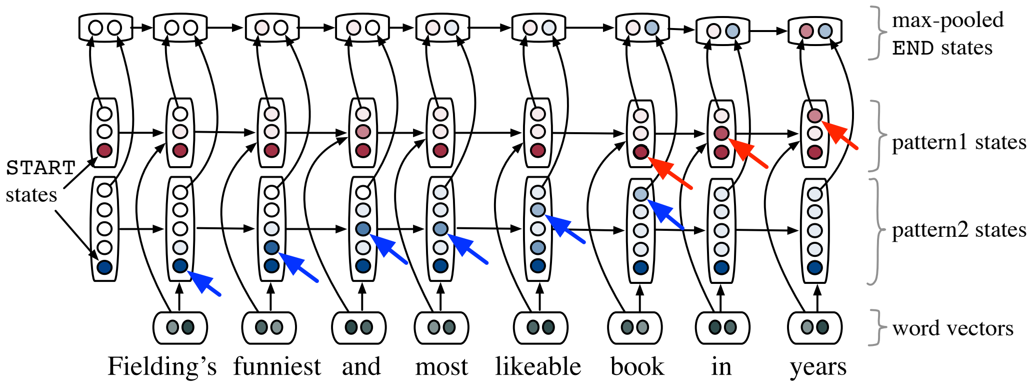


Figure 2: State activations of two patterns as they score a document. pattern1 (length three) matches on *"in years"*. pattern2 (length five) matches on *"funniest and most likeable book"*, using a self-loop to consume the token *"most"*. Active states in the best match are marked with arrow cursors.

retaining only the main path transitions. We also use the identity function as our encoder E (Equation 3), and use the max-sum semiring. With only main path transitions, the network will not match any span that is not exactly d tokens long. Using max-sum, spans of length d will be assigned the score:

$$s_{\text{span}}(\boldsymbol{x}_{i:i+d}) = \sum_{j=0}^{d-1} \mathbf{w}_j \cdot \mathbf{v}_{x_{i+j}} + b_j, \tag{9a}$$

$$= \mathbf{w}_{0:d} \cdot \mathbf{v}_{x_{i:i+d}} + \sum_{j=0}^{d-1} b_j, \tag{9b}$$

where $\mathbf{w}_{0:d} = [\mathbf{w}_0^\top; \ldots; \mathbf{w}_{d-1}^\top]^\top$, $\mathbf{v}_{x_{i:i+d}} = [\mathbf{v}_{x_i}^\top; \ldots; \mathbf{v}_{x_{i+d-1}}^\top]^\top$. Rearranged this way, we recognize the span score as an affine transformation of the concatenated word vectors $\mathbf{v}_{x_{i:i+d}}$. If we use k patterns, then together their span scores correspond to a linear filter with window size d and output dimension k.[10] A single pattern's score for a document is:

$$s_{\text{doc}}(\boldsymbol{x}) = \max_{1 \le i \le n-d+1} s_{\text{span}}(\boldsymbol{x}_{i:i+d}). \tag{10}$$

The $\max$ in Equation 10 is calculated for each pattern independently, corresponding exactly to element-wise max-pooling of the CNN's output layer.

Based on the equivalence between this impoverished version of SoPa and CNNs, we conclude that one-layer CNNs are learning an even more restricted class of WFSAs (linear-chain WFSAs) that capture only fixed-length patterns.

One notable difference between SoPa and arbitrary CNNs is that in general CNNs can use any filter (like an MLP over $\mathbf{v}_{x_{i:i+d}}$, for example). In contrast, in order to efficiently pool over flexible-length spans, SoPa is restricted to operations that follow the semiring laws.[11]

As a model that is more flexible than a one-layer CNN, but (arguably) less expressive than many RNNs, SoPa lies somewhere on the continuum between these two approaches. Continuing to study the bridge between CNNs and RNNs is an exciting direction for future research.

5 Experiments

To evaluate SoPa, we apply it to text classification tasks. Below we describe our datasets and baselines. More details can be found in Appendix A.

Datasets. We experiment with three binary classification datasets.

- **SST**. The Stanford Sentiment Treebank (Socher et al., 2013)[12] contains roughly 10K movie reviews from Rotten Tomatoes,[13] labeled on a scale of 1–5. We consider the binary task, which considers 1 and 2 as negative, and 4 and 5 as positive (ignoring 3s). It is worth noting that this dataset also contains syntactic phrase level annotations, providing a sentiment label to parts of

[10] This variant of SoPa has d bias parameters, which correspond to only a single bias parameter in a CNN. The redundant biases may affect optimization but are an otherwise unimportant difference.

[11] The max-sum semiring corresponds to a linear filter with max-pooling. Other semirings could potentially model more interesting interactions, but we leave this to future work.

[12] `https://nlp.stanford.edu/sentiment/index.html`

[13] `http://www.rottentomatoes.com`

sentences. In order to experiment in a realistic setup, we only consider the complete sentences, and ignore syntactic annotations at train or test time. The number of training/development/test sentences in the dataset is 6,920/872/1,821.

- **Amazon**. The Amazon Review Corpus (McAuley and Leskovec, 2013)[14] contains electronics product reviews, a subset of a larger review dataset. Each document in the dataset contains a review and a summary. Following Yogatama et al. (2015), we only use the reviews part, focusing on positive and negative reviews. The number of training/development/test samples is 20K/5K/25K.
- **ROC**. The ROC story cloze task (Mostafazadeh et al., 2016) is a story understanding task.[15] The task is composed of four-sentence story prefixes, followed by two competing endings: one that makes the joint five-sentence story coherent, and another that makes it incoherent. Following Schwartz et al. (2017), we treat it as a style detection task: we treat all "right" endings as positive samples and all "wrong" ones as negative, and we ignore the story prefix. We split the development set into train and development (of sizes 3,366 and 374 sentences, respectively), and take the test set as-is (3,742 sentences).

Reduced training data. In order to test our model's ability to learn from small datasets, we also randomly sample 100, 500, 1,000 and 2,500 SST training instances and 100, 500, 1,000, 2,500, 5,000, and 10,000 Amazon training instances. Development and test sets remain the same.

Baselines. We compare to four baselines: a BiLSTM, a one-layer CNN, DAN (a simple alternative to RNNs) and a feature-based classifier trained with hard-pattern features.

- **BiLSTM.** Bidirectional LSTMs have been successfully used in the past for text classification tasks (Zhou et al., 2016). We learn a one-layer BiLSTM representation of the document, and feed the average of all hidden states to an MLP.
- **CNN**. CNNs are particularly useful for text classification (Kim, 2014). We train a one-layer CNN with max-pooling, and feed the resulting representation to an MLP.
- **DAN**. We learn a deep averaging network with word dropout (Iyyer et al., 2015), a simple but strong text-classification baseline.
- **Hard**. We train a logistic regression classifier with hard-pattern features. Following Tsur et al. (2010), we replace low frequency words with a special wildcard symbol. We learn sequences of 1–6 concrete words, where any number of wildcards can come between two adjacent words. We consider words occurring with frequency of at least 0.01% of our training set as concrete words, and words occurring in frequency 1% or less as wildcards.[16]

Number of patterns. SoPa requires specifying the number of patterns to be learned, and their lengths. Preliminary experiments showed that the model doesn't benefit from more than a few dozen patterns. We experiment with several configurations of patterns of different lengths, generally considering 0, 10 or 20 patterns of each pattern length between 2–7. The total number of patterns learned ranges between 30–70.[17]

6 Results

Table 1 shows our main experimental results. In two of the cases (SST and ROC), SoPa outperforms all models. On Amazon, SoPa performs within 0.3 points of CNN and BiLSTM, and outperforms the other two baselines. The table also shows the number of parameters used by each model for each task. Given enough data, models with more parameters should be expected to perform better. However, SoPa performs better or roughly the same as a BiLSTM, which has 3–6 times as many parameters.

Figure 3 shows a comparison of all models on the SST and Amazon datasets with varying training set sizes. SoPa is substantially outperforming all baselines, in particular BiLSTM, on small datasets (100 samples). This suggests that SoPa is better fit to learn from small datasets.

Ablation analysis. Table 1 also shows an ablation of the differences between SoPa and CNN: max-product semiring with sigmoid vs. max-sum semiring with identity, self-loops, and ϵ-transitions. The last line is equivalent to a CNN with

[14]`http://riejohnson.com/cnn_data.html`

[15]`http://cs.rochester.edu/nlp/rocstories/`

[16]Some words may serve as both words and wildcards. See Davidov and Rappoport (2008) for discussion.

[17]The number of patterns and their length are hyperparameters tuned on the development data (see Appendix A).

Model	**ROC**	**SST**	**Amazon**
Hard	62.2 (4K)	75.5 (6K)	88.5 (67K)
DAN	64.3 (91K)	83.1 (91K)	85.4 (91K)
BiLSTM	65.2 (844K)	84.8 (1.5M)	**90.8** (844K)
CNN	64.3 (155K)	82.2 (62K)	90.2 (305K)
SoPa	**66.5** (255K)	**85.6** (255K)	90.5 (256K)
SoPa$_{ms_1}$	64.4	84.8	90.0
SoPa$_{ms_1}\setminus\{sl\}$	63.2	84.6	89.8
SoPa$_{ms_1}\setminus\{\epsilon\}$	64.3	83.6	89.7
SoPa$_{ms_1}\setminus\{sl,\epsilon\}$	64.0	85.0	89.5

Table 1: Test classification accuracy (and the number of parameters used). The bottom part shows our ablation results: SoPa: our full model. SoPa$_{ms_1}$: running with max-sum semiring (rather than max-product), with the identity function as our encoder E (see Equation 3). sl: self-loops, ϵ: ϵ transitions. The final row is equivalent to a one-layer CNN.

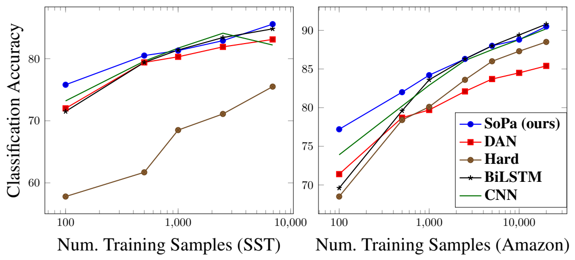


Figure 3: Test accuracy on SST and Amazon with varying number of training instances.

	Highest Scoring Phrases				
Patt. 1	thoughtful	,	reverent	portrait	of
	and	astonishingly	articulate	cast	of
	entertaining	,	thought-provoking	film	with
	gentle	,	mesmerizing	portrait	of
	poignant	and	uplifting	story	in
Patt. 2	's	ϵ	uninspired	story	.
	this	ϵ	bad	on	purpose
	this	ϵ	leaden	comedy	.
	a	ϵ	half-assed	film	.
	is	ϵ	clumsy ,$_{SL}$	the	writing
Patt. 3	mesmerizing	portrait	of	a	
	engrossing	portrait	of	a	
	clear-eyed	portrait	of	an	
	fascinating	portrait	of	a	
	self-assured	portrait	of	small	
Patt. 4	honest	,	and	enjoyable	
	soulful	, scathing$_{SL}$	and	joyous	
	unpretentious	, charming$_{SL}$	,	quirky	
	forceful	,	and	beautifully	
	energetic	,	and	surprisingly	
Patt. 5	is	deadly	dull		
	a	numbingly	dull		
	is	remarkably	dull		
	is	a	phlegmatic		
	an	utterly	incompetent		
Patt. 6	five	minutes			
	four	minutes			
	final	minutes			
	first	half-hour			
	fifteen	minutes			

Table 2: Six patterns of different lengths learned by SoPa on SST. Each group represents a single pattern p, and shows the five phrases in the training data that have the highest score for p. Columns represent pattern states. Words marked with $_{SL}$ are self-loops. ϵ symbols indicate ϵ-transitions. All other words are from main path transitions.

multiple window sizes. Interestingly, the most notable difference between SoPa and CNN is the semiring and encoder function, while ϵ transitions and self-loops have little effect on performance.[18]

7 Interpretability

We turn to another key aspect of SoPa—its interpretability. We start by demonstrating how we interpret a single pattern, and then describe how to interpret the decisions made by downstream classifiers that rely on SoPa—in this case, a sentence classifier. Importantly, these visualization techniques are equally applicable to CNNs.

Interpreting a single pattern. In order to visualize a pattern, we compute the pattern matching scores with each phrase in our training dataset, and select the k phrases with the highest scores. Table 2 shows examples of six patterns learned using the best SoPa model on the SST dataset, as represented by their five highest scoring phrases in the training set. A few interesting trends can be observed from these examples. First, it seems our patterns encode semantically coherent expressions. A large portion of them correspond to sentiment (the five top examples in the table), but others capture different semantics, e.g., time expressions.

Second, it seems our patterns are relatively soft, and allow lexical flexibility. While some patterns do seem to fix specific words, e.g., "of" in the first example or "minutes" in the last one, even in those cases some of the top matching spans replace these words with other, similar words ("with" and "half-hour", respectively). Encouraging SoPa to have more concrete words, e.g., by jointly learning the word vectors, might make SoPa useful in other contexts, particularly as a decoder. We defer this direction to future work.

Finally, SoPa makes limited but non-negligible use of self-loops and epsilon steps. Interestingly, the second example shows that one of the pat-

[18]Although SoPa does make use of them—see Section 7.

Analyzed Documents
it 's dumb , **but more importantly** , *it 's just not scary*
though moonlight mile is replete with **acclaimed actors and actresses** and tackles a subject that 's **potentially moving** , the movie is *too predictable* and *too self-conscious to reach a* level of **high drama**
While **its careful pace and** seemingly *opaque story* may not satisfy every moviegoer 's appetite, the film 's final scene is **soaringly , transparently moving**
unlike the speedy wham-bam effect *of most hollywood offerings* , character development – and more **importantly, character empathy – is at the heart of** italian for beginners .
the band 's courage in the face of official repression **is inspiring , especially for** aging *hippies* (this one included) .

Table 3: Documents from the SST training data. Phrases with the largest contribution toward a positive sentiment classification are in **bold green**, and the most negative phrases are in *italic orange*.

terns had an ϵ-transition at the same place in every phrase. This demonstrates a different function of ϵ-transitions than originally designed—they allow a pattern to effectively shorten itself, by learning a high ϵ-transition parameter for a certain state.

Interpreting a document. SoPa provides an interpretable representation of a document—a vector of the maximal matching score of each pattern with any span in the document. To visualize the decisions of our model for a given document, we can observe the patterns and corresponding phrases that score highly within it.

To understand which of the k patterns contributes most to the classification decision, we apply a leave-one-out method. We run the forward method of the MLP layer in SoPa k times, each time zeroing-out the score of a different pattern p. The difference between the resulting score and the original model score is considered p's contribution. We then consider the highest contributing patterns, and attach each one with its highest scoring phrase in that document. Table 3 shows example texts along with their most positive and negative contributing phrases.

8 Related Work

Weighted finite-state automata. WFSAs and hidden Markov models[19] were once popular in automatic speech recognition (Hetherington, 2004; Moore et al., 2006; Hoffmeister et al., 2012) and remain popular in morphology (Dreyer, 2011; Cotterell et al., 2015). Most closely related to this work, neural networks have been combined with weighted finite-state transducers to do morphological reinflection (Rastogi et al., 2016). These prior works learn a single FSA or FST, whereas our model learns a collection of simple but complementary FSAs, together encoding a sequence. We are the first to incorporate neural networks both *before* WFSAs (in their transition scoring functions), and *after* (in the function that turns their vector of scores into a final prediction), to produce an expressive model that remains interpretable.

[19] HMMs are a special case of WFSAs (Mohri et al., 2002).

Recurrent neural networks. The ability of RNNs to represent arbitrarily long sequences of embedded tokens has made them attractive to NLP researchers. The most notable variants, the long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU; Cho et al., 2014), have become ubiquitous in NLP algorithms (Goldberg, 2016). Recently, several works introduced simpler versions of RNNs, such as recurrent additive networks (Lee et al., 2017) and Quasi-RNNs (Bradbury et al., 2017). Like SoPa, these models can be seen as points along the bridge between RNNs and CNNs.

Other works have studied the expressive power of RNNs, in particular in the context of WFSAs or HMMs (Cleeremans et al., 1989; Giles et al., 1992; Visser et al., 2001; Chen et al., 2018). In this work we relate CNNs to WFSAs, showing that a one-layer CNN with max-pooling can be simulated by a collection of linear-chain WFSAs.

Convolutional neural networks. CNNs are prominent feature extractors in NLP, both for generating character-based embeddings (Kim et al., 2016), and as sentence encoders for tasks like text classification (Yin and Schütze, 2015) and machine translation (Gehring et al., 2017). Similarly to SoPa, several recently introduced variants of CNNs support varying window sizes by either allowing several fixed window sizes (Yin and Schütze, 2015) or by supporting non-consecutive n-gram matching (Lei et al., 2015; Nguyen and Grishman, 2016).

Neural networks and patterns. Some works used patterns as part of a neural network. Schwartz et al. (2016) used pattern contexts for estimating word embeddings, showing improved word similarity results compared to bag-of-word contexts. Shwartz et al. (2016) designed an LSTM representation for dependency patterns, using them to detect hypernymy relations. Here, we learn patterns as a neural version of WFSAs.

Interpretability. There have been several efforts to interpret neural models. The weights of the attention mechanism (Bahdanau et al., 2015) are often used to display the words that are most significant for making a prediction. LIME (Ribeiro et al., 2016) is another approach for visualizing neural models (not necessarily textual). Yogatama and Smith (2014) introduced structured sparsity, which encodes linguistic information into the regularization of a model, thus allowing to visualize the contribution of different bag-of-word features.

Other works jointly learned to encode text and extract the span which best explains the model's prediction (Yessenalina et al., 2010; Lei et al., 2016). Li et al. (2016) and Kádár et al. (2017) suggested a method that erases pieces of the text in order to analyze their effect on a neural model's decisions. Finally, several works presented methods to visualize deep CNNs (Zeiler and Fergus, 2014; Simonyan et al., 2014; Yosinski et al., 2015), focusing on visualizing the different layers of the network, mainly in the context of image and video understanding. We believe these two types of research approaches are complementary: inventing general purpose visualization tools for existing black-box models on the one hand, and on the other, designing models like SoPa that are interpretable by construction.

9 Conclusion

We introduced SoPa, a novel model that combines neural representation learning with WFSAs. We showed that SoPa is an extension of a one-layer CNN. It naturally models flexible-length spans with insertion and deletion, and it can be easily customized by swapping in different semirings. SoPa performs on par with or strictly better than four baselines on three text classification tasks, while requiring fewer parameters than the stronger baselines. On smaller training sets, SoPa outperforms all four baselines. As a simple version of an RNN, which is more expressive than one-layer CNNs, we hope that SoPa will encourage future research on the bridge between these two mechanisms. To facilitate such research, we release our implementation at `https://github.com/Noahs-ARK/soft_patterns`.

Acknowledgments

We thank Dallas Card, Elizabeth Clark, Peter Clark, Bhavana Dalvi, Jesse Dodge, Nicholas FitzGerald, Matt Gardner, Yoav Goldberg, Mark Hopkins, Vidur Joshi, Tushar Khot, Kelvin Luu, Mark Neumann, Hao Peng, Matthew E. Peters, Sasha Rush, Ashish Sabharwal, Minjoon Seo, Sofia Serrano, Swabha Swayamdipta, Chenhao Tan, Niket Tandon, Trang Tran, Mark Yatskar, Scott Yih, Vicki Zayats, Rowan Zellers, Luke Zettlemoyer, and several anonymous reviewers for their helpful advice and feedback. This work was supported in part by NSF grant IIS-1562364, by the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant ACI-1548562, and by the NVIDIA Corporation through the donation of a Tesla GPU.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Leonard E. Baum and Ted Petrie. 1966. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Network. In *Proc. of ICLR*.

Yining Chen, Sorcha Gilroy, Kevin Knight, and Jonathan May. 2018. Recurrent neural networks as weighted language recognizers. In *Proc. of NAACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Axel Cleeremans, David Servan-Schreiber, and James L McClelland. 1989. Finite state automata and simple recurrent networks. *Neural computation*, 1(3):372–381.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *TACL*, 3:433–447.

Dmitry Davidov and Ari Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proc. of ACL*.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proc. of COLING*.

Markus Dreyer. 2011. *A Non-parametric Model for the Discovery of Inflectional Paradigms from Plain Text Using Graphical Models over Strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD, USA.

Manfred Droste and Paul Gastin. 1999. The Kleene–Schützenberger theorem for formal power series in partially commuting variables. *Information and Computation*, 153(1):47–80.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper).

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. of ICML*.

C. Lee Giles, Clifford B Miller, Dong Chen, Hsing-Hen Chen, Guo-Zheng Sun, and Yee-Chun Lee. 1992. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *JAIR*, 57:345–420.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*.

Lee Hetherington. 2004. The MIT finite-state transducer toolkit for speech and language processing. In *Proc. of INTERSPEECH*.

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

Björn Hoffmeister, Georg Heigold, David Rybach, Ralf Schlüter, and Hermann Ney. 2012. WFST enabled solutions to ASR problems: Beyond HMM decoding. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:551–564.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Ákos Kádár, Grzegorz Chrupala, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43:761–780.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proc. of EMNLP*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proc. of AAAI*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Yann LeCun. 1998. Gradient-based Learning Applied to Document Recognition. In *Proc. of the IEEE*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. arXiv:1705.07393.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding CNNs for text: non-linear, non-consecutive convolutions. In *Proc. of EMNLP*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing Neural Predictions. In *Proc. of EMNLP*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding Neural Networks through Representation Erasure. arXiv:1612.08220.

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *Proc. of IJCAI*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proc. of RecSys*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Darren Moore, John Dines, Mathew Magimai-Doss, Jithendra Vepa, Octavian Cheng, and Thomas Hain. 2006. Juicer: A weighted finite-state transducer speech decoder. In *MLMI*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proc. of NAACL*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling Skip-Grams for Event Detection with Convolutional Neural Networks. In *Proc. of EMNLP*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proc. of NAACL*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *Proc. of KDD*.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proc. of ACL*.

Jacques Sakarovitch. 2009. Rational and recognisable power series. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 105–174. Springer Berlin Heidelberg, Berlin, Heidelberg.

M. P. Schützenberger. 1961. On the definition of a family of automata. *Information and Control*, 4(2):245–270.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proc. of CoNLL*.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2016. Symmetric patterns and coordinations: Fast and enhanced representations of verbs and adjectives. In *Proc. of NAACL*.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. In *Proc.of CoNLL*.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proc. of ACL*.

Hava T. Siegelmann and Eduardo D. Sontag. 1995. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. of ICLR Workshop*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. ICWSM–a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proc. of ICWSM*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proc. of EMNLP*.

Ingmar Visser, Maartje EJ Raijmakers, and Peter CM Molenaar. 2001. Hidden markov model interpretations of neural networks. In *Connectionist Models of Learning, Development and Evolution*, pages 197–206. Springer.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proc. of EMNLP*.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel Variable-Size Convolution for Sentence Classification. In *Proc. of CoNLL*.

Dani Yogatama, Lingpeng Kong, and Noah A. Smith. 2015. Bayesian optimization of text representations. In *Proc. of EMNLP*.

Dani Yogatama and Noah A. Smith. 2014. Linguistic structured sparsity in text categorization. In *Proc. of ACL*.

Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. In *Proc. of the ICML Deep Learning Workshop*.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proc. of ECCV*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proc. of COLING*.

Zero-shot Learning of Classifiers from Natural Language Quantification

Shashank Srivastava **Igor Labutov** **Tom Mitchell**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ssrivastava@cmu.edu ilabutov@cs.cmu.edu tom.mitchell@cmu.edu

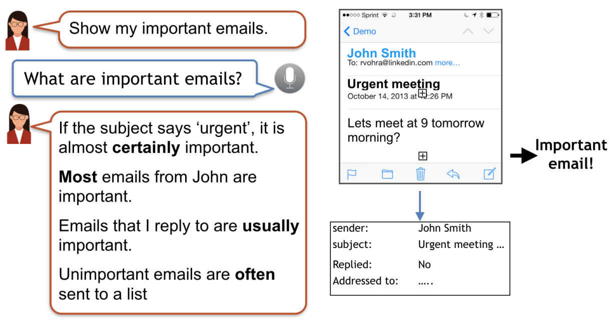
Abstract

Humans can efficiently learn new concepts using language. We present a framework through which a set of explanations of a concept can be used to learn a classifier without access to any labeled examples. We use semantic parsing to map explanations to probabilistic assertions grounded in latent class labels and observed attributes of unlabeled data, and leverage the differential semantics of linguistic quantifiers (e.g., 'usually' vs 'always') to drive model training. Experiments on three domains show that the learned classifiers outperform previous approaches for learning with limited data, and are comparable with fully supervised classifiers trained from a small number of labeled examples.

1 Introduction

As computer systems that interact with us in natural language become pervasive (e.g., Siri, Alexa, Google Home), they suggest the possibility of letting users teach machines in language. The ability to learn from language can enable a paradigm of ubiquitous machine learning, allowing users to teach personalized concepts (e.g., identifying 'important emails' or 'project-related emails') when limited or no training data is available.

In this paper, we take a step towards solving this problem by exploring the use of quantifiers to train classifiers from declarative language. For illustration, consider the hypothetical example of a user explaining the concept of an "important email" through natural language statements (Figure 1). Our framework takes a set of such natural language explanations describing a concept (e.g., "emails that I reply to are usually important") and a set of unlabeled instances as input, and produces


Show my important emails.

What are important emails?

If the subject says 'urgent', it is almost **certainly** important.

Most emails from John are important.

Emails that I reply to are **usually** important.

Unimportant emails are **often** sent to a list

Demo
John Smith
To: rvohra@linkedin.com more...
Urgent meeting
October 14, 2013 at 12:26 PM
Lets meet at 9 tomorrow morning?

sender:	John Smith
subject:	Urgent meeting ...
Replied:	No
Addressed to:	.....

Figure 1: Supervision from language can enable concept learning from limited or even no labeled examples. Our approach assumes the learner has sensors that can extract attributes from data, such as those listed in the table, and language that can refer to these sensors and their values.

a binary classifier (for important emails) as output. Our hypothesis is that language describing concepts encodes key properties that can aid statistical learning. These include specification of relevant attributes (e.g., whether an email was replied to), relationships between such attributes and concept labels (e.g., if a reply implies the class label of that email is 'important'), as well as the strength of these relationships (e.g., via quantifiers like 'often', 'sometimes', 'rarely'). We infer these properties automatically, and use the semantics of linguistic quantifiers to drive the training of classifiers *without labeled examples for any concept*. This is a novel scenario, where previous approaches in semi-supervised and constraint-based learning are not directly applicable. Those approaches require manual pre-specification of expert knowledge for model training. In our approach, this knowledge is automatically inferred from noisy natural language explanations from a user.

Our approach is summarized in the schematic in Figure 2. First, we map the set of natural language explanations of a concept to logical forms

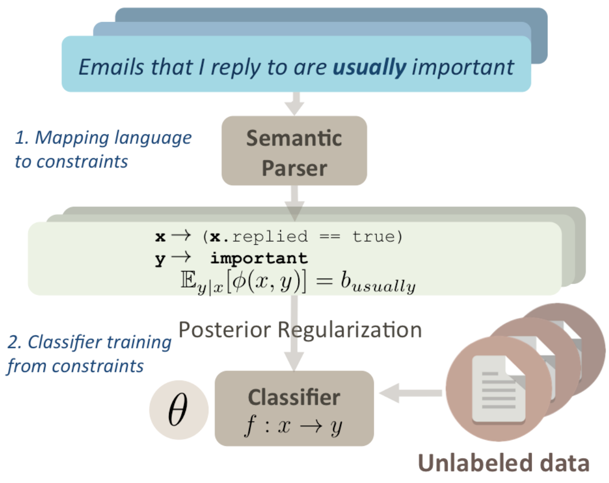

Emails that I reply to are usually important
1. Mapping language to constraints
Semantic Parser
x → (x.replied == true)
y → important
$\mathbb{E}_{y|x}[\phi(x,y)] = b_{usually}$
2. Classifier training from constraints
Posterior Regularization
θ
Classifier $f: x \to y$
Unlabeled data

Figure 2: Our approach to Zero-shot learning from Language. Natural language explanations on how to classify concept examples are parsed into formal constraints relating features to concept labels. The constraints are combined with unlabeled data, using posterior regularization to yield a classifier.

that identify the attributes mentioned in the explanation, and describe the information conveyed about the attribute and the concept label as a quantitative constraint. This mapping is done through semantic parsing. The logical forms denote quantitative constraints, which are probabilistic assertions about observable attributes of the data and unobserved concept labels. Here the strength of a constraint is assumed to be specified by a linguistic quantifier (such as 'all', 'some', 'few', etc., which reflect degrees of generality of propositions). Next, we train a classification model that can assimilate these constraints by adapting the posterior regularization framework (Ganchev et al., 2010).

Intuitively, this can be seen as defining an optimization problem, where the objective is to find parameter estimates for the classifier that do not simply fit the data, but also agree with the human provided natural language advice to the greatest extent possible. Since logical forms can be grounded in a variety of sensors and external resources, an explicit model of semantic interpretation conceptually allows the framework to subsume a flexible range of grounding behaviors. The main contributions of this work are:

1. We introduce the problem of zero-shot learning of classifiers from language, and present an approach towards this.
2. We develop datasets for zero-shot classification from natural descriptions, exhibiting tasks with various levels of difficulty.
3. We empirically show that coarse probability estimates to model linguistic quantifiers can effectively supervise model training across three domains of classification tasks.

2 Related Work

Many notable approaches have explored incorporation of background knowledge into the training of learning algorithms. However, none of them addresses the issue of learning from natural language. Prominent among these are the Constraint-driven learning (Chang et al., 2007a), Generalized Expectation (Mann and McCallum, 2010) and Posterior Regularization (Ganchev et al., 2010) and Bayesian Measurements (Liang et al., 2009) frameworks. All of these require domain knowledge to be manually programmed in before learning. Similarly, Probabilistic Soft Logic (Kimmig et al., 2012) allows users to specify rules in a logical language that can be used for reasoning over graphical models. More recently, multiple approaches have explored few-shot learning from perspective of term or attribute-based transfer (Lampert et al., 2014), or learning representations of instances as probabilistic programs (Lake et al., 2015).

Other work (Lei Ba et al., 2015; Elhoseiny et al., 2013) considers language terms such as colors and textures that can be directly grounded in visual meaning in images. Some previous work (Srivastava et al., 2017) has explored using language explanations for feature space construction in concept learning tasks, where the problem of learning to interpret language, and learning classifiers is treated jointly. However, this approach assumes availability of labeled data for learning classifiers. Also notable is recent work by Andreas et al. (2017), who propose using language descriptions as parameters to model structure in learning tasks in multiple settings. More generally, learning from language has also been previously explored in tasks such as playing games (Branavan et al., 2012), robot navigation (Karamcheti et al., 2017), etc.

Natural language quantification has been studied from multiple perspectives in formal logic (Barwise and Cooper, 1981), linguistics (Löbner, 1987; Bach et al., 2013) and cognitive psychology (Kurtzman and MacDonald, 1993). While quantification has traditionally been defined in set-theoretic terms in linguistic theories[1], our approach joins alternative

[1] e.g., *'some A are B'* $\Leftrightarrow A \cap B \neq \emptyset$

perspectives that represent quantifiers probabilistically (Moxey and Sanford, 1993; Yildirim et al., 2013). To the best of our knowledge, this is the first work to leverage the semantics of quantifiers to guide statistical learning models.

3 Learning Classifiers from Language

Our approach relies on first mapping natural language descriptions to quantitative constraints that specify statistical relationships between observable attributes of instances and their latent concept labels (Step 1 in Figure 2). These quantitative constraints are then imbued into the training of a classifier by guiding predictions from the learned models to concur with them (Step 2). We use semantic parsing to interpret sentences as quantitative constraints, and adapt the posterior regularization principle for our setting to estimate the classifier parameters. Next, we describe these steps in detail. Since learning in this work is largely driven by the semantics of linguistic quantifiers, we call our approach **Learning from Natural Quantification**, or **LNQ**.

3.1 Mapping language to constraints

A key challenge in learning from language is converting free-form language to representations that can be reasoned over, and grounded in data. For example, a description such as '*emails that I reply to are usually important*' may be converted to a mathematical assertion such as $P(important \mid replied : true) = 0.7$', which statistical methods can reason with. Here, we argue that this process can be automated for a large number of real-world descriptions. In interpreting statements describing concepts, we infer the following key elements:

1. *Feature* x, which is grounded in observed attributes of the data. For our example, 'emails replied to' can refer to a predicate such as `replied:true`, which can be evaluated in context of emails to indicate the whether an email was replied to. Incorporating compositional representations enables more complex reasoning. e.g., '*the subject of course-related emails usually mentions CS100*' can map to a composite predicate like '`isStringMatch(field:subject, stringVal('CS100'))`', which can be evaluated for different emails to reflect whether their subject mentions 'CS100'. Mapping language to executable feature functions has been shown to be effective (Srivastava et al., 2017). For sake of simplicity, here we assume that a statement refers to a single feature, but the method can be extended to handle more complex descriptions.
2. *Concept label* y, specifying the class of instances a statement refers to. For binary classes, this reduces to examples or non-examples of a concept. For our running example, y corresponds to the positive class of important emails.
3. *Constraint-type* asserted by the statement. We argue that most concept descriptions belong to one of three categories shown in Table 2, and these constitute our vocabulary of constraint types for this work. For our running example ('emails that I reply to are usually important'), the type corresponds to $P(y \mid x)$, since the syntax of the statement indicates an assertion conditioned on the feature indicating whether an email was replied to. On the other hand, an assertion such as 'I usually reply to important emails' indicates an assertion conditioned on the set important emails, and therefore corresponds to the type $P(x \mid y)$.
4. *Strength* of the constraint. We assume this to be specified by a quantifier. For our running example, this corresponds to the adverb 'usually'. In this work, by *quantifier* we specifically refer to frequency adverbs ('usually','rarely', etc.) and frequency determiners ('few', 'all', etc.).[2] Our thesis is that the semantics of quantifiers can be leveraged to make statistical assertions about relationships involving attributes and concept labels. One way to do this might be to simply associate point estimates of probability values, suggesting the fraction of truth values for assertions described with these quantifiers. Table 1 shows probability values we assign to some common frequency quantifiers for English. These values were set simply based on the authors' intuition about their semantics, and do not reflect any empirical distributions. See Figure 8 for empirical distributions corresponding to some linguistic quantifiers in our data. While these probability values maybe inaccurate, and the semantics of these quantifiers may also change based on context and the speaker, they can still serve as a strong signal for learning classifiers since they are not used as hard constraints, but serve to bias classifiers towards better generalization.

We use a semantic parsing model to map statements to formal semantic representations that specify these aspects. For example, the statement 'Emails that I reply to are usually important' is

[2]This is a significantly restricted definition, and does not address non-frequency determiners (e.g.,'the', 'only', etc.) or mass quantifiers (e.g. 'a lot', 'little'), among other categories.

Frequency quantifier	Probability
all, always, certainly, definitely	0.95
usually, normally, generally, likely, typically	0.70
most, majority	0.60
often, half	0.50
many	0.40
sometimes, frequently, some	0.30
few, occasionally	0.20
rarely, seldom	0.10
never	0.05

Table 1: Probability values we assign to common linguistic quantifiers (hyper-parameters for method)

mapped to a logical form like `(x→replied:true y→positive type:y|x quant:usually)`.

3.1.1 Semantic Parser components

Given a descriptive statement s, the parsing problem consists of predicting a logical form l that best represents its meaning. In turn, we formulate the probability of the logical form l as decomposing into three component factors: (i) probability of observing a feature and concept labels l_{xy} based on the text of the sentence, (ii) probability of the type of the assertion l_{type} based on the identified feature, concept label and syntactic properties of the sentence s, and (iii) identifying the linguistic quantifier, l_{quant}, in the sentence.

$$P(l \mid s) = P(l_{xy} \mid s)\ P(l_{type} \mid l_{xy}, s)\ P(l_{quant} \mid s)$$

We model each of the three components as follows: by using a traditional semantic parser for the first component, training a Max-Ent classifier for the constraint-type for the second component, and looking for an explicit string match to identify the quantifier for the third component.

Identifying features and concept labels, l_{xy}: For identifying the feature and concept label mentioned in a sentence, we presume a linear score $\mathcal{S}(s, l_{xy}) = w^T \psi(s, l_{xy})$ indicating the goodness of assigning a partial logical form, l_{xy}, to a sentence s. Here, $\psi(s, l_{xy}) \in \mathbb{R}^n$ are features that can depend on both the sentence and the partial logical form, and $w \in \mathbb{R}^n$ is a parameter weight-vector for this component. Following recent work in semantic parsing (Liang et al., 2011), we assume a loglinear distribution over interpretations of a sentence.

$$P(l_{xy} \mid s) \propto w^T \psi(s, l_{xy})$$

Provided data consisting of statements labeled with logical forms, the model can be trained via maximum likelihood estimation, and be used to predict interpretations for new statements. For training this component, we use a CCG semantic parsing formalism, and follow the feature-set from Zettlemoyer and Collins (2007), consisting of simple indicator features for occurrences of keywords and lexicon entries. This is also compatible with the semantic parsing formalism in Srivastava et al. (2017), whose data (and accompanying lexicon) are also used in our evaluation. For other datasets with predefined features, this component is learned easily from simple lexicons consisting of trigger words for features and labels.[3] This component is the only part of the parser that is domain-specific. We note that while this component assumes a domain-specific lexicon (and possibly statement annotated with logical forms), this effort is one-time-only, and will find re-use across the possibly large number of concepts in the domain (e.g., email categories).

Identifying assertion type, l_{type}: The principal novelty in our semantic parsing model is in identifying the type of constraint asserted by a statement. For this, we train a MaxEnt classifier, which uses positional and syntactic features based on the text-spans corresponding to feature and concept mentions to predict the constraint type. We extract the following features from a statement:
1. Boolean value indicating whether the text-span corresponding to the feature x precedes the text span for the concept label y.
2. Boolean value indicating if sentence is in passive (rather than active) voice, as identified by the occurrence of `nsubjpass` dependency relation.
3. Boolean value indicating whether head of the text-span for x is a noun, or a verb.
4. Features indicating the occurrence of conditional tokens ('if', 'then' and 'that') preceding or following text-spans for x and y.
5. Features indicating presence of a linguistic quantifier in a `det` or an `advmod` relation with syntactic head of x or y.

Since the constraint type is determined by syntactic and dependency parse features, this

[3]We also identify whether a feature x is negated, through the existence of a `neg` dependency relation with the head of its text-span. e.g., *Important emails are usually not deleted*

Type	Example description	Conversion to Expectation Constraint
$P(y \mid x)$	Emails that I reply to are usually important	$\mathbb{E}[\mathbb{I}_{y=important,reply(x):true}] - p_{usually} \times \mathbb{E}[\mathbb{I}_{reply(x):true}] = 0$
$P(x \mid y)$	I often reply to important emails	$\mathbb{E}[\mathbb{I}_{y=important,reply(x):true}] - p_{often} \times \mathbb{E}[\mathbb{I}_{y=important}] = 0$
$P(y)$	I rarely get important emails	Same as $P(y \vert x_0)$, where x_0 is a constant feature

Table 2: Common constraint-types, and their representation as expectations over feature values

component does not need to be retrained for new domains. In this work, we trained this classifier based on a manually annotated set of 80 sentences describing classes in the small UCI Zoo dataset (Lichman, 2013), and used this model for all experiments.

Identifying quantifiers, l_{quant}: Multiple linguistic quantifiers in a sentence are rare, and we simply look for the first occurrence of a linguistic quantifier in a sentence, i.e. $P(l_{quant}|s)$ is a deterministic function. We note that many real world descriptions of concepts lack an explicit quantifier. e.g., '*Emails from my boss are important*'. In this work, we ignore such statements for the purpose of training. Another treatment might be to models these statements as reflecting a default quantifier, but we do not explore this direction here. Finally, the decoupling of quantification from logical representation is a key decision. At the cost of linguistic coarseness, this allows modeling quantification irrespective of the logical representation (lambda calculus, predicate-argument structures, etc.).

3.2 Classifier training from constraints

In the previous section, we described how individual explanations can be mapped to probabilistic assertions about observable attributes (e.g., the statement 'Emails that I reply to are usually important' may map to $P(y = important \mid replied = true) = p_{usually}$). Here, we describe how a set of such assertions can be used in conjunction with unlabeled data to train classification models.

Our approach relies on having predictions from the classifier on a set of unlabeled examples ($X = \{x_1 \ldots x_n\}$) agree with human-provided advice (in form of constraints). The unobserved concept labels ($Y = \{y_1 \ldots y_n\}$) for the unlabeled data constitute *latent variables* for our method. The training procedure can be seen as iteratively inferring the latent concept labels for unlabeled examples so as to agree with the human advice, and updating the classification models by taking these labels as given. While there are multiple approaches for training statistical models with constraints on latent variables, here we use the Posterior Regularization (PR) framework. The PR objective can be used to optimize a latent variable model subject to a set of constraints, which specify preferences for values of the posterior distributions $p_\theta(Y \mid X)$.

$$J_Q(\theta) = \mathcal{L}(\theta) - \min_{q \in Q} KL(q \mid p_\theta(Y|X))$$

Here, the set Q represents a set of *preferred* posterior distributions over latent variables Y, and is defined as $Q := \{q_X(Y) : \mathbb{E}_q[\phi(X,Y)] \leq \mathbf{b}\}$. The overall objective consists of two components, representing how well does a model θ explain the data (likelihood term $\mathcal{L}(\theta)$), and how far it is from the set Q (KL-divergence term).

In our case, each parsed statement defines a probabilistic constraint. The conjunction of all such constraints defines Q (representing models that exactly agree with human-provided advice). Thus, optimizing the objective reflects a tension between choosing models that increase data likelihood, and emulating language advice.

Converting to PR constraints: The set of constraints that PR can handle can be characterized as bounds on expected values of functions (ϕ) of X and Y (or equivalently, from linearity of expectation, as linear inequalities over expected values of functions of X and Y). To use the framework, we need to ensure that each constraint type in our vocabulary can be expressed in such a form.

Following the plan in Table 2, each constraint type can be converted in an equivalent form $\mathbb{E}_q[\phi(X,Y)] = b$, compatible with PR. In particular, each of these constraint types in our vocabulary can be expressed as equations about expectation values of joint indicator functions of label assignments to instances and their attributes. To explain, consider the assertion $P(y = important \mid replied : true) = p_{usually}$. The probability on the LHS can be expressed as the empirical fraction $\frac{\sum_i \mathbb{E}[\mathbb{I}_{y_i=important,replied:true}]}{\sum_i \mathbb{E}[\mathbb{I}_{replied:true}]}$, which leads to the linear constraints seen in Table 2 (expected values in the table hide summations over instances for brevity). Here, $\mathbb{I}$ denote indicator functions. Thus, we can incorporate probability constraints into our adaptation of the PR scheme.

Learning and Inference: We choose a loglinear parameterization for the concept classifier.

$$p_\theta(y_i \mid x_i) \propto \exp(y\theta^T x)$$

The training of the classifier follows the modified EM procedure described in Ganchev et al. (2010). As proposed in the original work, we solve a relaxed version of the optimization that allows slack variables, and modifies the PR objective with a L_2 regularizer. This allows solutions even when the problem is over-constrained, and the set Q is empty (e.g. due to contradictory advice).

$$J'(\theta, q) = \mathcal{L}(\theta) - KL(q|p_\theta(Y|X)) \\ - \lambda \, ||\mathbb{E}_q[\phi(X,Y)] - b||^2$$

The key step in the training is the computation of the posterior regularizer in the E-step.

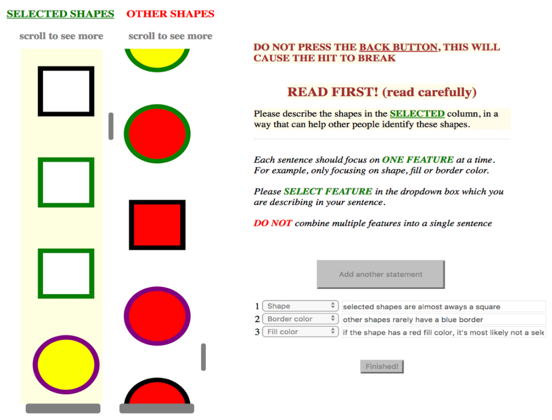
$$\operatorname*{argmin}_q KL(q \mid p_\theta) + \lambda \, ||\mathbb{E}_q[\phi(X,Y)] - b||^2$$

This objective is strictly convex, and all constraints are linear in q. We follow the optimization procedure from Bellare et al. (2009), whereby the minimization problem in the E-step can be efficiently solved through gradient steps in the dual space. In the M-step, we update the model parameters for the classifier based on label distributions q estimated in the E-step. This simply reduces to estimating the parameters θ for the logistic regression classifier, when class label probabilities are known. In all experiments, we run EM for 20 iterations and use a regularization coefficient of $\lambda = 0.1$.

4 Datasets

For evaluating our approach, we created datasets of classification tasks paired with descriptions of the classes, as well as used some existing resources. In this section, we summarize these steps.

Shapes data: To experiment with our approach in a wider range of controlled settings, part of our evaluation focuses on synthetic concepts. For this, we created a set of 50 shape classification tasks that exhibit a range of difficulty, and elicited language descriptions spanning a variety of quantifier expressions. The tasks require classifying geometric shapes with a set of predefined attributes (fill color, border, color, shape, size) into two concept-labels (abstractly named 'selected shape', and 'other'). The datasets were created through a generative process, where features x_i are conditionally independent given the concept-label. Each feature's conditional distribution is sampled from a symmetric



DO NOT PRESS THE BACK BUTTON, THIS WILL CAUSE THE HIT TO BREAK

READ FIRST! (read carefully)

Please describe the shapes in the SELECTED column, in a way that can help other people identify these shapes.

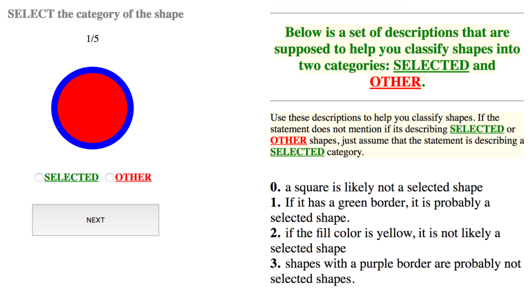
Each sentence should focus on ONE FEATURE at a time. For example, only focusing on shape, fill or border color.

Please SELECT FEATURE in the dropdown box which you are describing in your sentence.

DO NOT combine multiple features into a single sentence

1 Shape: selected shapes are almost aways a square
2 Border color: other shapes rarely have a blue border
3 Fill color: if the shape has a red fill color, it's most likely not a sele

(a) Statement generation task



Below is a set of descriptions that are supposed to help you classify shapes into two categories: SELECTED and OTHER.

Use these descriptions to help you classify shapes. If the statement does not mention if its describing SELECTED or OTHER shapes, just assume that the statement is describing a SELECTED category.

0. a square is likely not a selected shape
1. If it has a green border, it is probably a selected shape.
2. if the fill color is yellow, it is not likely a selected shape
3. shapes with a purple border are probably not selected shapes.

(b) Concept Quiz

Figure 3: Shapes data: Mechanical Turk tasks for (a) collecting concept descriptions, and (b) human evaluation from concept descriptions

Dirichlet distribution, and varying the concentration parameter α allows tuning the noise level of the generated datasets (quantified via their Bayes Optimal accuracy[4]). A dataset is then generated by sampling from these conditional distributions. We sample a total of 50 such datasets, consisting of 100 training and 100 test examples each, where each example is a shape and its assigned label.

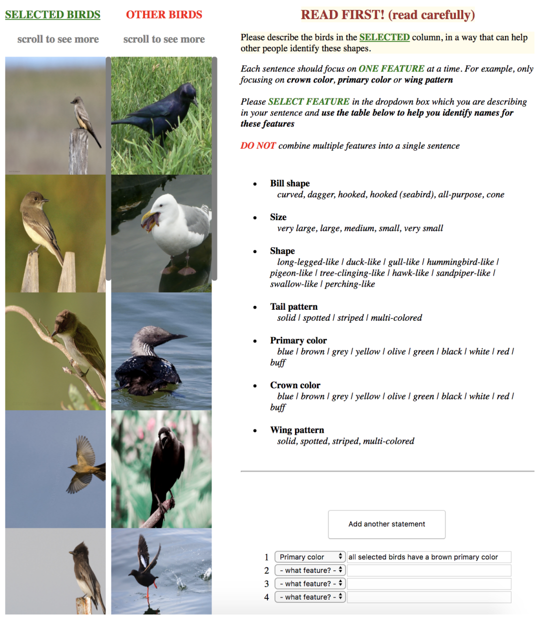
For each dataset, we then collected statements from Mechanical Turk workers that describe the concept. The task required turkers to study a sample of shapes presented on the screen for each of the two concept-labels (see Figure 3(a)). They were then asked to write a set of statements that would help others classify these shapes without seeing the data. In total, 30 workers participated in this task, generating a mean of 4.3 statements per dataset.

Email data: Srivastava et al. (2017) provide a dataset of language explanations from human users describing 7 categories of emails, as well as 1030 examples of emails belonging to those categories. While this work uses labeled examples, and focuses

[4]This is the accuracy of a theoretically optimal classifier, which knows the true distribution of the data and labels

Shapes: If a shape doesn't have a blue border, it is probably not a selected shape. Selected shapes occasionally have a yellow fill.
Emails: Emails that mention the word 'meet' in the subject are usually meeting requests Personal reminders almost always have the same recipient and sender
Birds: A specimen that has a striped crown is likely to be a selected bird. Birds in the other category rarely ever have dagger-shaped beaks

Table 3: Examples of explanations for each domain



READ FIRST! (read carefully)

Please describe the birds in the SELECTED column, in a way that can help other people identify these shapes.

*Each sentence should focus on ONE FEATURE at a time. For example, only focusing on **crown color**, **primary color** or **wing pattern***

*Please SELECT FEATURE in the dropdown box which you are describing in your sentence and **use the table below to help you identify names for these features***

DO NOT combine multiple features into a single sentence

- **Bill shape**
  curved, dagger, hooked, hooked (seabird), all-purpose, cone
- **Size**
  very large, large, medium, small, very small
- **Shape**
  long-legged-like | duck-like | gull-like | hummingbird-like | pigeon-like | tree-clinging-like | hawk-like | sandpiper-like | swallow-like | perching-like
- **Tail pattern**
  solid | spotted | striped | multi-colored
- **Primary color**
  blue | brown | grey | yellow | olive | green | black | white | red | buff
- **Crown color**
  blue | brown | grey | yellow | olive | green | black | white | red | buff
- **Wing pattern**
  solid, spotted, striped, multi-colored

1 Primary color — all selected birds have a brown primary color
2 - what feature? -
3 - what feature? -
4 - what feature? -

Figure 4: Statement generation task for Birds data

on mapping natural language explanations (∼30 explanations per email category) to compositional feature functions, we can also use statements in their data for evaluating our approach. While language quantifiers were not studied in the original work, we found about a third of the statements in this data to mention a quantifier.

Birds data: The CUB-200 dataset (Wah et al., 2011) contains images of birds annotated with observable attributes such as size, primary color, wing-patterns, etc. We selected a subset of the data consisting of 10 species of birds and 53 attributes (60 examples per species). Turkers were shown examples of birds from a species, and negative examples consisting of a mix of birds from other species, and were asked to describe the classes (similar to the Shapes data, see Figure 4). During the task, users also had access to a table enumerating groundable attributes they could refer to. In all, 60 workers participated, generating 6.1 statements on average.

Approach	Avg Accuracy	Labels	Descriptions
LNQ	0.751	no	yes
Bayes Optimal	0.831	–	–
FLGE+	0.659	no	yes
FLGE	0.598	no	yes
LR	0.737	yes	no
Random	0.524	–	–
Ablation:			
LNQ (coarse quant)	0.679	no	yes
LNQ (no quant)	0.545	no	yes
Human:			
Human teacher	0.802	yes	writes
Human learner	0.734	no	yes

Table 4: Classification performance on Shapes datasets (averaged over 50 classification tasks).

5 Experiments

Incorporating constraints from language has not been addressed before, and hence previous approaches for learning from limited data such as Mann and McCallum (2010); Chang et al. (2007b) would not directly work for this setting. Our baselines hence consist of extended versions of previous approaches that incorporate output from the parser, as well as fully supervised classifiers trained from a small number of labeled examples.

Classification performance: The top section in Table 4 summarizes performance of various classifiers on the Shape datasets, averaged over all 50 classification tasks. FLGE+ refers to a baseline

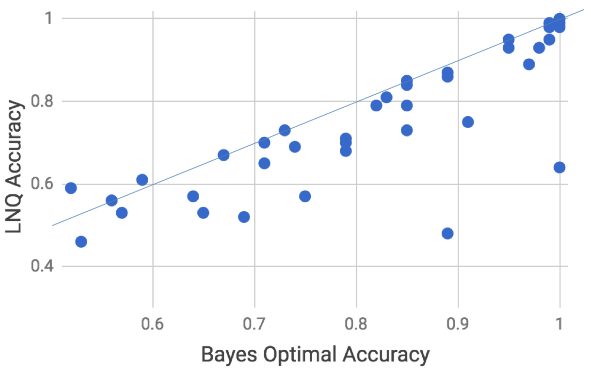


Figure 5: LNQ vs Bayes Optimal Classifier performance for Shape datasets. Each dot represents a dataset generated from a known distribution.

that uses the Feature Labeling through Generalized Expectation criterion, following the approach in Druck et al. (2008); Mann and McCallum (2010). The approach is based on labeling features are indicating specific class-labels, which corresponds to specifiying constraints of type $P(y|x)$[5]. While the original approach (Druck et al., 2008) sets this value to 0.9, we provide the method the quantitative probabilities used by LNQ. Since the original method cannot handle language descriptions, we also provide the approach the concept label y and feature x as identified by the parser. FLGE represents the version that is not provided quantifier probabilities. LR refers to a supervised logistic regression model trained on $n = 8$ randomly chosen labeled instances.[6] We note that LNQ performs substantially better than both FLGE+ and LR on average. This validates our modeling principle for learning classifiers from explanations alone, and also suggests value in our PR-based formulation, which can handle multiple constraint types. We further note that not using quantifier probabilities significantly deteriorates FLGE's performance.

Figure 5 provides a more detailed characterization of LNQ's performance. Each blue dot represents performance on a shape classification task. The horizontal axis represents the accuracy of the Bayes Optimal classifier, and the vertical represents accuracy of the LNQ approach. The blue line represents the trajectory for $x = y$, representing a perfect statistical classifier in the asymptotic case of infinite samples. We note that LNQ is effective in learning competent classifiers for all levels of hardness. Secondly, except for a small number of outliers, the approach works especially well for learning easy concepts (towards the right). From an error-analysis, we found that a majority of these errors are due to problems in parsing (e.g., missed negation, incorrect constraint type) or due to poor explanations from the teacher (bad grammar, or simply incorrect information).

Figure 6 shows results for email classification tasks. In the figure, LN* refers to the approach in Srivastava et al. (2017), which uses natural language descriptions to define compositional features for email classification, but does not incorporate

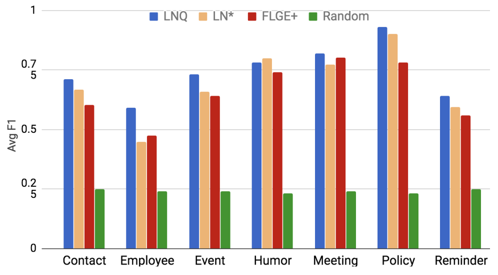


Figure 6: Classification performance (F1) on Email data. (LN* Results from Srivastava et al. (2017))

supervision from quantification. For this task, we found very few of the natural language descriptions to contain quantifiers for some of the individual email categories, making a direct comparison impractical. Thus in this case, we evaluate methods by combining supervision from descriptions in addition to 10 labeled examples (also in line with evaluation in the original paper). We note that additionally incorporating quantification (LNQ) consistently improves classification performance across email categories. On this task, LNQ improves upon FLGE+ and LN* for 6 of the 7 email categories.

Figure 7 shows classification results on the Birds data. Here, LR refers to a logistic regression model trained on n=10 examples. The trends in this case are similar, where LNQ consistently outperforms FLGE+, and is competitive with LR.

Ablating quantification: From Table 4, we further observe that the differential associative strengths of linguistic quantifiers are crucial for our method's classification performance. LNQ (no quant) refers to a variant that assigns the same probability value (average of values in Table 1), irrespective of quantifier. This yields a near random performance, which is what we'd expect if the learning is being driven by the differential strengths of quantifiers. LNQ (coarse quant) refers to a variant that rounds assigned quantifier probabilities in Table 1 to 0 or 1. (i.e., quantifiers such are *rarely* get mapped to 0, while *always* gets mapped to a probability of 1). While its performance (0.679) suggests that simple binary feedback is a substantial signal, the difference from the full model indicates value in using soft probabilities. On the other hand, in a sensitivity study, we found the performance of the approach to be robust to small changes in the probability values of quantifiers.

Comparison with human performance: For the Shapes data, we evaluated human teachers' own understanding of concepts they teach by evaluating

[5]In general, Generalized Expectation can also handle broader constraint types, similar to Posterior Regularization

[6]LNQ models are indistinct from LR w.r.t. parametrization, but trained to maximize a different objective. The choice of n here is arbitrary, but is roughly twice the number of explanations for each task in this domain

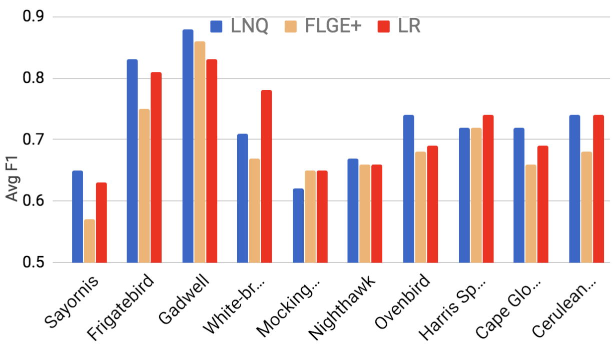


Figure 7: Classification performance on Birds data

them on a quiz based on predicting labels for examples from the test set (see Figure 3(b)). Second, we solicit additional workers that were not exposed to examples from the dataset, and present them only with the statements describing that data (created by a teacher), which is comparable supervision to what LNQ receives. We then evaluate their performance at the same task. From Table 4, we note that a human teacher's average performance is significantly worse ($p < 0.05$, Wilcoxon signed-rank test) than the Bayes Optimal classifier indicating that the teacher's own synthesis of concepts is noisy. The human learner performance is expectedly lower, but interestingly is also significantly worse than LNQ. While this might be potentially be caused by factors such as user fatigue, this might also suggest that automated methods can be better at reasoning with constraints than humans in certain scenarios. These results need to be validated through comprehensive experiments in more domains.

Empirical semantics of quantifiers: We can estimate the distributions of probability values for different quantifiers from our labeled data. For this, we aggregate sentences mentioning a quantifier, and calculate the empirical value of the (conditional) probability associated with the statement, leading to a set of probability values for each quantifier. Figure 8 shows empirical distributions of probability values for six quantifiers. We note that while a few estimates (e.g., 'rarely' and 'often') roughly align with pre-registered beliefs, others are somewhat off (e.g., 'likely' shows a much higher value), and yet others (e.g., 'sometimes') show a large spread of values to be meaningfully modeled as point values. LNQ's performance, inspite of this, shows strong stability in the approach. We don't use these empirical probabilities in experiments, (instead of pre-registered values), so as not to tune the hyperparameters to a specific dataset.

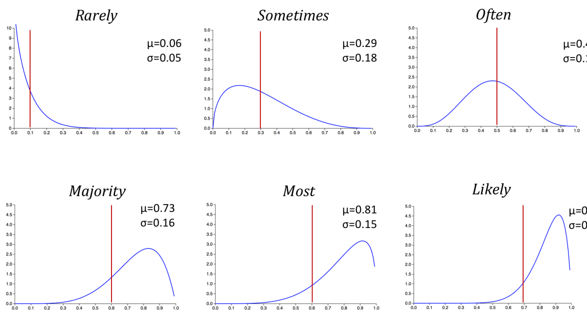


Figure 8: Empirical probability distributions for six quantifiers (Shapes data). Plots show Beta distributions with Method-of-Moment estimates. Red bars correspond to values from Table 1

Such estimates would not be available for a new task without labeled data. Further, using labeled data for estimating these probabilities, and then using the learned model for predicting labels would constitute overfitting, biasing evaluation.

6 Discussion and Future Work

Our approach is surprisingly effective in learning from free-form language. However, it does not address linguistic issues such as modifiers (e.g., *very* likely), nested quantification, etc. On the other hand, we found no instances of nested quantification in the data, suggesting that people might be primed to use simpler language when teaching. While we approximate quantifier semantics as absolute probability values, they may vary significantly based on the context, as shown by cognitive studies such as Newstead and Collis (1987). Future work can model how these parameters can be adapted in a task specific way (e.g., cases such as cancer prediction where base rates are small), and provide better models of quantifier semantics. e.g., as distributions, rather than point values.

Our approach is a step towards the idea of using language to guide learning of statistical models. This is an exciting direction, which contrasts with the predominant theme of using statistical learning methods to advance the field of NLP. We believe that language may have as much to help learning, as statistical learning has helped NLP.

Acknowledgments

This research was supported by the CMU - Yahoo! InMind project. The authors would also like to thank the anonymous reviewers for helpful comments and suggestions.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Learning with latent language. *CoRR* abs/1711.00482. http://arxiv.org/abs/1711.00482.

Elke Bach, Eloise Jelinek, Angelika Kratzer, and Barbara BH Partee. 2013. *Quantification in natural languages*, volume 54. Springer Science & Business Media.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and philosophy* 4(2):159–219.

Kedar Bellare, Gregory Druck, and Andrew McCallum. 2009. Alternating projections for learning with expectation constraints. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pages 43–50.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research* 43:661–704.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007a. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 280–287. http://www.aclweb.org/anthology/P07-1036.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007b. Guiding semi-supervision with constraint-driven learning. In *ACL*. pages 280–287.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 595–602.

Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. 2013. Write a classifier: Zero-shot learning using purely textual descriptions. In *The IEEE International Conference on Computer Vision (ICCV)*.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research* 11(Jul):2001–2049.

Siddharth Karamcheti, Edward C Williams, Dilip Arumugam, Mina Rhee, Nakul Gopalan, Lawson LS Wong, and Stefanie Tellex. 2017. A tale of two draggns: A hybrid approach for interpreting action-oriented and goal-oriented instructions. *arXiv preprint arXiv:1707.08668* .

Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*.

Howard S Kurtzman and Maryellen C MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition* 48(3):243–279.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(3):453–465.

Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *The IEEE International Conference on Computer Vision (ICCV)*.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 641–648.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 590–599.

M. Lichman. 2013. UCI machine learning repository. http://archive.ics.uci.edu/ml.

Sebastian Löbner. 1987. Quantification as a major module of natural language semantics. *Studies in discourse representation theory and the theory of generalized quantifiers* 8:53.

Gideon S Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of machine learning research* 11(Feb):955–984.

Linda M Moxey and Anthony J Sanford. 1993. Prior expectation and the interpretation of natural language quantifiers. *European Journal of Cognitive Psychology* 5(1):73–91.

Stephen E Newstead and Janet M Collis. 1987. Context and the interpretation of quantifiers of frequency. *Ergonomics* 30(10):1447–1462.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1528–1537. http://aclweb.org/anthology/D17-1161.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical report.

Ilker Yildirim, Judith Degen, Michael K Tanenhaus, and T Florian Jaeger. 2013. Linguistic variability and adaptation in quantifier meanings. In *CogSci*.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*. pages 678–687.

Sentence-State LSTM for Text Representation

Yue Zhang[1], **Qi Liu**[1] and **Linfeng Song**[2]
[1]Singapore University of Technology and Design
[2]Department of Computer Science, University of Rochester
{yue_zhang, qi_liu}@sutd.edu.sg, lsong10@cs.rochester.edu

Abstract

Bi-directional LSTMs are a powerful tool for text representation. On the other hand, they have been shown to suffer various limitations due to their sequential nature. We investigate an alternative LSTM structure for encoding text, which consists of a parallel state for each word. Recurrent steps are used to perform local and global information exchange between words simultaneously, rather than incremental reading of a sequence of words. Results on various classification and sequence labelling benchmarks show that the proposed model has strong representation power, giving highly competitive performances compared to stacked BiLSTM models with similar parameter numbers.

1 Introduction

Neural models have become the dominant approach in the NLP literature. Compared to handcrafted indicator features, neural sentence representations are less sparse, and more flexible in encoding intricate syntactic and semantic information. Among various neural networks for encoding sentences, bi-directional LSTMs (BiLSTM) (Hochreiter and Schmidhuber, 1997) have been a dominant method, giving state-of-the-art results in language modelling (Sundermeyer et al., 2012), machine translation (Bahdanau et al., 2015), syntactic parsing (Dozat and Manning, 2017) and question answering (Tan et al., 2015).

Despite their success, BiLSTMs have been shown to suffer several limitations. For example, their inherently sequential nature endows computation non-parallel within the same sentence (Vaswani et al., 2017), which can lead to a computational bottleneck, hindering their use in the industry. In addition, local ngrams, which have been shown a highly useful source of contextual information for NLP, are not explicitly modelled (Wang et al., 2016). Finally, sequential information flow leads to relatively weaker power in capturing long-range dependencies, which results in lower performance in encoding longer sentences (Koehn and Knowles, 2017).

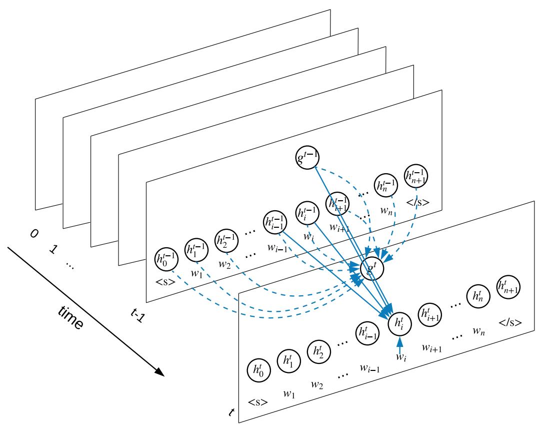


Figure 1: Sentence-State LSTM

We investigate an alternative recurrent neural network structure for addressing these issues. As shown in Figure 1, the main idea is to model the hidden states of all words simultaneously at each recurrent step, rather than one word at a time. In particular, we view the whole sentence as a single state, which consists of sub-states for individual words and an overall sentence-level state. To capture local and non-local contexts, states are updated recurrently by exchanging information between each other. Consequently, we refer to our model as sentence-state LSTM, or S-LSTM in short. Empirically, S-LSTM can give effective sentence encoding after 3 – 6 recurrent steps. In contrast, the number of recurrent steps necessary for BiLSTM scales with the size of the sentence.

At each recurrent step, information exchange is conducted between consecutive words in the sentence, and between the sentence-level state and each word. In particular, each word receives information from its predecessor and successor simultaneously. From an initial state without information exchange, each word-level state can obtain 3-gram, 5-gram and 7-gram information after 1, 2 and 3 recurrent steps, respectively. Being connected with every word, the sentence-level state vector serves to exchange non-local information with each word. In addition, it can also be used as a global sentence-level representation for classification tasks.

Results on both classification and sequence labelling show that S-LSTM gives better accuracies compared to BiLSTM using the same number of parameters, while being faster. We release our code and models at `https://github.com/leuchine/S-LSTM`, which include all baselines and the final model.

2 Related Work

LSTM (Graves and Schmidhuber, 2005) showed its early potentials in NLP when a neural machine translation system that leverages LSTM source encoding gave highly competitive results compared to the best SMT models (Bahdanau et al., 2015). LSTM encoders have since been explored for other tasks, including syntactic parsing (Dyer et al., 2015), text classification (Yang et al., 2016) and machine reading (Hermann et al., 2015). Bi-directional extensions have become a standard configuration for achieving state-of-the-art accuracies among various tasks (Wen et al., 2015; Ma and Hovy, 2016; Dozat and Manning, 2017). S-LSTMs are similar to BiLSTMs in their recurrent bi-directional message flow between words, but different in the design of state transition.

CNNs (Krizhevsky et al., 2012) also allow better parallelisation compared to LSTMs for sentence encoding (Kim, 2014), thanks to parallelism among convolution filters. On the other hand, convolution features embody only fix-sized local n-gram information, whereas sentence-level feature aggregation via pooling can lead to loss of information (Sabour et al., 2017). In contrast, S-LSTM uses a global sentence-level node to assemble and back-distribute local information in the recurrent state transition process, suffering less information loss compared to pooling.

Attention (Bahdanau et al., 2015) has recently been explored as a standalone method for sentence encoding, giving competitive results compared to Bi-LSTM encoders for neural machine translation (Vaswani et al., 2017). The attention mechanism allows parallelisation, and can play a similar role to the sentence-level state in S-LSTMs, which uses neural gates to integrate word-level information compared to hierarchical attention. S-LSTM further allows local communication between neighbouring words.

Hierarchical stacking of CNN layers (LeCun et al., 1995; Kalchbrenner et al., 2014; Papandreou et al., 2015; Dauphin et al., 2017) allows better interaction between non-local components in a sentence via incremental levels of abstraction. S-LSTM is similar to hierarchical attention and stacked CNN in this respect, incrementally refining sentence representations. However, S-LSTM models hierarchical encoding of sentence structure as a *recurrent* state transition process. In nature, our work belongs to the family of LSTM sentence representations.

S-LSTM is inspired by message passing over graphs (Murphy et al., 1999; Scarselli et al., 2009). Graph-structure neural models have been used for computer program verification (Li et al., 2016) and image object detection (Liang et al., 2016). The closest previous work in NLP includes the use of convolutional neural networks (Bastings et al., 2017; Marcheggiani and Titov, 2017) and DAG LSTMs (Peng et al., 2017) for modelling syntactic structures. Compared to our work, their motivations and network structures are highly different. In particular, the DAG LSTM of Peng et al. (2017) is a natural extension of tree LSTM (Tai et al., 2015), and is sequential rather than parallel in nature. To our knowledge, we are the first to investigate a graph RNN for encoding sentences, proposing parallel graph states for integrating word-level and sentence-level information. In this perspective, our contribution is similar to that of Kim (2014) and Bahdanau et al. (2015) in introducing a neural representation to the NLP literature.

3 Model

Given a sentence $\boldsymbol{s} = w_1, w_2, \ldots, w_n$, where w_i represents the ith word and n is the sentence length, our goal is to find a neural representation of $\boldsymbol{s}$, which consists of a hidden vector $\boldsymbol{h}_i$ for each input word w_i, and a global sentence-level hidden vector $\boldsymbol{g}$. Here $\boldsymbol{h}_i$ represents syntactic and semantic features for w_i under the sentential context, while $\boldsymbol{g}$ represents features for the whole sentence. Following previous work, we additionally add $\langle s\rangle$ and $\langle /s\rangle$ to the two ends of the sentence as w_0 and w_{n+1}, respectively.

3.1 Baseline BiLSTM

The baseline BiLSTM model consists of two LSTM components, which process the input in the forward left-to-right and the backward right-to-left directions, respectively. In each direction, the reading of input words is modelled as a recurrent process with a single hidden state. Given an initial value, the state changes its value recurrently, each time consuming an incoming word.

Take the forward LSTM component for example. Denoting the initial state as $\overrightarrow{\boldsymbol{h}}^0$, which is a model parameter, the recurrent state transition step for calculating $\overrightarrow{\boldsymbol{h}}^1, \ldots, \overrightarrow{\boldsymbol{h}}^{n+1}$ is defined as follows (Graves and Schmidhuber, 2005):

$$\begin{aligned}
\hat{\boldsymbol{i}}^t &= \sigma(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\overrightarrow{\boldsymbol{h}}^{t-1} + \boldsymbol{b}_i)\\
\hat{\boldsymbol{f}}^t &= \sigma(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\overrightarrow{\boldsymbol{h}}^{t-1} + \boldsymbol{b}_f)\\
\boldsymbol{o^t} &= \sigma(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\overrightarrow{\boldsymbol{h}}^{t-1} + \boldsymbol{b}_o)\\
\boldsymbol{u^t} &= tanh(\boldsymbol{W}_u\boldsymbol{x}_t + \boldsymbol{U}_u\overrightarrow{\boldsymbol{h}}^{t-1} + \boldsymbol{b}_u)\\
\boldsymbol{i}^t, \boldsymbol{f}^t &= softmax(\hat{\boldsymbol{i}}^t, \hat{\boldsymbol{f}}^t)\\
\boldsymbol{c}^t &= \boldsymbol{c}^{t-1} \odot \boldsymbol{f}^t + \boldsymbol{u}^t \odot \boldsymbol{i}^t\\
\overrightarrow{\boldsymbol{h}}^t &= \boldsymbol{o}^t \odot tanh(\boldsymbol{c}^t)
\end{aligned} \tag{1}$$

where $\boldsymbol{x}_t$ denotes the word representation of w_t; $\boldsymbol{i}^t$, $\boldsymbol{o}^t$, $\boldsymbol{f}^t$ and $\boldsymbol{u}^t$ represent the values of an input gate, an output gate, a forget gate and an actual input at time step t, respectively, which controls the information flow for a recurrent cell $\overrightarrow{\boldsymbol{c}}^t$ and the state vector $\overrightarrow{\boldsymbol{h}}^t$; $\boldsymbol{W}_x$, $\boldsymbol{U}_x$ and $\boldsymbol{b}_x$ ($x \in \{i, o, f, u\}$) are model parameters. σ is the sigmoid function.

The backward LSTM component follows the same recurrent state transition process as described in Eq 1. Starting from an initial state $\boldsymbol{h}^{n+1}$, which is a model parameter, it reads the input $\boldsymbol{x}_n$, $\boldsymbol{x}_{n-1}, \ldots, \boldsymbol{x}_0$, changing its value to $\overleftarrow{\boldsymbol{h}}^n$, $\overleftarrow{\boldsymbol{h}}^{n-1}$, $\ldots$, $\overleftarrow{\boldsymbol{h}}^0$, respectively. A separate set of parameters $\hat{\boldsymbol{W}}_x$, $\hat{\boldsymbol{U}}_x$ and $\hat{\boldsymbol{b}}_x$ ($x \in \{i, o, f, u\}$) are used for the backward component.

The BiLSTM model uses the concatenated value of $\overrightarrow{\boldsymbol{h}}^t$ and $\overleftarrow{\boldsymbol{h}}^t$ as the hidden vector for w_t:

$$\boldsymbol{h}^t = [\overrightarrow{\boldsymbol{h}}^t; \overleftarrow{\boldsymbol{h}}^t]$$

A single hidden vector representation $\boldsymbol{g}$ of the whole input sentence can be obtained using the final state values of the two LSTM components:

$$\boldsymbol{g} = [\overrightarrow{\boldsymbol{h}}^{n+1}; \overleftarrow{\boldsymbol{h}}^0]$$

Stacked BiLSTM Multiple layers of BiLTMs can be stacked for increased representation power, where the hidden vectors of a lower layer are used as inputs for an upper layer. Different model parameters are used in each stacked BiLSTM layer.

3.2 Sentence-State LSTM

Formally, an S-LSTM state at time step t can be denoted by:

$$\boldsymbol{H}^t = \langle \boldsymbol{h}_0^t, \boldsymbol{h}_1^t, \ldots, \boldsymbol{h}_{n+1}^t, \boldsymbol{g}^t\rangle,$$

which consists of a sub state $\boldsymbol{h}_i^t$ for each word w_i and a sentence-level sub state $\boldsymbol{g}^t$.

S-LSTM uses a recurrent state transition process to model information exchange between sub states, which enriches state representations incrementally. For the initial state $\boldsymbol{H}^0$, we set $\boldsymbol{h}_i^0 = \boldsymbol{g}^0 = \boldsymbol{h}^0$, where $\boldsymbol{h}^0$ is a parameter. The state transition from $\boldsymbol{H}^{t-1}$ to $\boldsymbol{H}^t$ consists of sub state transitions from $\boldsymbol{h}_i^{t-1}$ to $\boldsymbol{h}_i^t$ and from $\boldsymbol{g}^{t-1}$ to $\boldsymbol{g}^t$. We take an LSTM structure similar to the baseline BiLSTM for modelling state transition, using a recurrent cell $\boldsymbol{c}_i^t$ for each w_i and a cell $\boldsymbol{c}_g^t$ for $\boldsymbol{g}$.

As shown in Figure 1, the value of each $\boldsymbol{h}_i^t$ is computed based on the values of $\boldsymbol{x}_i$, $\boldsymbol{h}_{i-1}^{t-1}$, $\boldsymbol{h}_i^{t-1}$, $\boldsymbol{h}_{i+1}^{t-1}$ and $\boldsymbol{g}^{t-1}$, together with their corresponding cell values:

$$\begin{aligned}
\boldsymbol{\xi}_i^t &= [\boldsymbol{h}_{i-1}^{t-1}, \boldsymbol{h}_i^{t-1}, \boldsymbol{h}_{i+1}^{t-1}]\\
\hat{\boldsymbol{i}}_i^t &= \sigma(\boldsymbol{W}_i\boldsymbol{\xi}_i^t + \boldsymbol{U}_i\boldsymbol{x}_i + \boldsymbol{V}_i\boldsymbol{g}^{t-1} + \boldsymbol{b}_i)\\
\hat{\boldsymbol{l}}_i^t &= \sigma(\boldsymbol{W}_l\boldsymbol{\xi}_i^t + \boldsymbol{U}_l\boldsymbol{x}_i + \boldsymbol{V}_l\boldsymbol{g}^{t-1} + \boldsymbol{b}_l)\\
\hat{\boldsymbol{r}}_i^t &= \sigma(\boldsymbol{W}_r\boldsymbol{\xi}_i^t + \boldsymbol{U}_r\boldsymbol{x}_i + \boldsymbol{V}_r\boldsymbol{g}^{t-1} + \boldsymbol{b}_r)\\
\hat{\boldsymbol{f}}_i^t &= \sigma(\boldsymbol{W}_f\boldsymbol{\xi}_i^t + \boldsymbol{U}_f\boldsymbol{x}_i + \boldsymbol{V}_f\boldsymbol{g}^{t-1} + \boldsymbol{b}_f)\\
\hat{\boldsymbol{s}}_i^t &= \sigma(\boldsymbol{W}_s\boldsymbol{\xi}_i^t + \boldsymbol{U}_s\boldsymbol{x}_i + \boldsymbol{V}_s\boldsymbol{g}^{t-1} + \boldsymbol{b}_s)\\
\boldsymbol{o}_i^t &= \sigma(\boldsymbol{W}_o\boldsymbol{\xi}_i^t + \boldsymbol{U}_o\boldsymbol{x}_i + \boldsymbol{V}_o\boldsymbol{g}^{t-1} + \boldsymbol{b}_o)\\
\boldsymbol{u}_i^t &= tanh(\boldsymbol{W}_u\boldsymbol{\xi}_i^t + \boldsymbol{U}_u\boldsymbol{x}_i + \boldsymbol{V}_u\boldsymbol{g}^{t-1} + \boldsymbol{b}_u)\\
\boldsymbol{i}_i^t, \boldsymbol{l}_i^t, \boldsymbol{r}_i^t, \boldsymbol{f}_i^t, \boldsymbol{s}_i^t &= softmax(\hat{\boldsymbol{i}}_i^t, \hat{\boldsymbol{l}}_i^t, \hat{\boldsymbol{r}}_i^t, \hat{\boldsymbol{f}}_i^t, \hat{\boldsymbol{s}}_i^t)\\
\boldsymbol{c}_i^t &= \boldsymbol{l}_i^t \odot \boldsymbol{c}_{i-1}^{t-1} + \boldsymbol{f}_i^t \odot \boldsymbol{c}_i^{t-1} + \boldsymbol{r}_i^t \odot \boldsymbol{c}_{i+1}^{t-1}\\
&\quad + \boldsymbol{s}_i^t \odot \boldsymbol{c}_g^{t-1} + \boldsymbol{i}_i^t \odot \boldsymbol{u}_i^t\\
\boldsymbol{h}_i^t &= \boldsymbol{o}_t^i \odot tanh(\boldsymbol{c}_i^t)
\end{aligned} \tag{2}$$

where $\boldsymbol{\xi}_i^t$ is the concatenation of hidden vectors of a context window, and $\boldsymbol{l}_i^t$, $\boldsymbol{r}_i^t$, $\boldsymbol{f}_i^t$, $\boldsymbol{s}_i^t$ and $\boldsymbol{i}_i^t$ are gates that control information flow from $\boldsymbol{\xi}_i^t$ and $\boldsymbol{x}_i$ to $\boldsymbol{c}_i^t$. In particular, $\boldsymbol{i}_i^t$ controls information from the input $\boldsymbol{x}_i$; $\boldsymbol{l}_i^t$, $\boldsymbol{r}_i^t$, $\boldsymbol{f}_i^t$ and $\boldsymbol{s}_i^t$ control information from the left context cell $\boldsymbol{c}_{i-1}^{t-1}$, the right context cell $\boldsymbol{c}_{i+1}^{t-1}$, $\boldsymbol{c}_i^{t-1}$ and the sentence context cell $\boldsymbol{c}_g^{t-1}$, respectively. The values of $\boldsymbol{i}_i^t$, $\boldsymbol{l}_i^t$, $\boldsymbol{r}_i^t$, $\boldsymbol{f}_i^t$ and $\boldsymbol{s}_i^t$ are normalised such that they sum to $\mathbf{1}$. $\boldsymbol{o}_i^t$ is an output gate from the cell state $\boldsymbol{c}_i^t$ to the hidden state $\boldsymbol{h}_i^t$. $\boldsymbol{W}_x$, $\boldsymbol{U}_x$, $\boldsymbol{V}_x$ and $\boldsymbol{b}_x$ ($x \in \{i, o, l, r, f, s, u\}$) are model parameters. σ is the sigmoid function.

The value of $\boldsymbol{g}^t$ is computed based on the values of $\boldsymbol{h}_i^{t-1}$ for all $i \in [0..n+1]$:

$$
\begin{aligned}
\bar{\boldsymbol{h}} &= avg(\boldsymbol{h}_0^{t-1}, \boldsymbol{h}_1^{t-1}, \ldots, \boldsymbol{h}_{n+1}^{t-1}) \\
\hat{\boldsymbol{f}}_g^t &= \sigma(\boldsymbol{W}_g \boldsymbol{g}^{t-1} + \boldsymbol{U}_g \bar{\boldsymbol{h}} + \boldsymbol{b}_g) \\
\hat{\boldsymbol{f}}_i^t &= \sigma(\boldsymbol{W}_f \boldsymbol{g}^{t-1} + \boldsymbol{U}_f \boldsymbol{h}_i^{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{o}^t &= \sigma(\boldsymbol{W}_o \boldsymbol{g}^{t-1} + \boldsymbol{U}_o \bar{\boldsymbol{h}} + \boldsymbol{b}_o) \\
\boldsymbol{f}_0^t, \ldots, \boldsymbol{f}_{n+1}^t, \boldsymbol{f}_g^t &= softmax(\hat{\boldsymbol{f}}_0^t, \ldots, \hat{\boldsymbol{f}}_{n+1}^t, \hat{\boldsymbol{f}}_g^t) \\
\boldsymbol{c}_g^t &= \boldsymbol{f}_g^t \odot \boldsymbol{c}_g^{t-1} + \sum_i \boldsymbol{f}_i^t \odot \boldsymbol{c}_i^{t-1} \\
\boldsymbol{g}^t &= \boldsymbol{o}^t \odot tanh(\boldsymbol{c}_g^t)
\end{aligned}
\tag{3}
$$

where $\boldsymbol{f}_0^t, \ldots, \boldsymbol{f}_{n+1}^t$ and $\boldsymbol{f}_g^t$ are gates controlling information from $\boldsymbol{c}_0^{t-1}, \ldots, \boldsymbol{c}_{n+1}^{t-1}$ and $\boldsymbol{c}_g^{t-1}$, respectively, which are normalised. $\boldsymbol{o}^t$ is an output gate from the recurrent cell $\boldsymbol{c}_g^t$ to $\boldsymbol{g}^t$. $\boldsymbol{W}_x$, $\boldsymbol{U}_x$ and $\boldsymbol{b}_x$ ($x \in \{g, f, o\}$) are model parameters.

Contrast with BiLSTM The difference between S-LSTM and BiLSTM can be understood with respect to their recurrent states. While BiLSTM uses only one state in each direction to represent the subsequence from the beginning to a certain word, S-LSTM uses a structural state to represent the full sentence, which consists of a sentence-level sub state and $n + 2$ word-level sub states, simultaneously. Different from BiLSTMs, for which $\boldsymbol{h}^t$ at different time steps are used to represent $w_0, \ldots, w_{n+1}$, respectively, the word-level states $\boldsymbol{h}_i^t$ and sentence-level state $\boldsymbol{g}^t$ of S-LSTMs directly correspond to the goal outputs $\boldsymbol{h}_i$ and $\boldsymbol{g}$, as introduced in the beginning of this section. As t increases from 0, $\boldsymbol{h}_i^t$ and $\boldsymbol{g}^t$ are enriched with increasingly deeper context information.

From the perspective of information flow, BiLSTM passes information from one end of the sentence to the other. As a result, the number of time steps scales with the size of the input. In contrast, S-LSTM allows bi-directional information flow at each word simultaneously, and additionally between the sentence-level state and every word-level state. At each step, each $\boldsymbol{h}_i$ captures an increasing larger ngram context, while additionally communicating globally to all other $\boldsymbol{h}_j$ via $\boldsymbol{g}$. The optimal number of recurrent steps is decided by the end-task performance, and does not necessarily scale with the sentence size. As a result, S-LSTM can potentially be both more efficient and more accurate compared with BiLSTMs.

Increasing window size. By default S-LSTM exchanges information only between neighbouring words, which can be seen as adopting a 1-word window on each side. The window size can be extended to 2, 3 or more words in order to allow more communication in a state transition, expediting information exchange. To this end, we modify Eq 2, integrating additional context words to $\boldsymbol{\xi}_i^t$, with extended gates and cells. For example, with a window size of 2, $\boldsymbol{\xi}_i^t = [\boldsymbol{h}_{i-2}^{t-1}, \boldsymbol{h}_{i-1}^{t-1}, \boldsymbol{h}_i^{t-1}, \boldsymbol{h}_{i+1}^{t-1}, \boldsymbol{h}_{i+2}^{t-1}]$. We study the effectiveness of window size in our experiments.

Additional sentence-level nodes. By default S-LSTM uses one sentence-level node. One way of enriching the parameter space is to add more sentence-level nodes, each communicating with word-level nodes in the same way as described by Eq 3. In addition, different sentence-level nodes can communicate with each other during state transition. When one sentence-level node is used for classification outputs, the other sentence-level node can serve as hidden memory units, or latent features. We study the effectiveness of multiple sentence-level nodes empirically.

3.3 Task settings

We consider two task settings, namely classification and sequence labelling. For *classification*, $\boldsymbol{g}$ is fed to a *softmax* classification layer:

$$\boldsymbol{y} = softmax(\boldsymbol{W}_c \boldsymbol{g} + \boldsymbol{b}_c)$$

where $\boldsymbol{y}$ is the probability distribution of output class labels and $\boldsymbol{W}_c$ and $\boldsymbol{b}_c$ are model parameters. For *sequence labelling*, each $\boldsymbol{h}_i$ can be used as feature representation for a corresponding word w_i.

External attention It has been shown that summation of hidden states using attention (Bahdanau et al., 2015; Yang et al., 2016) give better accuracies compared to using the end states of BiLSTMs. We study the influence of attention on both S-LSTM and BiLSTM for *classification*. In particular, additive attention (Bahdanau

Dataset		Training		Development		Test	
		#sent	#words	#sent	#words	#sent	#words
Movie review (Pang and Lee, 2008)		8527	201137	1066	25026	1066	25260
Text Classification (Liu et al., 2017)	Books	1400	297K	200	59K	400	68K
	Electronics	1398	924K	200	184K	400	224K
	DVD	1400	1,587K	200	317K	400	404K
	Kitchen	1400	769K	200	153K	400	195K
	Apparel	1400	525K	200	105K	400	128K
	Camera	1397	1,084K	200	216K	400	260K
	Health	1400	742K	200	148K	400	175K
	Music	1400	1,176K	200	235K	400	276K
	Toys	1400	792K	200	158K	400	196K
	Video	1400	1,311K	200	262K	400	342K
	Baby	1300	855K	200	171K	400	221K
	Magazines	1370	1,033K	200	206K	400	264K
	Software	1315	1,143K	200	228K	400	271K
	Sports	1400	833K	200	183K	400	218K
	IMDB	1400	2,205K	200	507K	400	475K
	MR	1400	196K	200	41K	400	48K
POS tagging (Marcus et al., 1993)		39831	950011	1699	40068	2415	56671
NER (Sang et al., 2003)		14987	204567	3466	51578	3684	46666

Table 1: Dataset statistics

et al., 2015) is applied to the hidden states of input words for both BiLSTMs and S-LSTMs calculating a weighted sum

$$\boldsymbol{g} = \sum_t \alpha_t \boldsymbol{h}_t$$

where

$$\alpha_t = \frac{\exp \boldsymbol{u}^T \boldsymbol{\epsilon}_t}{\sum_i \exp \boldsymbol{u}^T \boldsymbol{\epsilon}_i}$$

$$\boldsymbol{\epsilon}_t = tanh(\boldsymbol{W}_\alpha \boldsymbol{h}_t + \boldsymbol{b}_\alpha)$$

Here $\boldsymbol{W}_\alpha$, $\boldsymbol{u}$ and $\boldsymbol{b}_\alpha$ are model parameters.

External CRF For *sequential labelling*, we use a CRF layer on top of the hidden vectors $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n$ for calculating the conditional probabilities of label sequences (Huang et al., 2015; Ma and Hovy, 2016):

$$P(\boldsymbol{Y}_1^n | \boldsymbol{h}, \boldsymbol{W}_s, \boldsymbol{b}_s) = \frac{\prod_{i=1}^n \psi_i(y_{i-1}, y_i, \boldsymbol{h})}{\sum_{\boldsymbol{Y}_1^{n\prime}} \prod_{i=1}^n \psi_i(y'_{i-1}, y'_i, \boldsymbol{h})}$$

$$\psi_i(y_{i-1}, y_i, \boldsymbol{h}) = exp(\boldsymbol{W}_s^{y_{i-1}, y_i} h_i + b_s^{y_{i-1}, y_i})$$

where $\boldsymbol{W}_s^{y_{i-1}, y_i}$ and $b_s^{y_{i-1}, y_i}$ are parameters specific to two consecutive labels y_{i-1} and y_i.

For training, standard log-likelihood loss is used with L_2 regularization given a set of gold-standard instances.

4 Experiments

We empirically compare S-LSTMs and BiLSTMs on different classification and sequence labelling tasks. All experiments are conducted using a GeForce GTX 1080 GPU with 8GB memory.

Model	Time (s)	Acc	# Param
+0 dummy node	56	81.76	7,216K
+1 dummy node	65	82.64	8,768K
+2 dummy node	76	82.24	10,321K
Hidden size 100	42	81.75	4,891K
Hidden size 200	54	82.04	6,002K
Hidden size 300	65	82.64	8,768K
Hidden size 600	175	81.84	17,648K
Hidden size 900	235	81.66	33,942K
Without ⟨s⟩, ⟨/s⟩	63	82.36	8,768K
With ⟨s⟩, ⟨/s⟩	65	82.64	8,768K

Table 2: Movie review DEV results of S-LSTM

4.1 Experimental Settings

Datasets. We choose the movie review dataset of Pang and Lee (2008), and additionally the 16 datasets of Liu et al. (2017) for classification evaluation. We randomly split the movie review dataset into training (80%), development (10%) and test (10%) sections, and the original split of Liu et al. (2017) for the 16 classification datasets.

For sequence labelling, we choose the Penn Treebank (Marcus et al., 1993) POS tagging task and the CoNLL (Sang et al., 2003) NER task as our benchmarks. For POS tagging, we follow the standard split (Manning, 2011), using sections 0 – 18 for training, 19 – 21 for development and 22 – 24 for test. For NER, we follow the standard split, and use the BIOES tagging scheme (Ratinov and Roth, 2009). Statistics of the four datasets are shown in Table 1.

Hyperparameters. We initialise word embeddings using GloVe (Pennington et al., 2014) 300 dimensional embeddings.[1] Embeddings are fine-tuned during model training for all tasks. Dropout (Srivastava et al., 2014) is applied to embedding hidden states, with a rate of 0.5. All models are optimised using the Adam optimizer (Kingma and Ba, 2014), with an initial learning rate of 0.001 and a decay rate of 0.97. Gradients are clipped at 3 and a batch size of 10 is adopted. Sentences with similar lengths are batched together. The L2 regularization parameter is set to 0.001.

4.2 Development Experiments

We use the movie review development data to investigate different configurations of S-LSTMs and BiLSTMs. For S-LSTMs, the default configuration uses $\langle s \rangle$ and $\langle /s \rangle$ words for augmenting words

[1]https://nlp.stanford.edu/projects/glove/

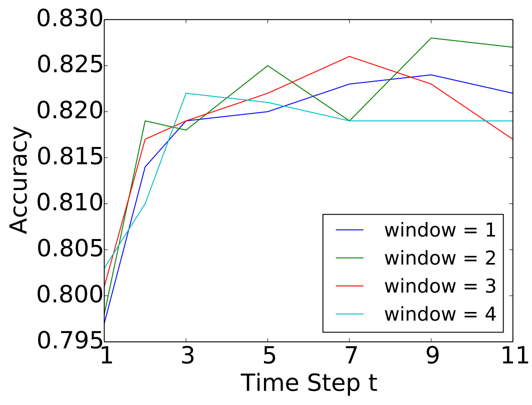


Figure 2: Accuracies with various window sizes and time steps on movie review development set

of a sentence. A hidden layer size of 300 and one sentence-level node are used.

Hyperparameters: Table 2 shows the development results of various S-LSTM settings, where Time refers to training time per epoch. Without the sentence-level node, the accuracy of S-LSTM drops to 81.76%, demonstrating the necessity of global information exchange. Adding one additional sentence-level node as described in Section 3.2 does not lead to accuracy improvements, although the number of parameters and decoding time increase accordingly. As a result, we use only 1 sentence-level node for the remaining experiments. The accuracies of S-LSTM increases as the hidden layer size for each node increases from 100 to 300, but does not further increase when the size increases beyond 300. We fix the hidden size to 300 accordingly. Without using $\langle s \rangle$ and $\langle /s \rangle$, the performance of S-LSTM drops from 82.64% to 82.36%, showing the effectiveness of having these additional nodes. Hyperparameters for BiLSTM models are also set according to the development data, which we omit here.

State transition. In Table 2, the number of recurrent state transition steps of S-LSTM is decided according to the best development performance. Figure 2 draws the development accuracies of S-LSTMs with various window sizes against the number of recurrent steps. As can be seen from the figure, when the number of time steps increases from 1 to 11, the accuracies generally increase, before reaching a maximum value. This shows the effectiveness of recurrent information exchange in S-LSTM state transition.

On the other hand, no significant differences are observed on the peak accuracies given by different window sizes, although a larger window size (e.g. 4) generally results in faster plateauing. This can be be explained by the intuition that information exchange between distant nodes can be achieved using more recurrent steps under a smaller window size, as can be achieved using fewer steps under a larger window size. Considering efficiency, we choose a window size of 1 for the remaining experiments, setting the number of recurrent steps to 9 according to Figure 2.

Model	**Time (s)**	**Acc**	**# Param**
LSTM	67	80.72	5,977K
BiLSTM	106	81.73	7,059K
2 stacked BiLSTM	207	81.97	9,221K
3 stacked BiLSTM	310	81.53	11,383K
4 stacked BiLSTM	411	81.37	13,546K
S-LSTM	65	82.64*	8,768K
CNN	34	80.35	5,637K
2 stacked CNN	40	80.97	5,717K
3 stacked CNN	47	81.46	5,808K
4 stacked CNN	51	81.39	5,855K
Transformer (N=6)	138	81.03	7,234K
Transformer (N=8)	174	81.86	7,615K
Transformer (N=10)	214	81.63	8,004K
BiLSTM+Attention	126	82.37	7,419K
S-LSTM+Attention	87	83.07*	8,858K

Table 3: Movie review development results

S-LSTM vs BiLSTM: As shown in Table 3, BiLSTM gives significantly better accuracies compared to uni-directional LSTM[2], with the training time per epoch growing from 67 seconds to 106 seconds. Stacking 2 layers of BiLSTM gives further improvements to development results, with a larger time of 207 seconds. 3 layers of stacked BiLSTM does not further improve the results. In contrast, S-LSTM gives a development result of 82.64%, which is significantly better compared to 2-layer stacked BiLSTM, with a smaller number of model parameters and a shorter time of 65 seconds.

We additionally make comparisons with stacked CNNs and hierarchical attention (Vaswani et al., 2017), shown in Table 3 (the CNN and Transformer rows), where N indicates the number of attention layers. CNN is the most efficient among all models compared, with the smallest model size. On the other hand, a 3-layer stacked CNN gives an accuracy of 81.46%, which is also

[2] $p < 0.01$ using t-test. For the remaining of this paper, we use the same measure for statistical significance.

Model	Accuracy	Train (s)	Test (s)
Socher et al. (2011)	77.70	–	–
Socher et al. (2012)	79.00	–	–
Kim (2014)	81.50	–	–
Qian et al. (2016)	81.50	–	–
BiLSTM	81.61	51	1.62
2 stacked BiLSTM	81.94	98	3.18
3 stacked BiLSTM	81.71	137	4.67
3 stacked CNN	81.59	31	1.04
Transformer (N=8)	81.97	89	2.75
S-LSTM	**82.45***	41	1.53

Table 4: Test set results on movie review dataset (* denotes significance in all tables).

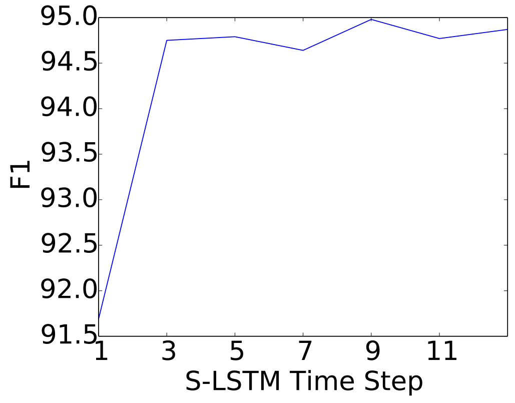
the lowest compared with BiLSTM, hierarchical attention and S-LSTM. The best performance of hierarchical attention is between single-layer and two-layer BiLSTMs in terms of both accuracy and efficiency. S-LSTM gives significantly better accuracies compared with both CNN and hierarchical attention.

Influence of external attention mechanism. Table 3 additionally shows the results of BiLSTM and S-LSTM when external attention is used as described in Section 3.3. Attention leads to improved accuracies for both BiLSTM and S-LSTM in classification, with S-LSTM still outperforming BiLSTM significantly. The result suggests that external techniques such as attention can play orthogonal roles compared with internal recurrent structures, therefore benefiting both BiLSTMs and S-LSTMs. Similar observations are found using external CRF layers for sequence labelling.

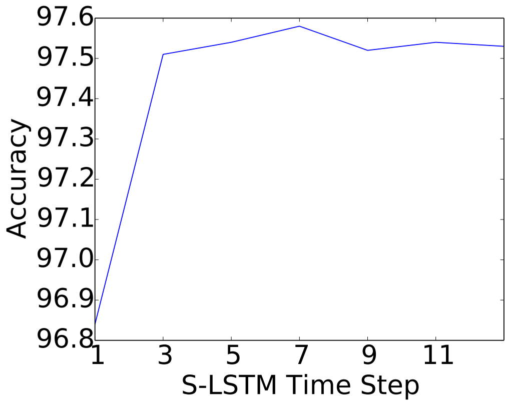
4.3 Final Results for Classification

The final results on the movie review and rich text classification datasets are shown in Tables 4 and 5, respectively. In addition to training time per epoch, test times are additionally reported. We use the best settings on the movie review development dataset for both S-LSTMs and BiLSTMs. The step number for S-LSTMs is set to 9.

As shown in Table 4, the final results on the movie review dataset are consistent with the development results, where S-LSTM outperforms BiLSTM significantly, with a faster speed. Observations on CNN and hierarchical attention are consistent with the development results. S-LSTM also gives highly competitive results when compared with existing methods in the literature.



(a) CoNLL03



(b) WSJ

Figure 3: Sequence labelling development results.

As shown in Table 5, among the 16 datasets of Liu et al. (2017), S-LSTM gives the best results on 12, compared with BiLSTM and 2 layered BiLSTM models. The average accuracy of S-LSTM is 85.6%, significantly higher compared with 84.9% by 2-layer stacked BiLSTM. 3-layer stacked BiLSTM gives an average accuracy of 84.57%, which is lower compared to a 2-layer stacked BiLSTM, with a training time per epoch of 423.6 seconds. The relative speed advantage of S-LSTM over BiLSTM is larger on the 16 datasets as compared to the movie review test test. This is because the average length of inputs is larger on the 16 datasets (see Section 4.5).

4.4 Final Results for Sequence Labelling

Bi-directional RNN-CRF structures, and in particular BiLSTM-CRFs, have achieved the state of the art in the literature for sequence labelling tasks, including POS-tagging and NER. We compare S-LSTM-CRF with BiLSTM-CRF for sequence labelling, using the same settings as decided on the movie review development experiments for both BiLSTMs and S-LSTMs. For the latter, we decide

Dataset	SLSTM	Time (s)	BiLSTM	Time (s)	2 BiLSTM	Time (s)
Camera	**90.02***	50 (2.85)	87.05	115 (8.37)	88.07	221 (16.1)
Video	**86.75***	55 (3.95)	84.73	140 (12.59)	85.23	268 (25.86)
Health	**86.5**	37 (2.17)	85.52	118 (6.38)	85.89	227 (11.16)
Music	**82.04***	52 (3.44)	78.74	185 (12.27)	80.45	268 (23.46)
Kitchen	**84.54***	40 (2.50)	82.22	118 (10.18)	83.77	225 (19.77)
DVD	**85.52***	63 (5.29)	83.71	166 (15.42)	84.77	217 (28.31)
Toys	85.25	39 (2.42)	85.72	119 (7.58)	**85.82**	231 (14.83)
Baby	**86.25***	40 (2.63)	84.51	125 (8.50)	85.45	238 (17.73)
Books	**83.44***	64 (3.64)	82.12	240 (13.59)	82.77	458 (28.82)
IMDB	**87.15***	67 (3.69)	86.02	248 (13.33)	86.55	486 (26.22)
MR	**76.2**	27 (1.25)	75.73	39 (2.27)	75.98	72 (4.63)
Appeal	85.75	35 (2.83)	86.05	119 (11.98)	**86.35***	229 (22.76)
Magazines	**93.75***	51 (2.93)	92.52	214 (11.06)	92.89	417 (22.77)
Electronics	**83.25***	47 (2.55)	82.51	195 (10.14)	82.33	356 (19.77)
Sports	**85.75***	44 (2.64)	84.04	172 (8.64)	84.78	328 (16.34)
Software	**87.75***	54 (2.98)	86.73	245 (12.38)	86.97	459 (24.68)
Average	**85.38***	47.30 (2.98)	84.01	153.48 (10.29)	84.64	282.24 (20.2)

Table 5: Results on the 16 datasets of Liu et al. (2017). Time format: train (test)

Model	Accuracy	Train (s)	Test (s)
Manning (2011)	97.28	–	–
Collobert et al. (2011)	97.29	–	–
Sun (2014)	97.36	–	–
Søgaard (2011)	97.50	–	–
Huang et al. (2015)	**97.55**	–	–
Ma and Hovy (2016)	**97.55**	–	–
Yang et al. (2017)	**97.55**	–	–
BiLSTM	97.35	254	22.50
2 stacked BiLSTM	97.41	501	43.99
3 stacked BiLSTM	97.40	746	64.96
S-LSTM	**97.55**	237	22.16

Table 6: Results on PTB (POS tagging)

Model	F1	Train (s)	Test (s)
Collobert et al. (2011)	89.59	–	–
Passos et al. (2014)	90.90	–	–
Luo et al. (2015)	91.20	–	–
Huang et al. (2015)	90.10	–	–
Lample et al. (2016)	90.94	–	–
Ma and Hovy (2016)	91.21	–	–
Yang et al. (2017)	91.26	–	–
Rei (2017)	86.26	–	–
Peters et al. (2017)	**91.93**	–	–
BiLSTM	90.96	82	9.89
2 stacked BiLSTM	91.02	159	18.88
3 stacked BiLSTM	91.06	235	30.97
S-LSTM	**91.57***	79	9.78

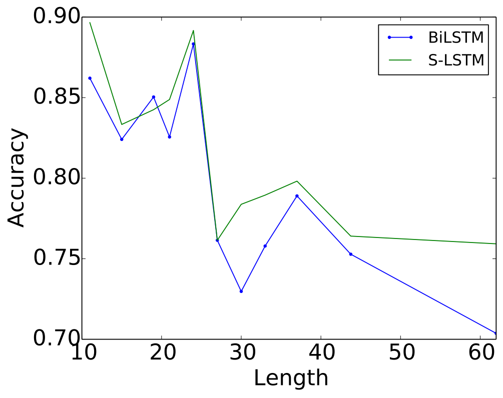
Table 7: Results on CoNLL03 (NER)

the number of recurrent steps on the respective development sets for sequence labelling. The POS accuracies and NER F1-scores against the number of recurrent steps are shown in Figure 3 (a) and (b), respectively. For POS tagging, the best step number is set to 7, with a development accuracy of 97.58%. For NER, the step number is set to 9, with a development F1-score of 94.98%.

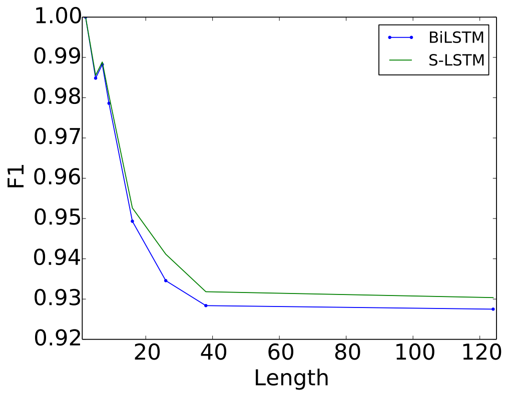
As can be seen in Table 6, S-LSTM gives significantly better results compared with BiLSTM on the WSJ dataset. It also gives competitive accuracies as compared with existing methods in the literature. Stacking two layers of BiLSTMs leads to improved results compared to one-layer BiLSTM, but the accuracy does not further improve with three layers of stacked LSTMs.

For NER (Table 7), S-LSTM gives an F1-score of 91.57% on the CoNLL test set, which is significantly better compared with BiLSTMs. Stacking more layers of BiLSTMs leads to slightly better F1-scores compared with a single-layer BiLSTM. Our BiLSTM results are comparable to the results reported by Ma and Hovy (2016) and Lample et al. (2016), who also use bidirectional RNN-CRF structures. In contrast, S-LSTM gives the best reported results under the same settings.

In the second section of Table 7, Yang et al. (2017) use cross-domain data, obtaining an F-score of 91.26%; Rei (2017) perform multi-task



(a) Movie review



(b) CoNLL03

Figure 4: Accuracies against sentence length.

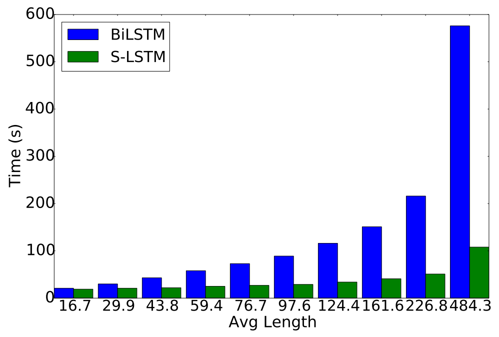


Figure 5: Time against sentence length.

learning using additional language model objectives, obtaining an F-score of 86.26%; Peters et al. (2017) leverage character-level language models, obtaining an F-score of 91.93%, which is the current best result on the dataset. All the three models are based on BiLSTM-CRF. On the other hand, these semi-supervised learning techniques are orthogonal to our work, and can potentially be used for S-LSTM also.

4.5 Analysis

Figure 4 (a) and (b) show the accuracies against the sentence length on the movie review and CoNLL datasets, respectively, where test samples are binned in batches of 80. We find that the performances of both S-LSTM and BiLSTM decrease as the sentence length increases. On the other hand, S-LSTM demonstrates relatively better robustness compared to BiLSTMs. This confirms our intuition that a sentence-level node can facilitate better non-local communication.

Figure 5 shows the training time per epoch of S-LSTM and BiLSTM on sentences with different lengths on the 16 classification datasets. To make these comparisons, we mix all training instances, order them by the size, and put them into 10 equal groups, the medium sentence lengths of which are shown. As can be seen from the figure, the speed advantage of S-LSTM is larger when the size of the input text increases, thanks to a fixed number of recurrent steps.

Similar to hierarchical attention (Vaswani et al., 2017), there is a relative disadvantage of S-LSTM in comparison with BiLSTM, which is that the memory consumption is relatively larger. For example, over the movie review development set, the actual GPU memory consumption by S-LSTM, BiLSTM, 2-layer stacked BiLSTM and 4-layer stacked BiLSTM are 252M, 89M, 146M and 253M, respectively. This is due to the fact that computation is performed in parallel by S-LSTM and hierarchical attention.

5 Conclusion

We have investigated S-LSTM, a recurrent neural network for encoding sentences, which offers richer contextual information exchange with more parallelism compared to BiLSTMs. Results on a range of classification and sequence labelling tasks show that S-LSTM outperforms BiLSTMs using the same number of parameters, demonstrating that S-LSTM can be a useful addition to the neural toolbox for encoding sentences.

The structural nature in S-LSTM states allows straightforward extension to tree structures, resulting in highly parallelisable tree LSTMs. We leave such investigation to future work. Next directions also include the investigation of S-LSTM to more NLP tasks, such as machine translation.

Acknowledge

We thank the anonymous reviewers for their constructive and thoughtful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP 2017*. Copenhagen, Denmark, pages 1957–1967.

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

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *ICML*. pages 933–941.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR 2017*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL 2015*. Beijing, China, pages 334–343.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 602–610.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*. pages 1693–1701.

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

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL 2014*. Baltimore, Maryland, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*. Doha, Qatar, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Vancouver, pages 28–39.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. pages 1097–1105.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 NAACL*. San Diego, California, pages 260–270.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *ICLR 2016*.

Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. 2016. Semantic object parsing with graph lstm. In *ECCV*. Springer, pages 125–143.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of ACL 2017*. Vancouver, Canada, pages 1–10.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of EMNLP 2015*. pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of ACL 2016*. Berlin, Germany, pages 1064–1074.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *CICLing*. Springer, pages 171–189.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP 2017*. Copenhagen, Denmark, pages 1506–1515.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *UAI*. Morgan Kaufmann Publishers Inc., pages 467–475.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

George Papandreou, Liang-Chieh Chen, Kevin Murphy, and Alan L Yuille. 2015. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv preprint arXiv:1502.02734* .

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*. Ann Arbor, Michigan, pages 78–86.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics* 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*. pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL 2017*. Vancouver, Canada, pages 1756–1765.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949* .

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*. pages 147–155.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of ACL 2017*. Vancouver, Canada, pages 2121–2130.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *NIPS*. pages 3859–3869.

Tjong Kim Sang, Erik F, and De Meulder Fien. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of HLT-NAACL 2003-Volume 4*. pages 142–147.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1):61–80.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP 2012*. pages 1201–1211.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP 2011*. pages 151–161.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of ACL 2011*. pages 48–52.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Xu Sun. 2014. Structure regularization for structured prediction. In *NIPS*. pages 2402–2410.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *InterSpeech*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL 2015*. Beijing, China, pages 1556–1566.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. pages 6000–6010.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016*. pages 2428–2437.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP 2015*. Lisbon, Portugal, pages 1711–1721.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR 2017*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL 2016*. pages 1480–1489.

Universal Language Model Fine-tuning for Text Classification

Jeremy Howard[*]
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder[*]
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al., 2015a; He et al., 2016; Huang et al., 2017).

Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection (Jindal and Liu, 2007; Ngai et al., 2011; Chu et al., 2012), emergency response (Caragea et al., 2011), and commercial document classification, such as for legal discovery (Roitblat et al., 2010).

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet models: The same 3-layer LSTM architecture—with the same hyperparameters and no additions other than tuned dropout hyperparameters—outperforms highly engineered models and trans-

[1] http://nlp.fast.ai/ulmfit.
[*] Equal contribution. Jeremy focused on the algorithm development and implementation, Sebastian focused on the experiments and writing.

fer learning approaches on six widely studied text classification tasks. On IMDb, with 100 labeled examples, ULMFiT matches the performance of training from scratch with $10\times$ and—given 50k unlabeled examples—with $100\times$ more data.

Contributions Our contributions are the following: 1) We propose Universal Language Model Fine-tuning (ULMFiT), a method that can be used to achieve CV-like transfer learning for any task for NLP. 2) We propose *discriminative fine-tuning*, *slanted triangular learning rates*, and *gradual unfreezing*, novel techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning. 3) We significantly outperform the state-of-the-art on six representative text classification datasets, with an error reduction of 18-24% on the majority of datasets. 4) We show that our method enables extremely sample-efficient transfer learning and perform an extensive ablation analysis. 5) We make the pretrained models and our code available to enable wider adoption.

2 Related work

Transfer learning in CV Features in deep neural networks in CV have been observed to transition from task-*specific* to *general* from the first to the last layer (Yosinski et al., 2014). For this reason, most work in CV focuses on transferring the last layers of the model (Long et al., 2015b). Sharif Razavian et al. (2014) achieve state-of-the-art results using features of an ImageNet model as input to a simple classifier. In recent years, this approach has been superseded by fine-tuning either the last (Donahue et al., 2014) or several of the last layers of a pretrained model and leaving the remaining layers frozen (Long et al., 2015a).

Hypercolumns In NLP, only recently have methods been proposed that go beyond transferring word embeddings. The prevailing approach is to pretrain embeddings that capture additional context via other tasks. Embeddings at different levels are then used as features, concatenated either with the word embeddings or with the inputs at intermediate layers. This method is known as hypercolumns (Hariharan et al., 2015) in CV[2] and is used by Peters et al. (2017), Peters et al. (2018), Wieting and Gimpel (2017), Conneau et al. (2017), and McCann et al. (2017) who use language modeling, paraphrasing, entailment, and Machine Translation (MT) respectively for pretraining. Specifically, Peters et al. (2018) require engineered custom architectures, while we show state-of-the-art performance with the same basic architecture across a range of tasks. In CV, hypercolumns have been nearly entirely superseded by end-to-end fine-tuning (Long et al., 2015a).

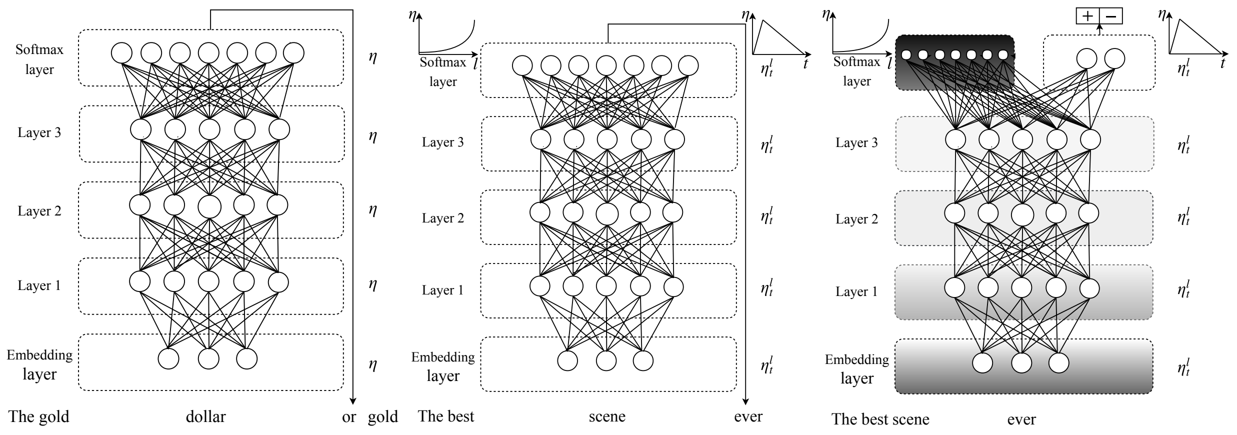
[2]A hypercolumn at a pixel in CV is the vector of activations of all CNN units above that pixel. In analogy, a hypercolumn for a word or sentence in NLP is the concatenation of embeddings at different layers in a pretrained model.

Multi-task learning A related direction is multi-task learning (MTL) (Caruana, 1993). This is the approach taken by Rei (2017) and Liu et al. (2018) who add a language modeling objective to the model that is trained jointly with the main task model. MTL requires the tasks to be trained from scratch every time, which makes it inefficient and often requires careful weighting of the task-specific objective functions (Chen et al., 2017).

Fine-tuning Fine-tuning has been used successfully to transfer between similar tasks, e.g. in QA (Min et al., 2017), for distantly supervised sentiment analysis (Severyn and Moschitti, 2015), or MT domains (Sennrich et al., 2015) but has been shown to fail between unrelated ones (Mou et al., 2016). Dai and Le (2015) also fine-tune a language model, but overfit with 10k labeled examples and require millions of in-domain documents for good performance. In contrast, ULMFiT leverages general-domain pretraining and novel fine-tuning techniques to prevent overfitting even with only 100 labeled examples and achieves state-of-the-art results also on small datasets.

3 Universal Language Model Fine-tuning

We are interested in the most general *inductive* transfer learning setting for NLP (Pan and Yang, 2010): Given a static source task $\mathcal{T}_S$ and *any* target task $\mathcal{T}_T$ with $\mathcal{T}_S \neq \mathcal{T}_T$, we would like to improve performance on $\mathcal{T}_T$. Language modeling can be seen as the ideal source task and a counterpart of ImageNet for NLP: It captures many facets of language relevant for downstream tasks, such as long-term dependencies (Linzen et al., 2016), hierarchical relations (Gulordava et al., 2018), and sentiment (Radford et al., 2017). In contrast to tasks like MT (McCann et al., 2017) and entailment (Conneau et al., 2017), it provides data in near-unlimited quantities for most domains and languages. Additionally, a pretrained LM can be easily adapted to the idiosyncrasies of a target



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

task, which we show significantly improves performance (see Section 5). Moreover, language modeling already is a key component of existing tasks such as MT and dialogue modeling. Formally, language modeling induces a hypothesis space $\mathcal{H}$ that should be useful for many other NLP tasks (Vapnik and Kotz, 1982; Baxter, 2000).

We propose Universal Language Model Fine-tuning (ULMFiT), which pretrains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques. The method is *universal* in the sense that it meets these practical criteria: 1) It works across tasks varying in document size, number, and label type; 2) it uses a single architecture and training process; 3) it requires no custom feature engineering or preprocessing; and 4) it does not require additional in-domain documents or labels.

In our experiments, we use the state-of-the-art language model AWD-LSTM (Merity et al., 2017a), a regular LSTM (with no attention, short-cut connections, or other sophisticated additions) with various tuned dropout hyperparameters. Analogous to CV, we expect that downstream performance can be improved by using higher-performance language models in the future.

ULMFiT consists of the following steps, which we show in Figure 1: a) General-domain LM pretraining (§3.1); b) target task LM fine-tuning (§3.2); and c) target task classifier fine-tuning (§3.3). We discuss these in the following sections.

3.1 General-domain LM pretraining

An ImageNet-like corpus for language should be large and capture general properties of language. We pretrain the language model on Wikitext-103 (Merity et al., 2017b) consisting of 28,595 preprocessed Wikipedia articles and 103 million words. Pretraining is most beneficial for tasks with small datasets and enables generalization even with 100 labeled examples. We leave the exploration of more diverse pretraining corpora to future work, but expect that they would boost performance. While this stage is the most expensive, it only needs to be performed once and improves performance and convergence of downstream models.

3.2 Target task LM fine-tuning

No matter how diverse the general-domain data used for pretraining is, the data of the target task will likely come from a different distribution. We thus fine-tune the LM on data of the target task. Given a pretrained general-domain LM, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target data, and it allows us to train a robust LM even for small datasets. We propose *discriminative fine-tuning* and *slanted triangular learning rates* for fine-tuning the LM, which we introduce in the following.

Discriminative fine-tuning As different layers capture *different types of information* (Yosinski et al., 2014), they should be fine-tuned to *different extents*. To this end, we propose a novel fine-tuning method, *discriminative fine-tuning*[3].

Instead of using the same learning rate for *all* layers of the model, discriminative fine-tuning allows us to tune *each* layer with different learning rates. For context, the regular stochastic gradient descent (SGD) update of a model's parameters θ at time step t looks like the following (Ruder, 2016):

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta) \tag{1}$$

where η is the learning rate and $\nabla_\theta J(\theta)$ is the gradient with regard to the model's objective function. For discriminative fine-tuning, we split the parameters θ into $\{\theta^1, \ldots, \theta^L\}$ where θ^l contains the parameters of the model at the l-th layer and L is the number of layers of the model. Similarly, we obtain $\{\eta^1, \ldots, \eta^L\}$ where η^l is the learning rate of the l-th layer.

The SGD update with discriminative fine-tuning is then the following:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta) \tag{2}$$

We empirically found it to work well to first choose the learning rate η^L of the last layer by fine-tuning only the last layer and using $\eta^{l-1} = \eta^l/2.6$ as the learning rate for lower layers.

Slanted triangular learning rates For adapting its parameters to task-specific features, we would like the model to quickly converge to a suitable region of the parameter space in the beginning of training and then refine its parameters. Using the same learning rate (LR) or an annealed learning rate throughout training is not the best way to achieve this behaviour. Instead, we propose *slanted triangular learning rates* (STLR), which first linearly increases the learning rate and then linearly decays it according to the following update schedule, which can be seen in Figure 2:

$$\begin{aligned} cut &= \lfloor T \cdot cut\_frac \rfloor \\ p &= \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t-cut}{cut\cdot(ratio-1)}, & \text{otherwise} \end{cases} \\ \eta_t &= \eta_{max} \cdot \frac{1 + p\cdot(ratio-1)}{ratio} \end{aligned} \tag{3}$$

where T is the number of training iterations[4], cut_frac is the fraction of iterations we increase the LR, cut is the iteration when we switch from increasing to decreasing the LR, p is the fraction of the number of iterations we have increased or will decrease the LR respectively, $ratio$ specifies how much smaller the lowest LR is from the maximum LR η_{max}, and η_t is the learning rate at iteration t. We generally use $cut\_frac = 0.1$, $ratio = 32$ and $\eta_{max} = 0.01$.

STLR modifies triangular learning rates (Smith, 2017) with a short increase and a long decay period, which we found key for good performance.[5] In Section 5, we compare against aggressive cosine annealing, a similar schedule that has recently been used to achieve state-of-the-art performance in CV (Loshchilov and Hutter, 2017).[6]

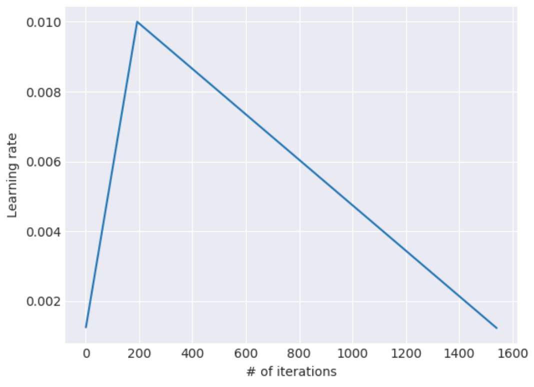


Figure 2: The slanted triangular learning rate schedule used for ULMFiT as a function of the number of training iterations.

3.3 Target task classifier fine-tuning

Finally, for fine-tuning the classifier, we augment the pretrained language model with two additional linear blocks. Following standard practice for CV classifiers, each block uses batch normalization (Ioffe and Szegedy, 2015) and dropout, with ReLU activations for the intermediate layer and a softmax activation that outputs a probability distribution over target classes at the last layer. Note that the parameters in these task-specific classifier layers are the only ones that are learned from scratch. The first linear layer takes as the input the pooled last hidden layer states.

Concat pooling The signal in text classification tasks is often contained in a few words, which may

[3] An unrelated method of the same name exists for deep Boltzmann machines (Salakhutdinov and Hinton, 2009).

[4]In other words, the number of epochs times the number of updates per epoch.

[5]We also credit personal communication with the author.

[6]While Loshchilov and Hutter (2017) use multiple annealing cycles, we generally found one cycle to work best.

occur anywhere in the document. As input documents can consist of hundreds of words, information may get lost if we only consider the last hidden state of the model. For this reason, we concatenate the hidden state at the last time step $\mathbf{h}_T$ of the document with both the max-pooled and the mean-pooled representation of the hidden states over as many time steps as fit in GPU memory $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$:

$$\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})] \quad (4)$$

where [] is concatenation.

Fine-tuning the target classifier is the most critical part of the transfer learning method. Overly aggressive fine-tuning will cause catastrophic forgetting, eliminating the benefit of the information captured through language modeling; too cautious fine-tuning will lead to slow convergence (and resultant overfitting). Besides discriminative fine-tuning and triangular learning rates, we propose *gradual unfreezing* for fine-tuning the classifier.

Gradual unfreezing Rather than fine-tuning all layers at once, which risks catastrophic forgetting, we propose to gradually unfreeze the model starting from the last layer as this contains the *least general* knowledge (Yosinski et al., 2014): We first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We then unfreeze the next lower frozen layer and repeat, until we fine-tune all layers until convergence at the last iteration. This is similar to '*chain-thaw*' (Felbo et al., 2017), except that we add a layer at a time to the set of 'thawed' layers, rather than only training a single layer at a time.

While discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing all are beneficial on their own, we show in Section 5 that they complement each other and enable our method to perform well across diverse datasets.

BPTT for Text Classification (BPT3C) Language models are trained with backpropagation through time (BPTT) to enable gradient propagation for large input sequences. In order to make fine-tuning a classifier for large documents feasible, we propose BPTT for Text Classification (BPT3C): We divide the document into fixed-length batches of size b. At the beginning of each batch, the model is initialized with the final state of the previous batch; we keep track of the hidden states for mean and max-pooling; gradients are back-propagated to the batches whose hidden states contributed to the final prediction. In practice, we use variable length backpropagation sequences (Merity et al., 2017a).

Dataset	Type	# classes	# examples
TREC-6	Question	6	5.5k
IMDb	Sentiment	2	25k
Yelp-bi	Sentiment	2	560k
Yelp-full	Sentiment	5	650k
AG	Topic	4	120k
DBpedia	Topic	14	560k

Table 1: Text classification datasets and tasks with number of classes and training examples.

Bidirectional language model Similar to existing work (Peters et al., 2017, 2018), we are not limited to fine-tuning a unidirectional language model. For all our experiments, we pretrain both a forward and a backward LM. We fine-tune a classifier for each LM independently using BPT3C and average the classifier predictions.

4 Experiments

While our approach is equally applicable to sequence labeling tasks, we focus on text classification tasks in this work due to their important real-world applications.

4.1 Experimental setup

Datasets and tasks We evaluate our method on six widely-studied datasets, with varying numbers of documents and varying document length, used by state-of-the-art text classification and transfer learning approaches (Johnson and Zhang, 2017; McCann et al., 2017) as instances of three common text classification tasks: sentiment analysis, question classification, and topic classification. We show the statistics for each dataset and task in Table 1.

Sentiment Analysis For sentiment analysis, we evaluate our approach on the binary movie review IMDb dataset (Maas et al., 2011) and on the binary and five-class version of the Yelp review dataset compiled by Zhang et al. (2015).

Question Classification We use the six-class version of the small TREC dataset (Voorhees and Tice, 1999) dataset of open-domain, fact-based questions divided into broad semantic categories.

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

Topic classification For topic classification, we evaluate on the large-scale AG news and DBpedia ontology datasets created by Zhang et al. (2015).

Pre-processing We use the same pre-processing as in earlier work (Johnson and Zhang, 2017; McCann et al., 2017). In addition, to allow the language model to capture aspects that might be relevant for classification, we add special tokens for upper-case words, elongation, and repetition.

Hyperparameters We are interested in a model that performs robustly across a diverse set of tasks. To this end, if not mentioned otherwise, we use the same set of hyperparameters across tasks, which we tune on the IMDb validation set. We use the AWD-LSTM language model (Merity et al., 2017a) with an embedding size of 400, 3 layers, 1150 hidden activations per layer, and a BPTT batch size of 70. We apply dropout of 0.4 to layers, 0.3 to RNN layers, 0.4 to input embedding layers, 0.05 to embedding layers, and weight dropout of 0.5 to the RNN hidden-to-hidden matrix. The classifier has a hidden layer of size 50. We use Adam with $\beta_1 = 0.7$ instead of the default $\beta_1 = 0.9$ and $\beta_2 = 0.99$, similar to (Dozat and Manning, 2017). We use a batch size of 64, a base learning rate of 0.004 and 0.01 for fine-tuning the LM and the classifier respectively, and tune the number of epochs on the validation set of each task[7]. We otherwise use the same practices used in (Merity et al., 2017a).

[7]On small datasets such as TREC-6, we fine-tune the LM only for 15 epochs without overfitting, while we can fine-tune longer on larger datasets. We found 50 epochs to be a good default for fine-tuning the classifier.

Baselines and comparison models For each task, we compare against the current state-of-the-art. For the IMDb and TREC-6 datasets, we compare against CoVe (McCann et al., 2017), a state-of-the-art transfer learning method for NLP. For the AG, Yelp, and DBpedia datasets, we compare against the state-of-the-art text categorization method by Johnson and Zhang (2017).

4.2 Results

For consistency, we report all results as error rates (lower is better). We show the test error rates on the IMDb and TREC-6 datasets used by McCann et al. (2017) in Table 2. Our method outperforms both CoVe, a state-of-the-art transfer learning method based on hypercolumns, as well as the state-of-the-art on both datasets. On IMDb, we reduce the error dramatically by 43.9% and 22% with regard to CoVe and the state-of-the-art respectively. This is promising as the existing state-of-the-art requires complex architectures (Peters et al., 2018), multiple forms of attention (McCann et al., 2017) and sophisticated embedding schemes (Johnson and Zhang, 2016), while our method employs a regular LSTM with dropout. We note that the language model fine-tuning approach of Dai and Le (2015) only achieves an error of 7.64 vs. 4.6 for our method on IMDb, demonstrating the benefit of transferring knowledge from a large ImageNet-like corpus using our fine-tuning techniques. IMDb in particular is reflective of real-world datasets: Its documents are generally a few

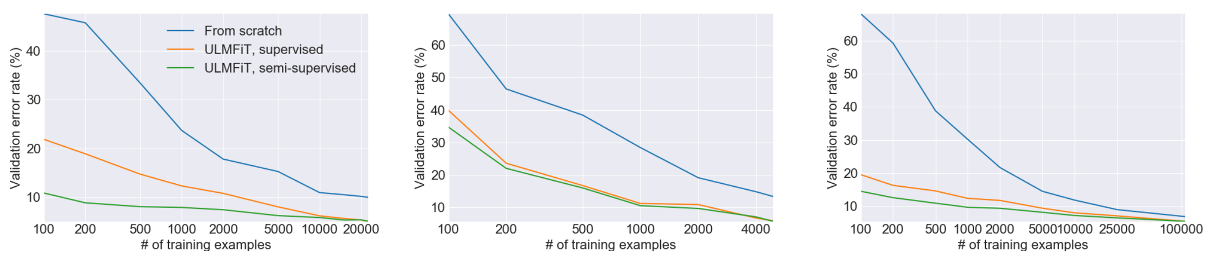


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

paragraphs long—similar to emails (e.g for legal discovery) and online comments (e.g for community management); and sentiment analysis is similar to many commercial applications, e.g. product response tracking and support email routing.

On TREC-6, our improvement—similar as the improvements of state-of-the-art approaches—is not statistically significant, due to the small size of the 500-examples test set. Nevertheless, the competitive performance on TREC-6 demonstrates that our model performs well across different dataset sizes and can deal with examples that range from single sentences—in the case of TREC-6—to several paragraphs for IMDb. Note that despite pretraining on more than two orders of magnitude less data than the 7 million sentence pairs used by McCann et al. (2017), we consistently outperform their approach on both datasets.

We show the test error rates on the larger AG, DBpedia, Yelp-bi, and Yelp-full datasets in Table 3. Our method again outperforms the state-of-the-art significantly. On AG, we observe a similarly dramatic error reduction by 23.7% compared to the state-of-the-art. On DBpedia, Yelp-bi, and Yelp-full, we reduce the error by 4.8%, 18.2%, 2.0% respectively.

5 Analysis

In order to assess the impact of each contribution, we perform a series of analyses and ablations. We run experiments on three corpora, IMDb, TREC-6, and AG that are representative of different tasks, genres, and sizes. For all experiments, we split off 10% of the training set and report error rates on this validation set with unidirectional LMs. We fine-tune the classifier for 50 epochs and train all methods but ULMFiT with early stopping.

Low-shot learning One of the main benefits of transfer learning is being able to train a model for a task with a small number of labels. We evaluate ULMFiT on different numbers of labeled examples in two settings: only labeled examples are used for LM fine-tuning ('*supervised*'); and all task data is available and can be used to fine-tune the LM ('*semi-supervised*'). We compare ULMFiT to training from scratch—which is necessary for hypercolumn-based approaches. We split off balanced fractions of the training data, keep the validation set fixed, and use the same hyperparameters as before. We show the results in Figure 3.

On IMDb and AG, supervised ULMFiT with only 100 labeled examples matches the performance of training from scratch with $10\times$ and $20\times$ more data respectively, clearly demonstrating the benefit of general-domain LM pretraining. If we allow ULMFiT to also utilize unlabeled examples (50k for IMDb, 100k for AG), at 100 labeled examples, we match the performance of training from scratch with $50\times$ and $100\times$ more data on AG and IMDb respectively. On TREC-6, ULMFiT significantly improves upon training from scratch; as examples are shorter and fewer, supervised and semi-supervised ULMFiT achieve similar results.

Pretraining	IMDb	TREC-6	AG
Without pretraining	5.63	10.67	5.52
With pretraining	**5.00**	**5.69**	**5.38**

Table 4: Validation error rates for ULMFiT with and without pretraining.

Impact of pretraining We compare using no pretraining with pretraining on WikiText-103 (Merity et al., 2017b) in Table 4. Pretraining is most useful for small and medium-sized datasets, which are most common in commercial applications. However, even for large datasets, pretraining improves performance.

LM	IMDb	TREC-6	AG
Vanilla LM	5.98	7.41	5.76
AWD-LSTM LM	**5.00**	**5.69**	**5.38**

Table 5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

Impact of LM quality In order to gauge the importance of choosing an appropriate LM, we compare a vanilla LM with the same hyperparameters without any dropout[8] with the AWD-LSTM LM with tuned dropout parameters in Table 5. Using our fine-tuning techniques, even a regular LM reaches surprisingly good performance on the larger datasets. On the smaller TREC-6, a vanilla LM without dropout runs the risk of overfitting, which decreases performance.

Impact of LM fine-tuning We compare no fine-tuning against fine-tuning the full model (Erhan et al., 2010) ('*Full*'), the most commonly used fine-tuning method, with and without discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*') in Table 6. Fine-tuning the LM is most beneficial for larger datasets. '*Discr*' and '*Stlr*' improve performance across all three datasets and are necessary on the smaller TREC-6, where regular fine-tuning is not beneficial.

Impact of classifier fine-tuning We compare training from scratch, fine-tuning the full model ('*Full*'), only fine-tuning the last layer ('*Last*') (Donahue et al., 2014), '*Chain-thaw*' (Felbo et al., 2017), and gradual unfreezing ('*Freez*'). We furthermore assess the importance of discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*'). We compare the latter to an alternative, aggressive cosine annealing schedule ('*Cos*') (Loshchilov and Hutter, 2017). We use a learning rate $\eta^L = 0.01$ for '*Discr*', learning rates of 0.001 and 0.0001 for the last and all other layers respectively for '*Chain-thaw*' as in (Felbo et al., 2017), and a learning rate of 0.001 otherwise. We show the results in Table 7.

[8]To avoid overfitting, we only train the vanilla LM classifier for 5 epochs and keep dropout of 0.4 in the classifier.

Classifier fine-tuning	IMDb	TREC-6	AG
From scratch	9.93	13.36	6.81
Full	6.87	6.86	5.81
Full + discr	4.57	6.21	5.62
Last	6.49	16.09	8.38
Chain-thaw	5.39	6.71	5.90
Freez	6.37	6.86	5.81
Freez + discr	5.39	5.86	6.04
Freez + stlr	5.04	6.02	5.35
Freez + cos	5.70	6.38	**5.29**
Freez + discr + stlr	**5.00**	**5.69**	5.38

Table 7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

Fine-tuning the classifier significantly improves over training from scratch, particularly on the small TREC-6. '*Last*', the standard fine-tuning method in CV, severely underfits and is never able to lower the training error to 0. '*Chain-thaw*' achieves competitive performance on the smaller datasets, but is outperformed significantly on the large AG. '*Freez*' provides similar performance as '*Full*'. '*Discr*' consistently boosts the performance of '*Full*' and '*Freez*', except for the large AG. Cosine annealing is competitive with slanted triangular learning rates on large data, but under-performs on smaller datasets. Finally, full ULMFiT classifier fine-tuning (bottom row) achieves the best performance on IMDB and TREC-6 and competitive performance on AG. Importantly, ULMFiT is the only method that shows excellent performance across the board—and is therefore the only *universal* method.

Classifier fine-tuning behavior While our results demonstrate that *how* we fine-tune the classifier makes a significant difference, fine-tuning for inductive transfer is currently under-explored in NLP as it mostly has been thought to be unhelpful (Mou et al., 2016). To better understand the fine-tuning behavior of our model, we compare the validation error of the classifier fine-tuned with ULMFiT and '*Full*' during training in Figure 4.

On all datasets, fine-tuning the full model leads to the lowest error comparatively early in training, e.g. already after the first epoch on IMDb.

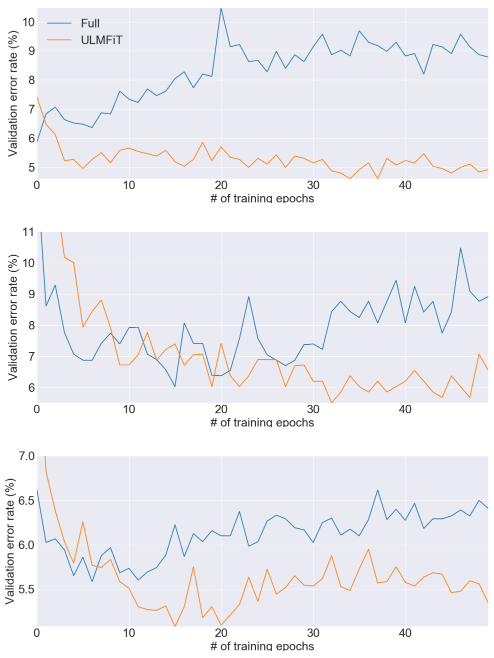


Figure 4: Validation error rate curves for fine-tuning the classifier with ULMFiT and '*Full*' on IMDb, TREC-6, and AG (top to bottom).

The error then increases as the model starts to overfit and knowledge captured through pretraining is lost. In contrast, ULMFiT is more stable and suffers from no such catastrophic forgetting; performance remains similar or improves until late epochs, which shows the positive effect of the learning rate schedule.

Impact of bidirectionality At the cost of training a second model, ensembling the predictions of a forward and backwards LM-classifier brings a performance boost of around 0.5–0.7. On IMDb we lower the test error from 5.30 of a single model to 4.58 for the bidirectional model.

6 Discussion and future directions

While we have shown that ULMFiT can achieve state-of-the-art performance on widely used text classification tasks, we believe that language model fine-tuning will be particularly useful in the following settings compared to existing transfer learning approaches (Conneau et al., 2017; McCann et al., 2017; Peters et al., 2018): a) NLP for non-English languages, where training data for supervised pretraining tasks is scarce; b) new NLP tasks where no state-of-the-art architecture exists; and c) tasks with limited amounts of labeled data (and some amounts of unlabeled data).

Given that transfer learning and particularly fine-tuning for NLP is under-explored, many future directions are possible. One possible direction is to improve language model pretraining and fine-tuning and make them more scalable: for ImageNet, predicting far fewer classes only incurs a small performance drop (Huh et al., 2016), while recent work shows that an alignment between source and target task label sets is important (Mahajan et al., 2018)—focusing on predicting a subset of words such as the most frequent ones might retain most of the performance while speeding up training. Language modeling can also be augmented with additional tasks in a multi-task learning fashion (Caruana, 1993) or enriched with additional supervision, e.g. syntax-sensitive dependencies (Linzen et al., 2016) to create a model that is more general or better suited for certain downstream tasks, ideally in a weakly-supervised manner to retain its universal properties.

Another direction is to apply the method to novel tasks and models. While an extension to sequence labeling is straightforward, other tasks with more complex interactions such as entailment or question answering may require novel ways to pretrain and fine-tune. Finally, while we have provided a series of analyses and ablations, more studies are required to better understand what knowledge a pretrained language model captures, how this changes during fine-tuning, and what information different tasks require.

7 Conclusion

We have proposed ULMFiT, an effective and extremely sample-efficient transfer learning method that can be applied to any NLP task. We have also proposed several novel fine-tuning techniques that in conjunction prevent catastrophic forgetting and enable robust learning across a diverse range of tasks. Our method significantly outperformed existing transfer learning techniques and the state-of-the-art on six representative text classification tasks. We hope that our results will catalyze new developments in transfer learning for NLP.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289.

References

Jonathan Baxter. 2000. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research* 12:149–198.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics* 45(1):440. https://doi.org/10.1109/IRPS.2011.5784441.

Cornelia Caragea, Nathan McNeese, Anuj Jaiswal, Greg Traylor, Hyun-Woo Kim, Prasenjit Mitra, Dinghao Wu, Andrea H Tapia, Lee Giles, Bernard J Jansen, et al. 2011. Classifying text messages for the haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks pages 1–10.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing* 9(6):811–824.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems (NIPS '15)* http://arxiv.org/abs/1511.01432.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*. pages 647–655.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR 2017*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11(Feb):625–660.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL-HLT 2018*.

Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 447–456.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. 2017. Densely Connected Convolutional Networks. In *Proceedings of CVPR 2017*.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. 2016. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614* .

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. pages 448–456.

Nitin Jindal and Bing Liu. 2007. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*. ACM, pages 1189–1190.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*. pages 526–534.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 562–570.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368* .

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of AAAI 2018*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015a. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3431–3440.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015b. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the 32nd International Conference on Machine learning (ICML '15)*. volume 37.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proceedings of the Internal Conference on Learning Representations 2017*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the Limits of Weakly Supervised Pretraining .

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017a. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* .

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017b. Pointer Sentinel Mixture Models. In *Proceedings of the International Conference on Learning Representations 2017*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* .

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? *Proceedings of 2016 Conference on Empirical Methods in Natural Language Processing* .

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems* 50(3):559–569.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL 2017*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL 2018*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444* .

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of ACL 2017*.

Herbert L Roitblat, Anne Kershaw, and Patrick Oot. 2010. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the Association for Information Science and Technology* 61(1):70–80.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* .

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep boltzmann machines. In *Artificial Intelligence and Statistics*. pages 448–455.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Aliaksei Severyn and Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* pages 464–469.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 806–813.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, pages 464–472.

Vladimir Naumovich Vapnik and Samuel Kotz. 1982. *Estimation of dependences based on empirical data*, volume 40. Springer-Verlag New York.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*. volume 1999, page 82.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*. pages 3320–3328.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016*.

Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement

Nina Poerner, Benjamin Roth & Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
poerner@cis.lmu.de

Abstract

The behavior of deep neural networks (DNNs) is hard to understand. This makes it necessary to explore post hoc explanation methods. We conduct the first comprehensive evaluation of explanation methods for NLP. To this end, we design two novel evaluation paradigms that cover two important classes of NLP problems: small context and large context problems. Both paradigms require no manual annotation and are therefore broadly applicable. We also introduce LIMSSE, an explanation method inspired by LIME that is designed for NLP. We show empirically that LIMSSE, LRP and DeepLIFT are the most effective explanation methods and recommend them for explaining DNNs in NLP.

1 Introduction

DNNs are complex models that combine linear transformations with different types of nonlinearities. If the model is deep, i.e., has many layers, then its behavior during training and inference is notoriously hard to understand.

This is a problem for both scientific methodology and real-world deployment. Scientific methodology demands that we understand our models. In the real world, a decision (e.g., "your blog post is offensive and has been removed") by itself is often insufficient; in addition, an explanation of the decision may be required (e.g., "our system flagged the following words as offensive"). The European Union plans to mandate that intelligent systems used for sensitive applications provide such explanations (European General Data Protection Regulation, expected 2018, cf. Goodman and Flaxman (2016)).

A number of post hoc explanation methods for DNNs have been proposed. Due to the complexity of the DNNs they explain, these methods are necessarily approximations and come with their own sources of error. At this point, it is not clear which of these methods to use when reliable explanations for a specific DNN architecture are needed.

Definitions. (i) A *task method* solves an NLP problem, e.g., a GRU that predicts sentiment.

(ii) An *explanation method* explains the behavior of a task method on a specific input. For our purpose, it is a function $\phi(t, k, \mathbf{X})$ that assigns real-valued relevance scores for a target class k (e.g., positive) to positions t in an input text $\mathbf{X}$ (e.g., "great food"). For this example, an explanation method might assign: $\phi(1, k, \mathbf{X}) > \phi(2, k, \mathbf{X})$.

(iii) An *(explanation) evaluation paradigm* quantitatively evaluates explanation methods for a task method, e.g., by assigning them accuracies.

Contributions. (i) We present novel evaluation paradigms for explanation methods for two classes of common NLP tasks (see §2). Crucially, *neither paradigm requires manual annotations* and our methodology is therefore broadly applicable.

(ii) Using these paradigms, we perform a comprehensive evaluation of explanation methods for NLP (§3). We cover the most important classes of task methods, RNNs and CNNs, as well as the recently proposed Quasi-RNNs.

(iii) We introduce LIMSSE (§3.6), an explanation method inspired by LIME (Ribeiro et al.,

tasks	sentiment analysis, morphological prediction, ...
task methods	CNN, GRU, LSTM, ...
explanation methods	LIMSSE, LRP, DeepLIFT, ...
evaluation paradigms	hybrid document, morphosyntactic agreement

Table 1: Terminology with examples.

lrp	From : *kolstad* @ cae.wisc.edu (Joel **Kolstad**) Subject : Re : Can Radio Freq . Be Used To Measure Distance ? [...] What is the difference between vertical and horizontal ? Gravity ? Does n't gravity pull down the photons and cause a *doppler* shift or something ? (Just kidding !)
$\mathrm{grad}^{\mathrm{L2}}_{1p}$	If you find faith to be honest , show me how . David The whole **denominational** mindset only causes more problems , sadly . (See section 7 for details .) Thank you . 'The Armenians just shot and shot . Maybe *coz* they 're *'quality'* cars ; -) 200 *posts/day* . [...]
$\mathrm{limsse}^{\mathrm{ms}}_{s}$	If you find faith to be honest , show me how . David The whole denominational mindset only causes more problems , sadly . (See section 7 for details .) Thank you . 'The **Armenians** just shot and shot . Maybe *coz* they 're *'quality'* cars ; -) 200 *posts/day* . [...]

Figure 1: **Top:** sci.electronics post (not hybrid). Underlined: Manual relevance ground truth. Green: evidence for sci.electronics. Task method: CNN. **Bottom:** hybrid newsgroup post, classified talk.politics.mideast. Green: evidence for talk.politics.mideast. Underlined: talk.politics.mideast fragment. Task method: QGRU. Italics: OOV. Bold: rmax position. See supplementary for full texts.

2016) that is designed for word-order sensitive task methods (e.g., RNNs, CNNs). We show empirically that LIMSSE, LRP (Bach et al., 2015) and DeepLIFT (Shrikumar et al., 2017) are the most effective explanation methods (§4): LRP and DeepLIFT are the most consistent methods, while LIMSSE wins the hybrid document experiment.

2 Evaluation paradigms

In this section, we introduce two novel evaluation paradigms for explanation methods on two types of common NLP tasks, *small context* tasks and *large context* tasks. Small context tasks are defined as those that can be solved by finding short, self-contained indicators, such as words and phrases, and weighing them up (i.e., tasks where CNNs with pooling can be expected to perform well). We design the *hybrid document paradigm* for evaluating explanation methods on small context tasks. Large context tasks require the correct handling of long-distance dependencies, such as subject-verb agreement.[1] We design the *morphosyntactic agreement paradigm* for evaluating explanation methods on large context tasks.

We could also use **human judgments** for evaluation. While we use Mohseni and Ragan (2018)'s manual relevance benchmark for comparison, there are two issues with it: (i) Due to the *cost of human labor*, it is limited in size and domain. (ii) More importantly, *a good explanation method should not reflect what humans attend to, but what task methods attend to.* For instance, the family name "Kolstad" has 11 out of its 13 appearances in the 20 newsgroups corpus in sci.electronics posts. Thus, task methods probably learn it as a sci.electronics indicator. Indeed, the explanation method in Fig 1 (top) marks "Kolstad" as relevant, but the human annotator does not.

2.1 Small context: Hybrid document paradigm

Given a collection of documents, hybrid documents are created by randomly concatenating document fragments. We assume that, on average, the most relevant input for a class k in a hybrid document is located in a fragment that stems from a document with gold label k. Hence, an explanation method succeeds if it places maximal relevance for k inside the correct fragment.

Formally, let x_t be a word inside hybrid document $\mathbf{X}$ that originates from a document $\mathbf{X}'$ with gold label $y(\mathbf{X}')$. x_t's gold label $y(\mathbf{X}, t)$ is set to $y(\mathbf{X}')$. Let $f(\mathbf{X})$ be the class assigned to the hybrid document by a task method, and let ϕ be an explanation method as defined above. Let $\mathrm{rmax}(\mathbf{X}, \phi)$ denote the position of the maximally relevant word in $\mathbf{X}$ for the predicted class $f(\mathbf{X})$. If this maximally relevant word comes from a document with the correct gold label, the explanation method is awarded a hit:

$$\mathrm{hit}(\phi, \mathbf{X}) = \mathbb{I}[y\big(\mathbf{X}, \mathrm{rmax}(\mathbf{X}, \phi)\big) = f(\mathbf{X})] \quad (1)$$

where $\mathbb{I}[P]$ is 1 if P is true and 0 otherwise. In Fig 1 (bottom), the explanation method $\mathrm{grad}^{\mathrm{L2}}_{1p}$ places rmax outside the correct (underlined) fragment. Therefore, it does not get a hit point, while $\mathrm{limsse}^{\mathrm{ms}}_{s}$ does.

The pointing game accuracy of an explanation method is calculated as its total number of hit points divided by the number of possible hit points. This is a form of the pointing game paradigm from computer vision (Zhang et al., 2016).

2.2 Large context: Morphosyntactic agreement paradigm

Many natural languages display morphosyntactic agreement between words v and w. A DNN that

[1] Consider deciding the number of *[verb]* in "the children in the green house said that the big telescope *[verb]*" vs. "the children in the green house who broke the big telescope *[verb]*". The local contexts of "children" or "*[verb]*" do not suffice to solve this problem, instead, the large context of the entire sentence has to be considered.

$\text{grad}^{\text{dot}}_{\int s}$	the **link** provided by the editor above [encourages ...]
lrp	the **link** provided by the editor above [encourages ...]
$\text{limsse}^{\text{bb}}$	the link provided by the **editor** above [encourages ...]

$\text{grad}^{\text{L2}}_{\int s}$	few if any events in **history** [are ...]
occ_1	few if any **events** in history [are ...]
$\text{limsse}^{\text{ms}}_{s}$	few if any **events** in history [are ...]

Figure 2: **Top:** verb context classified singular. Green: evidence for singular. Task method: GRU. **Bottom:** verb context classified plural. Green: evidence for plural. Task method: LSTM. Underlined: subject. Bold: rmax position.

predicts the agreeing feature in w should pay attention to v. For example, in the sentence "the children with the telescope are home", the number of the verb (plural for "are") can be predicted from the subject ("children") without looking at the verb. If the language allows for v and w to be far apart (Fig 3, top), successful task methods have to be able to handle large contexts.

Linzen et al. (2016) show that English verb number can be predicted by a unidirectional LSTM with accuracy $> 99\%$, based on left context alone. When a task method predicts the correct number, we expect successful explanation methods to place maximal relevance on the subject:

$$\text{hit}_{\text{target}}(\phi, \mathbf{X}) = \mathbb{I}[\text{rmax}(\mathbf{X}, \phi) = \text{target}(\mathbf{X})]$$

where $\text{target}(\mathbf{X})$ is the location of the subject, and rmax is calculated as above. Regardless of whether the prediction is correct, we expect rmax to fall onto a noun that has the predicted number:

$$\text{hit}_{\text{feat}}(\phi, \mathbf{X}) = \mathbb{I}[\text{feat}\big(\mathbf{X}, \text{rmax}(\mathbf{X}, \phi)\big) = f(\mathbf{X})]$$

where $\text{feat}(\mathbf{X}, t)$ is the morphological feature (here: number) of x_t. In Fig 2, rmax on "link" gives a $\text{hit}_{\text{target}}$ point (and a hit_{feat} point), rmax on "editor" gives a hit_{feat} point. $\text{grad}^{\text{L2}}_{\int s}$ does not get any points as "history" is not a plural noun.

Labels for this task can be automatically generated using part-of-speech taggers and parsers, which are available for many languages.

3 Explanation methods

In this section, we define the explanation methods that will be evaluated. For our purpose, explanation methods produce word relevance scores $\phi(t, k, \mathbf{X})$, which are specific to a given class k and a given input $\mathbf{X}$. $\phi(t, k, \mathbf{X}) > \phi(t', k, \mathbf{X})$ means that x_t contributed more than $x_{t'}$ to the task method's (potential) decision to classify $\mathbf{X}$ as k.

3.1 Gradient-based explanation methods

Gradient-based explanation methods approximate the contribution of some DNN input i to some output o with o's gradient with respect to i (Simonyan et al., 2014). In the following, we consider two output functions $o(k, \mathbf{X})$, the unnormalized class score $s(k, \mathbf{X})$ and the class probability $p(k|\mathbf{X})$:

$$s(k, \mathbf{X}) = \vec{w}_k \cdot \vec{h}(\mathbf{X}) + b_k \tag{2}$$

$$p(k|\mathbf{X}) = \frac{\exp\big(s(k, \mathbf{X})\big)}{\sum_{k'=1}^{K} \exp\big(s(k', \mathbf{X})\big)} \tag{3}$$

where k is the target class, $\vec{h}(\mathbf{X})$ the document representation (e.g., an RNN's final hidden layer), $\vec{w}_k$ (resp. b_k) k's weight vector (resp. bias).

The simple gradient of $o(k, \mathbf{X})$ w.r.t. i is:

$$\text{grad}_1(i, k, \mathbf{X}) = \frac{\partial o(k, \mathbf{X})}{\partial i} \tag{4}$$

grad_1 underestimates the importance of inputs that saturate a nonlinearity (Shrikumar et al., 2017). To address this, Sundararajan et al. (2017) integrate over all gradients on a linear interpolation $\alpha \in [0, 1]$ between a baseline input $\bar{\mathbf{X}}$ (here: all-zero embeddings) and $\mathbf{X}$:

$$\begin{aligned}\text{grad}_{\int}(i, k, \mathbf{X}) &= \int_{\alpha=0}^{1} \frac{\partial o(k, \bar{\mathbf{X}} + \alpha(\mathbf{X} - \bar{\mathbf{X}}))}{\partial i} \partial\alpha \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \frac{\partial o(k, \bar{\mathbf{X}} + \frac{m}{M}(\mathbf{X} - \bar{\mathbf{X}}))}{\partial i}\end{aligned} \tag{5}$$

where M is a big enough constant (here: 50).

In NLP, symbolic inputs (e.g., words) are often represented as one-hot vectors $\vec{x}_t \in \{1, 0\}^{|V|}$ and embedded via a real-valued matrix: $\vec{e}_t = \mathbf{M}\vec{x}_t$. Gradients are computed with respect to individual entries of $\mathbf{E} = [\vec{e}_1 \ldots \vec{e}_{|\mathbf{X}|}]$. Bansal et al. (2016) and Hechtlinger (2016) use the L2 norm to reduce vectors of gradients to single values:

$$\phi_{\text{grad}^{\text{L2}}}(t, k, \mathbf{X}) = ||\text{grad}(\vec{e}_t, k, \mathbf{E})|| \tag{6}$$

where $\text{grad}(\vec{e}_t, k, \mathbf{E})$ is a vector of elementwise gradients w.r.t. $\vec{e}_t$. Denil et al. (2015) use the dot product of the gradient vector and the embedding[2], i.e., the gradient of the "hot" entry in $\vec{x}_t$:

$$\phi_{\text{grad}^{\text{dot}}}(t, k, \mathbf{X}) = \vec{e}_t \cdot \text{grad}(\vec{e}_t, k, \mathbf{E}) \tag{7}$$

We use "grad_1" for Eq 4, "$\text{grad}_{\int}$" for Eq 5, "$_p$" for Eq 3, "$_s$" for Eq 2, "L2" for Eq 6 and "dot" for Eq 7. This gives us eight explanation methods: $\text{grad}^{\text{L2}}_{1s}$, $\text{grad}^{\text{L2}}_{1p}$, $\text{grad}^{\text{dot}}_{1s}$, $\text{grad}^{\text{dot}}_{1p}$, $\text{grad}^{\text{L2}}_{\int s}$, $\text{grad}^{\text{L2}}_{\int p}$, $\text{grad}^{\text{dot}}_{\int s}$, $\text{grad}^{\text{dot}}_{\int p}$.

[2]For $\text{grad}^{\text{dot}}_{\int}$, replace $\vec{e}_t$ with $\vec{e}_t - \bar{\vec{e}}_t$. Since our baseline embeddings are all-zeros, this is equivalent.

3.2 Layer-wise relevance propagation

Layer-wise relevance propagation (LRP) is a backpropagation-based explanation method developed for fully connected neural networks and CNNs (Bach et al., 2015) and later extended to LSTMs (Arras et al., 2017b). In this paper, we use Epsilon LRP (Eq 58, Bach et al. (2015)). Remember that the activation of neuron j, a_j, is the sum of weighted upstream activations, $\sum_i a_i w_{i,j}$, plus bias b_j, squeezed through some nonlinearity. We denote the pre-nonlinearity activation of j as a'_j. The relevance of j, $R(j)$, is distributed to upstream neurons i proportionally to the contribution that i makes to a'_j in the forward pass:

$$R(i) = \sum_j R(j) \frac{a_i w_{i,j}}{a'_j + \mathrm{esign}(a'_j)} \quad (8)$$

This ensures that relevance is conserved between layers, with the exception of relevance attributed to b_j. To prevent numerical instabilities, $\mathrm{esign}(a')$ returns $-\epsilon$ if $a' < 0$ and ϵ otherwise. We set $\epsilon = .001$. The full algorithm is:

$$R(L_{k'}) = s(k, \mathbf{X})\mathbb{I}[k' = k]$$

... recursive application of Eq 8 ...

$$\phi_{\mathrm{lrp}}(t, k, \mathbf{X}) = \sum_{j=1}^{\dim(\vec{e}_t)} R(e_{t,j})$$

where L is the final layer, k the target class and $R(e_{t,j})$ the relevance of dimension j in the t'th embedding vector. For $\epsilon \to 0$ and provided that all nonlinearities up to the unnormalized class score are relu, Epsilon LRP is equivalent to the product of input and raw score gradient (here: $\mathrm{grad}_{1s}^{\mathrm{dot}}$) (Kindermans et al., 2016). In our experiments, the second requirement holds only for CNNs.

Experiments by Ancona et al. (2017) (see §6) suggest that LRP does not work well for LSTMs if all neurons – including gates – participate in backpropagation. We therefore use Arras et al. (2017b)'s modification and treat sigmoid-activated gates as time step-specific weights rather than neurons. For instance, the relevance of LSTM candidate vector $\vec{g}_t$ is calculated from memory vector $\vec{c}_t$ and input gate vector $\vec{i}_t$ as

$$R(g_{t,d}) = R(c_{t,d}) \frac{g_{t,d} \cdot i_{t,d}}{c_{t,d} + \mathrm{esign}(c_{t,d})}$$

This is equivalent to applying Eq 8 while treating $\vec{i}_t$ as a diagonal weight matrix. The gate neurons in $\vec{i}_t$ do not receive any relevance themselves. See supplementary material for formal definitions of Epsilon LRP for different architectures.

3.3 DeepLIFT

DeepLIFT (Shrikumar et al., 2017) is another backpropagation-based explanation method. Unlike LRP, it does not explain $s(k, \mathbf{X})$, but $s(k, \mathbf{X}) - s(k, \bar{\mathbf{X}})$, where $\bar{\mathbf{X}}$ is some baseline input (here: all-zero embeddings). Following Ancona et al. (2018) (Eq 4), we use this backpropagation rule:

$$R(i) = \sum_j R(j) \frac{a_i w_{i,j} - \bar{a}_i w_{i,j}}{a'_j - \bar{a}'_j + \mathrm{esign}(a'_j - \bar{a}'_j)}$$

where $\bar{a}$ refers to the forward pass of the baseline. Note that the original method has a different mechanism for avoiding small denominators; we use esign for compatibility with LRP. The DeepLIFT algorithm is started with $R(L_{k'}) = \big(s(k, \mathbf{X}) - s(k, \bar{\mathbf{X}})\big)\mathbb{I}[k' = k]$. On gated (Q)RNNs, we proceed analogous to LRP and treat gates as weights.

3.4 Cell decomposition for gated RNNs

The cell decomposition explanation method for LSTMs (Murdoch and Szlam, 2017) decomposes the unnormalized class score $s(k, \mathbf{X})$ (Eq 2) into additive contributions. For every time step t, we compute how much of $\vec{c}_t$ "survives" until the final step T and contributes to $s(k, \mathbf{X})$. This is achieved by applying all future forget gates $\vec{f}$, the final $\tanh$ nonlinearity, the final output gate $\vec{o}_T$, as well as the class weights of k to $\vec{c}_t$. We call this quantity "net load of t for class k":

$$\mathrm{nl}(t, k, \mathbf{X}) = \vec{w}_k \cdot \Big(\vec{o}_T \odot \tanh\big(\big(\prod_{j=t+1}^{T} \vec{f}_j\big) \odot \vec{c}_t\big)\Big)$$

where $\odot$ and $\prod$ are applied elementwise. The relevance of t is its gain in net load relative to $t-1$: $\phi_{\mathrm{decomp}}(t, k, \mathbf{X}) = \mathrm{nl}(t, k, \mathbf{X}) - \mathrm{nl}(t-1, k, \mathbf{X})$. For GRU, we change the definition of net load:

$$\mathrm{nl}(t, k, \mathbf{X}) = \vec{w}_k \cdot \big(\big(\prod_{j=t+1}^{T} \vec{z}_j\big) \odot \vec{h}_t\big)$$

where $\vec{z}$ are GRU update gates.

3.5 Input perturbation methods

Input perturbation methods assume that the removal or masking of relevant inputs changes the output (Zeiler and Fergus, 2014). Omission-based methods remove inputs completely (Kádár et al., 2017), while occlusion-based methods replace them with a baseline (Li et al., 2016b). In computer vision, perturbations are usually applied to patches, as neighboring pixels tend to correlate (Zintgraf et al., 2017). To calculate the omit_N (resp. occ_N) relevance of word x_t, we delete (resp. occlude), one at a time, all N-grams that contain x_t, and average the change in the unnormalized class score from Eq 2:

$$\phi_{[\mathrm{omit}|\mathrm{occ}]_N}(t,k,\mathbf{X}) = \sum_{j=1}^{N} \big[s(k,[\vec{e}_1 \dots \vec{e}_{|\mathbf{X}|}]) \\ -s(k,[\vec{e}_1 \dots \vec{e}_{t-N-1+j}] \| \bar{\mathbf{E}} \| [\vec{e}_{t+j} \dots \vec{e}_{|\mathbf{X}|}])\big] \frac{1}{N}$$

where $\vec{e}_t$ are embedding vectors, $\|$ denotes concatenation and $\bar{\mathbf{E}}$ is either a sequence of length zero (ϕ_{omit}) or a sequence of N baseline (here: all-zero) embedding vectors (ϕ_{occ}).

3.6 LIMSSE: LIME for NLP

Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) is a framework for explaining predictions of complex classifiers. LIME approximates the behavior of classifier f in the neighborhood of input $\mathbf{X}$ with an interpretable (here: linear) model. The interpretable model is trained on samples $\mathbf{Z}_1 \dots \mathbf{Z}_N$ (here: $N = 3000$), which are randomly drawn from $\mathbf{X}$, with "gold labels" $f(\mathbf{Z}_1) \dots f(\mathbf{Z}_N)$.

Since RNNs and CNNs respect word order, we cannot use the bag of words sampling method from the original description of LIME. Instead, we introduce Local Interpretable Model-agnostic Substring-based Explanations (LIMSSE). LIMSSE uniformly samples a length l_n (here: $1 \le l_n \le 6$) and a starting point s_n, which define the substring $\mathbf{Z}_n = [\vec{x}_{s_n} \dots \vec{x}_{s_n+l_n-1}]$. To the linear model, $\mathbf{Z}_n$ is represented by a binary vector $\vec{z}_n \in \{0,1\}^{|\mathbf{X}|}$, where $z_{n,t} = \mathbb{I}[s_n \le t < s_n + l_n]$.

We learn a linear weight vector $\hat{\vec{v}}_k \in \mathbb{R}^{|\mathbf{X}|}$, whose entries are word relevances for k, i.e., $\phi_{\mathrm{limsse}}(t,k,\mathbf{X}) = \hat{v}_{k,t}$. To optimize it, we experiment with three loss functions. The first, which we will refer to as $\mathrm{limsse}^{\mathrm{bb}}$, assumes that our DNN is a total black box that delivers only a classification:

$$\hat{\vec{v}}_k = \underset{\vec{v}_k}{\mathrm{argmin}} \sum_n -\big[\log\big(\sigma(\vec{z}_n \cdot \vec{v}_k)\big)\mathbb{I}[f(\mathbf{Z}_n) = k] \\ + \log\big(1 - \sigma(\vec{z}_n \cdot \vec{v}_k)\big)\mathbb{I}[f(\mathbf{Z}_n) \neq k]\big]$$

where $f(\mathbf{Z}_n) = \mathrm{argmax}_{k'}\big(p(k'|\mathbf{Z}_n)\big)$. The black box approach is maximally general, but insensitive to the magnitude of evidence found in $\mathbf{Z}_n$. Hence, we also test magnitude-sensitive loss functions:

$$\hat{\vec{v}}_k = \underset{\vec{v}_k}{\mathrm{argmin}} \sum_n \big(\vec{z}_n \cdot \vec{v}_k - o(k,\mathbf{Z}_n)\big)^2$$

where $o(k,\mathbf{Z}_n)$ is one of $s(k,\mathbf{Z}_n)$ or $p(k|\mathbf{Z}_n)$. We refer to these as $\mathrm{limsse}^{\mathrm{ms}}_s$ and $\mathrm{limsse}^{\mathrm{ms}}_p$.

4 Experiments

4.1 Hybrid document experiment

For the hybrid document experiment, we use the 20 newsgroups corpus (topic classification) (Lang, 1995) and reviews from the 10th yelp dataset challenge (binary sentiment analysis)[3]. We train five DNNs per corpus: a bidirectional GRU (Cho et al., 2014), a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), a 1D CNN with global max pooling (Collobert et al., 2011), a bidirectional Quasi-GRU (QGRU), and a bidirectional Quasi-LSTM (QLSTM). The Quasi-RNNs are 1D CNNs with a feature-wise gated recursive pooling layer (Bradbury et al., 2017). Word embeddings are $\mathbb{R}^{300}$ and initialized with pre-trained GloVe embeddings (Pennington et al., 2014)[4]. The main layer has a hidden size of 150 (bidirectional architectures: 75 dimensions per direction). For the QRNNs and CNN, we use a kernel width of 5. In all five architectures, the resulting document representation is projected to 20 (resp. two) dimensions using a fully connected layer, followed by a softmax. See supplementary material for details on training and regularization.

After training, we sentence-tokenize the test sets, shuffle the sentences, concatenate ten sentences at a time and classify the resulting hybrid documents. Documents that are assigned a class that is not the gold label of at least one constituent word are discarded (yelp: $< 0.1\%$; 20 newsgroups: 14% - 20%). On the remaining documents, we use the explanation methods from §3 to find the maximally relevant word for each prediction. The random baseline samples the maximally relevant word from a uniform distribution.

For reference, we also evaluate on a **human judgment** benchmark (Mohseni and Ragan (2018), Table 2, C11-C15). It contains

[3]`www.yelp.com/dataset_challenge`
[4]`http://nlp.stanford.edu/data/glove.840B.300d.zip`

column	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27
	hybrid document experiment										man. groundtruth					morphosyntactic agreement experiment											
																$\text{hit}_{\text{target}}$				hit_{feat}							
	yelp					20 newsgroups					20 newsgroups					$f(\mathbf{X}) = y(\mathbf{X})$								$f(\mathbf{X}) \neq y(\mathbf{X})$			
ϕ	GRU	QGRU	LSTM	QLSTM	CNN	GRU	QGRU	LSTM	QLSTM	CNN	GRU	QGRU	LSTM	QLSTM	CNN	GRU	QGRU	LSTM	QLSTM	GRU	QGRU	LSTM	QLSTM	GRU	QGRU	LSTM	QLSTM
$\text{grad}^{\text{L2}}_{1s}$	.61	.68	.67	.70	.68	.45	.47	.25	.33	.79	.26	.31	.07	.18	.74	.48	.23	.63	.19	.52	.27	.73	.22	.09	.11	.19	.19
$\text{grad}^{\text{L2}}_{1p}$	.57	.67	.67	.70	.74	.40	.43	.26	.34	.70	.18	.35	.07	.13	.66	.48	.22	.63	.18	.53	.26	.73	.21	.09	.09	.18	.11
$\text{grad}^{\text{L2}}_{\int s}$	.71	.66	.69	.71	.70	.58	.32	.26	.21	.82	.23	.15	.11	.08	.76	.69	.67	.68	.51	.73	.70	.75	.55	.19	.22	.20	.20
$\text{grad}^{\text{L2}}_{\int p}$	.71	.70	.72	.71	.77	.56	.34	.30	.23	.81	.13	.08	.14	.01	$\underline{.78}$	.68	.77	.50	.70	.74	.82	.54	.78	.19	.21	.19	.30
$\text{grad}^{\text{dot}}_{1s}$	.88	.85	.81	.77	.86	.79	.76	.59	.72	$\underline{.89}$	$\underline{.80}$	.70	.14	.47	$\underline{.79}$	.81	.62	.73	.56	.85	.66	.81	.59	.42	.34	.46	.36
$\text{grad}^{\text{dot}}_{1p}$	$\underline{.92}$	.88	.84	.79	$\underline{.95}$	.78	.72	.59	.72	.81	.71	.59	.20	.44	.69	.79	.58	.74	.54	.83	.61	.81	.56	.41	.33	.46	.35
$\text{grad}^{\text{dot}}_{\int s}$	.84	$\underline{.90}$	.85	.87	.87	$\underline{.81}$	.68	.60	.68	$\underline{.89}$	$\underline{.82}$	.64	.21	.26	$\underline{.80}$	$\underline{.90}$	$\underline{.87}$	.78	.84	$\underline{.94}$	$\underline{.92}$	.83	.89	$\underline{.54}$	.51	.46	.52
$\text{grad}^{\text{dot}}_{\int p}$	.86	$\underline{.89}$	.84	$\underline{.89}$	$\underline{\textbf{.96}}$	$\underline{.80}$	.69	.62	.73	$\underline{.89}$	$\underline{.80}$	.53	.40	.54	$\underline{.78}$	$\underline{.87}$	$\underline{.85}$	.68	.84	$\underline{.93}$	$\underline{.92}$	.74	$\underline{.93}$	.53	.48	.42	.51
omit_1	.79	.82	.85	.87	.61	.78	.75	.54	.76	.82	$\underline{.80}$	.48	.33	.48	.65	.81	.81	.79	.80	.86	.87	.86	.84	.43	.45	.44	.45
omit_3	$\underline{.89}$	.80	$\underline{.89}$	$\underline{.88}$	.59	.79	.71	.72	$\underline{.81}$	.76	.77	.37	.36	.49	.61	.74	.77	.73	.73	.82	.84	.82	.79	.41	.45	.42	.46
omit_7	$\underline{.92}$	.88	$\underline{.91}$	$\underline{.91}$	.70	.79	.77	.77	$\underline{.84}$	.84	.77	.49	.44	.55	.65	.76	.80	.66	.74	.85	.88	.78	.80	.40	.48	.43	.47
occ_1	.80	.71	.74	.84	.61	.78	.73	.60	.77	.82	.77	.49	.19	.10	.65	$\underline{\textbf{.91}}$	$\underline{.85}$	$\underline{\textbf{.86}}$	$\underline{.86}$	$\underline{.94}$	.88	$\underline{.89}$	.88	.50	.44	.46	.47
occ_3	$\underline{.92}$	.61	$\underline{.93}$	.85	.59	.78	.63	.74	.74	.76	.74	.37	.32	.35	.61	.74	.73	.71	.72	.78	.76	.76	.76	.43	.37	.41	.43
occ_7	$\underline{.92}$	.77	$\underline{.93}$	$\underline{.90}$	.70	.78	.62	.74	.77	.84	.74	.35	.43	.39	.65	.64	.65	.63	.65	.73	.73	.72	.73	.36	.35	.39	.43
decomp	.79	.88	$\underline{.92}$	$\underline{.88}$	-	.75	.79	.77	.80	-	.54	.36	$\underline{.72}$	.51	-	.84	$\underline{.87}$	$\underline{\textbf{.86}}$	$\underline{.90}$	$\underline{.90}$	$\underline{.93}$	$\underline{\textbf{.92}}$	$\underline{\textbf{.96}}$	.52	$\underline{.58}$	$\underline{\textbf{.57}}$	$\underline{\textbf{.63}}$
lrp	$\underline{.92}$	.87	$\underline{.91}$	.84	.86	$\underline{.82}$	$\underline{.83}$	$\underline{.79}$	$\underline{.85}$	$\underline{.89}$	$\underline{\textbf{.85}}$	.72	$\underline{.74}$	$\underline{.81}$	$\underline{.79}$	$\underline{.90}$	$\underline{\textbf{.90}}$	$\underline{\textbf{.86}}$	$\underline{\textbf{.91}}$	$\underline{\textbf{.95}}$	$\underline{\textbf{.95}}$	$\underline{.91}$	$\underline{.95}$	$\underline{.58}$	$\underline{\textbf{.60}}$	$\underline{.52}$	$\underline{\textbf{.63}}$
deeplift	$\underline{.91}$	$\underline{.89}$	$\underline{\textbf{.94}}$	.85	.87	$\underline{.82}$	$\underline{.83}$	$\underline{.78}$	$\underline{.84}$	$\underline{.89}$	$\underline{.84}$	.72	.70	$\underline{.81}$	$\underline{.80}$	$\underline{\textbf{.91}}$	$\underline{\textbf{.90}}$	$\underline{.85}$	$\underline{\textbf{.91}}$	$\underline{\textbf{.95}}$	$\underline{\textbf{.95}}$	$\underline{.90}$	$\underline{.95}$	$\underline{\textbf{.59}}$	$\underline{.59}$	$\underline{.52}$	$\underline{\textbf{.63}}$
$\text{limsse}^{\text{bb}}$	.81	.82	.83	.84	.78	.78	.81	$\underline{.78}$	.80	.84	.52	.53	.53	.54	.57	.43	.41	.44	.42	.54	.51	.56	.52	.39	.43	.42	.41
$\text{limsse}^{\text{ms}}_{s}$	$\underline{\textbf{.94}}$	$\underline{\textbf{.94}}$	$\underline{.93}$	$\underline{\textbf{.93}}$	$\underline{.91}$	$\underline{\textbf{.85}}$	$\underline{\textbf{.87}}$	$\underline{\textbf{.83}}$	$\underline{\textbf{.86}}$	$\underline{.89}$	$\underline{\textbf{.85}}$	$\underline{\textbf{.84}}$	$\underline{\textbf{.76}}$	$\underline{\textbf{.84}}$	$\underline{\textbf{.82}}$	.62	.62	.67	.63	.75	.74	.82	.75	.52	.53	$\underline{.55}$	.53
$\text{limsse}^{\text{ms}}_{p}$	.87	.88	.85	.86	$\underline{.94}$	$\underline{\textbf{.85}}$	$\underline{.86}$	$\underline{\textbf{.83}}$	$\underline{\textbf{.86}}$	$\underline{\textbf{.90}}$	$\underline{.81}$	$\underline{.80}$	$\underline{.74}$	.76	.76	.62	.62	.67	.63	.75	.74	.82	.75	.51	.53	$\underline{.55}$	.53
random	.69	.67	.70	.69	.66	.20	.19	.22	.22	.21	.09	.09	.06	.06	.08	.27	.27	.27	.27	.33	.33	.33	.33	.12	.13	.12	.12
last	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	.66	.67	.66	.67	.76	.77	.76	.77	.21	.27	.25	.26
N	$7551 \leq N \leq 7554$					$3022 \leq N \leq 3230$					$137 \leq N \leq 150$					$N \approx 1400000$								$N \approx 20000$			

Table 2: Pointing game accuracies in hybrid document experiment (left), on manually annotated benchmark (middle) and in morphosyntactic agreement experiment (right). $\text{hit}_{\text{target}}$ (resp. hit_{feat}): maximal relevance on subject (resp. on noun with the predicted number feature). Bold: top explanation method. Underlined: within 5 points of top explanation method.

188 documents from the 20 newsgroups test set (classes sci.med and sci.electronics), with one manually created list of relevant words per document. We discard documents that are incorrectly classified (20% - 27%) and define: $\text{hit}(\phi, \mathbf{X}) = \mathbb{I}[\text{rmax}(\mathbf{X}, \phi) \in \text{gt}(\mathbf{X})]$, where $\text{gt}(\mathbf{X})$ is the manual ground truth.

4.2 Morphosyntactic agreement experiment

For the morphosyntactic agreement experiment, we use automatically annotated English Wikipedia sentences by Linzen et al. (2016)[5]. For our purpose, a sample consists of: all words preceding the verb: $\mathbf{X} = [x_1 \cdots x_T]$; part-of-speech (POS) tags: $\text{pos}(\mathbf{X}, t) \in \{\text{VBZ, VBP, NN, NNS}, \ldots\}$; and the position of the subject: $\text{target}(\mathbf{X}) \in [1, T]$. The number feature is derived from the POS:

$$\text{feat}(\mathbf{X}, t) = \begin{cases} \text{Sg} & \text{if } \text{pos}(\mathbf{X}, t) \in \{\text{VBZ, NN}\} \\ \text{Pl} & \text{if } \text{pos}(\mathbf{X}, t) \in \{\text{VBP, NNS}\} \\ \text{n/a} & \text{otherwise} \end{cases}$$

The gold label of a sentence is the number of its verb, i.e., $y(\mathbf{X}) = \text{feat}(\mathbf{X}, T + 1)$.

As task methods, we replicate Linzen et al. (2016)'s unidirectional LSTM ($\mathbb{R}^{50}$ randomly initialized word embeddings, hidden size 50). We also train unidirectional GRU, QGRU and QLSTM architectures with the same dimensionality. We use the explanation methods from §3 to find the most relevant word for predictions on the test set. As described in §2.2, explanation methods are awarded a $\text{hit}_{\text{target}}$ (resp. hit_{feat}) point if this word is the subject (resp. a noun with the predicted number feature). For reference, we use a random baseline as well as a baseline that assumes that the most relevant word directly precedes the verb.

5 Discussion

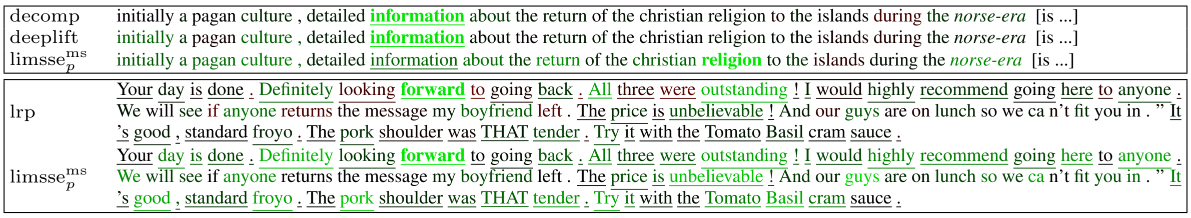
5.1 Explanation methods

Our experiments suggest that explanation methods for neural NLP differ in quality.

As in previous work (see §6), **gradient L2 norm** (grad^{L2}) performs poorly, especially on RNNs. We assume that this is due to its inability to distinguish relevances for and against k.

Gradient embedding dot product (grad^{dot}) is competitive on CNN (Table 2, $\text{grad}^{\text{dot}}_{1p}$ C05, $\text{grad}^{\text{dot}}_{1s}$ C10, C15), presumably because relu is linear on positive inputs, so gradients are exact in-

[5] www.tallinzen.net/media/rnn_agreement/agr_50_mostcommon_10K.tsv.gz



decomp	initially a pagan culture , detailed **information** about the return of the christian religion to the islands during the *norse-era* [is ...]
deeplift	initially a pagan culture , detailed **information** about the return of the christian religion to the islands during the *norse-era* [is ...]
$\text{limsse}_p^{\text{ms}}$	initially a pagan culture , detailed information about the return of the christian **religion** to the islands during the *norse-era* [is ...]

lrp	Your day is done . Definitely looking **forward** to going back . All three were outstanding ! I would highly recommend going here to anyone . We will see if anyone returns the message my boyfriend left . The price is unbelievable ! And our guys are on lunch so we ca n't fit you in . " It 's good , standard froyo . The pork shoulder was THAT tender . Try it with the Tomato Basil cram sauce .
$\text{limsse}_p^{\text{ms}}$	Your day is done . Definitely looking **forward** to going back . All three were outstanding ! I would highly recommend going here to anyone . We will see if anyone returns the message my boyfriend left . The price is unbelievable ! And our guys are on lunch so we ca n't fit you in . " It 's good , standard froyo . The pork shoulder was THAT tender . Try it with the Tomato Basil cram sauce .

Figure 3: **Top:** verb context classified singular. Task method: LSTM. **Bottom:** hybrid yelp review, classified positive. Task method: QLSTM.

stead of approximate. grad^{dot} also has decent performance for GRU ($\text{grad}^{\text{dot}}_{1p}$ C01, $\text{grad}^{\text{dot}}_{\int s}$ C{06, 11, 16, 20, 24}), perhaps because GRU hidden activations are always in [-1,1], where $\tanh$ and σ are approximately linear.

Integrated gradient ($\text{grad}_{\int}$) mostly outperforms simple gradient (grad_1), though not consistently (C01, C07). Contrary to expectation, integration did not help much with the failure of the gradient method on LSTM on 20 newsgroups ($\text{grad}^{\text{dot}}_1$ vs. $\text{grad}^{\text{dot}}_{\int}$ in C08, C13), which we had assumed to be due to saturation of $\tanh$ on large absolute activations in $\vec{c}$. Smaller intervals may be needed to approximate the integration, however, this means additional computational cost.

The gradient of $s(k, \mathbf{X})$ performs better or similar to the gradient of $p(k|\mathbf{X})$. The main exception is yelp ($\text{grad}^{\text{dot}}_{1s}$ vs. $\text{grad}^{\text{dot}}_{1p}$, C01-C05). This is probably due to conflation by $p(k|\mathbf{X})$ of evidence for k (numerator in Eq 3) and against competitor classes (denominator). In a two-class scenario, there is little incentive to keep classes separate, leading to information flow through the denominator. In future work, we will replace the two-way softmax with a one-way sigmoid such that $\phi(t, 0, \mathbf{X}) := -\phi(t, 1, \mathbf{X})$.

LRP and **DeepLIFT** are the most consistent explanation methods across evaluation paradigms and task methods. (The comparatively low pointing game accuracies on the yelp QRNNs and CNN (C02, C04, C05) are probably due to the fact that they explain $s(k, .)$ in a two-way softmax, see above.) On CNN (C05, C10, C15), LRP and $\text{grad}^{\text{dot}}_{1s}$ perform almost identically, suggesting that they are indeed quasi-equivalent on this architecture (see §3.2). On (Q)RNNs, modified LRP and DeepLIFT appear to be superior to the gradient method (lrp vs. $\text{grad}^{\text{dot}}_{1s}$, deeplift vs. $\text{grad}^{\text{dot}}_{\int s}$, C01-C04, C06-C09, C11-C14, C16-C27).

Decomposition performs well on LSTM, especially in the morphosyntactic agreement experiment, but it is inconsistent on other architectures. Gated RNNs have a long-term additive and a multiplicative pathway, and the decomposition method only detects information traveling via the additive one. Miao et al. (2016) show qualitatively that GRUs often reorganize long-term memory abruptly, which might explain the difference between LSTM and GRU. QRNNs only have additive recurrent connections; however, given that $\vec{c}_t$ (resp. $\vec{h}_t$) is calculated by convolution over several time steps, decomposition relevance can be incorrectly attributed inside that window. This likely is the reason for the stark difference between the performance of decomposition on QRNNs in the hybrid document experiment and on the manually labeled data (C07, C09 vs. C12, C14). Overall, we do not recommend the decomposition method, because it fails to take into account all routes by which information can be propagated.

Omission and occlusion produce inconsistent results in the hybrid document experiment. Shrikumar et al. (2017) show that perturbation methods can lack sensitivity when there are more relevant inputs than the "perturbation window" covers. In the morphosyntactic agreement experiment, omission is not competitive; we assume that this is because it interferes too much with syntactic structure. occ_1 does better (esp. C16-C19), possibly because an all-zero "placeholder" is less disruptive than word removal. But despite some high scores, it is less consistent than other explanation methods.

Magnitude-sensitive **LIMSSE** ($\text{limsse}^{\text{ms}}$) consistently outperforms black-box LIMSSE ($\text{limsse}^{\text{bb}}$), which suggests that numerical outputs should be used for approximation where possible. In the hybrid document experiment, magnitude-sensitive LIMSSE outperforms the other explanation methods (exceptions: C03, C05). However, it fails in the morphosyntactic agreement experiment (C16-C27). In fact, we expect LIMSSE to be unsuited for *large context* problems, as it cannot discover dependencies whose range is bigger than a given text sample. In Fig 3 (top), $\mathrm{limsse}_p^{\mathrm{ms}}$ highlights *any* singular noun without taking into account how that noun fits into the overall syntactic structure.

5.2 Evaluation paradigms

The assumptions made by our automatic evaluation paradigms have exceptions: (i) the correlation between fragment of origin and relevance does not always hold (e.g., a positive review may contain negative fragments, and will almost certainly contain neutral fragments); (ii) in morphological prediction, we cannot always expect the subject to be the only predictor for number. In Fig 2 (bottom) for example, "few" is a reasonable clue for plural despite not being a noun. This imperfect ground truth means that absolute pointing game accuracies should be taken with a grain of salt; but we argue that this does not invalidate them for comparisons.

We also point out that there are characteristics of explanations that may be desirable but are not reflected by the pointing game. Consider Fig 3 (bottom). Both explanations get hit points, but the lrp explanation appears "cleaner" than $\mathrm{limsse}_p^{\mathrm{ms}}$, with relevance concentrated on fewer tokens.

6 Related work

6.1 Explanation methods

Explanation methods can be divided into local and global methods (Doshi-Velez and Kim, 2017). Global methods infer general statements about what a DNN has learned, e.g., by clustering documents (Aubakirova and Bansal, 2016) or n-grams (Kádár et al., 2017) according to the neurons that they activate. Li et al. (2016a) compare embeddings of specific words with reference points to measure how drastically they were changed during training. In computer vision, Simonyan et al. (2014) optimize the input space to maximize the activation of a specific neuron. Global explanation methods are of limited value for explaining a specific prediction as they represent average behavior. Therefore, we focus on local methods.

Local explanation methods explain a decision taken for one specific input at a time. We have attempted to include all important local methods for NLP in our experiments (see §3). We do not address self-explanatory models (e.g., attention (Bahdanau et al., 2015) or rationale models (Lei et al., 2016)), as these are very specific architectures that may not be not applicable to all tasks.

6.2 Explanation evaluation

According to Doshi-Velez and Kim (2017)'s taxonomy of explanation evaluation paradigms, *application-grounded* paradigms test how well an explanation method helps real users solve real tasks (e.g., doctors judge automatic diagnoses); *human-grounded* paradigms rely on proxy tasks (e.g., humans rank task methods based on explanations); *functionally-grounded* paradigms work without human input, like our approach.

Arras et al. (2016) (cf. Samek et al. (2016)) propose a functionally-grounded explanation evaluation paradigm for NLP where words in a correctly (resp. incorrectly) classified document are deleted in descending (resp. ascending) order of relevance. They assume that the fewer words must be deleted to reduce (resp. increase) accuracy, the better the explanations. According to this metric, LRP (§3.2) outperforms $\mathrm{grad}^{\mathrm{L2}}$ on CNNs (Arras et al., 2016) and LSTMs (Arras et al., 2017b) on 20 newsgroups. Ancona et al. (2017) perform the same experiment with a binary sentiment analysis LSTM. Their graph shows occ_1, $\mathrm{grad}_1^{\mathrm{dot}}$ and $\mathrm{grad}_\int^{\mathrm{dot}}$ tied in first place, while LRP, DeepLIFT and the gradient L1 norm lag behind. Note that their treatment of LSTM gates in LRP / DeepLIFT differs from our implementation.

An issue with the word deletion paradigm is that it uses syntactically broken inputs, which may introduce artefacts (Sundararajan et al., 2017). In our hybrid document paradigm, inputs are syntactically intact (though semantically incoherent at the document level); the morphosyntactic agreement paradigm uses unmodified inputs.

Another class of functionally-grounded evaluation paradigms interprets the performance of a secondary task method, on inputs that are derived from (or altered by) an explanation method, as a proxy for the quality of that explanation method. Murdoch and Szlam (2017) build a rule-based classifier from the most relevant phrases in a corpus (task method: LSTM). The classifier based on decomp (§3.4) outperforms the gradient-based classifier, which is in line with our results. Arras et al. (2017a) build document representations by summing over word embeddings weighted by relevance scores (task method: CNN). They show that K-nearest neighbor performs better on document representations derived with LRP than on those derived with grad^{L2}, which also matches our results. Denil et al. (2015) condense documents by extracting top-K relevant sentences, and let the original task method (CNN) classify them. The accuracy loss, relative to uncondensed documents, is smaller for grad^{dot} than for heuristic baselines.

In the domain of human-based evaluation paradigms, Ribeiro et al. (2016) compare different variants of LIME (§3.6) by how well they help non-experts clean a corpus from words that lead to overfitting. Selvaraju et al. (2017) assess how well explanation methods help non-experts identify the more accurate out of two object recognition CNNs. These experiments come closer to real use cases than functionally-grounded paradigms; however, they are less scalable.

7 Summary

We conducted the first comprehensive evaluation of explanation methods for NLP, an important undertaking because there is a need for understanding the behavior of DNNs.

To conduct this study, we introduced evaluation paradigms for explanation methods for two classes of NLP tasks, small context tasks (e.g., topic classification) and large context tasks (e.g., morphological prediction). Neither paradigm requires manual annotations. We also introduced LIMSSE, a substring-based explanation method inspired by LIME and designed for NLP.

Based on our experimental results, we recommend LRP, DeepLIFT and LIMSSE for small context tasks and LRP and DeepLIFT for large context tasks, on all five DNN architectures that we tested. On CNNs and possibly GRUs, the (integrated) gradient embedding dot product is a good alternative to DeepLIFT and LRP.

8 Code

Our implementation of LIMSSE, the gradient, perturbation and decomposition methods can be found in our branch of the `keras` package: `www.github.com/NPoe/keras`. To re-run our experiments, see scripts in `www.github.com/NPoe/neural-nlp-explanation-experiment`. Our LRP implementation (same repository) is adapted from Arras et al. (2017b)[6].

[6] `https://github.com/ArrasL/LRP_for_LSTM`

References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2017. A unified view of gradient-based attribution methods for deep neural networks. In *Conference on Neural Information Processing System*, Long Beach, USA.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, Vancouver, Canada.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *First Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017a. What is relevant in a text document?: An interpretable machine learning approach. *PloS one*, 12(8):e0181142.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017b. Explaining recurrent neural network predictions in sentiment analysis. In *Eighth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168, Copenhagen, Denmark.

Malika Aubakirova and Mohit Bansal. 2016. Interpreting neural networks to improve politeness comprehension. In *Empirical Methods in Natural Language Processing*, page 2035–2041, Austin, USA.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, San Diego, USA.

Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task learning for deep text recommendations. In *ACM Conference on Recommender Systems*, pages 107–114, Boston, USA.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. In *International Conference on Learning Representations*, Toulon, France.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar.

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

Misha Denil, Alban Demiraj, and Nando de Freitas. 2015. Extraction of salient sentences from labelled documents. In *International Conference on Learning Representations*, San Diego, USA.

Finale Doshi-Velez and Been Kim. 2017. A roadmap for a rigorous science of interpretability. *CoRR*, abs/1702.08608.

Bryce Goodman and Seth Flaxman. 2016. European union regulations on algorithmic decision-making and a "right to explanation". In *ICML Workshop on Human Interpretability in Machine Learning*, pages 26–30, New York, USA.

Yotam Hechtlinger. 2016. Interpretation of prediction models using the input gradient. In *Conference on Neural Information Processing Systems*, Barcelona, Spain.

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

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. 2016. Investigating the influence of noise and distractors on the interpretation of neural networks. In *Conference on Neural Information Processing Systems*, Barcelona, Spain.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, Tahoe City, USA.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Empirical Methods in Natural Language Processing*, pages 107–117, Austin, USA.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. Visualizing and understanding neural models in NLP. In *NAACL-HLT*, pages 681–691, San Diego, USA.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Yajie Miao, Jinyu Li, Yongqiang Wang, Shi-Xiong Zhang, and Yifan Gong. 2016. Simplifying long short-term memory acoustic models for fast training and decoding. In *International Conference on Acoustics, Speech and Signal Processing*, pages 2284–2288.

Sina Mohseni and Eric D Ragan. 2018. A human-grounded evaluation benchmark for local explanations of machine learning. *CoRR*, abs/1801.05075.

W James Murdoch and Arthur Szlam. 2017. Automatic rule extraction from long short term memory networks. In *International Conference on Learning Representations*, Toulon, France.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, San Francisco, California.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 618–626, Honolulu, Hawaii.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153, Sydney, Australia.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*, Banff, Canada.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, Sydney, Australia.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833, Zürich, Switzerland.

Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. 2016. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559, Amsterdam, Netherlands.

Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. 2017. Visualizing deep neural network decisions: Prediction difference analysis. In *International Conference on Learning Representations*, Toulon, France.

Improving Text-to-SQL Evaluation Methodology

Catherine Finegan-Dollak[1*] **Jonathan K. Kummerfeld**[1*] **Li Zhang**[1]
Karthik Ramanathan[2] **Sesh Sadasivam**[1] **Rui Zhang**[3] **Dragomir Radev**[3]
Computer Science & Engineering[1] and School of Information[2] Department of Computer Science[3]
University of Michigan, Ann Arbor Yale University
{cfdollak,jkummerf}@umich.edu dragomir.radev@yale.edu

Abstract

To be informative, an evaluation must measure how well systems generalize to realistic unseen data. We identify limitations of and propose improvements to current evaluations of text-to-SQL systems. First, we compare human-generated and automatically generated questions, characterizing properties of queries necessary for real-world applications. To facilitate evaluation on multiple datasets, we release standardized and improved versions of seven existing datasets and one new text-to-SQL dataset. Second, we show that the current division of data into training and test sets measures robustness to variations in the way questions are asked, but only partially tests how well systems generalize to new queries; therefore, we propose a complementary dataset split for evaluation of future work. Finally, we demonstrate how the common practice of anonymizing variables during evaluation removes an important challenge of the task. Our observations highlight key difficulties, and our methodology enables effective measurement of future development.

1 Introduction

Effective natural language interfaces to databases (NLIDB) would give lay people access to vast amounts of data stored in relational databases. This paper identifies key oversights in current evaluation methodology for this task. In the process, we (1) introduce a new, challenging dataset, (2) standardize and fix many errors in existing datasets, and (3) propose a simple yet effective baseline system.[1]

[*] The first two authors contributed equally to this work.
[1] Code and data is available at https://github.com/jkkummerfeld/text2sql-data/

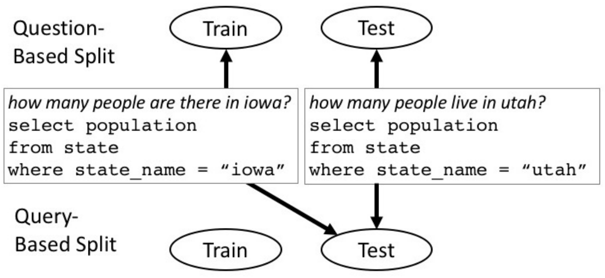

Question-Based Split
Train
Test
how many people are there in iowa?
select population
from state
where state_name = "iowa"
how many people live in utah?
select population
from state
where state_name = "utah"
Query-Based Split
Train
Test

Figure 1: Traditional question-based splits allow queries to appear in both train and test. Our query-based split ensures each query is in only one.

First, we consider query complexity, showing that human-written questions require more complex queries than automatically generated ones. To illustrate this challenge, we introduce *Advising*, a dataset of questions from university students about courses that lead to particularly complex queries.

Second, we identify an issue in the way examples are divided into training and test sets. The standard approach, shown at the top of Fig. 1, divides examples based on the text of each *question*. As a result, many of the queries in the test set are seen in training, albeit with different entity names and with the question phrased differently. This means metrics are mainly measuring robustness to the way a set of known SQL queries can be expressed in English—still a difficult problem, but not a complete test of ability to compose new queries in a familiar domain. We introduce a template-based slot-filling baseline that cannot generalize to new queries, and yet is competitive with prior work on multiple datasets. To measure robustness to new queries, we propose splitting based on the SQL *query*. We show that state-of-the-art systems with excellent performance on traditional question-based splits struggle on query-based splits. We also consider the common practice of variable anonymization, which removes a challenging form of ambiguity from the task. In the process, we apply extensive effort to standardize datasets and fix a range of errors.

Previous NLIDB work has led to impressive systems, but current evaluations provide an incomplete picture of their strengths and weaknesses. In this paper, we provide new and improved data, a new baseline, and guidelines that complement existing metrics, supporting future work.

2 Related Work

The task of generating SQL representations from English questions has been studied in the NLP and DB communities since the 1970s (Androutsopoulos et al., 1995). Our observations about evaluation methodology apply broadly to the systems cited below.

Within the DB community, systems commonly use pattern matching, grammar-based techniques, or intermediate representations of the query (Pazos Rangel et al., 2013). Recent work has explored incorporating user feedback to improve accuracy (Li and Jagadish, 2014). Unfortunately, none of these systems are publicly available, and many rely on domain-specific resources.

In the NLP community, there has been extensive work on semantic parsing to logical representations that query a knowledge base (Zettlemoyer and Collins, 2005; Liang et al., 2011; Beltagy et al., 2014; Berant and Liang, 2014), while work on mapping to SQL has recently increased (Yih et al., 2015; Iyer et al., 2017; Zhong et al., 2017). One of the earliest statistical models for mapping text to SQL was the PRECISE system (Popescu et al., 2003, 2004), which achieved high precision on queries that met constraints linking tokens and database values, attributes, and relations, but did not attempt to generate SQL for questions outside this class. Later work considered generating queries based on relations extracted by a syntactic parser (Giordani and Moschitti, 2012) and applying techniques from logical parsing research (Poon, 2013). However, none of these earlier systems are publicly available, and some required extensive engineering effort for each domain, such as the lexicon used by PRECISE.

More recent work has produced general purpose systems that are competitive with previous results and are also available, such as Iyer et al. (2017). We also adapt a logical form parser with a sequence to tree approach that makes very few assumptions about the output structure (Dong and Lapata, 2016).

One challenge for applying neural models to this task is annotating large enough datasets of question-query pairs. Recent work (Cai et al., 2017; Zhong et al., 2017) has automatically generated large datasets using templates to form random queries and corresponding natural-language-like questions, and then having humans rephrase the question into English. Another option is to use feedback-based learning, where the system alternates between training and making predictions, which a user rates as correct or not (Iyer et al., 2017). Other work seeks to avoid the data bottleneck by using end-to-end approaches (Yin et al., 2016; Neelakantan et al., 2017), which we do not consider here. One key contribution of this paper is standardization of a range of datasets, to help address the challenge of limited data resources.

3 Data

For our analysis, we study a range of text-to-SQL datasets, standardizing them to have a consistent SQL style.

ATIS (Price, 1990; Dahl et al., 1994) User questions for a flight-booking task, manually annotated. We use the modified SQL from Iyer et al. (2017), which follows the data split from the logical form version (Zettlemoyer and Collins, 2007).

GeoQuery (Zelle and Mooney, 1996) User questions about US geography, manually annotated with Prolog. We use the SQL version (Popescu et al., 2003; Giordani and Moschitti, 2012; Iyer et al., 2017), which follows the logical form data split (Zettlemoyer and Collins, 2005).

Restaurants (Tang and Mooney, 2000; Popescu et al., 2003) User questions about restaurants, their food types, and locations.

Scholar (Iyer et al., 2017) User questions about academic publications, with automatically generated SQL that was checked by asking the user if the output was correct.

Academic (Li and Jagadish, 2014) Questions about the Microsoft Academic Search (MAS) database, derived by enumerating every logical query that could be expressed using the search page of the MAS website and writing sentences to match them. The domain is similar to that of Scholar, but their schemas differ.

Yelp and IMDB (Yaghmazadeh et al., 2017) Questions about the Yelp website and the Internet Movie Database, collected from colleagues of the authors who knew the type of information in each database, but not their schemas.

WikiSQL (Zhong et al., 2017) A large collection of automatically generated questions about individual tables from Wikipedia, paraphrased by crowd workers to be fluent English.

Advising (This Work) Our dataset of questions over a database of course information at the University of Michigan, but with fictional student records. Some questions were collected from the EECS department Facebook page and others were written by CS students with knowledge of the database who were instructed to write questions they might ask in an academic advising appointment.

The authors manually labeled the initial set of questions with SQL. To ensure high quality, at least two annotators scored each question-query pair on a two-point scale for accuracy—did the query generate an accurate answer to the question?—and a three-point scale for helpfulness—did the answer provide the information the asker was probably seeking? Cases with low scores were fixed or removed from the dataset.

We collected paraphrases using Jiang et al. (2017)'s method, with manual inspection to ensure accuracy. For a given sentence, this produced paraphrases with the same named entities (e.g. course number EECS 123). To add variation, we annotated entities in the questions and queries with their types—such as course name, department, or instructor—and substituted randomly-selected values of each type into each paraphrase and its corresponding query. This combination of paraphrasing and entity replacement means an original question of "For next semester, who is teaching EECS 123?" can give rise to "Who teaches MATH 456 next semester?" as well as "Who's the professor for next semester's CHEM 789?"

3.1 SQL Canonicalization

SQL writing style varies. To enable consistent training and evaluation across datasets, we canonicalized the queries: (1) we alphabetically ordered fields in `SELECT`, tables in `FROM`, and constraints in `WHERE`; (2) we standardized table aliases in the form `<TABLE_NAME>alias<N>` for the `N`th use of the same table in one query; and (3) we standardized capitalization and spaces between symbols. We confirmed these changes do not alter the meaning of the queries via unit tests of the canonicalization code and manual inspection of the output. We also manually fixed some errors, such as ambiguous mixing of `AND` and `OR` (30 ATIS queries).

	Sets Identified	Affected Queries
ATIS	141	380
GeoQuery	17	39
Scholar	60	152

Table 1: Manually identified duplicate queries (different SQL for equivalent questions).

3.2 Variable Annotation

Existing SQL datasets do not explicitly identify which words in the question are used in the SQL query. Automatic methods to identify these variables, as used in prior work, do not account for ambiguities, such as words that could be either a city or an airport. To provide accurate anonymization, we annotated query variables using a combination of automatic and manual processing.

Our automatic process extracted terms from each side of comparison operations in SQL: one side contains quoted text or numbers, and the other provides a type for those literals. Often quoted text in the query is a direct copy from the question, while in some cases we constructed dictionaries to map common acronyms, like *american airlines*–`AA`, and times, like *2pm*–`1400`. The process flagged cases with ambiguous mappings, which we then manually processed. Often these were mistakes, which we corrected, such as missing constraints (e.g., *papers in 2015* with no date limit in the query), extra constraints (e.g., limiting to a single airline despite no mention in the question), inaccurate constraints (e.g., *more than 5* as `> 4`), and inconsistent use of *this year* to mean different years in different queries.

3.3 Query Deduplication

Three of the datasets had many duplicate queries (i.e., semantically equivalent questions with different SQL). To avoid this spurious ambiguity we manually grouped the data into sets of equivalent questions (Table 1). A second person manually inspected every set and ran the queries. Where multiple queries are valid, we kept them all, though only used the first for the rest of this work.

	Redundancy Measures						Complexity Measures							
	Question count	Unique query count	[1]/[2]	Queries / pattern μ	Queries / pattern Max	Pattern count	Tables / query μ	Tables / query Max	Unique tables / query μ	Unique tables / query Max	SELECTs / query μ	SELECTs / query Max	Nesting Depth μ	Nesting Depth Max
Advising	4570	211	21.7	20.3	90	174	3.2	9	3.0	9	1.23	6	1.18	4
ATIS	5280	947	5.6	7.0	870	751	6.4	32	3.8	12	1.79	8	1.39	8
GeoQuery	877	246	3.6	8.9	327	98	1.4	5	1.1	4	1.77	8	2.03	7
Restaurants	378	23	16.4	22.2	81	17	2.6	5	2.3	4	1.17	2	1.17	2
Scholar	817	193	4.2	5.6	71	146	3.3	6	3.2	6	1.02	2	1.02	2
Academic	196	185	1.1	2.1	12	92	3.2	10	3	6	1.04	3	1.04	2
IMDB	131	89	1.5	2.5	21	52	1.9	5	1.9	5	1.01	2	1.01	2
Yelp	128	110	1.2	1.4	11	89	2.2	4	2	4	1	1	1	1
WikiSQL	80,654	77,840	1.0	165.3	42,816	488	1	1	1	1	1	1	1	1

Table 2: Descriptive statistics for text-to-SQL datasets. Datasets in the first group are human-generated from the NLP community, in the second are human-generated from the DB community, and in the third are automatically-generated. [1]/[2] is Question count / Unique query count.

4 Evaluating on Multiple Datasets Is Necessary

For evaluation to be informative it must use data that is representative of real-world queries. If datasets have biases, robust comparisons of models will require evaluation on multiple datasets. For example, some datasets, such as ATIS and Advising, were collected from users and are task-oriented, while others, such as WikiSQL, were produced by automatically generating queries and engaging people to express the query in language. If these two types of datasets differ systematically, evaluation on one may not reflect performance on the other. In this section, we provide descriptive statistics aimed at understanding how several datasets differ, especially with respect to query redundancy and complexity.

4.1 Measures

We consider a range of measures that capture different aspects of data complexity and diversity:

Question / Unique Query Counts We measure dataset size and how many distinct queries there are when variables are anonymized. We also present the mean number of questions per unique query; a larger mean indicates greater redundancy.

SQL Patterns Complexity can be described as the answer to the question, "How many query-form patterns would be required to generate this dataset?" Fig. 2 shows an example of a pattern, which essentially abstracts away from the specific table and field names. Some datasets were generated from patterns similar to these, including WikiSQL and Cai et al. (2017). This enables the generation of large numbers of queries, but limits the variation between them to only that encompassed by their patterns. We count the number of patterns needed to cover the full dataset, where larger numbers indicate greater diversity. We also report mean queries per pattern; here, larger numbers indicate greater redundancy, showing that many queries fit the same mold.

```
SELECT <table-alias>.<field>
FROM <table> AS <table-alias>
WHERE <table-alias>.<field> = <literal>

SELECT RIVERalias0.RIVER_NAME
FROM RIVER AS RIVERalias0
WHERE RIVERalias0.TRAVERSE = "florida";

SELECT CITYalias0.CITY_NAME
FROM CITY AS CITYalias0
WHERE CITYalias0.STATE_NAME = "alabama";
```

Figure 2: An SQL pattern and example queries.

Counting Tables We consider the total number of tables and the number of unique tables mentioned in a query. These numbers differ in the event of self-joins. In both cases, higher values imply greater complexity.

Nesting A query with nested subqueries may be more complex than one without nesting. We count SELECT statements within each query to determine the number of sub-queries. We also report the depth of query nesting. In both cases, higher values imply greater complexity.

4.2 Analysis

The statistics in Table 2 show several patterns.

First, dataset size is not the best indicator of dataset diversity. Although WikiSQL contains fifteen times as many question-query pairs as ATIS, ATIS contains significantly more patterns than

WikiSQL; moreover, WikiSQL's queries are dominated by one pattern that is more than half of the dataset (`SELECT col AS result FROM table WHERE col = value`). The small, hand-curated datasets developed by the database community—Academic, IMDB, and Yelp—have noticeably less redundancy as measured by questions per unique query and queries per pattern than the datasets the NLP community typically evaluates on.

Second, human-generated datasets exhibit greater complexity than automatically generated data. All of the human-generated datasets except Yelp demonstrate at least some nesting. The average query from any of the human-generated datasets joins more than one table.

In particular, task-oriented datasets require joins and nesting. ATIS and Advising, which were developed with air-travel and student-advising tasks in mind, respectively, both score in the top three for multiple complexity scores.

To accurately predict performance on human-generated or task-oriented questions, it is thus necessary to evaluate on datasets that test the ability to handle nesting and joins. Training and testing NLP systems, particularly deep learning-based methods, benefits from large datasets. However, at present, the largest dataset available does not provide the desired complexity.

Takeaway: Evaluate on multiple datasets, some with nesting and joins, to provide a thorough picture of a system's strengths and weaknesses.

5 Current Data Splits Only Partially Probe Generalizability

It is standard best practice in machine learning to divide data into disjoint training, development, and test sets. Otherwise, evaluation on the test set will not accurately measure how well a model generalizes to new examples. The standard splits of GeoQuery, ATIS, and Scholar treat each pair of a natural language question and its SQL query as a single item. Thus, as long as each question-query pair appears in only one set, the test set is not tainted with training data. We call this a question-based data split.

However, many English questions may correspond to the same SQL query. If at least one copy of every SQL query appears in training, then the task evaluated is classification, not true semantic parsing, of the English questions. We can increase the number of distinct SQL queries by varying what entities our questions ask about; the queries for *what states border Texas* and *what states border Massachusetts* are not identical. Adding this variation changes the task from pure classification to classification plus slot-filling. Does this provide a true evaluation of the trained model's performance on unseen inputs?

It depends on what we wish to evaluate. If we want a system that answers questions within a particular domain, and we have a dataset that we are confident covers everything a user might want to know about that domain, then evaluating on the traditional question-based split tells us whether the system is robust to variation in how a request is expressed. But compositionality is an essential part of language, and a system that has trained on *What courses does Professor Smith teach?* and *What courses meet on Fridays?* should be prepared for *What courses that Professor Smith teaches meet on Fridays?* Evaluation on the question split does not tell us about a model's generalizable knowledge of SQL, or even its generalizable knowledge within the present domain.

To evaluate the latter, we propose a complementary new division, where no SQL query is allowed to appear in more than one set; we call this the *query split*. To generate a query split, we substitute variables for entities in each query in the dataset, as described in § 3.2. Queries that are identical when thus anonymized are treated as a single query and randomly assigned—with all their accompanying questions—to train, dev, or test. We include the original question split and the new query split labeling for the new Advising dataset, as well as ATIS, GeoQuery, and Scholar. For the much smaller Academic, IMDB, Restaurant, and Yelp datasets, we include question- and query- based buckets for cross validation.

5.1 Systems

Recently, a great deal of work has used variations on the seq2seq model. We compare performance of a basic seq2seq model (Sutskever et al., 2014), and seq2seq with attention over the input (Bahdanau et al., 2015), implemented with TensorFlow seq2seq (Britz et al., 2017). We also extend that model to include an attention-based copying option, similar to Jia and Liang (2016). Our output vocabulary for the decoder includes a special token, `COPY`. If `COPY` has the highest probability at step t, we replace it with the input token with the

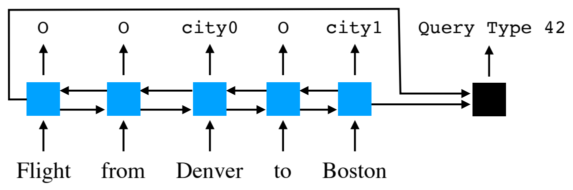


Figure 3: Baseline: blue boxes are LSTM cells and the black box is a feed-forward network. Outputs are the query template to use (right) and which tokens to fill it with (left).

max of the normalized attention scores. Our loss function is the sum of two terms: first, the categorical cross entropy for the model's probability distribution over the output vocabulary tokens; and second, the loss for word copying. When the correct output token is `COPY`, the second loss term is the categorical cross entropy of the distribution of attention scores at time t. Otherwise it is zero.

For comparison, we include systems from two recent papers. Dong and Lapata (2016) used an attention-based seq2tree model for semantic parsing of logical forms; we apply their code here to SQL datasets. Iyer et al. (2017) use a seq2seq model with automatic dataset expansion through paraphrasing and SQL templates.[2]

We could not find publicly available code for the non-neural text-to-SQL systems discussed in Section 2. Also, most of those approaches require development of specialized grammars or templates for each new dataset they are applied to, so we do not compare such systems.

5.2 New Template Baseline

In addition to the seq2seq models, we develop a new baseline system for text-to-SQL parsing which exploits repetitiveness in data. First, we automatically generate SQL templates from the training set. The system then makes two predictions: (1) which template to use, and (2) which words in the sentence should fill slots in the template. This system is not able to generalize beyond the queries in the training set, so it will fail completely on the new query-split data setting.

Fig. 3 presents the overall architecture, which we implemented in DyNet (Neubig et al., 2017). A bidirectional LSTM provides a prediction for each word, either `O` if the word is not used in the final query, or a symbol such as `city1` to indicate that it fills a slot. The hidden states of the LSTM at each end of the sentence are passed through a small feed-forward network to determine the SQL template to use. This architecture is simple and enables a joint choice of the tags and the template, though we do not explicitly enforce agreement.

To train the model, we automatically construct a set of templates and slots. Slots are determined based on the variables in the dataset, with each SQL variable that is explicitly given in the question becoming a slot. We can construct these templates because our new version of the data explicitly defines all variables, their values, and where they appear in both question and query.

For completeness, we also report on an oracle version of the template-based system (performance if it always chose the correct template from the train set and filled all slots correctly).

5.3 Oracle Entity Condition

Some systems, such as Dong and Lapata's model, are explicitly designed to work on anonymized data (i.e., data where entity names are replaced with a variable indicating their type). Others, such as attention-based copying models, treat identification of entities as an inextricable component of the text-to-SQL task. We therefore describe results on both the actual datasets with entities in place and a version anonymized using the variables described in § 3.2. We refer to the latter as the oracle entity condition.

5.4 Results and Analysis

We hypothesized that even a system unable to generalize can achieve good performance on question-based splits of datasets, and the results in Table 3 substantiate that for the NLP community's datasets. The template-based, slot-filling baseline was competitive with state-of-the-art systems for question split on the four datasets from the NLP community. The template-based oracle performance indicates that for these datasets anywhere from 70-100% accuracy on question-based split could be obtained by selecting a template from the training set and filling in the right slots.

For the three datasets developed by the databases community, the effect of question-query split is far less pronounced. The small sizes of these datasets cannot account for the difference,

[2] We enable Iyer et al. (2017)'s paraphrasing data augmentation, but not their template-based augmentation because templates do not exist for most of the datasets (though they also found it did not significantly improve performance). Note, on ATIS and Geo their evaluation assumed no ambiguity in entity identification, which is equivalent to our Oracle Entities condition (§5.3).

	Advising		ATIS		GeoQuery		Restaurants		Scholar		Academic		IMDB		Yelp	
Model	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q
No Variable Anonymization																
Baseline	**80**	0	46	0	57	0	95	0	52	0	0	0	0	0	1	0
seq2seq	6	0	8	0	27	7	47	0	19	0	6	7	1	0	0	0
+ Attention	29	0	46	18	63	21	**100**	2	33	0	71	64	7	3	2	2
+ Copying	70	0	**51**	**32**	**71**	20	**100**	4	**59**	5	**81**	**74**	**26**	**9**	**12**	4
D&L seq2tree	46	**2**	46	23	62	31	**100**	**11**	44	**6**	63	54	6	2	1	2
Iyer et al.	41	1	45	17	66	**40**	**100**	8	44	3	76	70	10	4	6	**6**
With Oracle Entities																
Baseline	89	0	56	0	56	0	95	0	66	0	0	0	7	0	8	0
seq2seq	21	0	14	0	49	14	71	6	23	0	10	9	6	0	12	9
+ Attention	88	0	57	23	73	31	100	32	71	4	77	74	44	17	33	28
D&L seq2tree	88	8	56	34	68	23	100	21	68	6	65	61	36	10	26	23
Iyer et al.	88	6	58	32	71	49	100	33	71	1	77	75	52	24	44	32
Baseline-Oracle	100	0	69	0	78	0	100	0	84	0	11	0	47	0	25	0

Table 3: Accuracy of neural text-to-SQL systems on English question splits ('?' columns) and SQL query splits ('Q' columns). The vertical line separates datasets from the NLP (left) and DB (right) communities. Results for Iyer et al. (2017) are slightly lower here than in the original paper because we evaluate on SQL output, not the database response.

since even the oracle baseline did not have much success on these question splits, and since the baseline was able to handle the small Restaurants dataset. Looking back at Section 4, however, we see that these are the datasets with the least redundancy in Table 2. Because their question:unique-query ratios are nearly 1:1, the question splits and query splits of these datasets were quite similar.

Reducing redundancy does not improve performance on query split, though; at most, it reduces the difference between performance on the two splits. IMDB and Yelp both show weak results on query split despite their low redundancy. Experiments on a non-redundant version of query split for Advising, ATIS, GeoQuery, and Restaurant that contained only one question for each query confirmed this: in each case, accuracy remained the same or declined relative to regular query split.

Having ruled out redundancy as a cause for the exceptional performance on Academic's query split, we suspect the simplicity of its questions and the compositionality of its queries may be responsible. Every question in the dataset begins *return me* followed by a phrase indicating the desired field, optionally followed by one or more constraints; for instance, *return me the papers by 'author_name0'* and *return me the papers by 'author_name0' on journal_name0.*

None of this, of course, is to suggest that question-based split is an easy problem, even on the NLP community's datasets. Except for the Advising and Restaurants datasets, even the oracle version of the template-based system is far from perfect. Access to oracle entities helps performance of non-copying systems substantially, as we would expect. Entity matching is thus a non-trivial component of the task.

But the query-based split is certainly more difficult than the question-based split. Across datasets and systems, performance suffered on query split. Access to oracle entities did not remove this effect.

Many of the seq2seq models do show some ability to generalize, though. Unlike the template-based baseline, many were able to eek out some performance on query split.

On question split, ATIS is the most difficult of the NLP datasets, yet on query split, it is among the easiest. To understand this apparent contradiction, we must consider what kinds of mistakes systems make and the contexts in which they appear. We therefore analyze the output of the attention-based-copying model in greater detail.

We categorize each output as shown in column one of Table 4. The "Correct" category is self-explanatory. "Entity problem only" means that the query would have been correct but for a mistake in one or more entity names. "Different template" means that the system output was the same as another query from the dataset but for the entity names; however, it did not match the correct query for this question. "No template match" contains both the most mundane and the most interesting errors. Here, the system output a query that is not copied from training data. Sometimes, this is a simple error, such as inserting an extra comma in the `WHERE` clause. Other times, it is recombining

		Advising		ATIS		GeoQuery		Scholar	
		Question	Query	Question	Query	Question	Query	Question	Query
Correct	Count	369	5	227	111	191	56	129	17
	μ Length	83.8	165.8	55.1	69.2	19.6	21.5	38.0	30.2
Entity problem	Count	10	0	1	6	5	0	5	0
	μ Length	111.8	N/A	28.0	71.3	17.2	N/A	42.6	N/A
Different template	Count	43	675	94	68	53	84	40	94
	μ Length	69.8	68.4	85.8	72.1	25.6	18.0	43.9	39.8
No template match	Count	79	25	122	162	30	42	44	204
	μ Length	88.8	90.5	113.8	92.2	29.7	25.0	42.1	41.6

Table 4: Types of errors by the attention-based copying model for question and query splits, with (Count)s of queries in each category, and the (μ Length) of gold queries in the category.

segments of queries it has seen into new queries. This is necessary but not sufficient model behavior in order to do well on the query split. In at least one case, this category includes a semantically equivalent query marked as incorrect by the exact-match accuracy metric.[3] Table 4 shows the number of examples from the test set that fell into each category, as well as the mean length of gold queries ("length") for each category.

Short queries are easier than long ones in the question-based condition. In most cases, length in "correct" is shorter than length in either "different template" or "no template match" categories.

In addition, for short queries, the model seems to prefer to copy a query it has seen before; for longer ones, it generates a new query. In every case but one, mean length in "different template" is less than in "No template match."

Interestingly, in ATIS and GeoQuery, where the model performs tolerably well on query split, the length for correct queries in query split is higher than the length for correct queries from the question split. Since, as noted above, recombination of template pieces (as we see in "no template match") is a necessary step for success on query split, it may be that longer queries have a higher probability of recombination, and therefore a better chance of being correct in query split. The data from Scholar does not support this position; however, note that only 17 queries were correct in Scholar query split, suggesting caution in making generalizations from this set.

These results also seem to indicate that our copying mechanism effectively deals with entity identification. Across all datasets, we see only a small number of entity-problem-only examples. However, comparing the rows from Table 3 for seq2seq+Copy at the top and seq2seq+Attention in the oracle entities condition, it is clear that having oracle entities provides a useful signal, with consistent gains in performance.

Takeaways: Evaluate on both question-based and query-based dataset splits. Additionally, variable anonymization noticeably decreases the difficulty of the task; thus, thorough evaluations should include results on datasets without anonymization.

5.5 Logic Variants

To see if our observations on query and question split performance apply beyond SQL, we also considered the logical form annotations for ATIS and GeoQuery (Zettlemoyer and Collins, 2005, 2007). We retrained Jia and Liang (2016)'s baseline and full system. Interestingly, we founnd limited impact on performance, measured with either logical forms or denotations. To understand why, we inspected the logical form datasets. In both ATIS and GeoQuery, the logical form version has a larger set of queries after variable identification. This seems to be because the logic abstracts away from the surface form less than SQL does. For example, these questions have the same SQL in our data, but different logical forms:

what state has the largest capital

```
(A, (state(A), loc(B, A), largest(B, capital(B))))
```

which state 's capital city is the largest

```
(A, largest(B, (state(A), capital(A, B), city(B))))
```

```
SELECT CITYalias0.STATE_NAME
FROM CITY AS CITYalias0
WHERE CITYalias0.POPULATION = (
  SELECT MAX( CITYalias1.POPULATION )
  FROM CITY AS CITYalias1 ,
       STATE AS STATEalias0
  WHERE STATEalias0.CAPITAL =
        CITYalias1.CITY_NAME ) ;
```

Other examples include variation in the logical form between sentences with *largest* and *largest*

[3]For the question *which of the states bordering pennsylvania has the largest population*, the gold query ranked the options by population and kept the top result, while the system output used a subquery to find the max population then selected states that had that population.

population even though the associated dataset only has population figures for cities (not area or any other measure of size). Similarly in ATIS, the logical form will add `(flight $0)` if the question mentions flights explicitly, making these two queries different, even though they convey the same user intent:

what flights do you have from bwi to sfo
i need a reservation from bwi to sfo

By being closer to a syntactic representation, the queries end up being more compositional, which encourages the model to learn more compositionality than the SQL models do.

6 Conclusion

In this work, we identify two issues in current datasets for mapping questions to SQL queries. First, by analyzing question and query complexity we find that human-written datasets require properties that have not yet been included in large-scale automatically generated query sets. Second, we show that the generalizability of systems is overstated by the traditional data splits. In the process we also identify and fix hundreds of mistakes across multiple datasets and homogenize the SQL query structures to enable effective multi-domain experiments.

Our analysis has clear implications for future work. Evaluating on multiple datasets is necessary to ensure coverage of the types of questions humans generate. Developers of future large-scale datasets should incorporate joins and nesting to create more human-like data. And new systems should be evaluated on both question- and query-based splits, guiding the development of truly general systems for mapping natural language to structured database queries.

Acknowledgments

We would like to thank Laura Wendlandt, Walter Lasecki, and Will Radford for comments on an earlier draft and the anonymous reviewers for their helpful suggestions. This material is based in part upon work supported by IBM under contract 4915012629. Any opinions, findings, conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of IBM.

References

I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. 1995. Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering*, 1(709):29–81.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR*, pages 1–15, San Diego, California.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Semantic parsing using distributional semantics and probabilistic logic. *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 7–11.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425.

Denny Britz, Anna Goldie, Minh-thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *ArXiv e-prints*.

Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, and Zijian Li. 2017. An encoder-decoder framework translating natural language to database queries. *ArXiv e-prints*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriber. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. *Proceedings of the workshop on Human Language Technology*, pages 43–48.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1:33–43.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In *COLING 2012*, pages 401–410.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963—-973, Vancouver, Canada.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding Task Design Trade-offs in Crowdsourced Paraphrase Collection.

In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 103–109, Vancouver, Canada.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. In *Proceedings of the VLDB Endowment*, pages 73–84.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 590–599, Portland, Oregon.

Arvind Neelakantan, Quoc V Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. *Proceedings of the ICLR*, pages 1–10.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Rodolfo A. Pazos Rangel, Juan Javier González Barbosa, Marco Antonio Aguirre Lam, José Antonio Martínez Flores, and Héctor J. Fraire Huacuja. 2013. *Natural language interfaces to databases: An analysis of the state of the art*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 933–943.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 141–147.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. *Proceedings of the 8th International Conference on Intelligent User Interfaces IUI 03*, pages 149–157.

Patti J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. *Proc. of the Speech and Natural Language Workshop*, pages 91–95.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated Construction of Database Interfaces: Integrating Statistical and Relational Learning for Semantic Parsing. *Proceedings of the Joint SIGDAT Conference on Emprical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type- and content-driven synthesis of SQL queries from natural language. *ArXiv e-prints*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural Enquirer: Learning to query tables in natural language. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 2308–2314.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database queries using inductive logic proramming. *Learning*, pages 1050–1055.

Luke Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form : Structured Classification with Probabilistic Categorial Grammars. *21st Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *ArXiv e-prints*, pages 1–12.

Semantic Parsing with Syntax- and Table-Aware SQL Generation

Yibo Sun[§]*, Duyu Tang[‡], Nan Duan[‡], Jianshu Ji[♮], Guihong Cao[♮],
Xiaocheng Feng[§], Bing Qin[§], Ting Liu[§], Ming Zhou[‡]
[§]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research Asia, Beijing, China
[♮]Microsoft AI and Research, Redmond WA, USA
{ybsun,xcfeng,qinb,tliu}@ir.hit.edu.cn
{dutang,nanduan,jianshuj,gucao,mingzhou}@microsoft.com

Abstract

We present a generative model to map natural language questions into SQL queries. Existing neural network based approaches typically generate a SQL query word-by-word, however, a large portion of the generated results is incorrect or not executable due to the mismatch between question words and table contents. Our approach addresses this problem by considering the structure of table and the syntax of SQL language. The quality of the generated SQL query is significantly improved through (1) learning to replicate content from column names, cells or SQL keywords; and (2) improving the generation of WHERE clause by leveraging the column-cell relation. Experiments are conducted on WikiSQL, a recently released dataset with the largest question-SQL pairs. Our approach significantly improves the state-of-the-art execution accuracy from 69.0% to 74.4%.

1 Introduction

We focus on semantic parsing that maps natural language utterances to executable programs (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011; Pasupat and Liang, 2015; Iyer et al., 2017; Iyyer et al., 2017). In this work, we regard SQL as the programming language, which could be executed on a table or a relational database to obtain an outcome. Datasets are the main driver of progress for statistical approaches in semantic parsing (Liang, 2016). Recently, Zhong et al. (2017) release WikiSQL, the largest hand-annotated semantic parsing dataset which is an order of magnitude larger than other datasets in terms of both the number of logical forms and the number of tables. Pointer network (Vinyals et al., 2015) based approach is developed, which generates a SQL query word-by-word through replicating from a word sequence consisting of question words, column names and SQL keywords. However, a large portion of generated results is incorrect or not executable due to the mismatch between question words and column names (or cells). This also reflects the real scenario where users do not always use exactly the same column name or cell content to express the question.

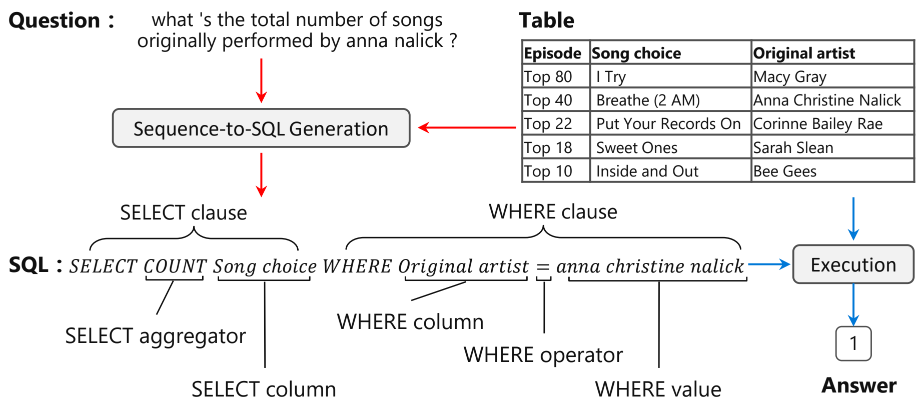
To address the aforementioned problem, we present a generative semantic parser that considers the structure of table and the syntax of SQL language. The approach is partly inspired by the success of structure/grammar driven neural network approaches in semantic parsing (Xiao et al., 2016; Krishnamurthy et al., 2017). Our approach is based on pointer networks, which encodes the question into continuous vectors, and synthesizes the SQL query with three channels. The model learns when to generate a column name, a cell or a SQL keyword. We further incorporate column-cell relation to mitigate the ill-formed outcomes.

We conduct experiments on WikiSQL. Results show that our approach outperforms existing systems, improving state-of-the-art execution accuracy to 74.4% and logical form accuracy to 60.7%. An extensive analysis reveals the advantages and limitations of our approach.

2 Task Formulation and Dataset

As shown in Figure 1, we focus on sequence-to-SQL generation in this work. Formally, the task takes a question q and a table t consisting of n col-

* Work is done during internship at Microsoft Research Asia.


Question :
what 's the total number of songs originally performed by anna nalick ?
Table
Sequence-to-SQL Generation
SELECT clause
WHERE clause
SQL : SELECT COUNT Song choice WHERE Original artist = anna christine nalick
Execution
SELECT aggregator
WHERE column
WHERE operator
SELECT column
WHERE value
1
Answer

Episode	Song choice	Original artist
Top 80	I Try	Macy Gray
Top 40	Breathe (2 AM)	Anna Christine Nalick
Top 22	Put Your Records On	Corinne Bailey Rae
Top 18	Sweet Ones	Sarah Slean
Top 10	Inside and Out	Bee Gees

Figure 1: An brief illustration of the task. The focus of this work is sequence-to-SQL generation.

umn names and $n \times m$ cells as the input, and outputs a SQL query y. We do not consider the join operation over multiple relational tables, which we leave in the future work.

We use WikiSQL (Zhong et al., 2017), the largest hand-annotated semantic parsing dataset to date which consists of 87,726 questions and SQL queries distributed across 26,375 tables from Wikipedia.

3 Related Work

Semantic Parsing. Semantic parsing aims to map natural language utterances to programs (e.g., logical forms), which will be executed to obtain the answer (denotation) (Zettlemoyer and Collins, 2005; Liang et al., 2011; Berant et al., 2013; Poon, 2013; Krishnamurthy and Kollar, 2013; Pasupat and Liang, 2016; Sun et al., 2016; Jia and Liang, 2016; Kočiský et al., 2016; Lin et al., 2017). Existing studies differ from (1) the form of the knowledge base, e.g. facts from Freebase, a table (or relational database), an image (Suhr et al., 2017; Johnson et al., 2017; Hu et al., 2017; Goldman et al., 2017) or a world state (Long et al., 2016); (2) the programming language, e.g. first-order logic, lambda calculus, lambda DCS, SQL, parameterized neural programmer (Yin et al., 2015; Neelakantan et al., 2016), or coupled distributed and symbolic executors (Mou et al., 2017); (3) the supervision used for learning the semantic parser, e.g. question-denotation pairs, binary correct/incorrect feedback (Artzi and Zettlemoyer, 2013), or richer supervision of question-logical form pairs (Dong and Lapata, 2016). In this work, we study semantic parsing over tables, which is critical for users to access relational databases with natural language, and could serve users' information need for structured data on the web. We use SQL as the programming language, which has a broad acceptance to programmers.

Natural Language Interface for Databases. Our work relates to the area of accessing database with natural language interface (Dahl et al., 1994; Brad et al., 2017). Popescu et al. (2003) use a parser to parse the question, and then use lexicon matching between question and the table column names/cells. Giordani and Moschitti (2012) parse the question with dependency parser, compose candidate SQL queries with heuristic rules, and use kernel based SVM ranker to rank the results. Li and Jagadish (2014) translate natural language utterances into SQL queries based on dependency parsing results, and interact with users to ensure the correctness of the interpretation process. Yaghmazadeh et al. (2017) build a semantic parser on the top of SEMPRE (Pasupat and Liang, 2015) to get a SQL sketch, which only has the SQL shape and will be subsequently completed based on the table content. Iyer et al. (2017) maps utterances to SQL queries through sequence-to-sequence learning. User feedbacks are incorporated to reduce the number of queries to be labeled. Zhong et al. (2017) develop an augmented pointer network, which is further improved with reinforcement learning for SQL sequence prediction. Xu et al. (2017) adopts a sequence-to-set model to predict WHERE columns, and uses an attentional model to predict the slots in where clause.

Different from (Iyer et al., 2017; Zhong et al., 2017), our approach leverages SQL syntax and table structure. Compared to (Popescu et al., 2003; Giordani and Moschitti, 2012; Yaghmazadeh et al., 2017), our approach is end-to-end learning and independent of a syntactic parser or manually designed templates. We are aware of existing studies that combine reinforcement learning and maximum likelihood estimation (MLE) (Guu et al., 2017; Mou et al., 2017; Liang et al., 2017). However, the focus of this work is the design of the neural architecture, despite we also implement an RL strategy (refer to §4.4).

Structure/Grammar Guided Neural Decoder Our approach could be viewed as an extension of the sequence-to-sequence learning (Sutskever et al., 2014; Bahdanau et al., 2015) with a tailored neural decoder driven by the characteristic of the target language (Yin and Neubig, 2017; Rabinovich et al., 2017). Methods with similar intuitions have been developed for language modeling (Dyer et al., 2016), neural machine translation (Wu et al., 2017) and lambda calculus based semantic parsing (Dong and Lapata, 2016; Krishnamurthy et al., 2017). The difference is that our model is developed for sequence-to-SQL generation, in which table structure and SQL syntax are considered.

4 Methodology

We first describe the background on pointer networks, and then present our approach that considers the table structure and the SQL syntax.

4.1 Background: Pointer Networks

Pointer networks is originally introduced by (Vinyals et al., 2015), which takes a sequence of elements as the input and outputs a sequence of discrete tokens corresponding to positions in the input sequence. The approach has been successfully applied in reading comprehension (Kadlec et al., 2016) for pointing to the positions of answer span from the document, and also in sequence-to-sequence based machine translation (Gulcehre et al., 2016) and text summarization (Gu et al., 2016) for replicating rare words from the source sequence to the target sequence.

The approach of Zhong et al. (2017) is based on pointer networks. The encoder is a recurrent neural network (RNN) with gated recurrent unit (GRU) (Cho et al., 2014), whose input is the concatenation of question words, words from column names and SQL keywords. The decoder is another GRU based RNN, which works in a sequential way and generates a word at each time step. The generation of a word is actually selectively replicating a word from the input sequence, the probability distribution of which is calculated with an attention mechanism (Bahdanau et al., 2015). The probability of generating the i-th word x_i in the input sequence at the t-th time step is calculated as Equation 1, where h_t^{dec} is the decoder hidden state at the t-th time step, h_i^{enc} is the encoder hidden state of the word x_i, W_a is the model parameter.

$$p(y_t = x_i | y_{<t}, x) \propto exp(W_a[h_t^{dec}; h_i^{enc}]) \quad (1)$$

It is worth to note that if a column name consists of multiple words (such as "original artist" in Figure 1), these words are separated in the input sequence. The approach has no guarantee that a multi-word column name could be successively generated, which would affect the executability of the generated SQL query.

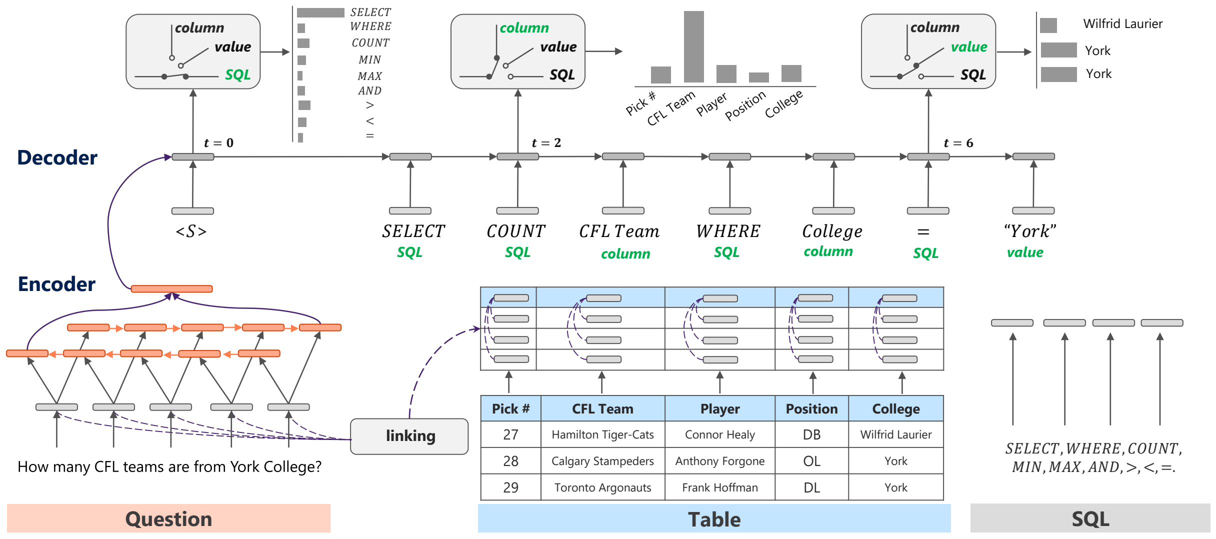
4.2 STAMP: Syntax- and Table- Aware seMantic Parser

Figure 2 illustrates an overview of the proposed model, which is abbreviated as STAMP. Different from Zhong et al. (2017), the word is not the basic unit to be generated in STAMP. As is shown, there are three "channels" in STAMP, among which the column channel predicts a column name, the value channel predicts a table cell and the SQL channel predicts a SQL keyword. Accordingly, the probability of generating a target token is formulated in Equation 2, where z_t stands for the channel selected by the switching gate, $p_z(\cdot)$ is the probability to choose a channel, and $p_w(\cdot)$ is similar to Equation 1 which is a probability distribution over the tokens from one of the three channels.

$$p(y_t | y_{<t}, x) = \sum_{z_t} p_w(y_t | z_t, y_{<t}, x) p_z(z_t | y_{<t}, x) \quad (2)$$

One advantage of this architecture is that it inherently addresses the problem of generating partial column name/cell because an entire column name/cell is the basic unit to be generated. Another advantage is that the column-cell relation and question-cell connection can be naturally integrated in the model, which will be described below.

Specifically, our encoder takes a question as the input. Bidirectional RNN with GRU unit is applied to the question, and the concatenation of both ends is used as the initial state of the decoder. Another bidirectional RNN is used to compute the representation of a column name (or a cell), in case that each unit contains multiple words (Dong



Pick #	CFL Team	Player	Position	College
27	Hamilton Tiger-Cats	Connor Healy	DB	Wilfrid Laurier
28	Calgary Stampeders	Anthony Forgone	OL	York
29	Toronto Argonauts	Frank Hoffman	DL	York

Figure 2: An illustration of the proposed approach. At each time step, a switching gate selects a channel to predict a column name (maybe composed of multiple words), a cell or a SQL keyword. The words in green below the SQL tokens stand for the results of the switching gate at each time step.

et al., 2015). Essentially, each channel is an attentional neural network. For cell and SQL channels, the input of the attention module only contains the decoder hidden state and the representation of the token to be calculated as follows,

$$p_w^{sql}(i) \propto exp(W_{sql}[h_t^{dec}; e_i^{sql}]) \qquad (3)$$

where e_i^{sql} stands for the representation of the i-th SQL keyword. As suggested by (Zhong et al., 2017), we also concatenate the question representation into the input of the column channel in order to improve the accuracy of the SELECT column. We implement the switching gate with a feed-forward neural network, in which the output is a $softmax$ function and the input is the decoder hidden state h_t^{dec}.

4.3 Improved with Column-Cell Relation

We further improve the STAMP model by considering the column-cell relation, which is important for predicting the WHERE clause.

On one hand, the column-cell relation could improve the prediction of SELECT column. We observe that a cell or a part of it typically appears at the question acting as the WHERE value, such as *"anna nalick"* for "*anna christine nalick*"). However, a column name might be represented with a totally different utterance, which is a "semantic gap". Supposing the question is "How many schools did player number 3 play at?" and the SQL query is "Select count School Club Team where No. = 3". We could see that the column names "School Club Team" and "No." are different from their corresponding utterances "schools", "number" in natural language question. Thus, table cells could be regarded as the pivot that connects the question and column names (the "linking" component in Figure 2). For instance, taking the question from Figure 2, the word "*York*" would help to predict the column name as "*College*" rather than "*Player*". There might be different possible ways to implement this intuition. We use cell information to enhance the column name representation in this work. The vector of a column name is further concatenated with a question-aware cell vector, which is weighted averaged over the cell vectors belonging to the same column. The probability distribution in the column channel is calculated as Equation 4. We use the number of cell words occurring in the question to measure the importance of a cell, which is further normalized through a $softmax$ function to yield the final weight $\alpha_j^{cell} \in [0, 1]$. An alternative measurement is to use an additional attention model whose input contains the question vector and the cell vector. We favor to the intuitive and efficient way in this work.

$$p_w^{col}(i) \propto exp(W_{col}[h_t^{dec}; h_i^{col}; \sum_{j \in col_i} \alpha_j^{cell} h_j^{cell}]) \qquad (4)$$

On the other hand, the column-cell relation could improve the prediction of the WHERE value. To yield an executable SQL, the WHERE value should be a cell that belongs to the same WHERE column[1]. Taking Figure 2 as an example, it should be avoided to predict a where clause like "*Player* = *York*" because the cell "*York*" does not belong to the column name "*Player*". To achieve this, we incorporate a global variable to memorize the last predicted column name. When the switching gate selects the value channel, the cell distribution is only calculated over the cells belonging to the last predicted column name. Furthermore, we incorporate an additional probability distribution over cells based on the aforementioned word co-occurrence between the question and cells, and weighted average two cell distributions, which is calculated as follows.

$$p_w^{cell}(j) = \lambda \hat{p}_w^{cell}(j) + (1-\lambda)\alpha_j^{cell} \qquad (5)$$

where $\hat{p}_w^{cell}(j)$ is the standard probability distribution obtained from the attentional neural network, and λ is a hyper parameter which is tuned on the dev set.

4.4 Improved with Policy Gradient

The model described so far could be conventionally learned via cross-entropy loss over question-SQL pairs. However, different SQL queries might be executed to yield the same result, and possible SQL queries of different variations could not be exhaustively covered in the training dataset. Two possible ways to handle this are (1) shuffling the WHERE clause to generate more SQL queries, and (2) using reinforcement learning (RL) which regards the correctness of the executed output as the goodness (reward) of the generated SQL query. We follow Zhong et al. (2017) and adopt a policy gradient based approach. We use a baseline strategy (Zaremba and Sutskever, 2015) to decrease the learning variance. The expected reward (Williams, 1992) for an instance is calculated as $\mathbb{E}(y_g) = \sum_{j=1}^{k} logp(y_j)R(y_j, y^g)$, where y^g is the ground truth SQL query, y_j is a generated SQL query, $p(y_j)$ is the probability of y_j being generated by our model, and k is the number of sampled SQL queries. $R(y_j, y^g)$ is the same reward function defined by Zhong et al. (2017), which is $+1$ if y_j is executed to yield the correct answer; -1 if y_j is a valid SQL query and is executed to get an incorrect answer; and -2 if y_j is not a valid SQL query. In this way, model parameters could be updated with policy gradient over question-answer pairs.

4.5 Training and Inference

As the WikiSQL data contains rich supervision of question-SQL pairs, we use them to train model parameters. The model has two cross-entropy loss functions, as given below. One is for the switching gate classifier (p_z) and another is for the attentional probability distribution of a channel (p_w).

$$l = -\sum_t logp_z(z_t|y_{<t}, x) - \sum_t logp_w(y_t|z_t, y_{<t}, x) \qquad (6)$$

Our parameter setting strictly follows Zhong et al. (2017). We represent each word using word embedding[2] (Pennington et al., 2014) and the mean of the sub-word embeddings of all the n-grams in the word (Hashimoto et al., 2016)[3]. The dimension of the concatenated word embedding is 400. We clamp the embedding values to avoid over-fitting. We set the dimension of encoder and decoder hidden state as 200. During training, we randomize model parameters from a uniform distribution with fan-in and fan-out, set batch size as 64, set the learning rate of SGD as 0.5, and update the model with stochastic gradient descent. Greedy search is used in the inference process. We use the model trained from question-SQL pairs as initialization and use RL strategy to fine-tune the model. SQL queries used for training RL are sampled based on the probability distribution of the model learned from question-SQL pairs. We tune the best model on the dev set and do inference on the test set for only once. This protocol is used in model comparison as well as in ablations.

5 Experiment

We conduct experiments on the WikiSQL dataset[4], which includes $61,297/9,145/17,284$ examples in the training/dev/test sets. Each instance consists of a question, a table, a SQL query and a result. Following Zhong et al. (2017), we use two

[1]This constraint is suitable in this work as we do not consider the nested query in the where clause, such as "*where College = select College from table*", which is also the case not included in the WikiSQL dataset. We leave generating nested SQL query in the future work.

[2]http://nlp.stanford.edu/data/glove.840B.300d.zip

[3]http://www.logos.t.u-tokyo.ac.jp/~hassy/publications/arxiv2016jmt/jmt_pre-trained_embeddings.tar.gz

[4]https://github.com/salesforce/WikiSQL

Methods	Dev		Test	
	Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
Attentional Seq2Seq	23.3%	37.0%	23.4%	35.9%
Aug.PntNet (Zhong et al., 2017)	44.1%	53.8%	43.3%	53.3%
Aug.PntNet (re-implemented by us)	51.5%	58.9%	52.1%	59.2%
Seq2SQL (no RL) (Zhong et al., 2017)	48.2%	58.1%	47.4%	57.1%
Seq2SQL (Zhong et al., 2017)	49.5%	60.8%	48.3%	59.4%
SQLNet (Xu et al., 2017)	–	69.8%	–	68.0%
Guo and Gao (2018)	–	71.1%	–	69.0%
STAMP (w/o cell)	58.6%	67.8%	58.0%	67.4%
STAMP (w/o column-cell relation)	59.3%	71.8%	58.4%	70.6%
STAMP	61.5%	74.8%	60.7%	74.4%
STAMP+RL	61.7%	75.1%	61.0%	74.6%

Table 1: Performances of different approaches on the WikiSQL dataset. Two evaluation metrics are logical form accuracy (Acc_{lf}) and execution accuracy (Acc_{ex}). Our model is abbreviated as (**STAMP**).

evaluation metrics. One metric is logical form accuracy (Acc_{lf}), which measures the percentage of the generated SQL queries that have exact string match with the ground truth SQL queries. Since different SQL queries might obtain the same result, another metric is execution accuracy (Acc_{ex}), which measures the percentage of the generated SQL queries that obtain the correct answer.

5.1 Model Comparisons

After released, WikiSQL dataset has attracted a lot of attentions from both industry and research communities. Zhong et al. (2017) develop following methods, including (1) Aug.PntNet which is an end-to-end learning pointer network approach; (2) Seq2SQL (no RL), in which two separate classifiers are trained for SELECT aggregator and SELECT column, separately; and (3) Seq2SQL, in which reinforcement learning is further used for model training. Results of tattentional sequence-to-sequence learning baseline (Attentional Seq2Seq) are also reported in (Zhong et al., 2017). Xu et al. (2017) develop SQLNet, which predicts SELECT clause and WHERE clause separately. Sequence-to-set neural architecture and column attention are adopted to predict the WHERE clause. Similarly, Guo and Gao (2018) develop tailored modules to handle three components of SQL queries, respectively. A parallel work from (Yu et al., 2018) obtains higher execution accuracy (82.6%) on WikiSQL, however, its model is slot-filling based which is designed specifically for the "select-aggregator-where" type and utilizes external knowledge base (such as Freebase) to tag the question words. We believe this mechanism could improve our model as well, we leave this as a potential future work.

Our model is abbreviated as (**STAMP**), which is short for Syntax- and Table- Aware seMantic Parser. The STAMP model in Table 1 stands for the model we describe in §4.2 plus §4.3. STAMP+RL is the model that is fine-tuned with the reinforcement learning strategy as described in §4.4. We implement a simplified version of our approach (w/o cell), in which WHERE values come from the question. Thus, this setting differs from Aug.PntNet in the generation of WHERE column. We also study the influence of the relation-cell relation (w/o column-cell relation) through removing the enhanced column vector, which is calculated by weighted averaging cell vectors.

From Table 1, we can see that STAMP performs better than existing systems on WikiSQL. Incorporating RL strategy does not significantly improve the performance. Our simplified model, STAMP (w/o cell), achieves better accuracy than Aug.PntNet, which further reveals the effects of the column channel. Results also demonstrate the effects of incorporating the column-cell relation, removing which leads to about 4% performance drop in terms of Acc_{ex}.

5.2 Model Analysis: Fine-Grained Accuracy

We analyze the STAMP model from different perspectives in this part.

Firstly, since SQL queries in WikiSQL consists of SELECT column, SELECT aggregator, and WHERE clause, we report the results with regard

Methods	Dev			Test		
	Acc$_{sel}$	Acc$_{agg}$	Acc$_{where}$	Acc$_{sel}$	Acc$_{agg}$	Acc$_{where}$
Aug.PntNet (reimplemented by us)	80.9%	89.3%	62.1%	81.3%	89.7%	62.1%
Seq2SQL (Zhong et al., 2017)	89.6%	90.0%	62.1%	88.9%	90.1%	60.2%
SQLNet (Xu et al., 2017)	91.5%	90.1%	74.1%	90.9%	90.3%	71.9%
Guo and Gao (2018)	92.5%	90.1%	74.7%	91.9%	90.3%	72.8%
STAMP (w/o cell)	89.9%	89.2%	72.1%	89.2%	89.3%	71.2%
STAMP (w/o column-cell relation)	89.3%	89.2%	73.2%	88.8%	89.2%	71.8%
STAMP	89.4%	89.5%	77.1%	88.9%	89.7%	76.0%
STAMP+RL	89.6%	89.7%	77.3%	90.0%	89.9%	76.3%

Table 2: Fine-grained accuracies on the WikiSQL dev and test sets. Accuracy (Acc$_{lf}$) is evaluated on SELECT column (Acc$_{sel}$) , SELECT aggregator (Acc$_{agg}$), and WHERE clause (Acc$_{where}$), respectively.

to more fine-grained evaluation metrics over these aspects. Results are given in Table 2, in which the numbers of Seq2SQL and SQLNet are reported in (Xu et al., 2017). We can see that the main improvement of STAMP comes from the WHERE clause, which is also the key challenge of the WikiSQL dataset. This is consistent with our primary intuition on improving the prediction of WHERE column and WHERE value. The accuracies of STAMP on SELECT column and SELECT aggregator are not as high as SQLNet. The main reason is that these two approaches train the SELECT clause separately while STAMP learns all these components in a unified paradigm.

5.3 Model Analysis: Difficulty Analysis

We study the performance of STAMP on different portions of the test set according to the difficulties of examples. We compare between Aug.PntNet (re-implemented by us) and STAMP. In this work, the difficulty of an example is reflected by the number of WHERE columns.

Method	#where	Dev	Test
	$= 1$	63.4%	63.8%
Aug.PntNet	$= 2$	51.0%	51.8%
	≥ 3	38.5%	38.1%
	$= 1$	80.9%	80.2%
STAMP	$= 2$	65.1%	65.4%
	≥ 3	44.1%	48.2%

Table 3: Execution accuracy (Acc$_{ex}$) on different groups of WikiSQL dev and test sets.

From Table 3, we can see that STAMP outperforms Aug.PntNet in all these groups. The accuracy decreases with the increase of the number of WHERE conditions.

5.4 Model Analysis: Executable Analysis

We study the percentage of executable SQL queries in the generated results. As shown in Table 4, STAMP significantly outperforms Aug.PntNet. Almost all the results of STAMP are executable. This is because STAMP avoids generating incomplete column names or cells, and guarantees the correlation between WHERE conditions and WHERE values in the table.

	Dev	Test
Aug.PntNet	77.9%	78.7%
STAMP	99.9%	99.9%

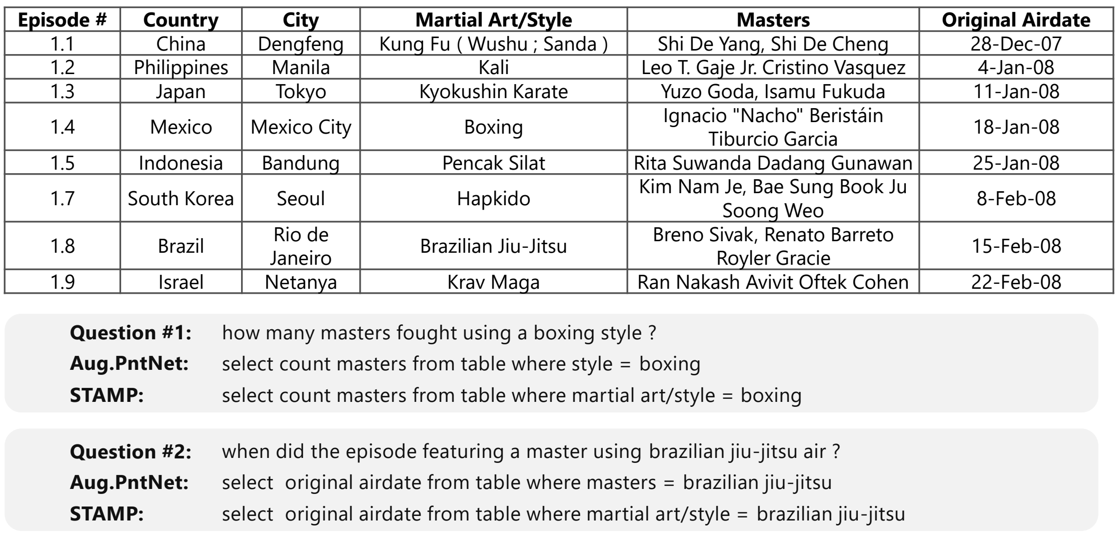
Table 4: Percentage of the executable SQL queries on WikiSQL dev and test sets.

5.5 Model Analysis: Case Study

We give a case study to illustrate the generated results by STAMP, with a comparison to Aug.PntNet. Results are given in Figure 3. In the first example, Aug.PntNet generates incomplete column name (“*style*”), which is addressed in STAMP through replicating an entire column name. In the second example, the WHERE value (“*brazilian jiu-jitsu*”) does not belong to the generated WHERE column (“*Masters*”) in Aug.PntNet. This problem is avoided in STAMP through incorporating the table content.

5.6 Error Analysis

We conduct error analysis on the dev set of WikiSQL to show the limitation of the STAMP model and where is the room for making further improvements. We analyze the 2,302 examples which are executed to wrong answers by the STAMP model, and find that 33.6% of them have wrong SE-



Episode #	Country	City	Martial Art/Style	Masters	Original Airdate
1.1	China	Dengfeng	Kung Fu (Wushu ; Sanda)	Shi De Yang, Shi De Cheng	28-Dec-07
1.2	Philippines	Manila	Kali	Leo T. Gaje Jr. Cristino Vasquez	4-Jan-08
1.3	Japan	Tokyo	Kyokushin Karate	Yuzo Goda, Isamu Fukuda	11-Jan-08
1.4	Mexico	Mexico City	Boxing	Ignacio "Nacho" Beristáin Tiburcio Garcia	18-Jan-08
1.5	Indonesia	Bandung	Pencak Silat	Rita Suwanda Dadang Gunawan	25-Jan-08
1.7	South Korea	Seoul	Hapkido	Kim Nam Je, Bae Sung Book Ju Soong Weo	8-Feb-08
1.8	Brazil	Rio de Janeiro	Brazilian Jiu-Jitsu	Breno Sivak, Renato Barreto Royler Gracie	15-Feb-08
1.9	Israel	Netanya	Krav Maga	Ran Nakash Avivit Oftek Cohen	22-Feb-08

Question #1:	how many masters fought using a boxing style ?
Aug.PntNet:	select count masters from table where style = boxing
STAMP:	select count masters from table where martial art/style = boxing

Question #2:	when did the episode featuring a master using brazilian jiu-jitsu air ?
Aug.PntNet:	select original airdate from table where masters = brazilian jiu-jitsu
STAMP:	select original airdate from table where martial art/style = brazilian jiu-jitsu

Figure 3: Case study on the dev set between Aug.PntNet and STAMP. These two questions are based on the same table. Each question is followed by the generated SQL queries from the two approaches.

LECT columns, 15.7% of them have a different number of conditions in the WHERE clause, and 53.7% of them have a different WHERE column set compared to the ground truth. Afterwards, we analyze a portion of randomly sampled dissatisfied examples. Consistent with the qualitative results, most problems come from column prediction, including both SELECT clause and WHERE clause. Even though the overall accuracy of the SELECT column prediction is about 90% and we also use cell information to enhance the column representation, this semantic gap is still the main bottleneck. Extracting and incorporating various expressions for a table column (i.e. relation in a relational database) might be a potential way to mitigate this problem. Compared to column prediction, the quality of cell prediction is much better because cell content typically (partially) appears in the question.

5.7 Transfers to WikiTableQuestions

WikiTableQuestions (Pasupat and Liang, 2015) is a widely used dataset for semantic parsing. To further test the performance of our approach, we conduct an additional transfer learning experiment. Firstly, we directly apply the STAMP model trained on WikiSQL to WikiTableQuestions, which is an unsupervised learning setting for the WikiTableQuestions dataset. Results show that the test accuracy of STAMP in this setting is 14.5%, which has a big gap between best systems on WikiTableQuestions, where Zhang et al. (2017) and Krishnamurthy et al. (2017) yield 43.3% and 43.7%, respectively. Furthermore, we apply the learnt STAMP model to generate SQL queries on natural language questions from WikiTableQuestions, and regard the generated SQL queries which could be executed to correct answers as additional pseudo question-SQL pairs. In this way, the STAMP model learnt from a combination of WikiSQL and pseudo question-SQL pairs could achieve 21.0% on the test set. We find that this big gap is caused by the difference between the two datasets. Among 8 types of questions in WikiTableQuestions, half of them including {"*Union*", "*Intersection*", "*Reverse*", "*Arithmetic*"} are not covered in the WikiSQL dataset. It is an interesting direction to leverage algorithms developed from two datasets to improve one another.

5.8 Discussion

Compared to slot-filling based models that restrict target SQL queries to fixed forms of "select-aggregator-where", our model is less tailored. We believe that it is easy to expand our model to generate nested SQL queries or JOIN clauses, which could also be easily trained with back-propagation if enough training instances of these SQL types are available. For example, we could incorporate a hierarchical "value" channel to handle nest queries. Let us suppose our decoder works horizontally that next generated token is at the right hand of the current token. Inspired by chunk-based decoder for neural machine translation (Ishiwatari et al., 2017), we could increase the depth of the "value" channel to generates tokens of a nested WHERE value along the vertical axis. During inference, an addition gating function might be necessary to determine whether to generate a nested query, followed by the generation of WHERE value. An intuitive way that extends our model to handle JOIN clauses is to add the 4th channel, which predicts a table from a collection of tables. Therefore, the decoder should learn to select one of the four channels at each time step. Accordingly, we need to add "from" as a new SQL keyword in order to generate SQL queries including "from xxxTable".

In terms of the syntax of SQL, the grammar we used in this work could be regarded as shallow syntax, such as three channels and column-cell relation. We do not use deep syntax, such as the sketch of SQL language utilized in some slot-filling models, because incorporating them would make the model clumpy. Instead, we let the model to learn the sequential and compositional relations of SQL queries automatically from data. Empirical results show that our model learns these patterns well.

6 Conclusion and Future Work

In this work, we develop STAMP, a Syntax- and Table- Aware seMantic Parser that automatically maps natural language questions to SQL queries, which could be executed on web table or relational dataset to get the answer. STAMP has three channels, and it learns to switch to which channel at each time step. STAMP considers cell information and the relation between cell and column name in the generation process. Experiments are conducted on the WikiSQL dataset. Results show that STAMP achieves the new state-of-the-art performance on WikiSQL. We conduct extensive experiment analysis to show advantages and limitations of our approach, and where is the room for others to make further improvements.

SQL language has more complicated queries than the cases included in the WikiSQL dataset, including (1) querying over multiple relational databases, (2) nested SQL query as condition value, (3) more operations such as "*group by*" and "*order by*", etc. In this work, the STAMP model is not designed for the first and second cases, but it could be easily adapted to the third case through incorporating additional SQL keywords and of course the learning of which requires dataset of the same type. In the future, we plan to improve the accuracy of the column prediction component. We also plan to build a large-scale dataset that considers more sophisticated SQL queries. We also plan to extend the approach to low-resource scenarios (Feng et al., 2018).

Acknowledgments

We thank Yaming Sun for her great help. We also would like to thank three anonymous reviewers for their valuable comments and suggestions. This research was partly supported by National Natural Science Foundation of China(No. 61632011 and No.61772156, and No.61472107).

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1:49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceeding of ICLR* .

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*. 5, page 6.

Florin Brad, Radu Cristian Alexandru Iacob, Ionel Alexandru Hosu, and Traian Rebedea. 2017. Dataset for a neural natural language interface for databases (nnlidb). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 906–914. http://www.aclweb.org/anthology/I17-1091.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 43–48.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 33–43. http://www.aclweb.org/anthology/P16-1004.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *IJCAI*. pages 1243–1249.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 199–209. http://www.aclweb.org/anthology/N16-1024.

Xiaocheng Feng, Xiachong Feng, Bing Qin, Zhangyin Feng, and Ting Liu. 2018. Improving low resource named entity recognition using cross-lingual knowledge transfer. In *IJCAI*.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to sql queries with generative parsers discriminatively reranked. In *COLING (Posters)*. pages 401–410.

Omer Goldman, Veronica Latcinnik, Udi Naveh, Amir Globerson, and Jonathan Berant. 2017. Weakly-supervised semantic parsing with abstract examples. *CoRR* abs/1711.05240. http://arxiv.org/abs/1711.05240.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1631–1640. http://www.aclweb.org/anthology/P16-1154.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 140–149. http://www.aclweb.org/anthology/P16-1014.

Tong Guo and Huilin Gao. 2018. Bidirectional attention for SQL generation. *CoRR* abs/1801.00076. http://arxiv.org/abs/1801.00076.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. pages 1051–1062.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587* .

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. *International Conference on Computer Vision (ICCV)*. .

Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. 2017. Chunk-based decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1901–1912. http://aclweb.org/anthology/P17-1174.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 963–973. http://aclweb.org/anthology/P17-1089.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1821–1831. http://aclweb.org/anthology/P17-1167.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 12–22. http://www.aclweb.org/anthology/P16-1002.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. *International Conference on Computer Vision (ICCV)*. .

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 908–918. http://www.aclweb.org/anthology/P16-1086.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1078–1087. https://aclweb.org/anthology/D16-1116.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1517–1527. https://www.aclweb.org/anthology/D17-1160.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics* 1:193–206.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1512–1523.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment* 8(1):73–84.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 23–33. http://aclweb.org/anthology/P17-1003.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM* 59(9):68–76.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. pages 590–599.

Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, Luke Zettlemoyer, and Michael D. Ernst. 2017. Program synthesis from natural language using recurrent neural networks. Technical Report UW-CSE-17-03-01, University of Washington Department of Computer Science and Engineering, Seattle, WA, USA.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1456–1465. http://www.aclweb.org/anthology/P16-1138.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2017. Coupling distributed and symbolic execution for natural language queries. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. pages 2518–2526. http://proceedings.mlr.press/v70/mou17a.html.

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a natural language interface with neural programmer. *arXiv preprint arXiv:1611.08945* .

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1470–1480. http://www.aclweb.org/anthology/P15-1142.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 23–32.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *ACL (1)*. pages 933–943.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, pages 149–157.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. *arXiv preprint arXiv:1704.07535* .

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 217–223. http://aclweb.org/anthology/P17-2034.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 771–782.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Annual Meeting-Association for computational Linguistics*. 1, page 960.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 698–707. http://aclweb.org/anthology/P17-1065.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1341–1350. http://www.aclweb.org/anthology/P16-1127.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436* .

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type-and content-driven synthesis of sql queries from natural language. *arXiv preprint arXiv:1702.01168* .

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables with natural language. *arXiv preprint arXiv:1512.00965* .

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696* .

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769* .

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521* 419.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*. pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*. pages 658–666.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*. pages 678–687.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1225–1234. https://www.aclweb.org/anthology/D17-1126.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* .

Multitask Parsing Across Semantic Representations

Daniel Hershcovich[1,2] **Omri Abend**[2] **Ari Rappoport**[2]
[1]The Edmond and Lily Safra Center for Brain Sciences
[2]School of Computer Science and Engineering
Hebrew University of Jerusalem
{danielh,oabend,arir}@cs.huji.ac.il

Abstract

The ability to consolidate information of different types is at the core of intelligence, and has tremendous practical value in allowing learning for one task to benefit from generalizations learned for others. In this paper we tackle the challenging task of improving semantic parsing performance, taking UCCA parsing as a test case, and AMR, SDP and Universal Dependencies (UD) parsing as auxiliary tasks. We experiment on three languages, using a uniform transition-based system and learning architecture for all parsing tasks. Despite notable conceptual, formal and domain differences, we show that multitask learning significantly improves UCCA parsing in both in-domain and out-of-domain settings. Our code is publicly available.[1]

1 Introduction

Semantic parsing has arguably yet to reach its full potential in terms of its contribution to downstream linguistic tasks, partially due to the limited amount of semantically annotated training data. This shortage is more pronounced in languages other than English, and less researched domains.

Indeed, recent work in semantic parsing has targeted, among others, Abstract Meaning Representation (AMR; Banarescu et al., 2013), bilexical Semantic Dependencies (SDP; Oepen et al., 2016) and Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013). While these schemes are formally different and focus on different distinctions, much of their semantic content is shared (Abend and Rappoport, 2017).

Multitask learning (MTL; Caruana, 1997) allows exploiting the overlap between tasks to effectively extend the training data, and has greatly advanced with neural networks and representation learning (see §2). We build on these ideas and propose a general transition-based DAG parser, able to parse UCCA, AMR, SDP and UD (Nivre et al., 2016). We train the parser using MTL to obtain significant improvements on UCCA parsing over single-task training in (1) in-domain and (2) out-of-domain settings in English; (3) an in-domain setting in German; and (4) an in-domain setting in French, where training data is scarce.

The novelty of this work is in proposing a general parsing and learning architecture, able to accommodate such widely different parsing tasks, and in leveraging it to show benefits from learning them jointly.

2 Related Work

MTL has been used over the years for NLP tasks with varying degrees of similarity, examples including joint classification of different arguments in semantic role labeling (Toutanova et al., 2005), and joint parsing and named entity recognition (Finkel and Manning, 2009). Similar ideas, of parameter sharing across models trained with different datasets, can be found in studies of domain adaptation (Blitzer et al., 2006; Daume III, 2007; Ziser and Reichart, 2017). For parsing, domain adaptation has been applied successfully in parser combination and co-training (McClosky et al., 2010; Baucom et al., 2013).

Neural MTL has mostly been effective in tackling formally similar tasks (Søgaard and Goldberg, 2016), including multilingual syntactic dependency parsing (Ammar et al., 2016; Guo et al., 2016), as well as multilingual (Duong et al., 2017), and cross-domain semantic parsing (Herzig and Berant, 2017; Fan et al., 2017).

Sharing parameters with a low-level task has

[1]http://github.com/danielhers/tupa

shown great benefit for transition-based syntactic parsing, when jointly training with POS tagging (Bohnet and Nivre, 2012; Zhang and Weiss, 2016), and with lexical analysis (Constant and Nivre, 2016; More, 2016). Recent work has achieved state-of-the-art results in multiple NLP tasks by jointly learning the tasks forming the NLP standard pipeline using a single neural model (Collobert et al., 2011; Hashimoto et al., 2017), thereby avoiding cascading errors, common in pipelines.

Much effort has been devoted to joint learning of syntactic and semantic parsing, including two CoNLL shared tasks (Surdeanu et al., 2008; Hajič et al., 2009). Despite their conceptual and practical appeal, such joint models rarely outperform the pipeline approach (Lluís and Màrquez, 2008; Henderson et al., 2013; Lewis et al., 2015; Swayamdipta et al., 2016, 2017).

Peng et al. (2017a) performed MTL for SDP in a closely related setting to ours. They tackled three tasks, annotated over the same text and sharing the same formal structures (bilexical DAGs), with considerable edge overlap, but differing in target representations (see §3). For all tasks, they reported an increase of 0.5-1 labeled F_1 points. Recently, Peng et al. (2018) applied a similar approach to joint frame-semantic parsing and semantic dependency parsing, using disjoint datasets, and reported further improvements.

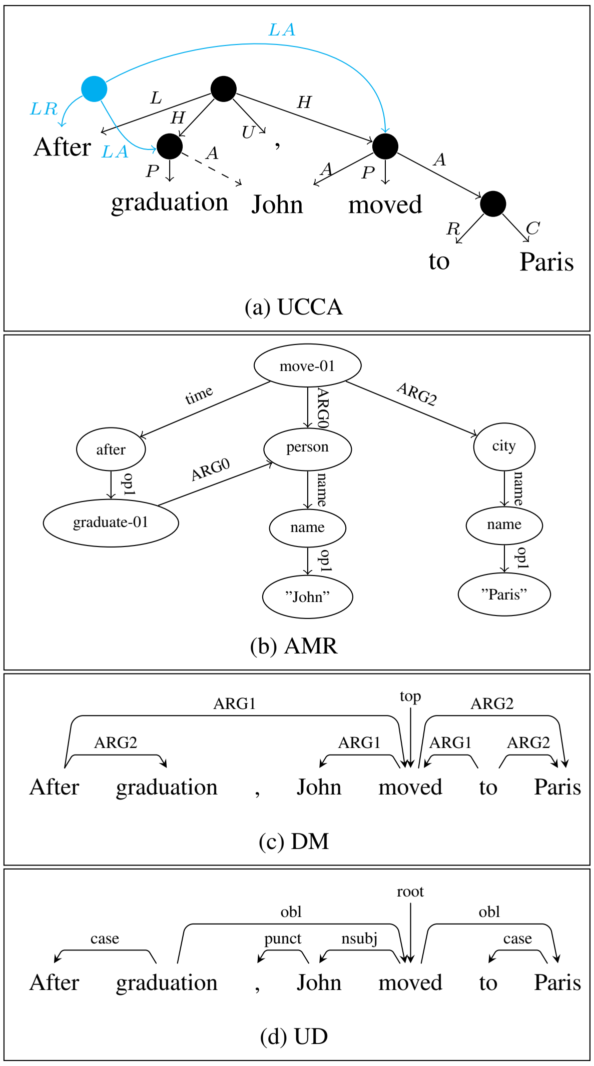


Figure 1: Example graph for each task. Figure 1a presents a UCCA graph. The dashed edge is remote, while the blue node and its outgoing edges represent inter-Scene linkage. Pre-terminal nodes and edges are omitted for brevity. Figure 1b presents an AMR graph. Text tokens are not part of the graph, and must be matched to concepts and constants by alignment. Variables are represented by their concepts. Figure 1c presents a DM semantic dependency graph, containing multiple roots: "After", "moved" and "to", of which "moved" is marked as *top*. Punctuation tokens are excluded from SDP graphs. Figure 1d presents a UD tree. Edge labels express syntactic relations.

3 Tackled Parsing Tasks

In this section, we outline the parsing tasks we address. We focus on representations that produce full-sentence analyses, i.e., produce a graph covering all (content) words in the text, or the lexical concepts they evoke. This contrasts with "shallow" semantic parsing, primarily semantic role labeling (SRL; Gildea and Jurafsky, 2002; Palmer et al., 2005), which targets argument structure phenomena using flat structures. We consider four formalisms: UCCA, AMR, SDP and Universal Dependencies. Figure 1 presents one sentence annotated in each scheme.

Universal Conceptual Cognitive Annotation. UCCA (Abend and Rappoport, 2013) is a semantic representation whose main design principles are ease of annotation, cross-linguistic applicability, and a modular architecture. UCCA represents the semantics of linguistic utterances as directed acyclic graphs (DAGs), where terminal (childless) nodes correspond to the text tokens, and non-terminal nodes to semantic units that participate in some super-ordinate relation. Edges are labeled, indicating the role of a child in the relation the parent represents. Nodes and edges belong to one of several *layers*, each corresponding to a "module" of semantic distinctions. UCCA's *foundational layer* (the only layer for which annotated data exists) mostly covers predicate-argument structure, semantic heads and inter-Scene relations.

UCCA distinguishes *primary* edges, corresponding to explicit relations, from *remote* edges (appear dashed in Figure 1a) that allow for a unit to participate in several super-ordinate relations. Primary edges form a tree in each layer, whereas remote edges enable reentrancy, forming a DAG.

Abstract Meaning Representation. AMR (Banarescu et al., 2013) is a semantic representation that encodes information about named entities, argument structure, semantic roles, word sense and co-reference. AMRs are rooted directed graphs, in which both nodes and edges are labeled. Most AMRs are DAGs, although cycles are permitted.

AMR differs from the other schemes we consider in that it does not anchor its graphs in the words of the sentence (Figure 1b). Instead, AMR graphs connect variables, concepts (from a predefined set) and constants (which may be strings or numbers). Still, most AMR nodes are alignable to text tokens, a tendency used by AMR parsers, which align a subset of the graph nodes to a subset of the text tokens (concept identification). In this work, we use pre-aligned AMR graphs.

Despite the brief period since its inception, AMR has been targeted by a number of works, notably in two SemEval shared tasks (May, 2016; May and Priyadarshi, 2017). To tackle its variety of distinctions and unrestricted graph structure, AMR parsers often use specialized methods. Graph-based parsers construct AMRs by identifying concepts and scoring edges between them, either in a pipeline fashion (Flanigan et al., 2014; Artzi et al., 2015; Pust et al., 2015; Foland and Martin, 2017), or jointly (Zhou et al., 2016). Another line of work trains machine translation models to convert strings into linearized AMRs (Barzdins and Gosko, 2016; Peng et al., 2017b; Konstas et al., 2017; Buys and Blunsom, 2017b). Transition-based AMR parsers either use dependency trees as pre-processing, then mapping them into AMRs (Wang et al., 2015a,b, 2016; Goodman et al., 2016), or use a transition system tailored to AMR parsing (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017). We differ from the above approaches in addressing AMR parsing using the same general DAG parser used for other schemes.

Semantic Dependency Parsing. SDP uses a set of related representations, targeted in two recent SemEval shared tasks (Oepen et al., 2014, 2015), and extended by Oepen et al. (2016). They correspond to four semantic representation schemes, referred to as DM, PAS, PSD and CCD, representing predicate-argument relations between content words in a sentence. All are based on semantic formalisms converted into bilexical dependencies—directed graphs whose nodes are text tokens. Edges are labeled, encoding semantic relations between the tokens. Non-content tokens, such as punctuation, are left out of the analysis (see Figure 1c). Graphs containing cycles have been removed from the SDP datasets.

We use one of the representations from the SemEval shared tasks: DM (DELPH-IN MRS), converted from DeepBank (Flickinger et al., 2012), a corpus of hand-corrected parses from LinGO ERG (Copestake and Flickinger, 2000), an HPSG (Pollard and Sag, 1994) using Minimal Recursion Semantics (Copestake et al., 2005).

Universal Dependencies. UD (Nivre et al., 2016, 2017) has quickly become the dominant dependency scheme for syntactic annotation in many languages, aiming for cross-linguistically consistent and coarse-grained treebank annotation. Formally, UD uses bilexical trees, with edge labels representing syntactic relations between words.

We use UD as an auxiliary task, inspired by previous work on joint syntactic and semantic parsing (see §2). In order to reach comparable analyses cross-linguistically, UD often ends up in annotation that is similar to the common practice in semantic treebanks, such as linking content words to content words wherever possible. Using UD further allows conducting experiments on languages other than English, for which AMR and SDP annotated data is not available (§7).

In addition to basic UD trees, we use the *enhanced*++ UD graphs available for English, which are generated by the Stanford CoreNLP converters (Schuster and Manning, 2016).[2] These include additional and augmented relations between content words, partially overlapping with the notion of remote edges in UCCA: in the case of control verbs, for example, a direct relation is added in enhanced++ UD between the subordinated verb and its controller, which is similar to the semantic schemes' treatment of this construction.

4 General Transition-based DAG Parser

All schemes considered in this work exhibit reentrancy and discontinuity (or non-projectivity), to varying degrees. In addition, UCCA and AMR

[2]http://github.com/stanfordnlp/CoreNLP

contain non-terminal nodes. To parse these graphs, we extend TUPA (Hershcovich et al., 2017), a transition-based parser originally developed for UCCA, as it supports all these structural properties. TUPA's transition system can yield any labeled DAG whose terminals are anchored in the text tokens. To support parsing into AMR, which uses graphs that are not anchored in the tokens, we take advantage of existing alignments of the graphs with the text tokens during training (§5).

First used for projective syntactic dependency tree parsing (Nivre, 2003), transition-based parsers have since been generalized to parse into many other graph families, such as (discontinuous) constituency trees (e.g., Zhang and Clark, 2009; Maier and Lichte, 2016), and DAGs (e.g., Sagae and Tsujii, 2008; Du et al., 2015). Transition-based parsers apply *transitions* incrementally to an internal state defined by a buffer B of remaining tokens and nodes, a stack S of unresolved nodes, and a labeled graph G of constructed nodes and edges. When a terminal state is reached, the graph G is the final output. A classifier is used at each step to select the next transition, based on features that encode the current state.

4.1 TUPA's Transition Set

Given a sequence of tokens $w_1, \ldots, w_n$, we predict a rooted graph G whose terminals are the tokens. Parsing starts with the root node on the stack, and the input tokens in the buffer.

The TUPA transition set includes the standard SHIFT and REDUCE operations, NODE$_X$ for creating a new non-terminal node and an X-labeled edge, LEFT-EDGE$_X$ and RIGHT-EDGE$_X$ to create a new primary X-labeled edge, LEFT-REMOTE$_X$ and RIGHT-REMOTE$_X$ to create a new remote X-labeled edge, SWAP to handle discontinuous nodes, and FINISH to mark the state as terminal.

Although UCCA contains nodes without any text tokens as descendants (called *implicit units*), these nodes are infrequent and only cover 0.5% of non-terminal nodes. For this reason we follow previous work (Hershcovich et al., 2017) and discard implicit units from the training and evaluation, and so do not include transitions for creating them.

In AMR, implicit units are considerably more common, as any unaligned concept with no aligned descendents is implicit (about 6% of the nodes). Implicit AMR nodes usually result from alignment errors, or from abstract concepts which

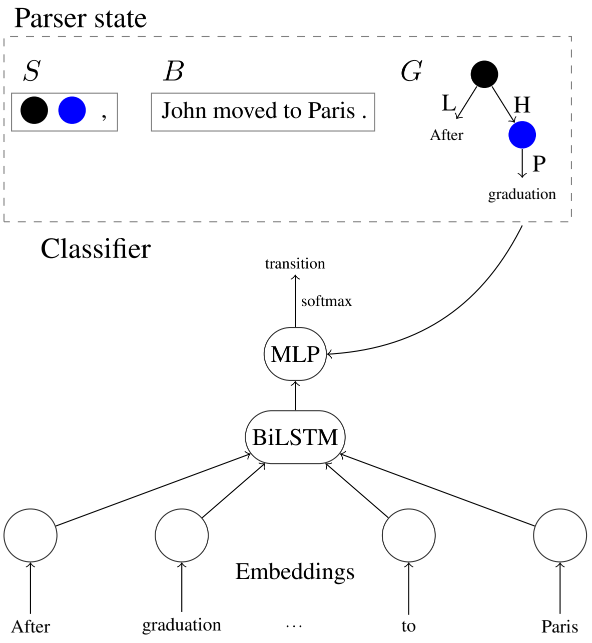


Figure 2: Illustration of the TUPA model, adapted from Hershcovich et al. (2017). Top: parser state. Bottom: BiLTSM architecture.

have no explicit realization in the text (Buys and Blunsom, 2017a). We ignore implicit nodes when training on AMR as well. TUPA also does not support node labels, which are ubiquitous in AMR but absent in UCCA structures (only edges are labeled in UCCA). We therefore only produce edge labels and not node labels when training on AMR.

4.2 Transition Classifier

To predict the next transition at each step, we use a BiLSTM with embeddings as inputs, followed by an MLP and a softmax layer for classification (Kiperwasser and Goldberg, 2016). The model is illustrated in Figure 2. Inference is performed greedily, and training is done with an oracle that yields the set of all optimal transitions at a given state (those that lead to a state from which the gold graph is still reachable). Out of this set, the actual transition performed in training is the one with the highest score given by the classifier, which is trained to maximize the sum of log-likelihoods of all optimal transitions at each step.

Features. We use the original TUPA features, representing the words, POS tags, syntactic dependency relations, and previously predicted edge labels for nodes in specific locations in the parser state. In addition, for each token we use embeddings representing the one-character prefix, three-character suffix, shape (capturing orthographic

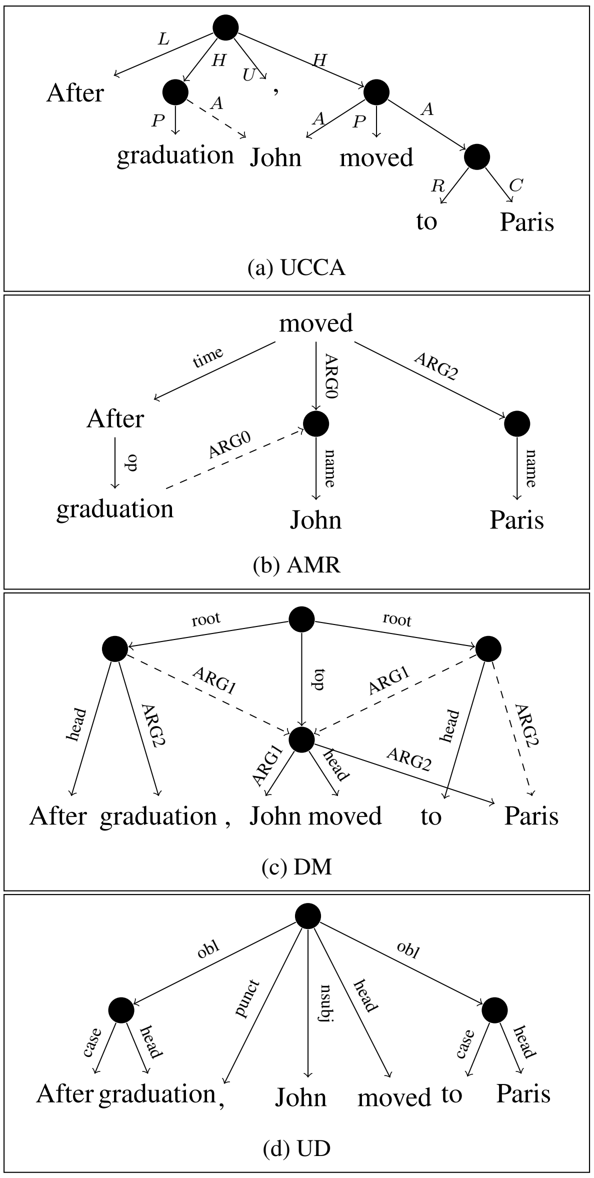


Figure 3: Graphs from Figure 1, after conversion to the unified DAG format (with pre-terminals omitted: each terminal drawn in place of its parent). Figure 3a presents a converted UCCA graph. Linkage nodes and edges are removed, but the original graph is otherwise preserved. Figure 3b presents a converted AMR graph, with text tokens added according to the alignments. Numeric suffixes of *op* relations are removed, and names collapsed. Figure 3c presents a converted SDP graph (in the DM representation), with intermediate non-terminal *head* nodes introduced. In case of reentrancy, an arbitrary reentrant edge is marked as remote. Figure 3d presents a converted UD graph. As in SDP, intermediate non-terminals and *head* edges are introduced. While converted UD graphs form trees, enhanced++ UD graphs may not.

features, e.g., "Xxxx"), and named entity type,[3] all provided by spaCy (Honnibal and Montani, 2018).[4] To the learned word vectors, we concatenate the 250K most frequent word vectors from fastText (Bojanowski et al., 2017),[5] pre-trained over Wikipedia and updated during training.

Constraints. As each annotation scheme has different constraints on the allowed graph structures, we apply these constraints separately for each task. During training and parsing, the relevant constraint set rules out some of the transitions according to the parser state. Some constraints are task-specific, others are generic. For example, in UCCA, a terminal may only have one parent. In AMR, a concept corresponding to a PropBank frame may only have the core arguments defined for the frame as children. An example of a generic constraint is that stack nodes that have been swapped should not be swapped again.[6]

5 Unified DAG Format

To apply our parser to the four target tasks (§3), we convert them into a unified DAG format, which is inclusive enough to allow representing any of the schemes with very little loss of information.[7]

The format consists of a rooted DAG, where the tokens are the terminal nodes. As in the UCCA format, edges are labeled (but not nodes), and are divided into *primary* and *remote* edges, where the primary edges form a tree (all nodes have at most one primary parent, and the root has none). Remote edges enable reentrancy, and thus together with primary edges form a DAG. Figure 3 shows examples for converted graphs. Converting UCCA into the unified format consists simply of removing linkage nodes and edges (see Figure 3a), which were also discarded by Hershcovich et al. (2017).

Converting bilexical dependencies. To convert DM and UD into the unified DAG format, we add a pre-terminal for each token, and attach the pre-terminals according to the original dependency edges: traversing the tree from the root down, for each head token we create a non-terminal parent with the edge label *head*, and add the node's dependents as children of the created non-terminal node (see Figures 3c and 3d). Since DM allows multiple roots, we form a single root node, whose

[3]See Supplementary Material for a full listing of features.

[4]`http://spacy.io`

[5]`http://fasttext.cc`

[6] To implement this constraint, we define a *swap index* for each node, assigned when the node is created. At initialization, only the root node and terminals exist. We assign the root a swap index of 0, and for each terminal, its position in the text (starting at 1). Whenever a node is created as a result of a NODE transition, its swap index is the arithmetic mean of the swap indices of the stack top and buffer head.

[7]See Supplementary Material for more conversion details.

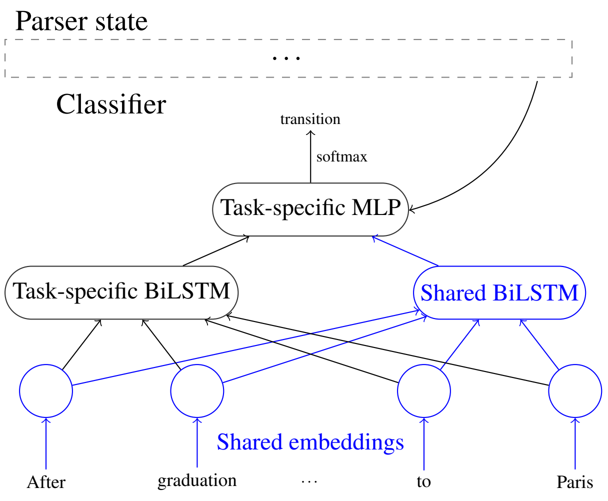


Figure 4: MTL model. Token representations are computed both by a task-specific and a shared BiLSTM. Their outputs are concatenated with the parser state embedding, identical to Figure 2, and fed into the task-specific MLP for selecting the next transition. Shared parameters are shown in blue.

children are the original roots. The added edges are labeled *root*, where top nodes are labeled *top* instead. In case of reentrancy, an arbitrary parent is marked as primary, and the rest as remote (denoted as dashed edges in Figure 3).

Converting AMR. In the conversion from AMR, node labels are dropped. Since alignments are not part of the AMR graph (see Figure 3b), we use automatic alignments (see §7), and attach each node with an edge to each of its aligned terminals.

Named entities in AMR are represented as a subgraph, whose *name*-labeled root has a child for each token in the name (see the two *name* nodes in Figure 1b). We collapse this subgraph into a single node whose children are the name tokens.

6 Multitask Transition-based Parsing

Now that the same model can be applied to different tasks, we can train it in a multitask setting. The fairly small training set available for UCCA (see §7) makes MTL particularly appealing, and we focus on it in this paper, treating AMR, DM and UD parsing as auxiliary tasks.

Following previous work, we share only some of the parameters (Klerke et al., 2016; Søgaard and Goldberg, 2016; Bollmann and Søgaard, 2016; Plank, 2016; Braud et al., 2016; Martínez Alonso and Plank, 2017; Peng et al., 2017a, 2018), leaving task-specific sub-networks as well. Concretely, we keep the BiLSTM used by TUPA for the main task (UCCA parsing), add a BiLSTM that is shared across all tasks, and replicate the MLP (feedforward sub-network) for each task. The BiLSTM outputs (concatenated for the main task) are fed into the task-specific MLP (see Figure 4). Feature embeddings are shared across tasks.

Unlabeled parsing for auxiliary tasks. To simplify the auxiliary tasks and facilitate generalization (Bingel and Søgaard, 2017), we perform unlabeled parsing for AMR, DM and UD, while still predicting edge labels in UCCA parsing. To support unlabeled parsing, we simply remove all labels from the EDGE, REMOTE and NODE transitions output by the oracle. This results in a much smaller number of transitions the classifier has to select from (no more than 10, as opposed to 45 in labeled UCCA parsing), allowing us to use no BiLSTMs and fewer dimensions and layers for task-specific MLPs of auxiliary tasks (see §7). This limited capacity forces the network to use the shared parameters for all tasks, increasing generalization (Martínez Alonso and Plank, 2017).

7 Experimental Setup

We here detail a range of experiments to assess the value of MTL to UCCA parsing, training the parser in single-task and multitask settings, and evaluating its performance on the UCCA test sets in both in-domain and out-of-domain settings.

Data. For UCCA, we use v1.2 of the English Wikipedia corpus (*Wiki*; Abend and Rappoport, 2013), with the standard train/dev/test split (see Table 1), and the *Twenty Thousand Leagues Under the Sea* corpora (*20K*; Sulem et al., 2015), annotated in English, French and German.[8] For English and French we use 20K v1.0, a small parallel corpus comprising the first five chapters of the book. As in previous work (Hershcovich et al., 2017), we use the English part only as an out-of-domain test set. We train and test on the French part using the standard split, as well as the German corpus (v0.9), which is a pre-release and still contains a considerable amount of noisy annotation. Tuning is performed on the respective development sets.

For AMR, we use LDC2017T10, identical to the dataset targeted in SemEval 2017 (May and Priyadarshi, 2017).[9] For SDP, we use the DM representation from the SDP 2016 dataset (Oepen

[8]http://github.com/huji-nlp/ucca-corpora
[9]http://catalog.ldc.upenn.edu/LDC2017T10

	English						**French**						**German**					
	# tokens			**# sentences**			**# tokens**			**# sentences**			**# tokens**			**# sentences**		
	train	**dev**	**test**	**train**	**dev**	**test**	**train**	**dev**	**test**	**train**	**dev**	**test**	**train**	**dev**	**test**	**train**	**dev**	**test**
UCCA																		
Wiki	128444	14676	15313	4268	454	503												
20K			12339			506	10047	1558	1324	413	67	67	79894	10059	42366	3429	561	2164
AMR	648950			36521														
DM	765025			33964														
UD	458277			17062			899163			32347			268145			13814		

Table 1: Number of tokens and sentences in the training, development and test sets we use for each corpus and language.

et al., 2016).[10] For Universal Dependencies, we use all English, French and German treebanks from UD v2.1 (Nivre et al., 2017).[11] We use the enhanced++ UD representation (Schuster and Manning, 2016) in our English experiments, henceforth referred to as UD^{++}. We use only the AMR, DM and UD training sets from standard splits.

While UCCA is annotated over Wikipedia and over a literary corpus, the domains for AMR, DM and UD are blogs, news, emails, reviews, and Q&A. This domain difference between training and test is particularly challenging (see §9). Unfortunately, none of the other schemes have available annotation over Wikipedia text.

Settings. We explore the following settings: (1) in-domain setting in English, training and testing on Wiki; (2) out-of-domain setting in English, training on Wiki and testing on 20K; (3) French in-domain setting, where available training dataset is small, training and testing on 20K; (4) German in-domain setting on 20K, with somewhat noisy annotation. For MTL experiments, we use unlabeled AMR, DM and UD^{++} parsing as auxiliary tasks in English, and unlabeled UD parsing in French and German.[12] We also report baseline results training only the UCCA training sets.

Training. We create a unified corpus for each setting, shuffling all sentences from relevant datasets together, but using only the UCCA development set F_1 score as the early stopping criterion. In each training epoch, we use the same number of examples from each task—the UCCA training set size. Since training sets differ in size, we sample this many sentences from each one. The model is implemented using DyNet (Neubig et al., 2017).[13]

[10]`http://sdp.delph-in.net/osdp-12.tgz`

[11]`http://hdl.handle.net/11234/1-2515`

[12]We did not use AMR, DM or UD^{++} in French and German, as these are only available in English.

[13]`http://dynet.io`

		Multitask		
Hyperparameter	**Single**	**Main**	**Aux**	**Shared**
Pre-trained word dim.	300			300
Learned word dim.	200			200
POS tag dim.	20			20
Dependency relation dim.	10			10
Named entity dim.	3			3
Punctuation dim.	1			1
Action dim.	3			3
Edge label dim.	20	20		
MLP layers	2	2	1	
MLP dimensions	50	50	50	
BiLSTM layers	2	2		2
BiLSTM dimensions	500	300		300

Table 2: Hyperparameter settings. Middle column shows hyperparameters used for the single-task architecture, described in §4.2, and right column for the multitask architecture, described in §6. **Main** refers to parameters specific to the main task—UCCA parsing (task-specific MLP and BiLSTM, and edge label embedding), **Aux** to parameters specific to each auxiliary task (task-specific MLP, but no edge label embedding since the tasks are unlabeled), and **Shared** to parameters shared among all tasks (shared BiLSTM and embeddings).

Hyperparameters. We initialize embeddings randomly. We use dropout (Srivastava et al., 2014) between MLP layers, and recurrent dropout (Gal and Ghahramani, 2016) between BiLSTM layers, both with $p = 0.4$. We also use word ($\alpha = 0.2$), tag ($\alpha = 0.2$) and dependency relation ($\alpha = 0.5$) dropout (Kiperwasser and Goldberg, 2016).[14] In addition, we use a novel form of dropout, *node dropout*: with a probability of 0.1 at each step, all features associated with a single node in the parser state are replaced with zero vectors. For optimization we use a minibatch size of 100, decaying all weights by 10^{-5} at each update, and train with stochastic gradient descent for N epochs with a learning rate of 0.1, followed by AMSGrad (Sashank J. Reddi, 2018) for N epochs with $\alpha = 0.001, \beta_1 = 0.9$ and $\beta_2 = 0.999$. We use $N = 50$ for English and German, and $N = 400$ for French. We found this training strategy better than using only one of the optimization methods,

[14]In training, the embedding for a feature value w is replaced with a zero vector with a probability of $\frac{\alpha}{\#(w)+\alpha}$, where $\#(w)$ is the number of occurrences of w observed.

	Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
English (in-domain)						
HAR17	74.4	72.7	73.5	47.4	51.6	49.4
Single	74.4	72.9	73.6	53	50	51.5
AMR	74.7	72.8	73.7	48.7$\star$	51.1	49.9
DM	75.7$\star$	73.9$\star$	74.8$\star$	54.9	53	**53.9**
UD^{++}	75$\star$	73.2	74.1$\star$	49	52.7	50.8
AMR + DM	75.6$\star$	73.9$\star$	74.7$\star$	49.9	53	51.4
AMR + UD^{++}	74.9	72.7	73.8	47.1	50	48.5
DM + UD^{++}	75.9$\star$	73.9$\star$	**74.9**$\star$	48	54.8	51.2
All	75.6$\star$	73.1	74.4$\star$	50.9	53.2	52

Table 3: Labeled precision, recall and F_1 (in %) for primary and remote edges, on the **Wiki** test set. $\star$ indicates significantly better than *Single*. HAR17: Hershcovich et al. (2017).

	Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**
English (out-of-domain)						
HAR17	68.7	68.5	68.6	38.6	18.8	25.3
Single	69	69	69	41.2	19.8	26.7
AMR	69.5	69.5	69.5	42.9	20.2	27.5
DM	70.7$\star$	70.7$\star$	70.7$\star$	42.7	18.6	25.9
UD^{++}	69.6	69.8$\star$	69.7	41.4	22	28.7
AMR + DM	70.7$\star$	70.2$\star$	70.5$\star$	45.8	19.4	27.3
AMR + UD^{++}	70.2$\star$	69.9$\star$	70$\star$	45.1	21.8	29.4
DM + UD^{++}	70.8$\star$	70.3$\star$	70.6$\star$	41.6	21.6	28.4
All	71.2$\star$	70.9$\star$	**71**$\star$	45.1	22	**29.6**
French (in-domain)						
Single	68.2	67	67.6	26	9.4	13.9
UD	70.3	70$\star$	**70.1**$\star$	43.8	13.2	20.3
German (in-domain)						
Single	73.3	71.7	72.5	57.1	17.7	27.1
UD	73.7$\star$	72.6$\star$	**73.2**$\star$	61.8	24.9$\star$	**35.5**$\star$

Table 4: Labeled precision, recall and F_1 (in %) for primary and remote edges, on the **20K** test sets. $\star$ indicates significantly better than *Single*. HAR17: Hershcovich et al. (2017).

similar to findings by Keskar and Socher (2017). We select the epoch with the best average labeled F_1 score on the UCCA development set. Other hyperparameter settings are listed in Table 2.

Evaluation. We evaluate on UCCA using labeled precision, recall and F_1 on primary and remote edges, following previous work (Hershcovich et al., 2017). Edges in predicted and gold graphs are matched by terminal yield and label. Significance testing of improvements over the single-task model is done by the bootstrap test (Berg-Kirkpatrick et al., 2012), with $p < 0.05$.

8 Results

Table 3 presents our results on the English in-domain Wiki test set. MTL with all auxiliary tasks and their combinations improves the primary F_1 score over the single task baseline. In most settings the improvement is statistically significant. Using all auxiliary tasks contributed less than just DM and UD^{++}, the combination of which yielded the best scores yet in in-domain UCCA parsing, with 74.9% F_1 on primary edges. Remote F_1 is improved in some settings, but due to the relatively small number of remote edges (about 2% of all edges), none of the differences is significant. Note that our baseline single-task model (*Single*) is slightly better than the current state-of-the-art (HAR17; Hershcovich et al., 2017), due to the incorporation of additional features (see §4.2).

Table 4 presents our experimental results on the 20K corpora in the three languages. For English out-of-domain, improvements from using MTL are even more marked. Moreover, the improvement is largely additive: the best model, using all three auxiliary tasks (*All*), yields an error reduction of 2.9%. Again, the single-task baseline is slightly better than HAR17.

The contribution of MTL is also apparent in French and German in-domain parsing: 3.7% error reduction in French (having less than 10% as much UCCA training data as English) and 1% in German, where the training set is comparable in size to the English one, but is noisier (see §7). The best MTL models are significantly better than single-task models, demonstrating that even a small training set for the main task may suffice, given enough auxiliary training data (as in French).

9 Discussion

Quantifying the similarity between tasks. Task similarity is an important factor in MTL success (Bingel and Søgaard, 2017; Martínez Alonso and Plank, 2017). In our case, the main and auxiliary tasks are annotated on different corpora from different domains (§7), and the target representations vary both in form and in content.

To quantify the domain differences, we follow Plank and van Noord (2011) and measure the L1 distance between word distributions in the English training sets and 20K test set (Table 5). All auxiliary training sets are more similar to 20K than Wiki is, which may contribute to the benefits observed on the English 20K test set.

As a measure of the formal similarity of the different schemes to UCCA, we use unlabeled F_1 score evaluation on both primary and remote edges (ignoring edge labels). To this end, we annotated 100 English sentences from Section 02 of the Penn Treebank Wall Street Journal (PTB WSJ). Anno-

	20K	AMR	DM	UD
Wiki	1.047	0.895	0.913	0.843
20K		0.949	0.971	0.904
AMR			0.757	0.469
DM				0.754

Table 5: L1 distance between dataset word distributions, quantifying domain differences in English (low is similar).

	Primary			**Remote**		
	UP	**UR**	**UF**	**UP**	**UR**	**UF**
AMR	53.8	15.6	24.2	7.3	5.5	6.3
DM	65	49.2	56	7.4	65.9	13.3
UD^{++}	82.7	84.6	83.6	12.5	12.7	12.6

Table 6: Unlabeled F_1 scores between the representations of the same English sentences (from PTB WSJ), converted to the unified DAG format, and annotated UCCA graphs.

tation was carried out by a single expert UCCA annotator, and is publicly available.[15] These sentences had already been annotated by the AMR, DM and PTB schemes,[16] and we convert their annotation to the unified DAG format.

Unlabeled F_1 scores between the UCCA graphs and those converted from AMR, DM and UD^{++} are presented in Table 6. UD^{++} is highly overlapping with UCCA, while DM less so, and AMR even less (cf. Figure 3).

Comparing the average improvements resulting from adding each of the tasks as auxiliary (see §8), we find AMR the least beneficial, UD^{++} second, and DM the most beneficial, in both in-domain and out-of-domain settings. This trend is weakly correlated with the formal similarity between the tasks (as expressed in Table 6), but weakly negatively correlated with the word distribution similarity scores (Table 5). We conclude that other factors should be taken into account to fully explain this effect, and propose to address this in future work through controlled experiments, where corpora of the same domain are annotated with the various formalisms and used as training data for MTL.

AMR, SDP and UD parsing. Evaluating the full MTL model (*All*) on the unlabeled auxiliary tasks yielded 64.7% unlabeled Smatch F_1 (Cai and Knight, 2013) on the AMR development set, when using oracle concept identification (since the auxiliary model does not predict node labels), 27.2% unlabeled F_1 on the DM development set, and 4.9% UAS on the UD development set. These poor results reflect the fact that model selection was based on the score on the UCCA development set, and that the model parameters dedicated to auxiliary tasks were very limited (to encourage using the shared parameters). However, preliminary experiments using our approach produced promising results on each of the tasks' respective English development sets, when treated as a single task: 67.1% labeled Smatch F_1 on AMR (adding a transition for implicit nodes and classifier for node labels), 79.1% labeled F_1 on DM, and 80.1% LAS F_1 on UD. For comparison, the best results on these datasets are 70.7%, 91.2% and 82.2%, respectively (Foland and Martin, 2017; Peng et al., 2018; Dozat et al., 2017).

10 Conclusion

We demonstrate that semantic parsers can leverage a range of semantically and syntactically annotated data, to improve their performance. Our experiments show that MTL improves UCCA parsing, using AMR, DM and UD parsing as auxiliaries. We propose a unified DAG representation, construct protocols for converting these schemes into the unified format, and generalize a transition-based DAG parser to support all these tasks, allowing it to be jointly trained on them.

While we focus on UCCA in this work, our parser is capable of parsing any scheme that can be represented in the unified DAG format, and preliminary results on AMR, DM and UD are promising (see §9). Future work will investigate whether a single algorithm and architecture can be competitive on all of these parsing tasks, an important step towards a joint many-task model for semantic parsing.

Acknowledgments

This work was supported by the Israel Science Foundation (grant no. 929/17), by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI). The first author was supported by a fellowship from the Edmond and Lily Safra Center for Brain Sciences. We thank Roi Reichart, Rotem Dror and the anonymous reviewers for their helpful comments.

[15] `http://github.com/danielhers/wsj`

[16] We convert the PTB format to UD^{++} v1 using Stanford CoreNLP, and then to UD v2 using Udapi: `http://github.com/udapi/udapi-python`.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL*, pages 77–89.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *TACL*, 4:431–444.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proc. of EMNLP*, pages 1699–1710.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proc. of EMNLP*, pages 1269–1275.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the Linguistic Annotation Workshop*.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proc. of SemEval*, pages 1143–1147.

Eric Baucom, Levi King, and Sandra Kübler. 2013. Domain adaptation for parsing. In *Proc. of RANLP*, pages 56–64.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proc. of EMNLP-CoNLL*, pages 995–1005.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proc. of EACL*, pages 164–169.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120–128.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP-CoNLL*, pages 1455–1465.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *Proc. of COLING*, pages 131–139.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *Proc. of COLING*, pages 1903–1913.

Jan Buys and Phil Blunsom. 2017a. Oxford at SemEval-2017 task 9: Neural AMR parsing with pointer-augmented attention. In *Proc. of SemEval*, pages 914–919.

Jan Buys and Phil Blunsom. 2017b. Robust incremental neural semantic graph parsing. In *Proc. of ACL*, pages 1215–1226.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*, pages 748–752.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41–75.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proc. of ACL*, pages 161–171.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proc. of LREC*, pages 591–600.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proc. of EACL*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*, pages 256–263.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proc. of CoNLL*, pages 20–30.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proc. of SemEval*, pages 927–931.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip Cohen, and Mark Johnson. 2017. Multilingual semantic parsing and code-switching. In *Proc. of CoNLL*, pages 379–389.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proc. of Workshop on Representation Learning for NLP*, pages 48–56.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proc. of NAACL-HLT*, pages 326–334.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proc. of ACL*, pages 1426–1436.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proc. of Workshop on Treebanks and Linguistic Theories*, pages 85–96.

William Foland and James H. Martin. 2017. Abstract Meaning Representation parsing using LSTM recurrent neural networks. In *Proc. of ACL*, pages 463–472.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proc. of ACL*, pages 1–11.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. Exploiting multi-typed treebanks for parsing with deep multi-task learning. *CoRR*, abs/1606.01161.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL*, pages 1–18.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proc. of EMNLP*, pages 1923–1933.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proc. of ACL*, pages 623–628.

Matthew Honnibal and Ines Montani. 2018. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Nitish Shirish Keskar and Richard Socher. 2017. Improving generalization performance by switching from Adam to SGD. *CoRR*, abs/1712.07628.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proc. of NAACL-HLT*, pages 1528–1533.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proc. of ACL*, pages 146–157.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proc. of EMNLP*, pages 1444–1454.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *Proc. of CoNLL*, pages 188–192.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proc. of Workshop on Discontinuous Structures in NLP*, pages 47–57.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *Proc. of EACL*, pages 44–53.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proc. of SemEval*, pages 1063–1073.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proc. of SemEval*, pages 536–545.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proc. of NAACL-HLT*, pages 28–36.

Amir More. 2016. Joint morpho-syntactic processing of morphologically rich languages in a transition-based framework. Master's thesis, The Interdisciplinary Center, Herzliya.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *CoRR*, abs/1701.03980.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*, pages 149–160.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà M, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phng Lê Hng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, pages 915–926.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, pages 63–72.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*, pages 2037–2048.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL-HLT*.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural AMR parsing. In *Proc. of EACL*, pages 366–375.

Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *Proc. of COLING*, pages 609–619.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *Proc. of ACL-HLT*, pages 1566–1576.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into Abstract Meaning Representation using syntax-based machine translation. In *Proc. of EMNLP*, pages 1143–1154.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proc. of COLING*, pages 753–760.

Sanjiv Kumar Sashank J. Reddi, Satyen Kale. 2018. On the convergence of Adam and beyond. *ICLR*.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proc. of LREC*. ELRA.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*, pages 231–235.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *Proc. of S2MT*, pages 11–22.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL*, pages 159–177.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*, pages 187–197.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *CoRR*, abs/1706.09528.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of ACL*, pages 589–596.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In *Proc. of SemEval*, pages 1173–1178.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proc. of ACL*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proc. of NAACL*, pages 366–375.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*, pages 1557–1566.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proc. of IWPT*, pages 162–171.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proc. of EMNLP*, pages 680–689.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proc. of CoNLL*, pages 400–410.

Character-Level Models versus Morphology in Semantic Role Labeling

Gözde Gül Şahin
Department of Computer Science
Technische Universität Darmstadt
Darmstadt, Germany
isguderg@itu.edu.tr

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, Scotland
steedman@inf.ed.ac.uk

Abstract

Character-level models have become a popular approach specially for their accessibility and ability to handle unseen data. However, little is known on their ability to reveal the underlying morphological structure of a word, which is a crucial skill for high-level semantic analysis tasks, such as semantic role labeling (SRL). In this work, we train various types of SRL models that use word, character and morphology level information and analyze how performance of characters compare to words and morphology for several languages. We conduct an in-depth error analysis for each morphological typology and analyze the strengths and limitations of character-level models that relate to out-of-domain data, training data size, long range dependencies and model complexity. Our exhaustive analyses shed light on important characteristics of character-level models and their semantic capability.

1 Introduction

Encoding of words is perhaps the most important step towards a successful end-to-end natural language processing application. Although word embeddings have been shown to provide benefit to such models, they commonly treat words as the smallest meaning bearing unit and assume that each word type has its own vector representation. This assumption has two major shortcomings especially for languages with rich morphology: (1) inability to handle unseen or out-of-vocabulary (OOV) word-forms (2) inability to exploit the regularities among word parts.

The limitations of word embeddings are particularly pronounced in sentence-level semantic tasks, especially in languages where word parts play a crucial role. Consider the Turkish sentences "*Köy+**lü-ler** (villagers) şehr+**e** (to town) geldi (came)*" and "*Sendika+**lı-lar** (union members) meclis+**e** (to council) geldi (came)*". Here the stems *köy (village)* and *sendika (union)* function similarly in semantic terms with respect to the verb *come* (as *the origin of the agents of the verb*), where *şehir (town)* and *meclis (council)* both function as *the end point*. These semantic similarities are determined by the common word parts shown in **bold**. However ortographic similarity does not always correspond to semantic similarity. For instance the ortographically similar words *knight* and *night* have large semantic differences. Therefore, for a successful semantic application, the model should be able to capture both the regularities, *i.e, morphological tags* and the irregularities, *i.e, lemmas* of the word.

Morphological analysis already provides the aforementioned information about the words. However access to useful morphological features may be problematic due to software licensing issues, lack of robust morphological analyzers and high ambiguity among analyses. Character-level models (CLM), being a cheaper and accessible alternative to morphology, have been reported as performing competitively on various NLP tasks (Ling et al., 2015; Plank et al., 2016; Lee et al., 2017). However the extent to which these tasks depend on morphology is small; and their relation to semantics is weak. Hence, little is known on their true ability to reveal the underlying morphological structure of a word and their semantic capabilities. Furthermore, their behaviour across languages from different families; and their limitations and strengths such as handling of long-range dependencies, reaction to model complexity or performance on out-of-domain data are unknown. Analyzing such issues is a key to fully understanding the character-level models.

To achieve this, we perform a case study on semantic role labeling (SRL), a sentence-level semantic analysis task that aims to identify predicate-argument structures and assign meaningful labels to them as follows:

[Villagers]$_{\text{comers}}$ came [to town]$_{\text{end point}}$

We use a simple method based on bidirectional LSTMs to train three types of base semantic role labelers that employ (1) words (2) characters and character sequences and (3) gold morphological analysis. The gold morphology serves as the upper bound for us to compare and analyze the performances of character-level models on languages of varying morphological typologies. We carry out an exhaustive error analysis for each language type and analyze the strengths and limitations of character-level models compared to morphology. In regard to the diversity hypothesis which states that *diversity* of systems in ensembles lead to further improvement, we combine character and morphology-level models and measure the performance of the ensemble to better understand how similar they are.

We experiment with several languages with varying degrees of morphological richness and typology: Turkish, Finnish, Czech, German, Spanish, Catalan and English. Our experiments and analysis reveal insights such as:

- CLMs provide great improvements over whole-word-level models despite not being able to match the performance of morphology-level models (MLMs) for *in-domain* datasets. However their performance surpass all MLMs on *out-of-domain* data,
- Limitations and strengths differ by morphological typology. Their limitations for agglutinative languages are related to rich *derivational morphology* and high *contextual ambiguity*; whereas for fusional languages they are related to *number of morphological tags* (morpheme ambiguity) ,
- CLMs can handle long-range dependencies equally well as MLMs,
- In presence of more training data, CLM's performance is expected to improve faster than of MLM.

2 Related Work

Neural SRL Methods: Neural networks have been first introduced to the SRL scene by Collobert et al. (2011), where they use a unified end-to-end convolutional network to perform various NLP tasks. Later, the combination of neural networks (LSTMs in particular) with traditional SRL features (categorical and binary) has been introduced (FitzGerald et al., 2015). Recently, it has been shown that careful design and tuning of deep models can achieve state-of-the-art with no or minimal syntactic knowledge for English and Chinese SRL. Although the architectures vary slightly, they are mostly based on a variation of bi-LSTMs. Zhou and Xu (2015); He et al. (2017) connect the layers of LSTM in an interleaving pattern where in (Wang et al., 2015; Marcheggiani et al., 2017) regular bi-LSTM layers are used. Commonly used features for the encoding layer are: pretrained word embeddings; distance from the predicate; predicate context; predicate region mark or flag; POS tag; and predicate lemma embedding. Only a few of the models (Marcheggiani et al., 2017; Marcheggiani and Titov, 2017) perform dependency-based SRL. Furthermore, all methods focus on languages with rich resources and less morphological complexity like English and Chinese.

Character-level Models: Character-level models have proven themselves useful for many NLP tasks such as language modeling (Ling et al., 2015; Kim et al., 2016), POS tagging (Santos and Zadrozny, 2014; Plank et al., 2016), dependency parsing (Dozat et al., 2017) and machine translation (Lee et al., 2017). However the number of comparative studies that analyze their relation to morphology are rather limited. Recently, Vania and Lopez (2017) presented a unified framework, where they investigated the performances of different subword units, namely characters, morphemes and morphological analysis on language modeling task. They experimented with languages of varying morphological typologies and concluded that the performance of character models can not yet match the morphological models, albeit very close. Similarly, Belinkov et al. (2017) analyzed how different word representations help learn better morphology and model rare words on a neural MT task and concluded that character-based representations are much better for learning morphology.

3 Method

Formally, we generate a label sequence $\vec{l}$ for each sentence and predicate pair: (s, p). Each $l_t \in \vec{l}$ is chosen from $\mathcal{L} = \{roles \cup nonrole\}$, where $roles$ are language-specific semantic roles (mostly consistent with PropBank) and $nonrole$ is a symbol to present tokens that are not arguments. Given θ as model parameters and g_t as gold label for t_{th} token, we find the parameters that minimize the negative log likelihood of the sequence:

$$\hat{\theta} = \arg\min_{\theta} \left(-\sum_{t=1}^{n} log(p(g_t|\theta, s, p)) \right) \quad (1)$$

Label probabilities, $p(l_t|\theta, s, p)$, are calculated with equations given below.First, the word encoding layer splits tokens into subwords via ρ function.

$$\rho(w) = s_0, s_1, .., s_n \quad (2)$$

As proposed by Ling et al. (2015), we treat words as a sequence of subword units. Then, the sequence is fed to a simple bi-LSTM network (Graves and Schmidhuber, 2005; Gers et al., 2000) and hidden states from each direction are weighted with a set of parameters which are also learned during training. Finally, the weighted vector is used as the word embedding given in Eq. 4.

$$hs_f, hs_b = \text{bi-LSTM}(s_0, s_1, .., s_n) \quad (3)$$
$$\vec{w} = W_f \cdot hs_f + W_b \cdot hs_b + b \quad (4)$$

There may be more than one predicate in the sentence so it is crucial to inform the network of which arguments we aim to label. In order to mark the predicate of interest, we concatenate a predicate flag pf_t to the word embedding vector.

$$\vec{x_t} = [\vec{w}; pf_t] \quad (5)$$

Final vector, $\vec{x_t}$ serves as an input to another bi-LSTM unit.

$$\vec{h_f}, \vec{h_b} = \text{bi-LSTM}(x_t) \quad (6)$$

Finally, the label distribution is calculated via softmax function over the concatenated hidden states from both directions.

$$p(\vec{l_t}|s, p) = softmax(W_l \cdot [\vec{h_f}; \vec{h_b}] + \vec{b_l}) \quad (7)$$

For simplicity, we assign the label with the highest probability to the input token. [1].

3.1 Subword Units

We use three types of units: (1) words (2) characters and character sequences and (3) outputs of morphological analysis. Words serve as a lower bound; while morphology is used as an upper bound for comparison. Table 1 shows sample outputs of various ρ functions. Here, *char* function

ρ	**word**	**output**
char	available	<-a-v-a-i-l-a-b-l-e->
char3	available	<av-ava-vai-ail-ila-lab-abl-ble-le>
morph-DEU	prächtiger	[*prächtig*;Pos;Nom;Sg;Masc]
morph-SPA	las	[*el*;postype=article;gen=f;num=p]
morph-CAT	la	[*el*;postype=article;gen=f;num=s]
morph-TUR	boyundaki	[*boy*;NOUN;A3sg;P3sg;Loc;DB;ADJ]
morph-FIN	tyhjyyttä	[*tyhjyys*;Case=Par;Number=Sing]
morph-CZE	si	[*se*;SubPOS=7;Num=X;Cas=3]

Table 1: Sample outputs of different ρ functions

simply splits the token into its characters. Similar to n-gram language models, *char3* slides a character window of width $n = 3$ over the token. Finally, gold morphological features are used as outputs of *morph-language*. Throughout this paper, we use *morph* and *oracle* interchangably, i.e., morphology-level models (MLM) have access to gold tags unless otherwise is stated. For all languages, *morph* outputs the *lemma* of the token followed by language specific morphological tags. As an exception, it outputs additional information for some languages, such as parts-of-speech tags for Turkish. Word segmenters such as Morfessor and Byte Pair Encoding (BPE) are other commonly used subword units. Due to low scores obtained from our preliminary experiments and unsatisfactory results from previous studies (Vania and Lopez, 2017), we excluded these units.

4 Experiments

We use the datasets distributed by LDC for Catalan (CAT), Spanish (SPA), German (DEU), Czech (CZE) and English (ENG) (Hajič et al., 2012b,a); and datasets made available by Haverinen et al. (2015); Şahin and Adalı (2017) for Finnish (FIN) and Turkish (TUR) respectively [2]. Datasets are

[1]Our implementation can be found at https://github.com/gozdesahin/Subword_Semantic_Role_Labeling

[2]Turkish PropBank is based on previous efforts (Atalay et al., 2003; Sulubacak et al., 2016; Sulubacak and Eryiğit, 2018; Oflazer et al., 2003; Şahin, 2016b,a)

	#sent	#token	#pred	#role	type
CZE	39K	653K	414K	51	F
ENG	39K	958K	179K	38	F
DEU	36K	649K	17K	9	F
SPA	14K	419K	44K	34	F
CAT	13K	384K	37K	35	F
FIN	12K	163K	27K	20	A
TUR	4K	39K	8K	26	A

Table 2: Training data statistics. A: Agglutinative, F: Fusional

provided with syntactic dependency annotations and semantic roles of verbal predicates. In addition, English supplies nominal predicates annotated with semantic roles and does not provide any morphological feature. Statistics for the training split for all languages are given in Table 2. Here, **#pred** is number of predicates, and **#role** refers to number distinct semantic roles that occur more than 10 times. More detailed statistics about the datasets can be found in Hajič et al. (2009); Haverinen et al. (2015); Şahin and Adalı (2017).

4.1 Experimental Setup

To fit the requirements of the SRL task and of our model, we performed the following:

Spanish, Catalan: Multiword expressions (MWE) are represented as a single token, *(e.g., Confederación_Francesa_del_Trabajo)*, that causes notably long character sequences which are hard to handle by LSTMs. For the sake of memory efficiency and performance, we used an abbreviation *(e.g., CFdT)* for each MWE during training and testing.

Finnish: Original dataset defines its own format of semantic annotation, such as 17:PBArgM_mod|19:PBArgM_mod meaning the node is an argument of 17_{th} and 19_{th} tokens with *ArgM-mod* (temporary modifier) semantic role. They have been converted into CoNLL-09 tabular format, where each predicate's arguments are given in a specific column.

Turkish: Words are splitted from derivational boundaries in the original dataset, where each inflectional group is represented as a separate token. We first merge boundaries of the same word, *i.e, tokens of the word*, then we use our own ρ function to split words into subwords.

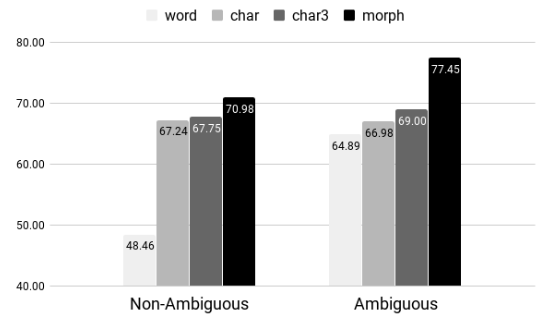
Training and Evaluation: We lowercase all tokens beforehand and place special start and end of the token characters. For all experiments, we initialized weight parameters orthogonally and used one layer bi-LSTMs both for subword composition and argument labeling with hidden size of 200. Subword embedding size is chosen as 200. We used gradient clipping and early stopping to prevent overfitting. Stochastic gradient descent is used as the optimizer. The initial learning rate is set to 1 and reduced by half if scores on development set do not improve after 3 epochs. We use the provided splits and evaluate the results with the official evaluation script provided by CoNLL-09 shared task. In this work (and in most of the recent SRL works), only the scores for argument labeling are reported, which may cause confusions for the readers while comparing with older SRL studies. Most of the early SRL work report combined scores (argument labeling with predicate sense disambiguation (PSD)). However, PSD is considered a simpler task with higher F1 scores [3]. Therefore, we believe omitting PSD helps us gain more useful insights on character level models.

5 Results and Analysis

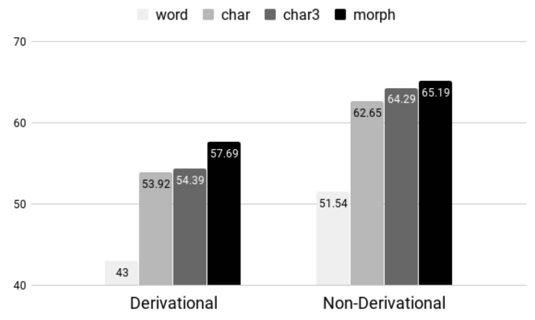
Our main results on test and development sets for models that use words, characters (*char*), character trigrams (*char3*) and morphological analyses (*morph*) are given in Table 3. We calculate *improvement over word (IOW)* for each subword model and *improvement over the best character model (IOC)* for the *morph*. *IOW* and *IOC* values are calculated on the test set.

The biggest improvement over the word baseline is achieved by the models that have access to morphology for all languages (except for English) as expected. Character trigrams consistently outperformed characters by a small margin. Same pattern is observed on the results of the development set. *IOW* has the values between 0% to 38% while *IOC* values range between 2%-10% dependending on the properties of the language and the dataset. We analyze the results separately for agglutinative and fusional languages and reveal the links between certain linguistic phenomena and the *IOC*, *IOW* values.

[3]For instance in English CoNLL-09 dataset, 87% of the predicates are annotated with their first sense, hence even a dummy classifier would achieve 87% accuracy. The best system from CoNLL-09 shared task reports 85.63 F1 on English evaluation dataset, however when the results of PSD are discarded, it drops down to 81.



(a) Finnish - Contextual ambiguity



(b) Turkish - Derivational morphology

Figure 1: Differences in model performances on agglutinative languages

	word	char		char3		morph		
	F1	F1	*IOW%*	F1	*IOW%*	F1	*IOW%*	*IOC%*
FIN	48.91	67.24	*37.46*	67.78	*38.58*	**71.15**	*45.47*	*4.97*
	51.65	66.82		67.08		**71.88**		
TUR	44.82	55.89	*24.68*	56.60	*26.28*	**59.38**	*32.48*	*4.91*
	43.14	54.48		55.41		**58.91**		
SPA	64.30	67.90	*5.61*	68.43	*6.42*	**69.39**	*7.92*	*2.25*
	64.53	67.64		67.64		**69.17**		
CAT	65.45	70.56	*7.82*	71.34	*9.00*	**73.24**	*11.90*	*2.66*
	65.67	70.43		70.48		**72.36**		
CZE	63.58	74.04	*16.45*	74.98	*17.93*	**80.66**	*26.87*	*7.58*
	72.69	74.58		75.59		**81.06**		
DEU	54.78	63.71	*16.29*	65.56	*19.68*	**69.35**	*26.58*	*5.77*
	53.76	62.75		63.70		**72.18**		
ENG	81.19	**81.61**	*0.52*	80.65	*-0.67*	-	-	-
	78.67	**79.22**		78.85		-	-	-

Table 3: F1 scores of word, character, character trigram and morphology models for argument labeling. Best F1 for each language is shown in **bold**. First row: results on test, Second row: results on development.

Agglutinative languages have many morphemes attached to a word like beads on a string. This leads to high number of OOV words and cause word lookup models to fail. Hence, the highest *IOW*s by character models are achieved on these languages: Finnish and Turkish. This language family has one-to-one morpheme to meaning mapping with small orthographic differences *(e.g., mış, miş, muş, müş for past perfect tense)*, that can be easily extracted from the data. Even though each morpheme has only one interpretation, each word (consisting of many morphemes) has usually more than one. For instance two possible analyses for the Turkish word “dolar” are (1) “dol+Verb+Positive+Aorist+3sg” *(it fills)*, (2) “dola+Verb+Positive+Aorist+3sg” *(he/she wraps)*. For a syntactic task, models are not obliged to learn the difference between the two; whereas for a semantic task like SRL, they are. We will refer to this issue as *contextual ambiguity*. Another important linguistic issue for agglutinative languages is the complex interaction between morphology and syntax, which is usually achieved via derivational morphemes. In other words, unlike *inflectional* morphemes that only give information on *word-level semantics*, derivational morphemes provide more clues on *sentence-level semantics*. The effects of these two phenomena on model performances is shown in Fig. 1. Scores given in Fig. 1 are absolute F1 scores for each model. For the analysis in Fig. 1a, we separately calculated F1 scores of each model on words that have been observed with at least two different set of morphological features (*ambiguous*), and one set of features (*non-ambiguous*). Due to the low number of ambiguous words in Turkish dataset ($\leq$100), it has been calculated for Finnish only. Similarly, for the derivational morphology analysis in Fig. 1b, we have separately calculated scores for sentences containing derived words (*derivational*), and simple sentences without any derivations. Both analyses show that access to gold morphological tags (*oracle*) provided big performance gains on arguments with contextual ambiguity and sentences with derived words. Moderate *IOC* signals that *char* and *char3* learns to imitate the “beads” and their “predictable order” on the string (in the absence of the aforementioned issues).

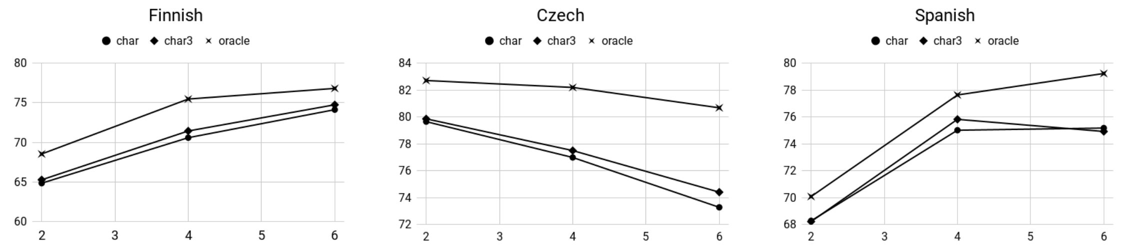


Figure 2: *x axis*: Number of morphological features; *y axis*: Targeted F1 scores

Fusional languages *may* have many morphemes in a word. Spanish and Catalan have relatively low morpheme per word ratio that results with low OOV% (5.63 and 5.40 for Spanish and Catalan respectively); whereas, German and Czech have OOV% of 7.93 and 7.98 (Hajič et al., 2009). We observe that *IOW* by character models are well aligned with OOV percentages of the datasets. Unlike agglutinative languages, single morpheme can serve multiple purposes in fusional languages. For instance, "o" (e.g., *habl-o*) may signal 1_{st} person singular present tense, or 3_{rd} person singular past tense. We count the number of surface forms with at least two different features and use their ratio *(#ambiguous forms/#total forms)* as a proxy to morphological complexity of the language. The *complexities* are approximated as 22%, 16%, 15% for Czech, Spanish and Catalan respectively; which are aligned with the observed *IOC*s. Since there is no unique morpheme to meaning mapping, generally multiple morphological tags are used to resolve the *morpheme ambiguity*. Therefore there is an indirect relation between the number of morphological tags used and the ambiguity of the word. To demonstrate this phenomena, we calculate targeted F1 scores on arguments with varying number of morphological features. Results using feature bins of [1-2], [3-4] and [5-6] are given in Fig. 2. As the number of features increase, the performance gap between oracle and character models grows dramatically for Czech and Spanish, while it stays almost fixed for Finnish. This finding suggests that high number of morphological tags signal the vagueness/complex cases in fusional languages where character models struggle; and also shows that the complexity can not be directly explained by number of morphological tags for agglutinative languages. German is known for having many compound words and compound lemmas that lead to high OOV% for lemma; and also is less ambiguous (9%). Therefore we would expect a lower *IOC*. However, the evaluation set consists only of 550 predicates and 1073 arguments, hence small changes in prediction lead to dramatic percentage changes.

5.1 Similarity between models

One way to infer similarity is to measure *diversity*. Consider a set of baseline models that are not diverse, i.e., making similar errors with similar inputs. In such a case, combination of these models would not be able to overcome the biases of the learners, hence the combination would not achieve a better result. In order to test if character and morphological models are *similar*, we combine them and measure the performance of the ensemble. Suppose that a prediction p_i is generated for each token by a model m_i, $i \in n$, then the final prediction is calculated from these predictions by:

$$p_{final} = f(p_0, p_1, .., p_n | \phi) \tag{8}$$

where f is the combining function with parameter ϕ. The simplest global approach is *averaging (AVG)*, where f is simply the mean function and p_is are the log probabilities. Mean function combines model outputs linearly, therefore ignores the nonlinear relation between base models/units. In order to exploit nonlinear connections, we learn the parameters ϕ of f via a simple linear layer followed by sigmoid activation. In other words, we train a new model that learns how to best combine the predictions from subword models. This ensemble technique is generally referred to as *stacking* or *stacked generalization (SG)*. [4]

Although not guaranteed, diverse models can be achieved by altering the input representation,

[4]To train the SG model, we have used one linear layer with 64 hidden units followed by sigmoid nonlinear activation. Weights are orthogonally initialized and optimized via adam algorithm with a learning rate of 0.02 for 25 epochs.

	char+char3			char+oracle			char3+oracle		
	Avg	SG	*IOB%*	Avg	SG	*IOB%*	Avg	SG	*IOB%*
Czech	76.24	76.26	**2.03**	80.36	81.06	*0.49*	80.57	81.10	*0.55*
Finnish	70.31	70.29	**4.58**	72.73	72.88	*2.42*	72.72	73.02	*2.62*
Turkish	59.43	59.39	**6.34**	61.98	62.07	*4.53*	60.56	60.74	*2.28*
Spanish	70.01	70.05	*3.16*	71.80	71.75	**3.47**	71.64	71.62	**3.24**
Catalan	72.79	72.71	*2.03*	74.80	74.82	**2.16**	75.15	75.18	**2.66**
German	66.84	66.97	*2.15*	71.02	71.16	**2.62**	71.31	71.25	**2.84**

Table 4: Results of ensembling via averaging (Avg) and stack generalization (SG). *IOB: Improvement Over Best of baseline models*

the learning algorithm, training data or the hyper-parameters. To ensure that the only factor contributing to the diversity of the learners is the input representation, all parameters, training data and model settings are left unchanged.

Our results are given in Table 4. *IOB* shows the improvement over the best of the baseline models in the ensemble. Averaging and stacking methods gave similar results, meaning that there is no immediate nonlinear relations between units. We observe two language clusters: (1) Czech and agglutinative languages (2) Spanish, Catalan, German and English. The common property of that separate clusters are (1) high OOV% and (2) relatively low OOV%. Amongst the first set, we observe that the improvement gained by character-morphology ensembles is higher (shown with green) than ensembles between characters and character trigrams (shown with red), whereas the opposite is true for the second set of languages. It can be interpreted as character level models being more similar to the morphology level models for the first cluster, i.e., languages with high OOV%, and characters and morphology being more diverse for the second cluster.

6 Limitations and Strengths

To expand our understanding and reveal the limitations and strengths of the models, we analyze their ability to handle long range dependencies, their relation with training data and model size; and measure their performances on out of domain data.

6.1 Long Range Dependencies

Long range dependency is considered as an important linguistic issue that is hard to solve. Therefore the ability to handle it is a strong performance indicator. To gain insights on this issue, we measure how models perform as the distance between the predicate and the argument increases. The unit of measure is number of tokens between the two; and argument is defined as the head of the argument phrase in accordance with dependency-based SRL task. For that purpose, we created bins of [0-4], [5-9], [10-14] and [15-19] distances. Then, we have calculate F1 scores for arguments in each bin. Due to low number of predicate-argument pairs in buckets, we could not analyze German and Turkish; and also the bin [15-19] is only used for Czech. Our results are shown in Fig. 3. We observe that either *char* or *char3* closely follows the *oracle* for all languages. The gap between the two does not increase with the distance, suggesting that the performance gap is not related to long range dependencies. In other words, both characters and the oracle handle long range dependencies equally well.

6.2 Training Data Size

We analyzed how *char3* and *oracle* models perform with respect to the training data size. For that purpose, we trained them on chunks of increasing size and evaluate on the provided test split. We used units of 2000 sentences for German and Czech; and 400 for Turkish. Results are shown in Fig. 4. Apparently as the data size increases, the performances of both models logarithmically increase - with a varying speed. To speak in statistical terms, we fit a logarithmic curve to the observed F1 scores (shown with transparent lines) and check the x coefficients, where x refers to the number of sentences. This coefficient can be considered as an approximation to the speed of growth with data size. We observe that the coefficient is higher for *char3* than *oracle* for all languages. It can be interpreted as: in the presence of more training data, *char3* may surpass the *oracle*; i.e., *char3* relies on data more than the *oracle*.

6.3 Out-of-Domain (OOD) Data

As part of the CoNLL09 shared task (Hajič et al., 2009), out of domain test sets are provided for

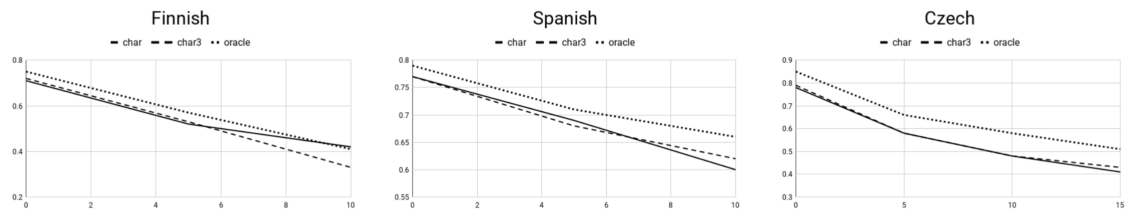


Figure 3: *X axis*: Distance between the predicate and the argument, *Y axis*: F1 scores on argument labels

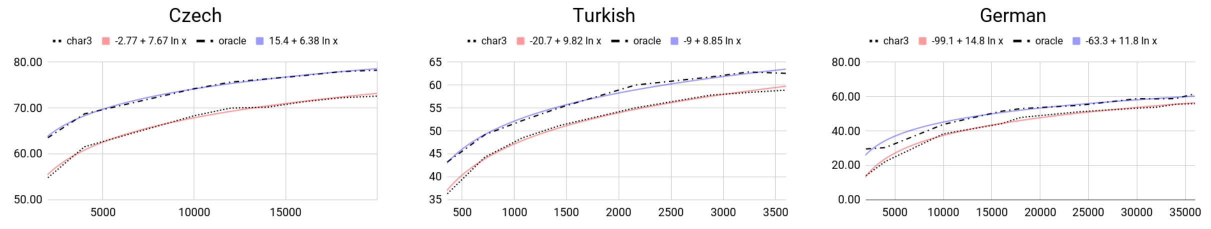


Figure 4: Performance of units w.r.t training data size. *X axis*: Number of sentences, *Y axis*: F1 score

	word	char	*IOW%*	char3	*IOW%*	oracle	*IOW%*	*IOC%*
CZE	69.97	72.98	*4.30*	**73.24**	*4.67*	72.28	*3.30*	*-1.31*
DEU	51.50	**57.05**	*10.78*	55.75	*8.24*	38.51	*-25.24*	*-45.17*
ENG	66.47	68.83	*0.70*	**70.22**	*0.23*	-	-	-

Table 5: F1 scores on out of domain data. Best scores are shown with **bold**.

three languages: Czech, German and English. We test our models trained on regular training dataset on these OOD data. The results are given in Table 5. Here, we clearly see that the best model has shifted from oracle to character based models. The dramatic drop in German oracle model is due to the high lemma OOV rate which is a consequence of keeping compounds as a single lemma. Czech oracle model performs reasonably however is unable to beat the generalization power of the *char3* model. Furthermore, the scores of the character models in Table 5 are higher than the best OOD scores reported in the shared task (Hajič et al., 2009); even though our main results on evaluation set are not (except for Czech). This shows that character-level models have increased robustness to out-of-domain data due to their ability to learn regularities among data.

6.4 Model Size

Throughout this paper, our aim was to gain insights on how models perform on different languages rather than scoring the highest F1. For this reason, we used a model that can be considered small when compared to recent neural SRL models and avoided parameter search. However,

		char3		**oracle**	
		F1	*I* (%)	F1	*I* (%)
Finnish	$\ell = 1$	67.78		71.15	
	$\ell = 2$	67.62	*-0.2*	75.71	*6.4*
Turkish	$\ell = 1$	56.60		59.38	
	$\ell = 2$	56.93	*0.5*	61.02	*2.7*
Spanish	$\ell = 1$	68.43		69.39	
	$\ell = 2$	69.30	*1.3*	71.56	*3.1*
Catalan	$\ell = 1$	71.34		73.24	
	$\ell = 2$	71.71	*0.5*	74.84	*2.2*

Table 6: Effect of layer size on model performances. *I*: Improvement over model with one layer.

we wonder how the models behave when given a larger network. To answer this question, we trained *char3* and *oracle* models with more layers for two fusional languages (Spanish, Catalan), and two agglutinative languages (Finnish, Turkish). The results given in Table 6 clearly shows that model complexity provides relatively more benefit to morphological models. This indicates that morphological signals help to extract more complex linguistic features that have semantic clues.

6.5 Predicted Morphological Tags

Although models with access to gold morphological tags achieve better F1 scores than character models, they can be less useful a in real-life scenario since they require gold tags at test time. To predict the performance of morphology-level models in such a scenario, we train the same models with the same parameters with predicted morphological features. Predicted tags

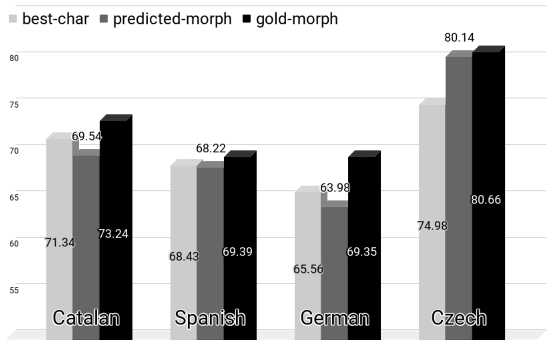


Figure 5: F1 scores for *best-char* (best of the CLMs) and model with predicted (*predicted-morph*) and gold morphological tags (*gold-morph*).

were only available for German, Spanish, Catalan and Czech. Our results given in Fig. 5, show that (except for Czech), predicted morphological tags are not as useful as characters alone.

7 Conclusion

Character-level neural models are becoming the *defacto* standard for NLP problems due to their accessibility and ability to handle unseen data. In this work, we investigated how they compare to models with access to gold morphological analysis, on a sentence-level semantic task. We evaluated their quality on *semantic role labeling* in a number of agglutinative and fusional languages. Our results lead to the following conclusions:

- For in-domain data, character-level models cannot yet match the performance of morphology-level models. However, they still provide considerable advantages over whole-word models,
- Their shortcomings depend on the morphology type. For agglutinative languages, their performance is limited on data with rich *derivational morphology* and high *contextual ambiguity* (morphological disambiguation); and for fusional languages, they struggle on tokens with high number of morphological tags,
- Similarity between character and morphology-level models is higher than the similarity within character-level (char and char-trigram) models on languages with high OOV%; and vice versa,
- Their ability to handle long-range dependencies is very similar to morphology-level models,
- They rely relatively more on training data size. Therefore, given more training data their performance will improve faster than morphology-level models,
- They perform *substantially* well on out of domain data, surpassing all morphology-level models. However, relatively less improvement is expected when model complexity is increased,
- They generally perform better than models that only have access to predicted/silver morphological tags.

8 Acknowledgements

Gözde Gül Şahin was a PhD student at Istanbul Technical University and a visiting research student at University of Edinburgh during this study. She was funded by Tübitak (The Scientific and Technological Research Council of Turkey) 2214-A scholarship during her visit to University of Edinburgh. She was granted access to CoNLL-09 Semantic Role Labeling Shared Task data by Linguistic Data Consortium (LDC). This work was supported by ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a Google Faculty award to Mark Steedman. We would like to thank Adam Lopez for fruitful discussions, guidance and support during the first author's visit.

References

Nart Bedin Atalay, Kemal Oflazer, and Bilge Say. 2003. The Annotation Process in the Turkish Treebank. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora, LINC at EACL 2003, Budapest, Hungary, April 13-14, 2003*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017. What do Neural Machine Translation Models Learn about Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 861–872.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research* 12:2461–2505.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* pages 20–30.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic Role Labeling with Neural Network Factors. In *EMNLP*. pages 960–970.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* 12(10):2451–2471.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL '09, pages 1–18.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Adam Meyers, Jan Štěpánek, Joakim Nivre, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2012a. 2009 CoNLL Shared Task Part 1 LDC2012T04. Web Download.

Jan Hajič, Maria A. Martí, Lluis Marquez, Joakim Nivre, Jan Štěpánek, Sebastian Padó, and Pavel Straňák. 2012b. 2009 CoNLL Shared Task Part 1 LDC2012T03. Web Download.

Katri Haverinen, Jenna Kanerva, Samuel Kohonen, Anna Missila, Stina Ojala, Timo Viljanen, Veronika Laippala, and Filip Ginter. 2015. The Finnish Proposition Bank. *Language Resources and Evaluation* 49(4):907–926.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-Aware Neural Language Models. In *AAAI*. pages 2741–2749.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *TACL* 5:365–378.

Wang Ling, Tiago Luis, Luis Marujo, Ramon F Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*. pages 1520–1530.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A Simple and Accurate Syntax-Agnostic Neural Model for Dependency-based Semantic Role Labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1507–1516.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In *Treebanks*, Springer, pages 261–277.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Gözde Gül Şahin and Eşref Adalı. 2017. Annotation of semantic roles for the Turkish Proposition Bank. *Language Resources and Evaluation* pages 1–34.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Umut Sulubacak and Gülşen Eryiğit. 2018. Implementing Universal Dependency, Morphology and Multiword Expression Annotation Standards for Turkish Language Processing. *Turkish Journal of Electrical Engineering Computer Sciences* pages 1–23.

Umut Sulubacak, Tuğba Pamay, and Gülşen Eryiğit. 2016. IMST: A Revisited Turkish Dependency Treebank. In *Proceedings of the 1st International Conference on Turkic Computational Linguistics (TurCLing) at CICLing, Konya, Turkey, 2016*.

Clara Vania and Adam Lopez. 2017. From Characters to Words to in Between: Do We Capture Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 2016–2027.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese Semantic Role Labeling with Bidirectional Recurrent Neural Networks. In *EMNLP*. pages 1626–1631.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 1127–1137.

Gözde Gül Şahin. 2016a. Framing of Verbs for Turkish PropBank. In *In Proceedings of 1st International Conference on Turkic Computational Linguistics, TurCLing*.

Gözde Gül Şahin. 2016b. Verb Sense Annotation for Turkish PropBank via Crowdsourcing. In *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Revised Selected Papers, Part I*. pages 496–506.

AMR Parsing as Graph Prediction with Latent Alignment

Chunchuan Lyu[1] **Ivan Titov**[1,2]
[1]ILCC, School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam

Abstract

Abstract meaning representations (AMRs) are broad-coverage sentence-level semantic representations. AMRs represent sentences as rooted labeled directed acyclic graphs. AMR parsing is challenging partly due to the lack of annotated alignments between nodes in the graphs and words in the corresponding sentences. We introduce a neural parser which treats alignments as latent variables within a joint probabilistic model of concepts, relations and alignments. As exact inference requires marginalizing over alignments and is infeasible, we use the variational auto-encoding framework and a continuous relaxation of the discrete alignments. We show that joint modeling is preferable to using a pipeline of align and parse. The parser achieves the best reported results on the standard benchmark (74.4% on LDC2016E25).

1 Introduction

Abstract meaning representations (AMRs) (Banarescu et al., 2013) are broad-coverage sentence-level semantic representations. AMR encodes, among others, information about semantic relations, named entities, co-reference, negation and modality. The semantic representations can be regarded as rooted labeled directed acyclic graphs (see Figure 1). As AMR abstracts away from details of surface realization, it is potentially beneficial in many semantic related NLP tasks, including text summarization (Liu et al., 2015; Dohare and Karnick, 2017), machine translation (Jones et al., 2012) and question answering (Mitra and Baral, 2016).

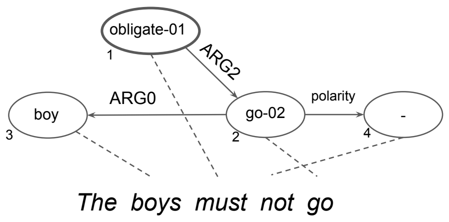


Figure 1: An example of AMR, the dashed lines denote latent alignments, *obligate-01* is the root. Numbers indicate depth-first traversal order.

AMR parsing has recently received a lot of attention (e.g., (Flanigan et al., 2014; Artzi et al., 2015; Konstas et al., 2017)). One distinctive aspect of AMR annotation is the lack of explicit alignments between nodes in the graph (*concepts*) and words in the sentences. Though this arguably simplified the annotation process (Banarescu et al., 2013), it is not straightforward to produce an effective parser without relying on an alignment. Most AMR parsers (Damonte et al., 2017; Flanigan et al., 2016; Werling et al., 2015; Wang and Xue, 2017; Foland and Martin, 2017) use a pipeline where the aligner training stage precedes training a parser. The aligners are not directly informed by the AMR parsing objective and may produce alignments suboptimal for this task.

In this work, we demonstrate that the alignments can be treated as latent variables in a joint probabilistic model and induced in such a way as to be beneficial for AMR parsing. Intuitively, in our probabilistic model, every node in a graph is assumed to be aligned to a word in a sentence: each concept is predicted based on the corresponding RNN state. Similarly, graph edges (i.e. relations) are predicted based on representations of concepts and aligned words (see Figure 2). As alignments are latent, exact inference requires marginalizing over latent alignments, which is infeasible. Instead we use variational inference, specifically the variational autoencoding framework of Kingma and Welling (2014). Using discrete latent variables in deep learning has proven to be challenging (Mnih and Gregor, 2014; Bornschein and Bengio, 2015). We use a continuous relaxation of the alignment problem, relying on the recently introduced Gumbel-Sinkhorn construction (Mena et al., 2018). This yields a computationally-efficient approximate method for estimating our joint probabilistic model of concepts, relations and alignments.

We assume injective alignments from concepts to words: every node in the graph is aligned to a single word in the sentence and every word is aligned to at most one node in the graph. This is necessary for two reasons. First, it lets us treat concept identification as sequence tagging at test time. For every word we would simply predict the corresponding concept or predict *NULL* to signify that no concept should be generated at this position. Secondly, Gumbel-Sinkhorn can only work under this assumption. This constraint, though often appropriate, is problematic for certain AMR constructions (e.g., named entities). In order to deal with these cases, we re-categorized AMR concepts. Similar recategorization strategies have been used in previous work (Foland and Martin, 2017; Peng et al., 2017).

The resulting parser achieves 74.4% Smatch score on the standard test set when using LDC2016E25 training set,[1] an improvement of 3.4% over the previous best result (van Noord and Bos, 2017). We also demonstrate that inducing alignments within the joint model is indeed beneficial. When, instead of inducing alignments, we follow the standard approach and produce them on preprocessing, the performance drops by 0.9% Smatch. Our main contributions can be summarized as follows:

- we introduce a joint probabilistic model for alignment, concept and relation identification;
- we demonstrate that a continuous relaxation can be used to effectively estimate the model;
- the model achieves the best reported results.[2]

[1]The standard deviation across multiple training runs was 0.16%.

[2]The code can be accessed from `https://github.com/ChunchuanLv/AMR_AS_GRAPH_PREDICTION`

2 Probabilistic Model

In this section we describe our probabilistic model and the estimation technique. In section 3, we describe preprocessing and post-processing (including concept re-categorization, sense disambiguation, wikification and root selection).

2.1 Notation and setting

We will use the following notation throughout the paper. We refer to words in the sentences as $\mathbf{w} = (w_1, \dots, w_n)$, where n is sentence length, $w_k \in \mathcal{V}$ for $k \in \{1 \dots, n\}$. The concepts (i.e. labeled nodes) are $\mathbf{c} = (c_1, \dots, c_m)$, where m is the number of concepts and $c_i \in \mathcal{C}$ for $i \in \{1 \dots, m\}$. For example, in Figure 1, $\mathbf{c} = (\textit{obligate}, \textit{go}, \textit{boy}, \text{-})$.[3] Note that senses are predicted at post-processing, as discussed in Section 3.2 (i.e. *go* is labeled as *go-02*).

A relation between 'predicate concept' i and 'argument concept' j is denoted by $r_{ij} \in \mathcal{R}$; it is set to *NULL* if j is not an argument of i. In our example, $r_{2,3} = \textit{ARG0}$ and $r_{1,3} = \textit{NULL}$. We will use R to denote all relations in the graph.

To represent alignments, we will use $\mathbf{a} = \{a_1, \dots, a_m\}$, where $a_i \in \{1, \dots, n\}$ returns the index of a word aligned to concept i. In our example, $a_1 = 3$.

All three model components rely on bidirectional LSTM encoders (Schuster and Paliwal, 1997). We denote states of BiLSTM (i.e. concatenation of forward and backward LSTM states) as $\mathbf{h}_k \in \mathbb{R}^d$ $(k \in \{1, \dots, n\})$. The sentence encoder takes pre-trained fixed word embeddings, randomly initialized lemma embeddings, part-of-speech and named-entity tag embeddings.

2.2 Method overview

We believe that using discrete alignments, rather than attention-based models (Bahdanau et al., 2015) is crucial for AMR parsing. AMR banks are a lot smaller than parallel corpora used in machine translation (MT) and hence it is important to inject a useful inductive bias. We constrain our alignments from concepts to words to be injective. First, it encodes the observation that concepts are mostly triggered by single words (especially, after re-categorization, Section 3.1). Second, it implies

[3]The probabilistic model is invariant to the ordering of concepts, though the order affects the inference algorithm (see Section 2.5). We use depth-first traversal of the graph to generate the ordering.

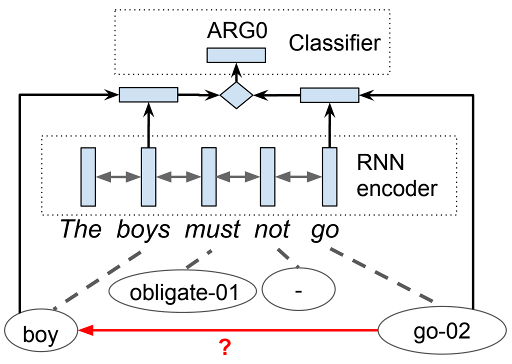


Figure 2: Relation identification: predicting a relation between *boy* and *go-02* relying on the two concepts and corresponding RNN states.

that each word corresponds to at most one concept (if any). This encourages competition: alignments are mutually-repulsive. In our example, *obligate* is not lexically similar to the word *must* and may be hard to align. However, given that other concepts are easy to predict, alignment candidates other than *must* and *the* will be immediately ruled out. We believe that these are the key reasons for why attention-based neural models do not achieve competitive results on AMR (Konstas et al., 2017) and why state-of-the-art models rely on aligners. Our goal is to combine best of two worlds: to use alignments (as in state-of-the-art AMR methods) and to induce them while optimizing for the end goal (similarly to the attention component of encoder-decoder models).

Our model consists of three parts: (1) the concept identification model $P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w})$; (2) the relation identification model $P_\phi(R|\mathbf{a},\mathbf{w},\mathbf{c})$ and (3) the alignment model $Q_\psi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$.[4] Formally, (1) and (2) together with the uniform prior over alignments $P(\mathbf{a})$ form the generative model of AMR graphs. In contrast, the alignment model $Q_\psi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$, as will be explained below, is approximating the intractable posterior $P_{\theta,\phi}(\mathbf{a}|\mathbf{c},R,\mathbf{w})$ within that probabilistic model.

In other words, we assume the following model for generating the AMR graph:

$$P_{\theta,\phi}(\mathbf{c},R|\mathbf{w}) = \sum_{\mathbf{a}} P(\mathbf{a}) P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w}) P_\phi(R|\mathbf{a},\mathbf{w},\mathbf{c})$$

$$= \sum_{\mathbf{a}} P(\mathbf{a}) \prod_{i=1}^{m} P(c_i|\mathbf{h}_{a_i}) \prod_{i,j=1}^{m} P(r_{ij}|\mathbf{h}_{a_i},\mathbf{c}_i,\mathbf{h}_{a_j},\mathbf{c}_j)$$

AMR concepts are assumed to be generated conditional independently relying on the BiLSTM states and surface forms of the aligned words. Similarly, relations are predicted based only on AMR concept embeddings and LSTM states corresponding to words aligned to the involved concepts. Their combined representations are fed into a bi-affine classifier (Dozat and Manning, 2017) (see Figure 2).

The expression involves intractable marginalization over all valid alignments. As standard in variational autoencoders, VAEs (Kingma and Welling, 2014), we lower-bound the log-likelihood as

$$\begin{aligned} &\log P_{\theta,\phi}(\mathbf{c},R|\mathbf{w}) \\ &\geq E_Q[\log P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w}) P_\phi(R|\mathbf{a},\mathbf{w},\mathbf{c})] \\ &- D_{KL}(Q_\psi(\mathbf{a}|\mathbf{c},R,\mathbf{w}) || P(\mathbf{a})), \end{aligned} \quad (1)$$

where $Q_\psi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$ is the variational posterior (aka the inference network), $E_Q[\ldots]$ refers to the expectation under $Q_\psi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$ and D_{KL} is the Kullback-Liebler divergence. In VAEs, the lower bound is maximized both with respect to model parameters (θ and ϕ in our case) and the parameters of the inference network (ψ). Unfortunately, gradient-based optimization with discrete latent variables is challenging. We use a continuous relaxation of our optimization problem, where real-valued vectors $\hat{\mathbf{a}}_i \in \mathbb{R}^n$ (for every concept i) approximate discrete alignment variables a_i. This relaxation results in low-variance estimates of the gradient using the parameterization trick (Kingma and Welling, 2014), and ensures fast and stable training. We will describe the model components and the relaxed inference procedure in detail in sections 2.6 and 2.7.

Though the estimation procedure requires the use of the relaxation, the learned parser is straightforward to use. Given our assumptions about the alignments, we can independently choose for each word w_k ($k = 1, \ldots, m$) the most probably concept according to $P_\theta(c|\mathbf{h}_k)$. If the highest scoring option is *NULL*, no concept is introduced. The relations could then be predicted relying on $P_\phi(R|\mathbf{a},\mathbf{w},\mathbf{c})$. This would have led to generating inconsistent AMR graphs, so instead we search for the highest scoring valid graph (see Section 3.2). Note that the alignment model Q_ψ is not used at test time and only necessary to train accurate concept and relation identification models.

[4] θ, ϕ and ψ denote all parameters of the models.

2.3 Concept identification model

The concept identification model chooses a concept c (i.e. a labeled node) conditioned on the aligned word k or decides that no concept should be introduced (i.e. returns *NULL*). Though it can be modeled with a softmax classifier, it would not be effective in handling rare or unseen words. First, we split the decision into estimating the probability of concept category $\tau(c) \in \mathcal{T}$ (e.g. 'number', 'frame') and estimating the probability of the specific concept within the chosen category. Second, based on a lemmatizer and training data[5] we prepare one candidate concept e_k for each word k in vocabulary (e.g., it would propose *want* if the word is *wants*). Similar to Luong et al. (2015), our model can then either copy the candidate e_k or rely on the softmax over potential concepts of category τ. Formally, the concept prediction model is defined as

$$P_\theta(c|\mathbf{h}_k, w_k) = P(\tau(c)|\mathbf{h}_k, w_k) \times \frac{[[e_k = c]] \times \exp(\mathbf{v}_{copy}^T \mathbf{h_k}) + \exp(\mathbf{v}_c^T \mathbf{h_k})}{Z(\mathbf{h_k}, \theta)},$$

where the first multiplicative term is a softmax classifier over categories (including *NULL*); $\mathbf{v}_{copy}, \mathbf{v}_c \in \mathbb{R}^d$ (for $c \in \mathcal{C}$) are model parameters; $[[\ldots]]$ denotes the indicator function and equals 1 if its argument is true and 0, otherwise; $Z(\mathbf{h}, \theta)$ is the partition function ensuring that the scores sum to 1.

2.4 Relation identification model

We use the following arc-factored relation identification model:

$$P_\phi(R|\mathbf{a}, \mathbf{w}, \mathbf{c}) = \prod_{i,j=1}^{m} P(r_{ij}|\mathbf{h}_{a_i}, \mathbf{c}_i, \mathbf{h}_{a_j}, \mathbf{c}_j) \quad (2)$$

Each term is modeled in exactly the same way:

1. for both endpoints, embedding of the concept c is concatenated with the RNN state $\mathbf{h}$;

2. they are linearly projected to a lower dimension separately through $M_h(\mathbf{h}_{a_i} \circ c_i) \in \mathbb{R}^{d_f}$ and $M_d(\mathbf{h}_{a_j} \circ c_j) \in \mathbb{R}^{d_f}$, where $\circ$ denotes concatenation;

3. a log-linear model with bilinear scores $M_h(\mathbf{h}_{a_i} \circ c_i)^T C_r M_d(\mathbf{h}_{a_j} \circ c_j)$, $C_r \in \mathbb{R}^{d_f \times d_f}$ is used to compute the probabilities.

[5]See supplementary materials.

In the above discussion, we assumed that BiLSTM encodes a sentence once and the BiLSTM states are then used to predict concepts and relations. In semantic role labeling, the task closely related to the relation identification stage of AMR parsing, a slight modification of this approach was shown more effective (Zhou and Xu, 2015; Marcheggiani et al., 2017). In that previous work, the sentence was encoded by a BiLSTM once per each predicate (i.e. verb) and the encoding was in turn used to identify arguments of that predicate. The only difference across the re-encoding passes was a binary flag used as input to the BiLSTM encoder at each word position. The flag was set to 1 for the word corresponding to the predicate and to 0 for all other words. In that way, BiLSTM was encoding the sentence specifically for predicting arguments of a given predicate. Inspired by this approach, when predicting label r_{ij} for $j \in \{1, \ldots m\}$, we input binary flags $\mathbf{p}_1, \ldots \mathbf{p}_n$ to the BiLSTM encoder which are set to 1 for the word indexed by a_i ($\mathbf{p}_{a_i} = 1$) and to 0 for other words ($\mathbf{p}_j = 0$, for $j \neq a_i$). This also means that BiLSTM encoders for predicting relations and concepts end up being distinct. We use this multi-pass approach in our experiments.[6]

2.5 Alignment model

Recall that the alignment model is only used at training, and hence it can rely both on input (states $\mathbf{h}_1, \ldots, \mathbf{h}_n$) and on the list of concepts $c_1, \ldots, c_m$.

Formally, we add $(m-n)$ *NULL* concepts to the list.[7] Aligning a word to any *NULL*, would correspond to saying that the word is not aligned to any 'real' concept. Note that each one-to-one alignment (i.e. permutation) between n such concepts and n words implies a valid injective alignment of n words to m 'real' concepts. This reduction to permutations will come handy when we turn to the Gumbel-Sinkhorn relaxation in the next section. Given this reduction, from now on, we will assume that $m = n$.

As with sentences, we use a BiLSTM model to encode concepts $\mathbf{c}$, where $\mathbf{g}_i \in \mathcal{R}^{d_g}$, $i \in \{1, \ldots, n\}$. We use a globally-normalized align-

[6]Using the vanilla one-pass model from equation (2) results in 1.4% drop in Smatch score.

[7]After re-categorization (Section 3.1), $m \geq n$ holds for most cases. For exceptions, we append *NULL* to the sentence.

ment model:

$$Q_\psi(\mathbf{a}|\mathbf{c}, R, \mathbf{w}) = \frac{\exp(\sum_{i=1}^{n} \varphi(\mathbf{g}_i, \mathbf{h}_{a_i}))}{Z_\psi(\mathbf{c}, \mathbf{w})},$$

where $Z_\psi(\mathbf{c}, \mathbf{w})$ is the intractable partition function and the terms $\varphi(\mathbf{g}_i, \mathbf{h}_{a_i})$ score each alignment link according to a bilinear form

$$\varphi(\mathbf{g}_i, \mathbf{h}_{a_i}) = \mathbf{g}_i^T B \mathbf{h}_{a_i}, \qquad (3)$$

where $B \in \mathbb{R}^{d_g \times d}$ is a parameter matrix.

2.6 Estimating model with Gumbel-Sinkhorn

Recall that our learning objective (1) involves expectation under the alignment model. The partition function of the alignment model $Z_\psi(\mathbf{c}, \mathbf{w})$ is intractable, and it is tricky even to draw samples from the distribution. Luckily, the recently proposed relaxation (Mena et al., 2018) lets us circumvent this issue. First, note that exact samples from a categorical distribution can be obtained using the perturb-and-max technique (Papandreou and Yuille, 2011). For our alignment model, it would correspond to adding independent noise to the score for every possible alignment and choosing the highest scoring one:

$$\mathbf{a}^\star = \operatorname*{argmax}_{\mathbf{a} \in \mathcal{P}} \sum_{i=1}^{n} \varphi(\mathbf{g}_i, \mathbf{h}_{a_i}) + \epsilon_{\mathbf{a}}, \qquad (4)$$

where $\mathcal{P}$ is the set of all permutations of n elements, $\epsilon_{\mathbf{a}}$ is a noise drawn independently for each $\mathbf{a}$ from the fixed Gumbel distribution ($\mathcal{G}(0,1)$). Unfortunately, this is also intractable, as there are $n!$ permutations. Instead, in perturb-and-max an approximate schema is used where noise is assumed factorizable. In other words, first noisy scores are computed as $\hat{\varphi}(\mathbf{g}_i, \mathbf{h}_{a_i}) = \varphi(\mathbf{g}_i, \mathbf{h}_{a_i}) + \epsilon_{i,a_i}$, where $\epsilon_{i,a_i} \sim \mathcal{G}(0,1)$ and an approximate sample is obtained by $\mathbf{a}^\star = \operatorname{argmax}_{\mathbf{a}} \sum_{i=1}^{n} \hat{\varphi}(\mathbf{g}_i, \mathbf{h}_{a_i})$,

Such sampling procedure is still intractable in our case and also non-differentiable. The main contribution of Mena et al. (2018) is approximating this argmax with a simple differentiable computation $\hat{\mathbf{a}} = S_t(\Phi, \Sigma)$ which yields an approximate (i.e. relaxed) permutation. We use Φ and Σ to denote the $n \times n$ matrices of alignment scores $\varphi(\mathbf{g}_i, \mathbf{h}_k)$ and noise variables ϵ_{ik}, respectively. Instead of returning index a_i for every concept i, it would return a (peaky) distribution over words $\hat{\mathbf{a}}_i$. The peakiness is controlled by the temperature parameter t of Gumbel-Sinkhorn which balances smoothness ('differentiability') vs. bias of the estimator. For further details and the derivation, we refer the reader to the original paper (Mena et al., 2018).

Note that Φ is a function of the alignment model Q_ψ, so we will write Φ_ψ in what follows. The variational bound (1) can now be approximated as

$$\begin{aligned} E_{\Sigma \sim \mathcal{G}(0,1)} & [\log P_\theta(c|S_t(\Phi_\psi, \Sigma), \mathbf{w}) \\ & + \log P_\phi(R|S_t(\Phi_\psi, \Sigma), \mathbf{w}, \mathbf{c})] \\ & - D_{KL}(\frac{\Phi_\psi + \Sigma}{t} || \frac{\Sigma}{t_0}) \end{aligned} \qquad (5)$$

Following Mena et al. (2018), the original KL term from equation (1) is approximated by the KL term between two $n \times n$ matrices of i.i.d. Gumbel distributions with different temperature and mean. The parameter t_0 is the 'prior temperature'.

Using the Gumbel-Sinkhorn construction unfortunately does not guarantee that $\sum_i \hat{\mathbf{a}}_{ij} = 1$. To encourage this equality to hold, and equivalently to discourage overlapping alignments, we add another regularizer to the objective (5):

$$\Omega(\hat{\mathbf{a}}, \lambda) = \lambda \sum_j \max(\sum_i (\hat{\mathbf{a}}_{ij}) - 1, 0). \qquad (6)$$

Our final objective is fully differentiable with respect to all parameters (i.e. θ, ϕ and ψ) and has low variance as sampling is performed from the fixed non-parameterized distribution, as in standard VAEs.

2.7 Relaxing concept and relation identification

One remaining question is how to use the soft input $\hat{\mathbf{a}} = S_t(\Phi_\psi, \Sigma)$ in the concept and relation identification models in equation (5). In other words, we need to define how we compute $P_\theta(c|S_t(\Phi_\psi, \Sigma), \mathbf{w})$ and $P_\phi(R|S_t(\Phi_\psi, \Sigma), \mathbf{w}, \mathbf{c})$.

The standard technique would be to pass to the models expectations under the relaxed variables $\sum_{k=1}^{n} \hat{\mathbf{a}}_{ik} \mathbf{h}_k$, instead of the vectors $\mathbf{h}_{a_i}$ (Maddison et al., 2017; Jang et al., 2017). This is what we do for the relation identification model. We use this approach also to relax the one-hot encoding of the predicate position ($\mathbf{p}$, see Section 2.4).

However, the concept prediction model $\log P_\theta(c|S_t(\Phi_\psi, \Sigma), \mathbf{w})$ relies on the pointing mechanism, i.e. directly exploits the words $\mathbf{w}$ rather than relies only on biLSTM states $\mathbf{h}_k$. So

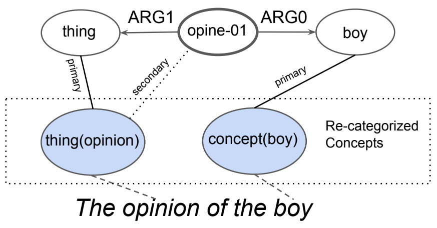


Figure 3: An example of re-categorized AMR. AMR graph at the top, re-categorized concepts in the middle, and the sentence is at the bottom.

instead we treat $\hat{\mathbf{a}}_i$ as a prior in a hierarchical model:

$$\log P_\theta(c_i|\hat{\mathbf{a}}_\mathbf{i}, \mathbf{w}) \approx \log \sum_{k=1}^{n} \hat{\mathbf{a}}_{ik} P_\theta(c_i|a_i = k, \mathbf{w}) \quad (7)$$

As we will show in our experiments, a softer version of the loss is even more effective:

$$\log P_\theta(c_i|\hat{\mathbf{a}}_\mathbf{i}, \mathbf{w}) \approx \log \sum_{k=1}^{n} (\hat{\mathbf{a}}_{ik} P_\theta(c_i|a_i = k, \mathbf{w}))^\alpha, \quad (8)$$

where we set the parameter $\alpha = 0.5$. We believe that using this loss encourages the model to more actively explore the alignment space. Geometrically, the loss surface shaped as a ball in the 0.5-norm space would push the model away from the corners, thus encouraging exploration.

3 Pre- and post-pocessing

3.1 Re-Categorization

AMR parsers often rely on a pre-processing stage, where specific subgraphs of AMR are grouped together and assigned to a single node with a new compound category (e.g., Werling et al. (2015); Foland and Martin (2017); Peng et al. (2017)); this transformation is reversed at the post-processing stage. Our approach is very similar to the Factored Concept Label system of Wang and Xue (2017), with one important difference that we unpack our concepts before the relation identification stage, so the relations are predicted between original concepts (all nodes in each group share the same alignment distributions to the RNN states). Intuitively, the goal is to ensure that concepts rarely lexically triggered (e.g., *thing* in Figure 3) get grouped together with lexically triggered nodes. Such 'primary' concepts get encoded in the category of the concept (the set of categories is τ, see also section 2.3). In Figure 3, the re-categorized concept *thing(opinion)* is produced from *thing* and *opine-01*. We use *concept* as the dummy category type. There are 8 templates in our system which extract re-categorizations for fixed phrases (e.g. *thing(opinion)*), and a deterministic system for grouping lexically flexible, but structurally stable sub-graphs (e.g., named entities, *have-rel-role-91* and *have-org-role-91* concepts).

Details of the re-categorization procedure and other pre-processing are provided in appendix.

3.2 Post-processing

For post-processing, we handle sense-disambiguation, wikification and ensure legitimacy of the produced AMR graph. For sense disambiguation we pick the most frequent sense for that particular concept ('-01', if unseen). For wikification we again look-up in the training set and default to "-". There is certainly room for improvement in both stages. Our probability model predicts edges conditional independently and thus cannot guarantee the connectivity of AMR graph, also there are additional constraints which are useful to impose. We enforce three constraints: (1) specific concepts can have only one neighbor (e.g., 'number' and 'string'; see appendix for details); (2) each predicate concept can have at most one argument for each relation $r \in \mathcal{R}$; (3) the graph should be connected. Constraint (1) is addressed by keeping only the highest scoring neighbor. In order to satisfy the last two constraints we use a simple greedy procedure. First, for each edge, we pick-up the highest scoring relation and edge (possibly *NULL*). If the constraint (2) is violated, we simply keep the highest scoring edge among the duplicates and drop the rest. If the graph is not connected (i.e. constraint (3) is violated), we greedily choose edges linking the connected components until the graph gets connected (MSCG in Flanigan et al. (2014)).

Finally, we need to select a root node. Similarly to relation identification, for each candidate concept c_i, we concatenate its embedding with the corresponding LSTM state ($\mathbf{h}_{a_i}$) and use these scores in a softmax classifier over all the concepts.

Model	Data	Smatch
JAMR (Flanigan et al., 2016)	R1	67.0
AMREager (Damonte et al., 2017)	R1	64.0
CAMR (Wang et al., 2016)	R1	66.5
SEQ2SEQ + 20M (Konstas et al., 2017)	R1	62.1
Mul-BiLSTM (Foland and Martin, 2017)	R1	70.7
Ours	R1	**73.7**
Neural-Pointer (Buys and Blunsom, 2017)	R2	61.9
ChSeq (van Noord and Bos, 2017)	R2	64.0
ChSeq + 100K (van Noord and Bos, 2017)	R2	71.0
Ours	R2	**74.4** ± 0.16

Table 1: Smatch scores on the test set. R2 is LDC2016E25 dataset, and R1 is LDC2015E86 dataset. Statistics on R2 are over 8 runs.

Models	A' 17	C' 16	J' 16	Ch' 17	Ours
Dataset	R1	R1	R1	R2	R2
Smatch	64	63	67	71	**74.4**±0.16
Unlabeled	69	69	69	74	**77.1**±0.10
No WSD	65	64	68	72	**75.5**±0.12
Reentrancy	41	41	42	**52**	**52.3**±0.43
Concepts	83	80	83	82	**85.9**±0.11
NER	83	75	79	79	**86.0**±0.46
Wiki	64	0	75	65	**75.7**±0.30
Negations	48	18	45	**62**	58.4±1.32
SRL	56	60	60	66	**69.8**±0.24

Table 2: F1 scores on individual phenomena. A'17 is AMREager, C'16 is CAMR, J'16 is JAMR, Ch'17 is ChSeq+100K. Ours are marked with standard deviation.

Metric	Pre-Align	R1	Pre-Align	R2 mean
Smatch	72.8	73.7	73.5	**74.4**
Unlabeled	75.3	76.3	76.1	**77.1**
No WSD	73.8	74.7	74.6	**75.5**
Reentrancy	50.2	50.6	**52.6**	52.3
Concepts	85.4	85.5	85.5	**85.9**
NER	85.3	84.8	85.3	**86.0**
Wiki	66.8	75.6	67.8	**75.7**
Negations	56.0	57.2	56.6	**58.4**
SRL	68.8	68.9	**70.2**	69.8

Table 3: F1 scores of on subtasks. Scores on ablations are averaged over 2 runs. The left side results are from LDC2015E86 and right results are from LDC2016E25.

4 Experiments and Discussion

4.1 Data and setting

We primarily focus on the most recent LDC2016E25 (R2) dataset, which consists of 36521, 1368 and 1371 sentences in training, development and testing sets, respectively. The earlier LDC2015E86 (R1) dataset has been used by much of the previous work. It contains 16833 training sentences, and same sentences for development and testing as R2.[8]

We used the development set to perform model selection and hyperparameter tuning. The hyperparameters, as well as information about embeddings and pre-processing, are presented in the supplementary materials.

We used Adam (Kingma and Ba, 2014) to optimize the loss (5) and to train the root classifier. Our best model is trained fully jointly, and we do early stopping on the development set scores. Training takes approximately 6 hours on a single GeForce GTX 1080 Ti with Intel Xeon CPU E5-2620 v4.

4.2 Experiments and discussion

We start by comparing our parser to previous work (see Table 1). Our model substantially outperforms all the previous models on both datasets. Specifically, it achieves 74.4% Smatch score on LDC2016E25 (R2), which is an improvement of 3.4% over character seq2seq model relying on silver data (van Noord and Bos, 2017). For LDC2015E86 (R1), we obtain 73.7% Smatch score, which is an improvement of 3.0% over the previous best model, multi-BiLSTM parser of Foland and Martin (2017).

In order to disentangle individual phenomena, we use the AMR-evaluation tools (Damonte et al., 2017) and compare to systems which reported these scores (Table 2). We obtain the highest scores on most subtasks. The exception is negation detection. However, this is not too surprising as many negations are encoded with morphology, and character models, unlike our word-level model, are able to capture predictive morphological features (e.g., detect prefixes such as "un-" or "im-").

Now, we turn to ablation tests (see Table 3). First, we would like to see if our latent alignment framework is beneficial. In order to test this, we create a baseline version of our system ('pre-align') which relies on the JAMR aligner (Flani-

[8]Annotation in R2 has also been slightly revised.

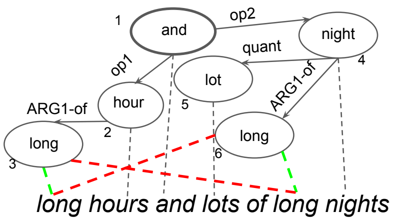


Figure 4: When modeling concepts alone, the posterior probability of the correct (green) and wrong (red) alignment links will be the same.

Ablation	Concepts	SRL	Smatch
2 stages	85.6	68.9	73.6
2 stages, tune align	85.6	69.2	73.9
Full model	**85.9**	**69.8**	**74.4**

Table 4: Ablation studies: effect of joint modeling (all on R2). Scores on ablations are averaged over 2 runs. The first two models load the same concept and alignment model before the second stage.

gan et al., 2014), rather than induces alignments as latent variables. Recall that in our model we used training data and a lemmatizer to produce candidates for the concept prediction model (see Section 2.3, the copy function). In order to have a fair comparison, if a concept is not aligned after JAMR, we try to use our copy function to align it. If an alignment is not found, we make the alignment uniform across the unaligned words. In preliminary experiments, we considered alternatives versions (e.g., dropping concepts unaligned by JAMR or dropping concepts unaligned after both JAMR and the matching heuristic), but the chosen strategy was the most effective. These scores of pre-align are superior to the results from Foland and Martin (2017) which also relies on JAMR alignments and uses BiLSTM encoders. There are many potential reasons for this difference in performance. For example, their relation identification model is different (e.g., single pass, no bi-affine modeling), they used much smaller networks than us, they use plain JAMR rather than a combination of JAMR and our copy function, they use a different recategorization system. These results confirm that we started with a strong basic model, and that our variational alignment framework provided further gains in performance.

Now we would like to confirm that joint training of alignments with both concepts and relations is beneficial. In other words, we would like to see if alignments need to be induced in such a way as to benefit the relation identification task. For this ablation we break the full joint training into two stages. We start by jointly training the alignment model and the concept identification model. When these are trained, we optimizing the relation model but keep the concept identification model and alignment models fixed ('2 stages' in see Table 4). When compared to our joint model ('full model'), we observe a substantial drop in Smatch score (-0.8%). In another version ('2 stages, tune align') we also use two stages but we fine-tune the alignment model on the second stage. This approach appears slightly more accurate but still -0.5% below the full model. In both cases, the drop is more substantial for relations ('SRL'). In order to see why relations are potentially useful in learning alignments, consider Figure 4. The example contains duplicate concepts *long*. The concept prediction model factorizes over concepts and does not care which way these duplicates are aligned: correctly (green edges) or not (red edges). Formally, the true posterior under the concept-only model in '2 stages' assigns exactly the same probability to both configurations, and the alignment model Q_ψ will be forced to mimic it (even though it relies on an LSTM model of the graph). The spurious ambiguity will have a detrimental effect on the relation identification stage.

Ablation	Concepts	SRL	Smatch
No Sinkhorn	85.7	69.3	73.8
No Sinkhorn reg	85.6	69.5	74.2
No soft loss	85.2	69.1	73.7
Full model	**85.9**	**69.8**	**74.4**

Table 5: Ablation studies: alignment modeling and relaxation (all on R2). Scores on ablations are averaged over 2 runs.

It is interesting to see the contribution of other modeling decisions we made when modeling and relaxing alignments. First, instead of using Gumbel-Sinkhorn, which encourages mutually-repulsive alignments, we now use a factorized alignment model. Note that this model ('No Sinkhorn' in Table 5) still relies on (relaxed) discrete alignments (using Gumbel softmax) but does not constrain the alignments to be injective. A substantial drop in performance indicates that the prior knowledge about the nature of alignments appears beneficial. Second, we remove the additional regularizer for Gumbel-Sinkhorn approximation (equation (6)). The performance drop in

Smatch score ('No Sinkhorn reg') is only moderate. Finally, we show that using the simple hierarchical relaxation (equation (7)) rather than our softer version of the loss (equation (8)) results in a substantial drop in performance ('No soft loss', -0.7% Smatch). We hypothesize that the softer relaxation favors exploration of alignments and helps to discover better configurations.

5 Additional Related Work

Alignment performance has been previously identified as a potential bottleneck affecting AMR parsing (Damonte et al., 2017; Foland and Martin, 2017). Some recent work has focused on building aligners specifically for training their parsers (Werling et al., 2015; Wang and Xue, 2017). However, those aligners are trained independently of concept and relation identification and only used at pre-processing.

Treating alignment as discrete variables has been successful in some sequence transduction tasks with neural models (Yu et al., 2017, 2016). Our work is similar in that we also train discrete alignments jointly but the tasks, the inference framework and the decoders are very different.

The discrete alignment modeling framework has been developed in the context of traditional (i.e. non-neural) statistical machine translation (Brown et al., 1993). Such translation models have also been successfully applied to semantic parsing tasks (e.g., (Andreas et al., 2013)), where they rivaled specialized semantic parsers from that period. However, they are considerably less accurate than current state-of-the-art parsers applied to the same datasets (e.g., (Dong and Lapata, 2016)).

For AMR parsing, another way to avoid using pre-trained aligners is to use seq2seq models (Konstas et al., 2017; van Noord and Bos, 2017). In particular, van Noord and Bos (2017) used character level seq2seq model and achieved the previous state-of-the-art result. However, their model is very data demanding as they needed to train it on additional 100K sentences parsed by other parsers. This may be due to two reasons. First, seq2seq models are often not as strong on smaller datasets. Second, recurrent decoders may struggle with predicting the linearized AMRs, as many statistical dependencies are highly non-local.

6 Conclusions

We introduced a neural AMR parser trained by jointly modeling alignments, concepts and relations. We make such joint modeling computationally feasible by using the variational autoencoding framework and continuous relaxations. The parser achieves state-of-the-art results and ablation tests show that joint modeling is indeed beneficial.

We believe that the proposed approach may be extended to other parsing tasks where alignments are latent (e.g., parsing to logical form (Liang, 2016)). Another promising direction is integrating character seq2seq to substitute the copy function. This should also improve the handling of negation and rare words. Though our parsing model does not use any linearization of the graph, we relied on LSTMs and somewhat arbitrary linearization (depth-first traversal) to encode the AMR graph in our alignment model. A better alternative would be to use graph convolutional networks (Marcheggiani and Titov, 2017; Kipf and Welling, 2017): neighborhoods in the graph are likely to be more informative for predicting alignments than the neighborhoods in the graph traversal.

Acknowledgments

We thank Marco Damonte, Shay Cohen, Diego Marcheggiani and Wilker Aziz for helpful discussions as well as anonymous reviewers for their suggestions. The project was supported by the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518).

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 47–52.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking.

Jörg Bornschein and Yoshua Bengio. 2015. Reweighted wake-sleep. *International Conference on Learning Representations*.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Jan Buys and Phil Blunsom. 2017. Oxford at semeval-2017 task 9: Neural amr parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 914–919. Association for Computational Linguistics.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An Incremental Parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 536–546.

Shibhansh Dohare and Harish Karnick. 2017. Text Summarization using Abstract Meaning Representation. *arXiv preprint arXiv:1706.01678*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. *International Conference on Learning Representations*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 Task 8: Graph-based AMR Parsing with Infinite Ramp Loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

William Foland and James H. Martin. 2017. Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–472, Vancouver, Canada. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*.

Bevan K. Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *COLING*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *International Conference on Learning Representations*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman M. Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *HLT-NAACL*.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A Simple and Accurate Syntax-Agnostic Neural Model for Dependency-based Semantic Role Labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1507–1516, Copenhagen, Denmark. Association for Computational Linguistics.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning Latent Permutations with Gumbel-Sinkhorn Networks. *International Conference on Learning Representations*. Accepted as poster.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*. AAAI press.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the International Conference on Machine Learning*.

Rik van Noord and Johan Bos. 2017. Neural Semantic Parsing by Character-based Translation: Experiments with Abstract Meaning Representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

George Papandreou and Alan L Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 193–200. IEEE.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the Data Sparsity Issue in Neural AMR Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429.

M. Schuster and K.K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 Task 8: An Extended Transition-based AMR Parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178, San Diego, California. Association for Computational Linguistics.

Chuan Wang and Nianwen Xue. 2017. Getting the Most out of AMR Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust Subgraph Generation Improves Abstract Meaning Representation Parsing. In *ACL*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017. The Neural Noisy Channel. In *International Conference on Learning Representations*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online Segment to Segment Neural Transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1127–1137.

Accurate SHRG-Based Semantic Parsing

Yufei Chen, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{yufei.chen,ws,wanxiaojun}@pku.edu.cn

Abstract

We demonstrate that an SHRG-based parser can produce semantic graphs much more accurately than previously shown, by relating synchronous production rules to the syntacto-semantic composition process. Our parser achieves an accuracy of 90.35 for EDS (89.51 for DMRS) in terms of ELEMENTARY DEPENDENCY MATCH, which is a 4.87 (5.45) point improvement over the best existing data-driven model, indicating, in our view, the importance of linguistically-informed derivation for data-driven semantic parsing. This accuracy is equivalent to that of English Resource Grammar guided models, suggesting that (recurrent) neural network models are able to effectively learn deep linguistic knowledge from annotations.

1 Introduction

Graph-structured semantic representations, e.g. Semantic Dependency Graphs (SDG; Clark et al., 2002; Ivanova et al., 2012), Elementary Dependency Structure (EDS; Oepen and Lønning, 2006), Abstract Meaning Representation (AMR; Banarescu et al., 2013), Dependency-based Minimal Recursion Semantics (DMRS; Copestake, 2009), and Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), provide a lightweight yet effective way to encode rich semantic information of natural language sentences (Kuhlmann and Oepen, 2016). Parsing to semantic graphs has been extensively studied recently.

At the risk of oversimplifying, work in this area can be divided into three types, according to how much structural information of a target graph is explicitly modeled. Parsers of the first type throw an input sentence into a sequence-to-sequence model and leverage the power of deep learning technologies to obtain auxiliary symbols to transform the output sequence into a graph (Peng et al., 2017b; Konstas et al., 2017). The strategy of the second type is to gradually generate a graph in a greedy search fashion (Zhang et al., 2016; Buys and Blunsom, 2017). Usually, a transition system is defined to handle graph construction. The last solution explicitly associates each basic part with a target graph score, and casts parsing as the search for the graphs with highest sum of partial scores (Flanigan et al., 2014; Cao et al., 2017). Although many parsers achieve encouraging results, they are very hard for linguists to interpret and understand, partially because they do not explicitly model the syntacto-semantic composition process which is a significant characteristic of natural languages.

In theory, Synchronous Hyperedge Replacement Grammar (SHRG; Drewes et al., 1997) provides a mathematically sound framework to construct semantic graphs. In practice, however, initial results on the utility of SHRG for semantic parsing were somewhat disappointing (Peng et al., 2015; Peng and Gildea, 2016). In this paper, we show that the performance that can be achieved by an SHRG-based parser is far higher than what has previously been demonstrated. We focus here on relating SHRG rules to the syntacto-semantic composition process because we feel that information about syntax-semantics interface has been underexploited in the data-driven parsing architecture. We demonstrate the feasibility of inducing a high-quality, linguistically-informed SHRG from compositional semantic annotations licensed by English Resource Grammar (ERG; Flickinger, 2000), dubbed English Resource Semantics[1] (ERS). Coupled with RNN-based pars-

[1] http://moin.delph-in.net/ErgSemantics

Model	Grammar	SDG	EDS	DMRS
Data-driven	NO	89.4	85.48	84.16
ERG-based	Unification	92.80	89.58	89.64
SHRG-based	Rewriting	- -	90.39	89.51

Table 1: Parsing accuracy of the best existing grammar-free and -based models as well as our SHRG-based model. Results are copied from (Oepen et al., 2015; Peng et al., 2017a; Buys and Blunsom, 2017).

ing techniques, we build a robust SHRG parser that is able to produce semantic analysis for all sentences. Our parser achieves an accuracy of 90.35 for EDS and 89.51 for DMRS in terms of ELEMENTARY DEPENDENCY MATCH (EDM) which outperforms the best existing grammar-free model (Buys and Blunsom, 2017) by a significant margin (see Table 1). This marked result affirms the value of modeling the syntacto-semantic composition process for semantic parsing.

On sentences that can be parsed by ERG-guided parsers, e.g. PET[2] or ACE[3], significant accuracy gaps between ERG-guided parsers and data-driven parsers are repeatedly reported (see Table 1). The main challenge for ERG-guided parsing is limited coverage. Even for treebanking on WSJ sentences from PTB, such a parser lacks analyses for c.a. 11% of sentences (Oepen et al., 2015). Our parser yields equivalent accuracy to ERG-guided parsers and equivalent coverage, full-coverage in fact, to data-driven parsers. We see this investigation as striking a balance between data-driven and grammar-driven parsing. It is not our goal to argue against the use of unification grammar in high-performance deep linguistic processing. Nevertheless, we do take it as a reflection of two points: (1) (recurrent) neural network models are able to effectively learn deep linguistic knowledge from annotations; (2) practical parsing may benefit from transforming a model-theoretic grammar into a generative-enumerative grammar.

The architecture of our parser has potential uses beyond establishing a strong string-to-graph parser. Our grammar extraction algorithm has some freedom to induce different SHRGs following different linguistic hypothesis, and allows some issues in theoretical linguistics to be empirically investigated. In this paper, we examine the

[2]http://pet.opendfki.de/

[3]http://sweaglesw.org/linguistics/ace/

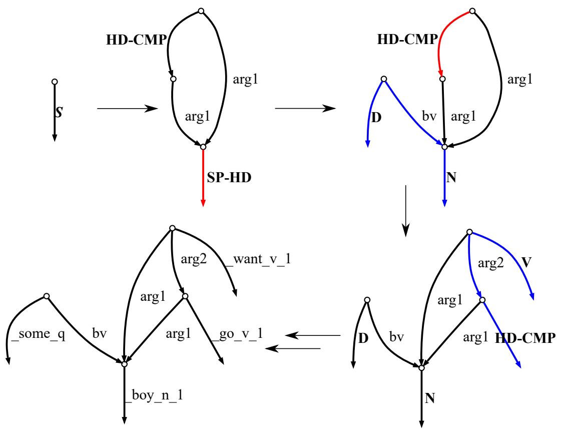


Figure 1: A partial rewriting process of HRG on the semantic graph associated with "Some boys want to go." Lowercase symbols indicate terminal edges, while bold, uppercase symbols indicate nonterminal edges. Red edges are the hyperedges that will be replaced in the next step, while the blue edges in the next step constitute their corresponding RHS graphs.

lexicalist/constructivist hypothesis, a divide across a variety of theoretical frameworks, in an empirical setup. The lexicalist tradition traces its origins to Chomsky (1970) and is widely accepted by various computational grammar formalisms, including CCG, LFG, HPSG and LTAG. A lexicalist approach argues that the lexical properties of words determine their syntactic and semantic behaviors. The constructivist perspective, e.g. Borer's Exo-Skeletal approach (2005b; 2005a; 2013), emphasizes the role of syntax in constructing meanings. In this paper, we focus on lexicalist and constructivist hypotheses for syntacto-semantic composition. We present our computation-oriented analysis in §6. Under the architecture of our neural parser, a construction grammar works much better than a lexicalized grammar.

Our parser is available at https://github.com/draplater/hrg-parser/.

2 Hyperedge Replacement Grammar

Hyperedge replacement grammar (HRG) is a context-free rewriting formalism for graph generation (Drewes et al., 1997). An edge-labeled, directed hypergraph is a tuple $H = \langle V, E, l, X \rangle$, where V is a finite set of nodes, and $E \subseteq V^+$ is a finite set of hyperedges. A hyperedge is an extension of a normal edge which can connect to more than two nodes or only one node. $l : E \rightarrow L$

Algorithm 1 Hyperedge Replacement Grammar Extraction Algorithm

```
Require: Input syntactic tree $T$, hypergraph $g$
 1: RULES $\leftarrow \{\}$
 2: for tree node $n$ in postorder traversal of $T$ do
Ensure: Rewriting rule of node $n$ is A $\rightarrow$ B + C, spans of node A, B, C are SPAN-A, SPAN-B, SPAN-C
 3:     SPANS $\leftarrow$ {SPAN-A, SPAN-B, SPAN-C}
 4:     C-EDGES $\leftarrow \{e|e \in \text{EDGES}(g) \wedge \text{SPAN}(e) \in \text{SPANS}\}$
 5:     ALL-NODES $\leftarrow \{s|s \in \text{NODES}(g) \wedge \exists e \in \text{C-EDGES s.t. } s \in \text{NODES}(e)\}$
 6:     S-EDGES $\leftarrow \{e|e \in \text{EDGES}(g) \wedge e \text{ is structual edge} \wedge \forall s \in \text{NODES}(e) \implies s \in \text{C-EDGES}\}$
 7:     ALL-EDGES = C-EDGES $\cup$ S-EDGES
 8:     INTERNAL-NODES $\leftarrow \{\}$
 9:     EXTERNAL-NODES $\leftarrow \{\}$
10:     for node $s$ in ALL-NODES do
11:         if $\forall e \in \text{EDGES}(g), s \in \text{NODES}(e) \implies e \in \text{ALL-EDGES}$ then
12:             INTERNAL-NODES $\leftarrow$ INTERNAL-NODES $\cup \{s\}$
13:         else
14:             EXTERNAL-NODES $\leftarrow$ EXTERNAL-NODES $\cup \{s\}$
15:         end if
16:     end for
17:     RULES $\leftarrow$ RULES $\cup$ {(A, ALL-EDGES, INTERNAL-NODES, EXTERNAL-NODES)}
18: end for
```

assigns a label from a finite set L to each edge. $X \in V^*$ defines an ordered list of nodes, i.e., **external nodes**, which specify the connecting parts when replacing a hyperedge.

An HRG $G = \langle N, T, P, S \rangle$ is a graph rewriting system, where N and T are two disjoint finite sets of nonterminal and terminal symbols respectively. $S \in N$ is the start symbol. P is a finite set of productions of the form $A \rightarrow R$, where the left hand side (LHS) $A \in N$, and the right hand side (RHS) R is a hypergraph with edge labels over $N \cup T$. The rewriting process replaces a nonterminal hyperedge with the graph fragment specified by a production's RHS, attaching each external node to the matched node of the corresponding LHS. An example is shown in Figure 1. Following Chiang et al. (2013), we make the nodes only describe connections between edges and store no other information.

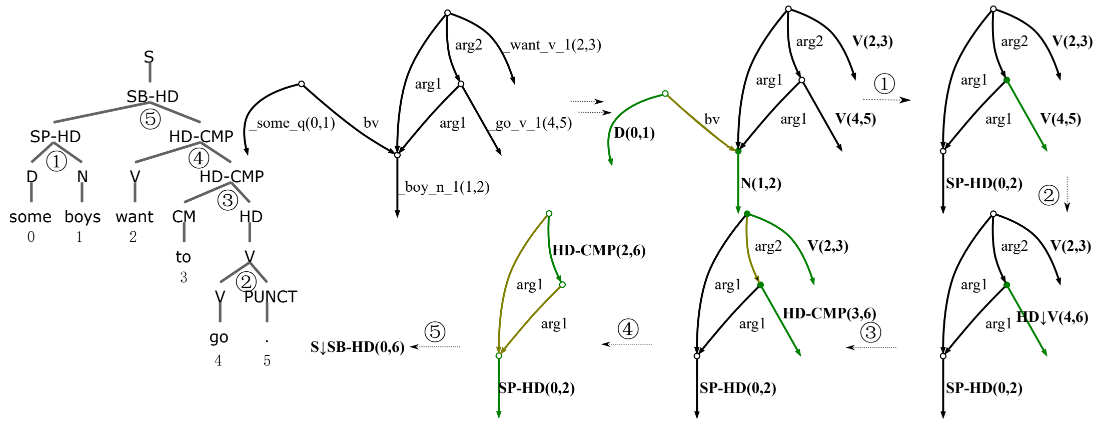
A synchronous grammar defines mappings between different grammars. Here we focus on relating a string grammar, CFG in our case, to a graph grammar, i.e., HRG. SHRG can be represented as tuple $G = \langle N, T, T', P, S \rangle$. N is a finite set of nonterminal symbols in both CFG and HRG. T' and T are finite sets of terminal symbols in CFG and HRG, respectively. $S \in N$ is the start symbol. P is a finite set of productions of the form $A \rightarrow \langle R, R', \sim \rangle$, where $A \in N$, R is a hypergraph fragment with edge labels over $N \cup T$, and R' is a symbol sequence over $N \cup T'$. $\sim$ is a mapping between the nonterminals in R and R'. When a coherent CFG derivation is ready, we can *interpret* it using the corresponding HRG and get a semantic graph.

3 Grammar Extraction

3.1 Graph Representations for ERS

ERS are richly detailed semantic representations produced by the ERG, a hand-crafted, linguistically-motivated HPSG grammar for English. Beyond basic predicate–argument structures, ERS also includes other information about various complex phenomena such as the distinction between scopal and non-scopal arguments, conditionals, comparatives, and many others. ERS are in the formalism of Minimal Recursion Semantics (MRS; Copestake et al., 2005), but can be expressed in different ways. Semantic graphs, including EDS and DMRS, can be reduced from the standard feature structure encoded representations, with or without a loss of information. In this paper, we conduct experiments on ERS data, but our grammar extraction algorithm and the parser are not limited to ERS.

One distinguished characteristic of ERS is that the construction of ERS strictly follows the prin-



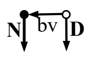
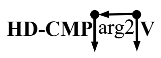
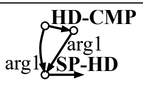
	①	②	③	④	⑤
Shared LHS	SP-HD	HD↓V	HD-CMP	HD-CMP	S↓SP-HD
RHS (syntax)	D + N	V + PUNCT	CM + HD↓V	V + HD-CMP	SP-HD + HD-CMP
RHS (semantics)					

Figure 2: The grammar extraction process of the running example. Conceptual edges which are directly aligned with the syntactic rules are painted in green. The span-based alignment is shown in the parentheses. Structural edges that connect conceptual edges are painted in brown. Green edges and brown edges together form the subgraph, which acts as RHS in the HRG rule. External nodes are represented as solid dots.

ciple of compositionality (Bender et al., 2015). A precise syntax-semantics interface is introduced to guarantee compositionality and therefore all meaning units can be traced back to linguistic signals, including both lexical and constructional ones. Take Figure 2 for example. Every concept, e.g. the existence quantifier _some_q, is associated with a surface string. We favor such correspondence not because it eases extraction of SHRGs, but because we emphasize sentence meanings that are from forms. The connection between syntax (sentence form) and semantics (word and sentence meaning) is fundamental to the study of language.

3.2 The Algorithm

We introduce a novel SHRG extraction algorithm, which requires and only requires alignments between conceptual edges and surface strings. A tree is also required, but this tree does not have to be a gold-standard syntactic tree. All trees that are compatible with an alignment can be used. The syntactic part of DeepBank is a phrase structure which describes HPSG derivation. The vast majority of syntactic rules in DeepBank are binary, and the rest are unary. In §5, we report evaluation results based on DeepBank trees.

A conceptual graph is composed by two kinds of edges: 1) **conceptual edges** that carry semantic concept information and are connected with only one node, and 2) **structural edges** that build relationships among concepts by connecting nodes. The grammar extraction process repeatedly replaces a subgraph with a nonterminal hyperedge, defining the nonterminal symbol as LHS and the subgraph as RHS. The key problem is to identify an appropriate subgraph in each step. To this end, we take advantage of DeepBank's accurate and fine-grained alignments between the surface string in syntactic tree and concepts in semantic graphs.

To extract the HRG rule synchronized with the syntactic rewriting rule A → B + C, we assume that conceptual edges sharing common spans with A, B or C are in the same subgraph. This subgraph acts as the RHS of the HRG rule. We make the extraction process go in the direction of postorder traversal of the syntactic tree, to ensure that all sub-spans of A, B or C are already replaced with hyperedges. We then add the structural edges that connect the above conceptual edges to RHS. After the subgraph is identified, it is easy to distinguish between internal nodes and external nodes.

If all edges connected to a node are in the subgraph, this node is an internal node. Otherwise, it is external node. Finally, the subgraph is replaced with a nonterminal edge. Algorithm 1 presents a precise demonstration and Figure 2 illustrates an example.

4 A Neural SHRG Parser

Under the SHRG formalism, semantic parsing can be divided into two steps: syntactic parsing and semantic interpretation. Syntactic parsing utilizes the CFG part to get a derivation that is shared by the HRG part. At one derivation step, there may be more than one HRG rule applicable. In this case, we need a semantic disambiguation model to choose a good one.

4.1 Syntactic Parsing

Following the LSTM-Minus approach proposed by Cross and Huang (2016), we build a constituent parser with a CKY decoder. We denote the output vectors of forward and backward LSTM as $\boldsymbol{f}_i$ and $\boldsymbol{b}_i$. The feature $\boldsymbol{s}_{i,j}$ of a span (i, j) can be calculated from the differences of LSTM encodings:

$$\boldsymbol{s}_{i,j} = (\boldsymbol{f}_j - \boldsymbol{f}_i) \oplus (\boldsymbol{b}_i - \boldsymbol{b}_j)$$

The operator $\oplus$ indicates the concatenation of two vectors. Constituency parsing can be regarded as predicting scores for spans and labels, and getting the best syntactic tree with dynamic programming. Following Stern et al. (2017)'s approach, We calculate the span scores $\textsc{Score}_{\text{span}}(i, j)$ and labels scores $\textsc{Score}_{\text{label}}(i, j, l)$ from $\boldsymbol{s}_{i,j}$ with multilayer perceptrons (MLPs):

$$\textsc{Score}_{\text{span}}(i, j) = \text{MLP}_{\text{span}}(\boldsymbol{s}_{i,j})$$

$$\textsc{Score}_{\text{label}}(i, j, l) = \text{MLP}_{\text{label}}(\boldsymbol{s}_{i,j})[l]$$

$\boldsymbol{x}[i]$ indicates the ith element of a vector $\boldsymbol{x}$. We condense the unary chains into one label to ensure that only one rule is corresponds with a specific span. Because the construction rules from DeepBank are either unary or binary, we do not deal with binarization.

Because the SHRG synchronizes at rule level, we need to restrict the parser to ensure that the output agrees with the known rules. The restriction can be directly added into the CKY decoder. To simplify the semantic interpretation process, we add extra label information to enrich the nonterminals in CFG rules. In particular, we consider the count of external nodes of a corresponding HRG rule. For example, the LHS of rule ④ in Figure 2 will be labeled as "`HD-CMP#2`", since the RHS of its HRG counterpart has two external nodes.

4.2 Semantic Interpretation

When a phrase structure tree, i.e., a derivation tree, T is available, semantic interpretation can be regarded as *translating* T to the derivation of graph construction by assigning a corresponding HRG rule to each syntactic counterpart. Our approach to finding the optimal HRG rule combination $\hat{R} = \{r_1, r_2, ...\}$ from the search space $\mathcal{R}(T)$:

$$\hat{R} = \text{argmax}_{R \in \mathcal{R}(T)} \textsc{Score}(R|T) \qquad (1)$$

To solve this optimization problem, we implement a greedy search decoder and a bottom-up beam search decoder. The final semantic graph G is read off from $\hat{R}$.

4.2.1 Greedy Search Model

In this model, we assume that each HRG rule is selected independently of the others. The score of G is defined as the sum of all rule scores:

$$\textsc{Score}(R = \{r_1, r_2, ...\}|T) = \sum_{r \in R} \textsc{Score}(r|T)$$

The maximization of the graph score can be decomposed into the maximization of each rule score. $\textsc{Score}(r|T)$ can be calculated in many ways. Count-based approach is the simplest one, where the rule score is estimated by its frequency in the training data. We also evaluate a sophisticated scoring method, i.e., training a classifier based on rule embedding:

$$\textsc{Score}(r|T) = \text{MLP}(\boldsymbol{s}_{i,j} \oplus \boldsymbol{r})$$

Inspired by the bag-of-words model, we represent the rule as bag of edge labels. The i-th position in $\boldsymbol{r}$ indicates the number of times the i-th label appears in the rule.

4.2.2 Bottom-Up Beam Search Model

We can also leverage structured prediction to approximate $\textsc{Score}(R|T)$ and employ principled decoding algorithms to solve the optimization problem (1). We propose a factorization model to assign scores to the graph and subgraphs in the intermediate state. The score of a certain graph can be seen as the sum of each factor score.

$$\textsc{Score}(R|T) = \sum_{i \in \textsc{Part}(R,T)} \textsc{ScorePart}(i)$$

We use predicates and arguments as factors for scoring. There are two kinds of factors: 1) A conceptual edge aligned with span (i, j) taking predicate name p. We use the span embedding $\boldsymbol{s}_{i,j}$ as features, and scoring with non-linear transformation:

$$\textsc{ScorePart}_{\text{pred}}(i, j, p) = \text{MLP}_{\text{pred}}(\boldsymbol{s}_{i,j})[p]$$

2) A structural edge with label L connects with predicates p_a and p_b, which are aligned with spans (i_1, j_1) and (i_2, j_2) respectively. We use the span embedding $\boldsymbol{s}_{i_1,j_1}$, $\boldsymbol{s}_{i_2,j_2}$ and random initialized predicate embedding $\boldsymbol{p}_a$, $\boldsymbol{p}_b$ as features, and scoring with non-linear transformation:

$$\textsc{ScorePart}_{\text{arg}}(i_1, j_1, i_2, j_2, \boldsymbol{p}_a, \boldsymbol{p}_b, L)$$
$$= \text{MLP}_{\text{arg}}(\boldsymbol{s}_{i_1,j_1} \oplus \boldsymbol{s}_{i_2,j_2} \oplus \boldsymbol{p}_a \oplus \boldsymbol{p}_b)[L]$$

We assign a beam to each node in the syntactic tree. To ensure that we always get a subgraph which does not contain any nonterminal edges during the search process, we perform the beam search in the bottom-up direction. We only reserve top k subgraphs in each beam. Figure 3 illustrates the process.

4.3 Training

The objective of training is to make the score of the correct graph higher than incorrect graphs. We use the score difference between the correct graph R_g and the highest scoring incorrect graph as the loss:

$$\text{loss} = \max_{\hat{R} \neq R_g} \textsc{Score}(\hat{R}|T) - \textsc{Score}(R_g|T)$$

Following (Kiperwasser and Goldberg, 2016)'s experience of loss augmented inference, in order to update graphs which have high model scores but are very wrong, we augment each factor belonging to the gold graph by adding a penalty term c to its score. Finally the *loss* term is:

$$\text{loss} = \textsc{Score}(R_g|T) - \sum_{i \in \textsc{Part}(R_g,T)} c -$$
$$\max(\textsc{Score}(\hat{R}|T) - \sum_{i \in \textsc{Part}(\hat{R},T) \cap \textsc{Part}(R_g,T)} c)$$

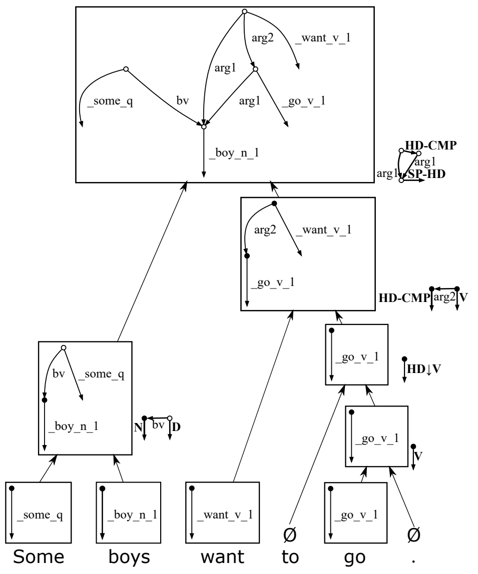


Figure 3: The semantic interpretation process. The interpretation performs bottom-up beam search to get a bunch of high-scored subgraphs for each node in the derivation tree.

5 Experiments

5.1 Set-up

DeepBank is an annotation of the Penn TreeBank Wall Street Journal which is annotated under the formalism of HPSG. We use DeepBank version 1.1, corresponding to ERG 1214, and use the *standard* data split. Therefore the numeric performance can be directly compared to results reported in Buys and Blunsom (2017). We use the pyDelphin library to extract DMRS and EDS graphs and use the tool provided by jigsaw[4] to separate punctuation marks from the words they attach to. We use DyNet[5] to implement our neural models, and automatic batch technique (Neubig et al., 2017) in DyNet to perform mini-batch gradient descent training. The detailed network hyper-parameters can be seen in Table 2. The same pre-trained word embedding as (Kiperwasser and Goldberg, 2016) is employed.

5.2 Results of Grammar Extraction

DeepBank provides fine-grained syntactic trees with rich information. For example, the label `SP-HD_HC_C` denotes that this is a "head+specifier" construction, where the semantic head is also the syntactic head. But there

[4] www.coli.uni-saarland.de/~yzhang/files/jigsaw.jar

[5] https://github.com/clab/dynet

Hyperparamter	**Value**
Batch size	32
Pre-trained word embedding dimension	100
Random-initialized word embedding dimension	150
LSTM Layer count	2
LSTM dimension (each direction)	250
MLP hidden layer count	1
MLP hidden layer dimension	250
penalty term c	1

Table 2: Hyperparamters used in the experiments.

	#EP	#Rule			#Instance
		Fine	Coarse	Unlabeled	
EDS	1	49689	14234	1476	676817
	2	9616	3424	488	64708
	3	2739	1486	280	11195
	4	1059	732	248	2071
	5+	508	418	251	655
DMRS	1	50668	15745	2688	657999
	2	11428	4418	896	79888
	3	3576	1929	465	14237
	4	1237	873	299	2561
	5+	669	557	297	901

Table 3: Statistics of SHRG rules with different label type by the count of external points in EDS and DMRS representations.

is also the potential for data sparseness. In our experiments, we extract SHRG with three kinds of labels: fine-grained labels, coarse-grained labels and single Xs (meaning unlabeled parsing). The fine-grained labels are the original labels, namely fine-grained construction types. We use the part before the first underscore of each label, e.g. SP-HD, as a coarse-grained label. The coarse-grained labels are more like the highly generalized rule schemata proposed by Pollard and Sag (1994). Some statistics are shown in Table 3.

Instead of using gold-standard trees to extract a synchronous grammar, we also tried randomly-generated alignment-compatible trees. The result is shown in Table 4. Gold standard trees exhibit a low entropy, indicating a high regularity.

5.3 Results of Syntactic Parsing

In addition to the standard evaluation method for phrase-structure parsing, we find a more suitable measurement, i.e. condensed score, for our task. Because we condense unary rule chains into one label and extract synchronous grammar under this condensed syntactic tree, it is better to calculate the correctness of the condensed label rather than a single label. The additional label "#N" that indicates the number of external points is also considered in our condensed score evaluation method. The result is shown in Table 5.

Tree Type	1	2	3	4	5+
Gold	1476	488	280	248	251
Fuzzy 1	12710	7591	7963	6578	8998
Fuzzy 2	13606	7355	7228	6090	9112
Fuzzy 3	12278	8228	8462	7039	9946

Table 4: Comparison of grammars extracted from unlabeled gold trees and randomly-generated alignment-compatible trees (”fuzzy” trees).

Label	Standard				Condensed	
	P	R	F	POS	BCKT	POS
Fine	90.81	91.19	91.00	94.40	87.09	92.98
Coarse	90.78	91.24	91.01	98.30	87.93	95.98

Table 5: Accuracy of syntactic parsing under different labels on development data. We add the count of external nodes of corresponding HRG rule. "POS" concerns the prediction of preterminals, while "BCKT" denotes bracketing.

5.4 Results of Semantic Interpretation

Dridan and Oepen (2011) proposed the EDM metric to evaluate the performance the ERS-based graphs. EDM uses the alignment between the nodes in a graph and the spans in a string to detect the common parts between two graphs. It converts the predicate and predicate–argument relationship to comparable triples and calculates the correctness in these triples. A predicate of label L and span S is denoted as triple (S, NAME, L) and a relationship R between the predicate labelled P and argument labelled A is denoted as triple (P, R, A). We calculate the F_1 value of the total triples as EDM score. Similarity, we compute the F_1 score of only predicate triples and only the relation triples as EDM_P and EDM_A.

We reuse the word embeddings and bidirectional LSTM in the trained syntactic parsing model to extract span embedding $\boldsymbol{s}_{i,j}$. The results of the count-based model, rule embedding model and structured model with beam decoder are summarized in Table 6. We report the standard EDM metrics. The count-based model can achieve considerably good results, showing the correctness of our grammar extraction method. We also try different labels for the syntactic trees. The results

Model	EDM$_P$	EDM$_A$	EDM
Count Based	90.12	81.96	86.03
Rule Embedding	93.41	84.84	89.11
Beam Search	93.48	87.88	90.67

Table 6: The EDM score on EDS development data with different model: count based greedy search, rule embedding greedy search and beam search. We use syntactic trees with coarse-grained labels.

Data	Label	EDM$_P$	EDM$_A$	EDM
EDS	Fine	92.70	87.77	90.23
	Coarse	93.48	87.88	90.67
DMRS	Fine	92.52	86.47	89.46
	Coarse	93.60	86.62	90.07

Table 7: Accuracy on the development data under different labels of syntactic tree and beam search.

are shown in Table 7. Models based on coarse-grained labels achieve optimal performance. The results on test set of EDS data are shown in Table 8. We achieve state-of-the-art performance with a remarkable improvement over Buys and Blunsom (2017)'s neural parser.

6 On Syntax-Semantics Interface

In this paper, we empirically study the lexicalist/constructivist hypothesis, a divide across a variety of theoretical frameworks, taking semantic parsing as a case study. Although the original grammar that guides the annotation of ERS data, namely ERG, is highly lexicalized in that the majority of information is encoded in lexical entries (or lexical rules) as opposed to being represented in constructions (i.e., rules operating on phrases), our grammar extraction algorithm has some freedom to induce different SHRGs that choose between the lexicalist and constructivist approaches. We modify algorithm 1 to follow the key insights of the lexicalist approach. This is done by considering all outgoing edges when finding the subgraph of the lexical rules. The differences between two kinds of grammars is shown in Table 9.

Different grammars allow the lexicalist/constructivist issue in theoretical linguistics to be empirically examined. The comparison of the counts of rules in each grammar is summarized in Table 11, from which we can see that the sizes of the grammars are comparable. However, the parsing results are quite different, as shown

	Model	EDM$_P$	EDM$_A$	EDM
EDS	Buys and Blunsom	88.14	82.20	85.48
	ACE	91.82	86.92	89.58
	Ours	93.15	87.59	90.35
DMRS	Buys and Blunsom	87.54	80.10	84.16
	ACE	92.08	86.77	89.64
	Ours	93.11	86.01	89.51

Table 8: Accuracy on the test set. We use syntactic trees of coarse-grained labels and beam search.

in Table 10. A construction grammar works much better than a lexicalized grammar under the architecture of our neural parser. We take this comparison as informative since lexicalist approaches are more widely accepted by various computational grammar formalisms, including CCG, LFG, HPSG and LTAG.

We think that the success of applying SHRG to resolve semantic parsing highly relies on the compositionality nature of ERS' sentence-level semantic annotation. This is the property that makes sure the extracted rules are consistent and regular. Previous investigation by Peng et al. (2015) on SHRG-based semantic parsing utilizes AMR-Bank which lacks this property to some extent (see Bender et al.'s argument). We think this may be one reason for the disappointing parsing performance. Think about the AMR graph associated "John wants Bob to believe that he saw him." The AMR's annotation for co-reference is a kind of non-compositional, speaker meaning, and results in grammar sparseness.

7 On Deep Linguistic Knowledge

Semantic annotations have a tremendous impact on semantic parsing. In parallel with developing new semantic annotations, e.g. AMRBank, there is a resurgence of interest in exploring existing annotations grounded under deep grammar formalisms, such as semantic analysis provided by ERS (Flickinger, 2000). In stark contrast, it seems that only the annotation results gain interests, but not the core annotation engine—knowledge-extensive grammar.

The tendency to continually ignore the positive impact of precision grammar on semantic parsing is somewhat strange. For sentences that can be parsed by an ERG-guided parser, there is a significant accuracy gap which is repeatedly reported. See Table 1 for recent results. The main challenges for precision grammar-guided parsing are unsat-

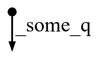
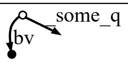
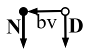
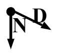
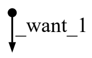
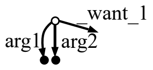
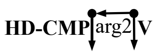
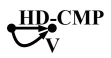
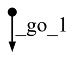
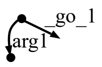
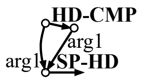
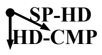
Lexicon	Construction	Lexicalized	CFG Counterpart	Construction	Lexicalized
some			SP-HD → D + N		
want			HD-CMP → V + HD-CMP		
go			S↓SP-HD → SP-HD + HD-CMP		

Table 9: Rules of lexicalized and construction grammars that are extracted from the running example.

Grammar	EDM$_P$	EDM$_A$	EDM
Construction	93.48	87.88	90.67
Lexicalized	92.14	81.05	86.63

Table 10: The EDM score on EDS development data with construction grammar and lexicalized grammar using syntax trees of coarse-grained labels and beam search.

Grammar	1	2	3	4	5+
Construction	14234	3424	1486	732	418
Lexicalized	11653	5938	2358	396	11

Table 11: Comparison of the construction grammar and the lexicalized grammar extracted from EDS data. We use syntax trees of coarse-grained labels.

isfactory coverage and efficiency that limit their uses in NLP applications. Even for treebanking on newswire data, i.e., Wall Street Journal data from Penn TreeBank (Marcus et al., 1993), ERG lacks analyses for c.a. 11% of sentences (Oepen et al., 2015). For text data from the web, e.g. tweets, this problem is even more serious. Moreover, checking all possible linguistic constraints makes a grammar-guided parser too slow for many realistic NLP applications. Robustness and efficiency, thus, are two major problems for practical NLP applications.

Recent encouraging progress achieved with purely data-driven models helps resolve the above two problems. Nevertheless, it seems too radical to remove all explicit linguistic knowledge about the syntacto-semantic composition process, the key characteristics of natural languages. In this paper, we introduce a neural SHRG-based semantic parser that strikes a balance between data-driven and grammar-guided parsing. We encode deep linguistic knowledge partially in a symbolic way and partially in a statistical way. It is worth noting that the symbolic system is a derivational, generative-enumerative grammar, while the origin of the data source is grounded under a representational, model-theoretic grammar. While grammar writers may favor the convenience provided by a unification grammar formalism, a practical parser may re-use algorithms by another formalism by *translating* grammars through *data*. Experiments also suggest that (recurrent) neural network models are able to effectively gain some deep linguistic knowledge from annotations.

8 Conclusion

The advantages of using graph grammars to resolve semantic parsing is clear in concept but underexploited in practice. Here, we have shown ways to improve SHRG-based string-to-semantic-graph parsing. Especially, we emphasize the importance of modeling syntax-semantic interface and the compositional property of semantic annotations. Just like recent explorations on many other NLP tasks, we also show that neural network models are very powerful to advance deep language understanding.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61772036, 61331011) and the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Weiwei Sun is the corresponding author.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 228–238. http://www.aclweb.org/anthology/P13-1023.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann A. Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics, IWCS 2015, 15-17 April, 2015, Queen Mary University of London, London, UK*. pages 239–249. http://aclweb.org/anthology/W/W15/W15-0128.pdf.

H. Borer. 2005a. *In Name Only*. Hagit Borer. Oxford University Press. https://books.google.com/books?id=cAEmAQAAIAAJ.

H. Borer. 2005b. *The Normal Course of Events*. Hagit Borer. Oxford University Press. https://books.google.com/books?id=M48UPLst_MQC.

H. Borer. 2013. *Structuring Sense: Volume III: Taking Form*. Borer, Hagit. OUP Oxford. https://books.google.com/books?id=tUkGAQAAQBAJ.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1215–1226. http://aclweb.org/anthology/P17-1112.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 2110–2120. http://aclweb.org/anthology/P17-1193.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with Hyperedge Replacement Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 924–932. http://www.aclweb.org/anthology/P13-1091.

Noam Chomsky. 1970. Remarks on nominalization. In R. A. Jacobs and P. S. Rosenbaum, editors, *Readings in English Transformational Grammar*, Waltham, MA, pages 170–221.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*. pages 327–334. http://www.aclweb.org/anthology/P02-1042.pdf.

Ann Copestake. 2009. ***Invited Talk:*** slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Association for Computational Linguistics, Athens, Greece, pages 1–9. http://www.aclweb.org/anthology/E09-1001.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* pages 281–332.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1–11. https://aclweb.org/anthology/D16-1001.

F. Drewes, H.-J. Kreowski, and A. Habel. 1997. Hyperedge Replacement Graph Grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific Publishing Co., Inc., River Edge, NJ, USA, pages 95–162. http://dl.acm.org/citation.cfm?id=278918.278927.

Rebecca Dridan and Stephan Oepen. 2011. Parser evaluation using elementary dependency matching. In *Proceedings of the 12th International Conference on Parsing Technologies*. Dublin, Ireland, pages 225–230.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1426–1436. http://www.aclweb.org/anthology/P14-1134.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Nat. Lang. Eng.* 6(1):15–28.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea, pages 2–11.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 146–157. http://aclweb.org/anthology/P17-1014.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics* 42(4):819–827.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017. On-the-fly operation batching in dynamic computation graphs. In *Advances in Neural Information Processing Systems*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresová. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*. European Language Resources Association (ELRA), Genoa, Italy. ACL Anthology Identifier: L06-1214.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 2037–2048. http://aclweb.org/anthology/P17-1186.

Xiaochang Peng and Daniel Gildea. 2016. Uofr at semeval-2016 task 8: Learning Synchronous Hyperedge Replacement Grammar for AMR parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1185–1189. http://www.aclweb.org/anthology/S16-1183.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 32–41. http://www.aclweb.org/anthology/K15-1004.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 366–375. http://www.aclweb.org/anthology/E17-1035.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 818–827. http://aclweb.org/anthology/P17-1076.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.

Using Intermediate Representations to Solve Math Word Problems

Danqing Huang[1]*, **Jin-Ge Yao**[2], **Chin-Yew Lin**[2], **Qingyu Zhou**[3], **and Jian Yin**[1]
{huangdq2@mail2,issjyin@mail}.sysu.edu.cn
{Jinge.Yao,cyl}@microsoft.com
qyzhou@hit.edu.cn
[1] The School of Data and Computer Science, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2]Microsoft Research [3] Harbin Institute of Technology

Abstract

To solve math word problems, previous statistical approaches attempt at learning a direct mapping from a problem description to its corresponding equation system. However, such mappings do not include the information of a few higher-order operations that cannot be explicitly represented in equations but are required to solve the problem. The gap between natural language and equations makes it difficult for a learned model to generalize from limited data. In this work we present an intermediate meaning representation scheme that tries to reduce this gap. We use a sequence-to-sequence model with a novel attention regularization term to generate the intermediate forms, then execute them to obtain the final answers. Since the intermediate forms are latent, we propose an iterative labeling framework for learning by leveraging supervision signals from both equations and answers. Our experiments show using intermediate forms outperforms directly predicting equations.

1 Introduction

There is a growing interest in math word problem solving (Kushman et al., 2014; Koncel-Kedziorski et al., 2015; Huang et al., 2017; Roy and Roth, 2018). It requires reasoning with respect to sets of numbers or variables, which is an essential capability in many other natural language understanding tasks. Consider the math problems shown in Table 1. To solve the problems, one needs to know how many numbers to be summed up (e.g. "2 numbers/3 numbers"), and the relation between variables ("the first/second number"). However, an equation system does not encode these information explicitly. For example, an equation represents "the sum of 2 numbers" as $(x + y)$ and "the sum of 3 numbers" as $(x + y + z)$. This makes it difficult to generalize to cases unseen from data (e.g. "the sum of 100 numbers").

1) <u>**The sum of 2 numbers**</u> is 18. **The first number** is 4 more than **the second number**. Find the two numbers. **Equations:** $x + y = 18, x = y + 4$
2) <u>**The sum of 3 numbers**</u> is 15. **The larger number** is 4 times **the smallest** and **the middle number** is 5. What are the numbers? **Equations:** $x + y + z = 15, x = 4 * z, y = 5$

Table 1: Math word problems. Equations have lost the information of *count, max, ordinal* operations.

This paper presents a new intermediate meaning representation scheme for solving math problems, aiming at closing the semantic gap between natural language and equations. To generate the intermediate forms, we adapt a sequence-to-sequence (seq2seq) network following recent work that tries to generate equations from problem descriptions for this task. Wang et al. (2017) have shown that seq2seq models have the power to generate equations of which problem types do not exist in training data. In this paper, we propose a new method which adds an extra meaning representation and generate an intermediate form as output. Additionally, we observe that the attention weights of the seq2seq model repetitively concentrates on numbers in the problem description. To address the issue, we further propose to use a form of attention regularization.

To train the model without explicit annotations of intermediate forms, we propose an iterative la-

*Work done while this author was an intern at Microsoft Research.

beling framework to leverage signals from both equations and their solutions. We first derive possible intermediate forms with ambiguity using the gold-standard equation systems, and use these forms for training to get a pre-trained model. Then we iteratively refine the intermediate forms using the learned model and the signals from the gold-standard answers.

We conduct experiments on two publicly available math problem datasets. Our experimental results show that using the intermediate forms for training performs significantly better than directly mapping problems to equation systems. Furthermore, our iterative labeling framework creates better labeled data with intermediate forms for training, which leads to improved performance.

To summarize, our contributions include:

- We present a new intermediate meaning representation scheme for solving math problems.
- We design an iterative labeling framework to automatically augment training data with intermediate meaning representation.
- We propose using attention regularization in training to address the issue of incorrect attention in the seq2seq model.
- We verify the effectiveness of our proposed solutions by conducting experiments and analysis on real-world datasets.

2 Meaning Representation

In this section, we will compare meaning representations for solving math problems and introduce the proposed intermediate meaning representation.

2.1 Meaning Representations for Math Problem Solving

We first discuss two meaning representation schemes for math problem solving.

An **equation system** is a collection of one or more equations involving the same set of variables, which should be considered as highly abstractive symbolic representation.

The **Dolphin Language** is introduced by Shi et al. (2015). It contains about 35 math-related classes and over 200 math-related functions, with additional classes and functions automatically mined from Freebase.

Unfortunately, these representation schemes do not generalize well. Consider the two problems listed in Table 2. They belong to the same type of problems asking about the summation of consecutive integers. However, their meaning representations are very different in the Dolphin language and in equations. On one hand, the Dolphin language aligns too closely with natural utterances. Since the math problem descriptions are diverse in using various nouns and verbs, Dolphin language may represent the same type of problems differently. On the other hand, an equation system does not explicitly represent useful problem solving information such as "*number of variables*" and "*numbers are consecutive*"

2.2 Intermediate Meaning Representation

To bridge the semantic gap between the two meaning representations, we present a new intermediate meaning representation scheme for math problem solving. It consists of 6 *classes* and 23 *functions*. Here a *class* is the set of entities with the same semantic properties and can be inherited (e.g. $2 \in int$, $int \sqsubseteq num$). A *function* is comprised of a name, a list of arguments with corresponding types, and a return type. For example, there are two overloaded definitions for the function *math#sum* (Table 3). These forms can be constructed by recursively applying joint operations on functions with class type constraints. Our representation scheme attempts to borrow the explicit use of higher-order functions from the Dolphin language, while avoiding to be too specific. Meanwhile, the intermediate forms are not as concise as the equation systems (Table 2). We leave more detailed definitions to the supplement material due to space limit.

3 Problem Statement

Given a math word problem p, our goal is to predict its answer A_p. For each problem we have annotations of both the equation system E_p and the answer A_p available for training. The latent intermediate form will be denoted as LF_p.

We formulate math problem solving as a sequence prediction task, taking the sequence of words in a math problem as input and generating a sequence of tokens in its corresponding intermediate form as output. We then execute the intermediate form to obtain the final answer. We evaluate the task using answer accuracy on two publicly

Problem 1: Find three consecutive integers with a sum of 267.
Dolphin Language: vf.find(cat('integers'), count:3, adj.consecutive, (math#sum(pron.that, 267, det.a)))
Equation: $x + (x + 1) + (x + 2) = 267$
This work: math#consecutive(3), math#sum(cnt: 3) = 267
Problem 2: What are 5 consecutive numbers total 95?
Dolphin Language: wh.vf.math.total((cat('numbers'), count:5, pron.what, adj.consecutive), 95)
Equation: $x + (x + 1) + (x + 2) + (x + 3) + (x + 4) = 95$
This work: math#consecutive(5), math#sum(cnt: 5) = 95

Table 2: Different representations for math problems. Dolphin language is detailed ('all words'). Equation system is coarse that it represents many functions implicitly, such as "*count*", "*consecutive*".

Classes
int, float, num, unk, var, list
Functions
ret:int count($1:list): number of variables in $1
ret:var max($1:list): variable of max value in $1
ret:var math#product($1,$2:var): $1 times $2
ret:var math#sum($1:list): sum of variables in $1
ret:var math#sum(cnt:$1:int): sum of $1 unks
Example
Four times the sum of three and a number is 10. -> math#product(4, math#sum(3, m))=10

Table 3: Examples of classes and functions in our intermediate representation. "ret" stands for return type. $1, $2 are arguments with its types.

available math word problem datasets[1]:

- **Number Word Problem** (NumWord) is created by Shi et al. (2015). It contains 1,878 number word problems (verbally expressed number problems, such as the examples in Table 1). Its linear subset (subset of problems that can be solved by linear equation systems) has 986 problems, only involving four basic operations $\{+, -, *, /\}$.

- **Dolphin18K** is created by Huang et al. (2016). It contains 18,711 math word problems collected from Yahoo! Answers[2]. Since it contains some problems without equations, we only use the subset of 10,644 problems which are paired with their equation systems.

[1]Other small datasets with 4 basic operations $\{+, -, *, /\}$ and only one unknown variable are considered as subsets of our datasets.

[2]https://answers.yahoo.com/

4 Model

In this section, we describe (1) the basic sequence-to-sequence model, and (2) attention regularization.

4.1 Sequence-to-Sequence RNN Model

Our baseline model is based on sequence-to-sequence learning (Sutskever et al., 2014) with attention (Bahdanau et al., 2015) and copy mechanism (Gulcehre et al., 2016; Gu et al., 2016).

Encoder: The encoder is implemented as a single-layer bidirectional RNN with gated recurrent units (GRUs). It reads words one-by-one from the input problem, producing a sequence of hidden states $h_i = [h_i^F, h_i^B]$ with:

$$h_i^F = GRU(\phi^{in}(x_i), h_{i-1}^F), \quad (1)$$

$$h_i^B = GRU(\phi^{in}(x_i), h_{i+1}^B), \quad (2)$$

where ϕ^{in} maps each input word x_i to a fixed-dimensional vector.

Decoder with Copying: At each decoding step j, the decoder receives the word embedding of the previous word, and an attention function is applied to attend over the input words as follows:

$$e_{ji} = v^T \tanh(W_h h_i + W_s s_j + b_{attn}), \quad (3)$$

$$a_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})}, \quad (4)$$

$$c_j = \sum_{i=1}^{m} a_{ji} h_i, \quad (5)$$

where s_j is the decoder hidden state. Intuitively, a_{ji} defines the probability distribution of attention over the input words. They are computed from the unnormalized attention scores e_{ji}. c_j is the context vector, which is the weighted sum of the encoder hidden states.

At each step, the model has to decide whether to *generate a word* from target vocabulary or to *copy a number* from the problem description. The generation probability p_{gen} is modeled by:

$$p_{gen} = \sigma(w_c^T c_j + w_s^T s_j + b_{ptr}), \quad (6)$$

where w_c, w_s and b_{ptr} are model parameters. Next, p_{gen} is used as a soft switch: with probability p_{gen} the model decides to generate from the decoder state. The probability distribution over all words in the vocabulary is:

$$P_{RNN} = \text{softmax}(W[s_j, c_j] + b); \quad (7)$$

with probability $1 - p_{gen}$ the model decides to directly copy an input word according to its attention weight. This leads to the final distribution of decoder state outputs:

$$P(w_j = w|\cdot) = p_{gen}P_{RNN}(w) + (1 - p_{gen})a_{ji} \quad (8)$$

4.2 Attention Regularization

In preliminary experiments, we observed that the attention weights in the baseline model repetitively concentrate on the numbers in the math problem description (will be discussed in later sections with Figure 1(a)). To address this issue, we regularize the accumulative attention weights for each input token using a rectified linear unit (ReLU) layer, leading to the regularization term:

$$\text{AttReg} = \sum_i \text{ReLU}(\sum_{j=0}^{T} a_{ji} - 1), \quad (9)$$

where $\text{ReLU}(x) = \max(x, 0)$. This term penalizes the accumulated attention weights on specific locations if it exceeds 1. Adding this term to the primary loss to get the final objective function:

$$Loss = -\sum_i \log p(\mathbf{y}^i|\mathbf{x}^i; \theta) + \lambda * \text{AttReg} \quad (10)$$

where λ is a hyper-parameter that controls the contribution of attention regularization in the loss.

The format of our attention regularization term resembles the coverage mechanism used in neural machine translation (Tu et al., 2016; Cohn et al., 2016), which encourages the coverage or fertility control for input tokens.

5 Iterative Labeling

Since explicit annotations of our intermediate forms do not exist, we propose an iterative labeling framework for training.

5.1 Deriving Latent Forms From Equations

We use the annotated equation systems to derive possible latent forms. First we define some simple rules that map an expression to our intermediate form. For example, we use regular expressions to match numbers and unknown variables. Example rules are shown in Table 4 (see Section 2 of the Supplement Material for all rules).

Regex/Rules	**Class/Function**
`\-?[0-9\.]+`	num
`[a-z]`	unk
`<num>|<unk>`	var
`(<var>\+)+<var>`	math#sum($1:list)
`(<unk>\+)+<unk>` $1=count of unk	math#sum (cnt:$1:int)

Table 4: Example rules for deriving latent forms from equation system.

5.2 Ambiguity in Derivation

For one equation system, several latent form derivations are possible. Take the following math problem as an example:

*Find **3** consecutive integers that **3** times the sum of the first and the third is 79.*

Given the annotation of its equation $3 * (x + (x + 2)) = 79$, there are two possible latent intermediate forms:
1) math#consecutive(3), math#product(3, math#sum(ordinal(1), ordinal(3)))=79
2) math#consecutive(3), math#product(3, math#sum(min(), max()))=79

There exist two types of ambiguities: a) **operator ambiguity**. $(x + 2)$ may correspond to the operator "*ordinal(3)*" or "*max()*"; b) **alignment ambiguity**. For each "3" in the intermediate form, it is unclear which "3" in the input to be copied. Therefore, we may derive multiple intermediate forms with spurious ones for a training problem.

We can see from Table 5 that both datasets we used have the issue of ambiguity, containing about 20% of problems with operator ambiguity and 10% of problems with alignment ambiguity.

5.3 Iterative Labeling

To address the issue of ambiguity, we perform an iterative procedure where we search for correct intermediate forms to refine the training data. The

Dataset	Ambiguous		Ambig. #LF
	oper	align	(per prob)
NumWord (Linear)	28.0%	10.2%	3.67
NumWord (All)	26.9%	9.5%	4.29
Dolphin18K	35.9%	9.6%	3.86

Table 5: Statistics of latent forms on two datasets. The percentage of problems with operator and alignment ambiguity is shown in the 2*nd* and 3*rd* columns respectively. We also show the average number of intermediate forms of problems with derivation ambiguity in the rightmost column.

intuition is that a better model will lead to more correct latent form outputs, and more correct latent forms in training data will lead to a better model.

Algorithm 1 Iterative Labeling

```
Require:
  (1) Tuples of (math problem description, equation system, answer) $D_n = \{(p_i, E_{p_i}, A_{p_i})\}$
  (2) Possible latent forms $P_{LF} = \{(p_0, LF^1_{p_0}), (p_0, LF^2_{p_0}), ..., (p_n, LF^m_{p_n})\}$
  (3) Beam size $B$
  (4) training iterations $N_{iter}$, pre-training iterations $N_{pre}$
  Procedure:
  for $iter = 1$ to $N_{iter}$ do
      if $iter < N_{pre}$ then
          $\theta \leftarrow$ MLE with $P_{LF}$
      else
          for $(p, LF)$ in $P_{LF}$ do
              C = Decode $B$ latent forms given $p$
              for j in 1...B do
                  if Ans($C_j$) is correct then
                      $LF \Leftarrow C_j$
                      break
          $\theta \leftarrow$ MLE with relabeled $P_{LF}$
```

Algorithm 1 describes our training procedure. As pre-training, we first update our model by maximum likelihood estimation (MLE) with all possible latent forms for N_{pre} iterations. Ambiguous and wrong latent forms may appear at this stage. This pre-training is to ensure faster convergence and a more stable model. After N_{pre} iterations, iterative labeling starts. We decode on each training instance with beam search. We declare C_j to be the *consistent form* in the beam if it can be executed to yield the correct answer. Therefore we can relabel the latent form LF with C_j for problem p and use the new pairs for training. If there is no consistent form in the beam, we keep it unchanged. With iterative labeling, we update our model by MLE with relabeled latent forms. There are two conditions of N_{pre} to consider:
(1) $N_{pre} = 0$, the training starts iterative labeling without pre-training.
(2) $N_{pre} = N_{iter}$, the training is pure MLE without iterative labeling.

6 Experiments

In this section, we compare our method against several strong baseline systems.

6.1 Experiment Setting

Following previous work, experiments are done in 5-fold cross validation: in each run, 20% is used for testing, 70% for training and 10% for validation.

Representation To make the task easier with less auxiliary nuisances (e.g. bracket pairs), we represent the intermediate forms in Polish notation. [3]

Implementation details The dimension of encoder hidden state, decoder hidden state and embeddings are 100 in NumWord, 512 in Dolphin18K. All model parameters are initialized randomly with Gaussian distribution. The hyper-parameter λ for the weight of attention regularization is set to 1.0 on NumWord and 0.4 on Dolphin18K. We use SGD optimizer with decaying learning rate initialized as 0.5. Dropout rate is set to 0.5. The stopping criterion for training is validation accuracy with the maximum number of iterations no more than 150. The vocabulary consists of words observed no less than N times in training set. We set $N = 1$ for NumWord and $N = 5$ for Dolphin18K. The beam size is set to 20 in the decoding stage. For iterative training, we first train a model for $N_{pre} = 50$ iterations for pre-training. We tune the hyper-parameters on a separate dev set.

We consider the following models for comparisons:

- **Wang et al. (2017):** a seq2seq model with attention mechanism. As preprocessing, it replaces numbers in the math problem with tokens $\{n_1, n_2, ...\}$. It generates **equation**

[3]https://en.wikipedia.org/wiki/Polish_notation

as output and recovers $\{n_1, n_2, ...\}$ to corresponding numbers in the post-processing.

- **Seq2Seq_Equ:** we implement a seq2seq model with attention and copy mechanism. Different from Wang et al. (2017), it has the ability to copy numbers from problem description.
- **Shi et al. (2015):** a rule-based system. It parses math problems into Dolphin language trees with predefined grammars and reasons across trees to get the equations with rules. We report numbers from their paper as the Dolphin language is not publicly available.
- **Huang et al. (2017):** the current state-of-the-art model on Dolphin18K. It is a feature-based model. It generates candidate equations and find the most probable equation by ranking with predefined features.

6.2 Results

Overall results are shown in Table 6. From the table, we can see that our final model (**Seq2Seq_LF+AttReg+Iter**) outperforms the neural-based baseline models (Wang et al. (2017)[4] and **Seq2Seq_Equ**). On Number word problem dataset, our model already outperforms the state-of-the-art feature-based model (Huang et al., 2017) by 40.8% and is comparable to the ruled-based model (Shi et al., 2015)[5].

Advantage of intermediate forms: From the first two rows, we can see that the seq2seq model which is trained to generate intermediate forms (**Seq2Seq_LF**) greatly outperforms the same model trained to generate equations (**Seq2Seq_Equ**). The use of intermediate forms helps more on NumWord than on Dolphin18K. This result is expected as the Dolphin18K dataset is more challenging, containing many other types of difficulties discussed in Section 6.3.

Effect of Attention Regularization: Attention regularization improves the seq2seq model on the two datasets as expected. Figure 1 shows an example. The attention regularization does meet the expectation: the alignments in Fig 1(b) are less concentrated on the numbers in the input and more importantly and alignments are more reasonable. For example, when generating "*math#product*" in the output, the attention is now correctly focused on the input token "*times*".

[4]We re-implement this since it is not publicly available.

[5]The system reports precision and recall. Since all the problems have answers, its recall equals to our accuracy.

Effect of Iterative Labeling: We can see from Table 6 that iterative labeling clearly contributes to the accuracy increase on the two datasets. Now we compare the performance with and without pre-training in Table 7. When $N_{pre} = 0$ in Algorithm 1, the model starts iterative labeling from the first iteration without pre-training. We find that training with pre-training is substantially better, as the model without pre-training can be unstable and may generate misleading spurious candidate forms.

Next, we compare the performance with pure MLE training on NumWord (Linear) in Figure 2. The difference is that after 50 iterations of MLE training, iterative labeling would refine the latent forms of training data. In pure MLE training, the accuracy converges after 130 iterations. By using iterative labeling, the model achieves the accuracy of 61.6% at $110th$ iterations, which is faster to converge and leads to better performance.

Furthermore, to check whether iterative labeling actually resolves ambiguities in the intermediate forms of the training data, we manually sample 100 math problems with derivation ambiguity. **78%** of them are relabeled with correct latent forms as we have checked. From Table 8, we can see the latent form of one training problem is iteratively refined to the correct one.

6.3 Model Comparisons

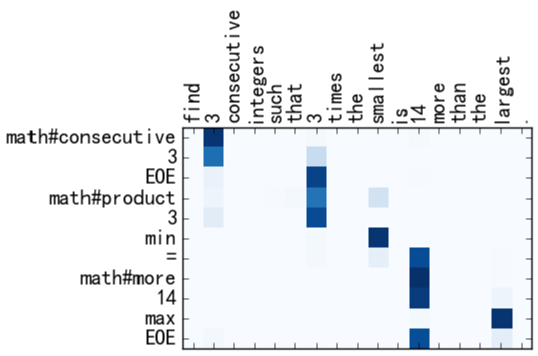
To explore the generalization ability of the neural approach and better guide our future work, we compare the problems solved by our neural-based model with the rule-based model (Shi et al., 2015) and the feature-based model (Huang et al., 2017).

Neural-based v. Rule-based: On NumWord (ALL), **41.6%** of problems can be solved by both models. **15.5%** can only be solved by our neural model, while the rule-based model generates an empty or a wrong semantic tree due to the limitations of the predefined grammar. The neural model is more consistent with flexible word order and insertion of lexical items (e.g. rule-based model cannot handle the extra word 'whole' in "Find two consecutive **whole** numbers").

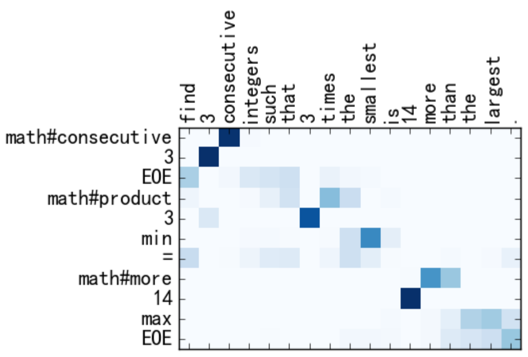
Neural-based v. Feature-based: On Dolphin18K, **9.2%** of problems can be solved by both models. **7.6%** can only be solved by our neural model, which indicates that the neural model

Models	NumWord (Linear)	NumWord (ALL)	Dolphin18K (Linear)
Wang et al. (2017)	19.7%	14.6%	10.2%
Seq2Seq_Equ	26.8%	20.1%	13.1%
Seq2Seq_LF	50.8%	45.2%	13.9%
Seq2Seq_LF+AttReg	56.7%	54.0%	15.1%
Seq2Seq_LF+AttReg+Iter	61.6%	57.1%	16.8%
Shi et al. (2015)	63.6%	60.2%	n/a
Huang et al. (2017)	20.8%	n/a	28.4%

Table 6: Performances on two datasets. "**LF**" means that the model generates latent intermediate forms instead of equation systems. "**AttReg**" means attention regularization. "**Iter**" means iterative labeling. "**n/a**" means that the model does not run on the dataset.



(a) **seq2seq_LF**



(b) **seq2seq_LF+AttReg**

Figure 1: Example alignments for one problem (darker color represents higher attention score).

	NumWord (Linear)	NumWord (ALL)	Dolphin18K (Linear)
-pre	58.1%	54.9%	14.9%
+pre	61.6%	57.1%	16.8%

Table 7: Performance with and without pre-training in iterative labeling.

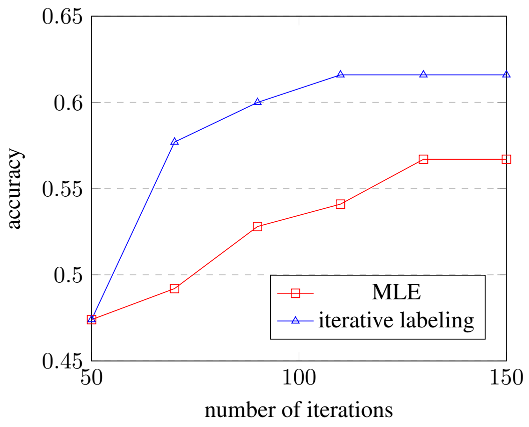


Figure 2: Accuracy with different iterations of training on NumWord (Linear).

can capture novel features that the feature-based model is missing.

While our neural model is complementary to the above mentioned models, we observe two main types of errors (more examples are shown in the supplementary material):

1. Natural language variations: Same type of problems can be described in different scenarios. The two problems: (1) "What is 10 minus 2?" and (2) "John has 10 apples. How many apples does John have after giving Mary 2 apples", lead to the same equation $x = 10 - 2$ but with very different descriptions. With limited size of data, we could not be expected to cover all possible ways to ask the same underlining math problems. Although the feature-based model has considered this with some features (e.g. POS Tag), the challenge is not well-addressed.

2. Nested operations: Some problems require multiple nested operations (e.g. "I think of a number, double it, add 3, multiply the answer by 3 and then add on the original number"). The rule-based model performs more consistently on this.

Training Problem:
Find **2_0** *consecutive integers which the first number is* **2_1** *more than* **2_2** *times the second number.*
Intermediate form in 1*st* iteration
(✗) math#consecutive(**2_0**), ordinal(1) = math#sum("**2_0**", math#product("**2_0**", "**max()**")
Intermediate form in 51*st* iteration
(✗) math#consecutive(**2_0**), ordinal(1) = math#sum(**2_1**, math#product("**2_0**", ordinal(2))
Intermediate form in 101*st* iteration
(✓) math#consecutive(**2_0**), ordinal(1) = math#sum(**2_1**, math#product(**2_2**, ordinal(2))

Table 8: Instance check of intermediate form for one math problem in several training iterations. **2_0** means the the first '2' in the input and so on. Tokens with quote marks mean that they are incorrect.

7 Related Work

Our work is related to two research areas: math word problem solving and semantic parsing.

7.1 Math Word Problem Solving

There are two major components in this task: (1) meaning representation; (2) learning framework.

Semantic Representation With the annotation of equation system, most approaches attempt at learning a direct mapping from math problem description to an equation system. There are other approaches considering an intermediate representation that bridges the semantic gap between natural language and equation system. Bakman (2007) defines a table of schema (e.g. Transfer-In-Place, Transfer-In-Ownership) with associated formulas in natural utterance. A math problem can be mapped into a list of schema instantiations, then converted to equations. Liguda and Pfeiffer (2012) use augmented semantic network to represent math problems, where nodes represent concepts of quantities and edges represent transition states. Shi et al. (2015) design a new meaning representation language called Dolphin Language (DOL) with over 200 math-related functions and more additional noun functions. With predefined rules, these approaches accept limited well-format input sentences. Inspired by these representations, our work describes a new formal language which is more compact and is effective in facilitating better machine learning performance.

Learning Framework In rule-based approaches (Bakman, 2007; Liguda and Pfeiffer, 2012; Shi et al., 2015), they map math problem description into structures with predefined grammars and rules.

Feature-based approaches contain two stages: (1) generate equation candidates; They either replace numbers of existing equations in the training data as new equations (Kushman et al., 2014; Zhou et al., 2015; Upadhyay et al., 2016), or enumerate possible combinations of math operators and numbers and variables (Koncel-Kedziorski et al., 2015), which leads to intractably huge search space. (2) predict equation with features. For example, Hosseini et al. (2014) design features to classify verbs to addition or subtraction. Roy and Roth (2015); Roy et al. (2016) leverage the tree structure of equations. Mitra and Baral (2016); Roy and Roth (2018) design features for a few math concepts (e.g. Part-Whole, Comparison). Roy and Roth (2017) focus on the dependencies between number units. These approaches requires manual feature design and the features may be difficult to be generalized to other tasks.

Recently, there are a few works trying to build an end-to-end system with neural models. Ling et al. (2017) consider multiple-choice math problems and use a seq2seq model to generate rationale and the final choice (i.e. A, B, C, D). Wang et al. (2017) apply a seq2seq model to generate equations with the constraint of single unknown variable. Similarly, we use the seq2seq model but with novel attention regularization to address incorrect attention weights in the seq2seq model.

7.2 Semantic Parsing

Our work is also related to the classic settings of learning executable semantic parsers from indirect supervision (Clarke et al., 2010; Liang et al., 2011; Artzi and Zettlemoyer, 2011, 2013; Berant et al., 2013; Pasupat and Liang, 2016). Maximum marginal likelihood with beam search (Kwiatkowski et al., 2013; Pasupat and Liang, 2016; Ling et al., 2017) is traditionally used. It maximizes the marginal likelihood of all consistent logical forms being observed. Recently reinforcement learning (Guu et al., 2017; Liang et al., 2017) has also been considered, which maximizes the expected reward over all possible logical forms. Different from them, we only consider one single consistent latent form per training instance by leveraging training signals from both the answer and the equation system, which should be more efficient for our task.

8 Conclusion

This paper presents an intermediate meaning representation scheme for math problem solving that bridges the semantic gap between natural language and equation systems. To generate intermediate forms, we propose a seq2seq model with novel attention regularization. Without explicit annotations of latent forms, we design an iterative labeling framework for training. Experimental result shows that using intermediate forms is more effective than directly using equations. Furthermore, our iterative labeling effectively resolves ambiguities and leads to better performances.

As shown in the error analysis, same types of problems can have different natural language expressions. In the future, we will focus on tackling this challenge. In addition, we plan to expand the coverage of our meaning representation to support more mathematic concepts.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61472453, U1401256, U1501252, U1611264,U1711261,U1711262). Thanks to the anonymous reviewers for their helpful comments and suggestions.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *Transactions of the Association for Computational Linguistics*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conferene on Learning Representation*.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. Http://arxiv.org/abs/math/0701393.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the worlds response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang.

2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luku Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Chen Liang, Jonathan Berant, Quoc Le, Kennet D.Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *Natural Language Processing and Information Systems. International Conference on Applications of Natural Language to Information Systems (NLDB-2012)*, pages 247–252.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the 2017 Conference on Association for the Advancement of Artificial Intelligence*.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. In *Transactions of the Association for Computational Linguistic*.

Subhro Roy and Subhro Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. The Association for Computational Linguistics.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing: Mapping sentences to grounded equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Shuming Shi, Wang Yuehui, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural model for math word problem problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Discourse Representation Structure Parsing

Jiangming Liu   **Shay B. Cohen**   **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Jiangming.Liu@ed.ac.uk, scohen@inf.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

We introduce an open-domain neural semantic parser which generates formal meaning representations in the style of Discourse Representation Theory (DRT; Kamp and Reyle 1993). We propose a method which transforms Discourse Representation Structures (DRSs) to trees and develop a structure-aware model which decomposes the decoding process into three stages: basic DRS structure prediction, condition prediction (i.e., predicates and relations), and referent prediction (i.e., variables). Experimental results on the Groningen Meaning Bank (GMB) show that our model outperforms competitive baselines by a wide margin.

1 Introduction

Semantic parsing is the task of mapping natural language to machine interpretable meaning representations. A variety of meaning representations have been adopted over the years ranging from functional query language (FunQL; Kate et al. 2005) to dependency-based compositional semantics (λ-DCS; Liang et al. 2011), lambda calculus (Zettlemoyer and Collins, 2005), abstract meaning representations (Banarescu et al., 2013), and minimal recursion semantics (Copestake et al., 2005).

Existing semantic parsers are for the most part data-driven using annotated examples consisting of utterances and their meaning representations (Zelle and Mooney, 1996; Wong and Mooney, 2006; Zettlemoyer and Collins, 2005). The successful application of encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) to a variety of NLP tasks has provided strong impetus to treat semantic parsing as a sequence transduction problem where an utterance is mapped to a target meaning representation in string format (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiskỳ et al., 2016). The fact that meaning representations do not naturally conform to a linear ordering has also prompted efforts to develop recurrent neural network architectures tailored to tree or graph-structured decoding (Dong and Lapata, 2016; Cheng et al., 2017; Yin and Neubig, 2017; Alvarez-Melis and Jaakkola, 2017; Rabinovich et al., 2017; Buys and Blunsom, 2017)

Most previous work focuses on building semantic parsers for question answering tasks, such as querying a database to retrieve an answer (Zelle and Mooney, 1996; Cheng et al., 2017), or conversing with a flight booking system (Dahl et al., 1994). As a result, parsers trained on query-based datasets work on restricted domains (e.g., restaurants, meetings; Wang et al. 2015), with limited vocabularies, exhibiting limited compositionality, and a small range of syntactic and semantic constructions. In this work, we focus on open-domain semantic parsing and develop a general-purpose system which generates formal meaning representations in the style of Discourse Representation Theory (DRT; Kamp and Reyle 1993).

DRT is a popular theory of meaning representation designed to account for a variety of linguistic phenomena, including the interpretation of pronouns and temporal expressions within and across sentences. Advantageously, it supports meaning representations for entire texts rather than isolated sentences which in turn can be translated into first-order logic. The Groningen Meaning Bank (GMB; Bos et al. 2017) provides a large collection of English texts annotated with Discourse Representation Structures (see Figure 1 for an example). GMB integrates various levels of semantic annotation (e.g., anaphora, named entities, thematic roles, rhetorical relations) into a unified formalism providing expressive meaning representations for open-domain texts.

We treat DRT parsing as a structure prediction problem. We develop a method to transform DRSs to tree-based representations which can be further linearized to bracketed string format. We examine a series of encoder-decoder models (Bahdanau et al., 2015) differing in the way tree-

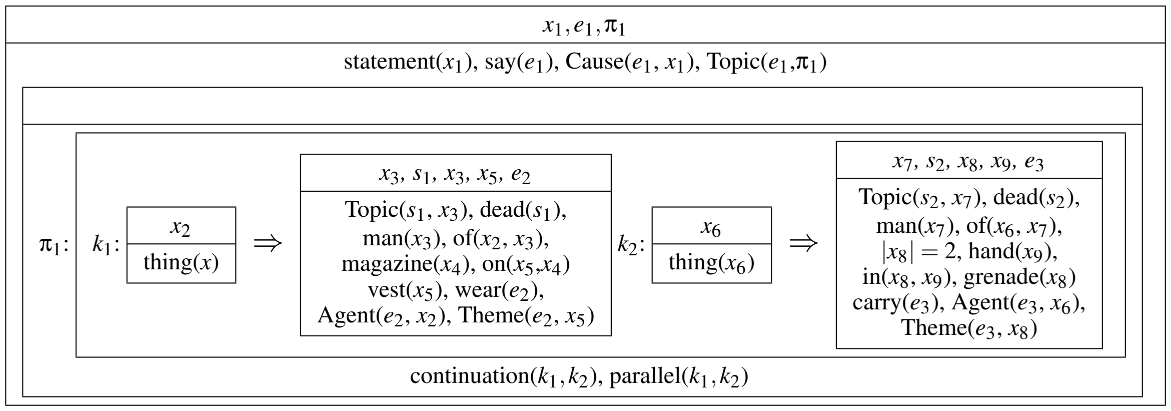

x_1, e_1, π_1
statement(x_1), say(e_1), Cause(e_1, x_1), Topic(e_1,π_1)
π_1:
k_1:
x_2
thing(x)
$\Rightarrow$
x_3, s_1, x_3, x_5, e_2
Topic(s_1, x_3), dead(s_1), man(x_3), of(x_2, x_3), magazine(x_4), on(x_5,x_4) vest(x_5), wear(e_2), Agent(e_2, x_2), Theme(e_2, x_5)
k_2:
x_6
thing(x_6)
$\Rightarrow$
x_7, s_2, x_8, x_9, e_3
Topic(s_2, x_7), dead(s_2), man(x_7), of(x_6, x_7), $|x_8| = 2$, hand(x_9), in(x_8, x_9), grenade(x_8) carry(e_3), Agent(e_3, x_6), Theme(e_3, x_8)
continuation(k_1,k_2), parallel(k_1,k_2)

Figure 1: DRT meaning representation for the sentence *The statement says each of the dead men wore magazine vests and carried two hand grenades.*

structured logical forms are generated and show that a structure-aware decoder is paramount to open-domain semantic parsing. Our proposed model decomposes the decoding process into three stages. The first stage predicts the structure of the meaning representation omitting details such as predicates or variable names. The second stage fills in missing predicates and relations (e.g., thing, Agent) conditioning on the natural language input and the previously predicted structure. Finally, the third stage predicts variable names based on the input and the information generated so far.

Decomposing decoding into these three steps reduces the complexity of generating logical forms since the model does not have to predict deeply nested structures, their variables, and predicates all at once. Moreover, the model is able to take advantage of the GMB annotations more efficiently, e.g., examples with similar structures can be effectively used in the first stage despite being very different in their lexical make-up. Finally, a piecemeal mode of generation yields more accurate predictions; since the output of every decoding step serves as input to the next one, the model is able to refine its predictions taking progressively more global context into account. Experimental results on the GMB show that our three-stage decoder outperforms a vanilla encoder-decoder model and a related variant which takes shallow structure into account, by a wide margin.

Our contributions in this work are three-fold: an open-domain semantic parser which yields discourse representation structures; a novel end-to-end neural model equipped with a structured decoder which decomposes the parsing process into three stages; a DRS-to-tree conversion method which transforms DRSs to tree-based representations allowing for the application of structured decoders as well as sequential modeling. We release our code[1] and tree formatted version of the GMB in the hope of driving further research in open-domain semantic parsing.

2 Discourse Representation Theory

In this section we provide a brief overview of the representational semantic formalism used in the GMB. We refer the reader to Bos et al. (2017) and Kamp and Reyle (1993) for more details.

Discourse Representation Theory (DRT; Kamp and Reyle 1993) is a general framework for representing the meaning of sentences and discourse which can handle multiple linguistic phenomena including anaphora, presuppositions, and temporal expressions. The basic meaning-carrying units in DRT are Discourse Representation Structures (DRSs), which are recursive formal meaning structures that have a model-theoretic interpretation and can be translated into first-order logic (Kamp and Reyle, 1993). Basic DRSs consist of discourse referents (e.g., x, y) representing entities in the discourse and discourse conditions (e.g., man(x), magazine(y)) representing information about discourse referents. Following conventions in the DRT literature, we visualize DRSs in a box-like format (see Figure 1).

GMB adopts a variant of DRT that uses a neo-Davidsonian analysis of events (Kipper et al., 2008), i.e., events are first-order entities characterized by one-place predicate symbols (e.g., say(e_1) in Figure 1). In addition, it follows Projective Discourse Representation Theory (PDRT; Venhuizen et al. 2013) an extension of DRT specifically developed to account for the interpretation of presuppositions and related projection phenomena

[1] https://github.com/EdinburghNLP/EncDecDRSparsing

(e.g., conventional implicatures). In PDRT, each basic DRS introduces a label, which can be bound by a pointer indicating the interpretation site of semantic content. To account for the rhetorical structure of texts, GMB adopts Segmented Discourse Representation Theory (SDRT; Asher and Lascarides 2003). In SDRT, discourse segments are linked with rhetorical relations reflecting different characteristics of textual coherence, such as temporal order and communicative intentions (see continuation(k_1, k_2) in Figure 1).

More formally, DRSs are expressions of type $\langle exp_e \rangle$ (denoting individuals or discourse referents) and $\langle exp_t \rangle$ (i.e., truth values):

$$\langle exp_e \rangle ::= \langle ref \rangle, \quad \langle exp_t \rangle ::= \langle drs \rangle | \langle sdrs \rangle, \tag{1}$$

discourse referents $\langle ref \rangle$ are in turn classified into six categories, namely common referents (x_n), event referents (e_n), state referents (s_n), segment referents (k_n), proposition referents (π_n), and time referents (t_n). $\langle drs \rangle$ and $\langle sdrs \rangle$ denote basic and segmented DRSs, respectively:

$$\langle drs \rangle ::= \langle pvar \rangle : \boxed{\begin{array}{c} (\langle pvar \rangle, \langle ref \rangle))^* \\ \hline (\langle pvar \rangle, \langle condition \rangle))^* \end{array}}, \tag{2}$$

$$\langle sdrs \rangle ::= \boxed{\begin{array}{c} k_1 : \langle exp_t \rangle, k_2 : \langle exp_t \rangle \\ \hline coo(k_1, k_2) \end{array}} \; | \; \boxed{\begin{array}{c} k_1 : \langle exp_t \rangle \\ k_2 : \langle exp_t \rangle \\ \hline sub(k_1, k_2) \end{array}}, \tag{3}$$

Basic DRSs consist of a set of referents ($\langle ref \rangle$) and conditions ($\langle condition \rangle$), whereas segmented DRSs are *recursive* structures that combine two $\langle exp_t \rangle$ by means of coordinating (*coo*) or subordinating (*sub*) relations. DRS conditions can be basic or complex:

$$\langle condition \rangle ::= \langle basic \rangle | \langle complex \rangle, \tag{4}$$

Basic conditions express properties of discourse referents or relations between them:

$$\begin{aligned} \langle basic \rangle ::= \; & \langle sym_1 \rangle(\langle exp_e \rangle) \mid \langle sym_2 \rangle(\langle exp_e \rangle, \langle exp_e \rangle) \\ & \mid \langle exp_e \rangle = \langle exp_e \rangle \mid \langle exp_e \rangle = \langle num \rangle \\ & \mid timex(\langle exp_e \rangle, \langle sym_0 \rangle) \\ & \mid named(\langle exp_e \rangle, \langle sym_0 \rangle, class). \end{aligned} \tag{5}$$

where $\langle sym_n \rangle$ denotes n-place predicates, $\langle num \rangle$ denotes cardinal numbers, *timex* expresses temporal information (e.g., timex(x_7, 2005) denotes the year 2005), and *class* refers to named entity classes (e.g., location).

Complex conditions are unary or binary. Unary conditions have one DRS as argument and represent negation ($\neg$) and modal operators expressing necessity ($\Box$) and possibility ($\Diamond$). Condition

sections	# doc	# sent	# token	avg
00-99	10,000	62,010	1,354,149	21.84
20-99	7,970	49,411	1,078,953	21.83
10-19	1,038	6,483	142,344	21.95
00-09	992	6,116	132,852	21.72

Table 1: Statistics on the GMB (avg denotes the average number of tokens per sentence).

$\langle ref \rangle : \langle exp_t \rangle$ represents verbs with propositional content (e.g., factive verbs). Binary conditions are conditional statements ($\rightarrow$) and questions.

$$\begin{aligned} \langle complex \rangle &::= \langle unary \rangle \mid \langle binary \rangle, \\ \langle unary \rangle &::= \neg\langle exp_t \rangle \mid \Box\langle exp_t \rangle | \Diamond\langle exp_t \rangle | \langle ref \rangle : \langle exp_t \rangle \\ \langle binary \rangle &::= \langle exp_t \rangle \rightarrow \langle exp_t \rangle | \langle exp_t \rangle \vee \langle exp_t \rangle | \langle exp_t \rangle ? \langle exp_t \rangle \end{aligned} \tag{6}$$

3 The Groningen Meaning Bank Corpus

Corpus Creation DRSs in GMB were obtained from Boxer (Bos, 2008, 2015), and then refined using expert linguists and crowdsourcing methods. Boxer constructs DRSs based on a pipeline of tools involving POS-tagging, named entity recognition, and parsing. Specifically, it relies on the syntactic analysis of the C&C parser (Clark and Curran, 2007), a general-purpose parser using the framework of Combinatory Categorial Grammar (CCG; Steedman 2001). DRSs are obtained from CCG parses, with semantic composition being guided by the CCG syntactic derivation.

Documents in the GMB were collected from a variety of sources including *Voice of America* (a newspaper published by the US Federal Government), the Open American National Corpus, Aesop's fables, humorous stories and jokes, and country descriptions from the *CIA World Factbook*. The dataset consists of 10,000 documents each annotated with a DRS. Various statistics on the GMB are shown in Table 1. Bos et al. (2017) recommend sections 20–99 for training, 10–19 for tuning, and 00–09 for testing.

DRS-to-Tree Conversion As mentioned earlier, DRSs in the GMB are displayed in a box-like format which is intuitive and easy to read but not particularly amenable to structure modeling. In this section we discuss how DRSs were post-processed and simplified into a tree-based format, which served as input to our models.

The GMB provides DRS annotations per-document. Our initial efforts have focused on sentence-level DRS parsing which is undoubtedly a necessary first step for more global semantic representations. It is relatively, straightforward to obtain sentence-level DRSs from document-level annotations since referents and conditions are indexed to tokens. We match each sentence in a document with the DRS whose content bears the same indices as the tokens occurring in the sentence. This matching process yields 52,268 sentences for training (sections 20–99), 5,172 sentences for development (sections 10–19), (development), and 5,440 sentences for testing (sections 00–09).

In order to simplify the representation, we omit referents in the top part of the DRS (e.g., x_1, e_1 and π_1 in Figure 1) but preserve them in conditions without any information loss. Also we ignore pointers to DRSs since this information is implicitly captured through the typing and co-indexing of referents. Definition (1) is simplified to:

$$\langle drs \rangle ::= \text{DRS}(\langle condition \rangle^*), \tag{7}$$

where DRS() denotes a basic DRS. We also modify discourse referents to SDRSs (e.g., k_1, k_2 in Figure 1) which we regard as elements bearing scope over expressions $\langle exp_t \rangle$ and add a 2-place predicate $\langle sym_2 \rangle$ to describe the discourse relation between them. So, definition (3) becomes:

$$\begin{aligned} \langle sdrs \rangle ::= &\text{SDRS}((\langle ref \rangle(\langle exp_t \rangle))^* \\ &(\langle sym_2 \rangle(\langle ref \rangle, \langle ref \rangle))^*), \end{aligned} \tag{8}$$

where SDRS() denotes a segmented DRS, and $\langle ref \rangle$ are segment referents.

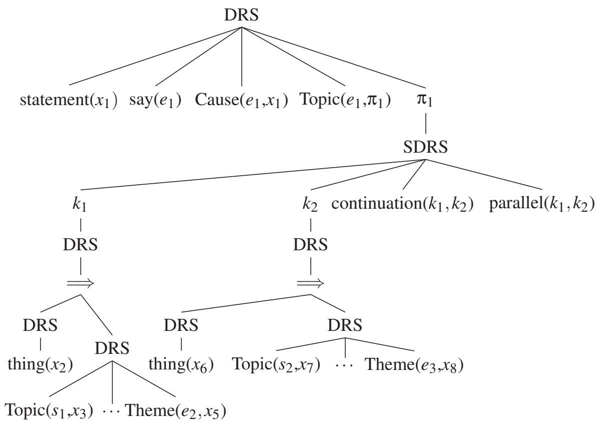
We treat cardinal numbers $\langle num \rangle$ and $\langle sym_0 \rangle$ in relation *timex* as constants. We introduce the binary predicate "card" to represent cardinality (e.g., $|x_8| = 2$ is card(x_8,NUM)). We also simplify $\langle exp_e \rangle = \langle exp_e \rangle$ to eq($\langle exp_e \rangle, \langle exp_e \rangle$) using the binary relation "eq" (e.g., $x_1 = x_2$ becomes eq(x_1,x_2)). Moreover, we ignore *class* in *named* and transform *named*($\langle exp_e \rangle, \langle sym_0 \rangle$, *class*) into $\langle sym_1 \rangle(\langle exp_e \rangle)$ (e.g., named(x_2, mongolia, *geo*) becomes mongolia(x_2)). Consequently, basic conditions (see definition (5)) are simplified to:

$$\langle basic \rangle ::= \langle sym_1 \rangle(\langle exp_e \rangle) | \langle sym_2 \rangle(\langle exp_e \rangle, \langle exp_e \rangle) \tag{9}$$

Analogously, we treat *unary* and *binary* conditions as scoped functions, and definition (6) becomes:

$$\begin{aligned} \langle unary \rangle ::= &\ \neg \mid \Box \mid \Diamond \mid \langle ref \rangle(\langle exp_t \rangle) \\ \langle binary \rangle ::= &\ \rightarrow \mid \vee \mid ?(\langle exp_t \rangle, \langle exp_t \rangle), \end{aligned} \tag{10}$$

Following the transformations described above, the DRS in Figure 1 is converted into the tree in



DRS(statement(x_1) say(e_1) Cause(e_1,x_1) Topic(e_1,π_1) π_1(SDRS(k_1 (DRS ($\Longrightarrow$(DRS(thing(x_2)) DRS (Topic(s_1,x_3) dead(s_1) man(x_3) of(x_2,x_3) magazine(x_4) on(x_5,x_4) vest(x_5) wear(e_2) Agent(e_2,x_2) Theme(e_2,x_5))))) k_2(DRS $\Longrightarrow$(DRS(thing(x_6)) DRS(Topic(s_2,x_7) dead(s_2) man(x_7) of(x_6,x_7) card(x_8,NUM) hand(x_9) in(x_8,x_9) carry(e_3) Agent(e_3,x6) Theme(e3,x8))))) continuation(k_1,k_2) parallel(k_1,k_2)

Figure 2: Tree-based representation (top) of the DRS in Figure 1 and its linearization (bottom).

Figure 2, which can be subsequently linearized into a PTB-style bracketed sequence. It is important to note that the conversion does not diminish the complexity of DRSs. The average tree width in the training set is 10.39 and tree depth is 4.64.

4 Semantic Parsing Models

We present below three encoder-decoder models which are increasingly aware of the structure of the DRT meaning representations. The models take as input a natural language sentence X represented as $w_1, w_2, \ldots, w_n$, and generate a sequence $Y = (y_1, y_2, ..., y_m)$, which is a linearized tree (see Figure 2 bottom), where n is the length of the sentence, and m the length of the generated DRS sequence. We aim to estimate $p(Y|X)$, the conditional probability of the semantic parse tree Y given natural language input X:

$$p(Y|X) = \prod_j p(y_j | Y_1^{j-1}, X_1^n)$$

4.1 Encoder

An encoder is used to represent the natural language input X into vector representations. Each token in a sentence is represented by a vector x_k which is the concatenation of randomly initialized embeddings e_{w_i}, pre-trained word embeddings $\bar{e}_{w_i}$, and lemma embeddings e_{l_i}: $x_k = \tanh([e_{w_i}; \bar{e}_{w_i}; e_{l_i}] * W_1 + b_1)$, where $W_1 \in \mathbb{R}^{\mathcal{D}}$ and $\mathcal{D}$ is a shorthand for $(d_w + d_p + d_l) \times d_{input}$ (subscripts w, p, and l denote the dimensions of word embeddings, pre-trained embeddings, and lemma embeddings, respectively); $b_1 \in \mathbb{R}^{d_{input}}$ and the symbol ; denotes concatenation. Embeddings e_{w_i}

and e_{l_i} are randomly initialized and tuned during training, while $\bar{e}_{w_i}$ are fixed.

We use a bidirectional recurrent neural network with long short-term memory units (bi-LSTM; Hochreiter and Schmidhuber 1997) to encode natural language sentences:

$$[h_{e_1} : h_{e_n}] = \text{bi-LSTM}(x_1 : x_n),$$

where h_{e_i} denotes the hidden representation of the encoder, and x_i refers to the input representation of the ith token in the sentence. Table 2 summarizes the notation used throughout this paper.

4.2 Sequence Decoder

We employ a sequential decoder (Bahdanau et al., 2015) as our baseline model with the architecture shown in Figure 3(a). Our decoder is a (forward) LSTM, which is conditionally initialized with the hidden state of the encoder, i.e., we set $h_{d_0} = h_{e_n}$ and $c_{d_0} = c_{e_n}$, where c is a memory cell:

$$h_{d_j} = \text{LSTM}(e_{y_{j-1}}),$$

where h_{d_j} denotes the hidden representation of y_j, e_{y_j} are randomly initialized embeddings tuned during training, and y_0 denotes the start of sequence.

The decoder uses the contextual representation of the encoder together with the embedding of the previously predicted token to output the next token from the vocabulary V:

$$s_j = [h_{ct_j}; e_{y_{j-1}}] * W_2 + b_2,$$

where $W_2 \in \mathbb{R}^{(d_{enc}+d_y)\times|V|}$, $b_2 \in \mathbb{R}^{|V|}$, d_{enc} and d_y are the dimensions of the encoder hidden unit and output representation, respectively, and h_{ct_j} is obtained using an attention mechanism:

$$h_{ct_j} = \sum_{i=1}^{n} \beta_{ji} h_{e_i},$$

where the weight β_{ji} is computed by:

$$\beta_{ji} = \frac{e^{f(h_{d_j}, h_{e_i})}}{\sum_k e^{f(h_{d_j}, h_{e_k})}},$$

and f is the dot-product function. We obtain the probability distribution over the output tokens as:

$$p_j = p(y_j | Y_1^{j-1}, X_1^n) = \textsc{SoftMax}(s_j)$$

Symbol	Description
X; Y	sequence of words; outputs
w_i; y_i	the ith word; output
X_i^j; Y_i^j	word; output sequence from position i to j
e_{w_i}; e_{y_i}	random embedding of word w_i; of output y_i
$\bar{e}_{w_i}$	fixed pretrained embedding of word w_i
e_{l_i}	random embedding for lemma l_i
d_w	dimension of random word embedding
d_p	dimension of pretrained word embedding
d_l	the dimension of random lemma embedding
d_{input}	input dimension of encoder
d_{enc}; d_{dec}	hidden dimension of encoder; decoder
W_i	matrix of model parameters
b_i	vector of model parameters
x_i	representation of ith token
h_{e_i}	hidden representation of ith token
c_{e_i}	memory cell of ith token in encoder
h_{d_i}	hidden representation of ith token in decoder
c_{d_i}	memory cell of ith token in decoder
s_j	score vector of jth output in decoder
h_{ct_j}	context representation of jth output
β_j^i	alignment from jth output to ith token
o_j^i	copy score of jth output from ith token
$\hat{}$	indicates tree structure (e.g. $\hat{Y}, \hat{y}_i, \hat{s}_j$)
$\bar{}$	indicates DRS conditions (e.g. $\bar{Y}, \bar{y}_i, \bar{s}_j$)
$\dot{}$	indicates referents (e.g. $\dot{Y}, \dot{y}_i, \dot{s}_j$)

Table 2: Notation used throughout this paper.

4.3 Shallow Structure Decoder

The baseline decoder treats all conditions in a DRS uniformly and has no means of distinguishing between conditions corresponding to tokens in a sentence (e.g., the predicate say(e_1) refers to the verb *said*) and semantic relations (e.g., Cause(e_1,x_1)). Our second decoder attempts to take this into account by distinguishing conditions which are local and correspond to words in a sentence from items which are more global and express semantic content (see Figure 3(b)). Specifically, we model sentence specific conditions using a copying mechanism, and all other conditions $\mathcal{G}$ which do not correspond to sentential tokens (e.g., thematic roles, rhetorical relations) with an insertion mechanism.

Each token in a sentence is assigned a copying score o_{ji}:

$$o_{ji} = h_{d_j}^\top W_3 h_{e_i},$$

where subscript ji denotes the ith token at jth time step, and $W_3 \in \mathbb{R}^{d_{dec}\times d_{enc}}$. All other conditions $\mathcal{G}$ are assigned an insertion score:

$$s_j = [h_{ct_j}; e_{y_{j-1}}] * W_4 + b_4,$$

where $W_4 \in \mathbb{R}^{(d_{enc}+d_y)\times|\mathcal{G}|}$, $b_4 \in \mathbb{R}^{|\mathcal{G}|}$, and h_{ct_j} are the same with the baseline decoder. We obtain the probability distribution over output tokens as:

$$p_j = p(y_j | Y_1^{j-1}, X_1^n) = \textsc{SoftMax}([o_j; s_j])$$

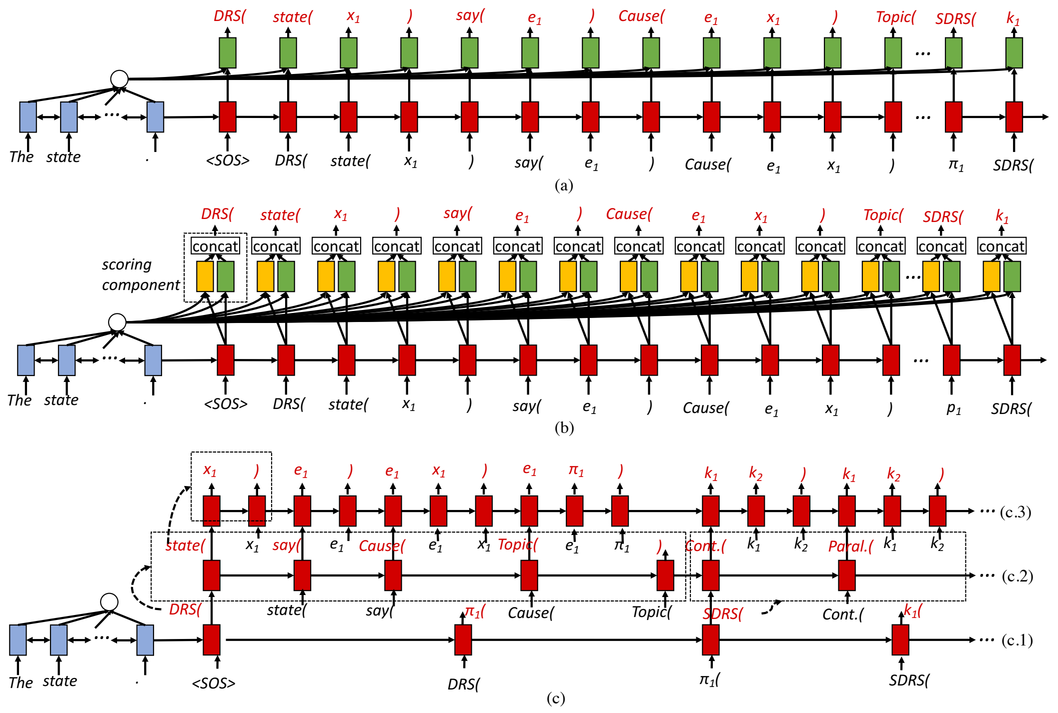


Figure 3: (a) baseline model; (b) shallow structure model; (c) deep structure model (scoring components are not displayed): (c.1) predicts DRS structure, (c.2) predicts conditions, and (c.3) predicts referents. Blue boxes are encoder hidden units, red boxes are decoder LSTM hidden units, green and yellow boxes represent copy and insertion scores, respectively.

4.4 Deep Structure Decoder

As explained previously, our structure prediction problem is rather challenging: the length of a bracketed DRS is nearly five times longer than its corresponding sentence. As shown in Figure 1, a bracketed DRS, $y_1, y_2, ..., y_n$ consists of three parts: internal structure $\hat{Y} = \hat{y}_1, \hat{y}_2, ...\hat{y}_t$ (e.g., DRS(π_1(SDRS(k_1(DRS($\rightarrow$(DRS( )DRS( ))) k_2(DRS($\rightarrow$(DRS( ) DRS ( ))))))))), conditions $\bar{Y} = \bar{y}_1, \bar{y}_2, ..., \bar{y}_r$ (e.g., statement, say, Topic), and referents $\dot{Y} = \dot{y}_1, \dot{y}_2, ..., \dot{y}_v$ (e.g., x_1, e_1, π_1), where $t + r * 2 + v = n$.[2]

Our third decoder (see Figure 3(c)) first predicts the structural make-up of the DRS, then the conditions, and finally their referents in an end-to-end framework. The probability distribution of structured output Y given natural language input X is rewritten as:

$$\begin{aligned} p(Y|X) = & \; p(\hat{Y}, \bar{Y}, \dot{Y}|X) \\ = & \; \prod_j p(\hat{y}_j|\hat{Y}_1^{j-1}, X) \\ & \times \prod_j p(\bar{y}_j|\bar{Y}_1^{j-1}, \hat{Y}_1^{j'}, X) \\ & \times \prod_j p(\dot{y}_j|\dot{Y}_1^{j-1}, \bar{Y}_1^{j'}, \hat{Y}_1^{j''}, X) \end{aligned} \quad (11)$$

where $\hat{Y}_1^{j-1}$, $\bar{Y}_1^{j-1}$, and $\dot{Y}_1^{j-1}$ denote the tree structure, conditions, and referents predicted so far. $\hat{Y}_1^{j'}$ denotes the structure predicted *before* conditions $\bar{y}_j$; $\hat{Y}_1^{j''}$ and $\bar{Y}_1^{j'}$ are the structures and conditions predicted *before* referents $\dot{y}_j$. We next discuss how each decoder is modeled.

Structure Prediction To model basic DRS structure we apply the shallow decoder discussed in Section 4.3 and also shown in Figure 3(c.1). Tokens in such structures correspond to parent nodes in a tree; in other words, they are all inserted from $\mathcal{G}$, and subsequently predicted tokens are only scored with the insert score, i.e., $\hat{s}_i = s_i$. The hidden units of the decoder are:

$$\hat{h}_{d_j} = \text{LSTM}(e_{\hat{y}_{j-1}}),$$

And the probabilistic distribution over structure denoting tokens is:

$$p(y_j|Y_1^{j-1}, X) = \text{SoftMax}(\hat{s}_j)$$

Condition Prediction DRS conditions are generated by taking previously predicted structures into account, e.g., when "DRS(" or "SDRS(" are predicted, their conditions will be generated next. By mapping j to (k, m_k), the sequence of conditions can be rewritten as $\bar{y}_1, \ldots, \bar{y}_j, \ldots, \bar{y}_r = \bar{y}_{(1,1)}, \bar{y}_{(1,2)}, \ldots, \bar{y}_{(k,m_k)}, \ldots,$ where $\bar{y}_{(k,m_k)}$ is m_kth

[2]Each condition has one and only one right bracket.

condition of structure token $\hat{y}_k$. The corresponding hidden units $\hat{h}_{d_k}$ act as conditional input to the decoder. Structure denoting tokens (e.g., "DRS(" or "SDRS(") are fed into the decoder one by one to generate the corresponding conditions as:

$$e_{\bar{y}_{(k,0)}} = \hat{h}_{d_k} * W_5 + b_5,$$

where $W_5 \in \mathbb{R}^{d_{dec} \times d_y}$ and $b_5 \in \mathbb{R}^{d_y}$. The hidden unit of the conditions decoder is computed as:

$$\bar{h}_{d_j} = \bar{h}_{d_{(k,m_k)}} = \text{LSTM}(e_{\bar{y}_{(k,m_k-1)}}),$$

Given hidden unit $\bar{h}_{d_j}$, we obtain the copy score $\bar{o}_j$ and insert score $\bar{s}_j$. The probabilistic distribution over conditions is:

$$p(\bar{y}_j | \bar{Y}_1^{j-1}, \hat{Y}_1^{j'}, X) = \text{SoftMax}([\bar{o}_j; \bar{s}_j])$$

Referent Prediction Referents are generated based on the structure and conditions of the DRS. Each condition has at least one referent. Similar to condition prediction, the sequence of referents can be rewritten as $\dot{y}_1, \ldots, \dot{y}_j, \ldots, \dot{y}_v = \dot{y}_{(1,1)}, \dot{y}_{(1,2)}, \ldots, \dot{y}_{(k,m_k)}, \ldots$ The hidden units of the conditions decoder are fed into the referent decoder $e_{\dot{y}_{(k,0)}} = \bar{h}_{d_k} * W_6 + b_6$, where $W_6 \in \mathbb{R}^{d_{dec} \times d_y}$, $b_6 \in \mathbb{R}^{d_y}$. The hidden unit of the referent decoder is computed as:

$$\dot{h}_{d_j} = \dot{h}_{d_{(k,m_k)}} = \text{LSTM}(e_{\dot{y}_{(k,m_k-1)}}),$$

All referents are inserted from $\mathcal{G}$, given hidden unit $\dot{h}_{d_j}$ (we only obtain the insert score $\dot{s}_j$). The probabilistic distribution over predicates is:

$$p(\dot{y}_j | \dot{Y}_1^{j-1}, \bar{Y}_1^{j'}, \hat{Y}_1^{j''}, X) = \text{SoftMax}(\dot{s}_j).$$

Note that a single LSTM is adopted for structure, condition and referent prediction. The mathematic symbols are summarized in Table 2.

4.5 Training

The models are trained to minimize a cross-entropy loss objective with ℓ_2 regularization:

$$L(\theta) = -\sum_j \log p_j + \frac{\lambda}{2} ||\theta||^2,$$

where θ is the set of parameters, and λ is a regularization hyper-parameter ($\lambda = 10^{-6}$). We used stochastic gradient descent with Adam (Kingma and Ba, 2014) to adjust the learning rate.

5 Experimental Setup

Settings Our experiments were carried out on the GMB following the tree conversion process discussed in Section 3. We adopted the training, development, and testing partitions recommended in Bos et al. (2017). We compared the three models introduced in Section 4, namely the baseline sequence decoder, the shallow structured decoder and the deep structure decoder. We used the same empirical hyper-parameters for all three models. The dimensions of word and lemma embeddings were 64 and 32, respectively. The dimensions of hidden vectors were 256 for the encoder and 128 for the decoder. The encoder used two hidden layers, whereas the decoder only one. The dropout rate was 0.1. Pre-trained word embeddings (100 dimensions) were generated with Word2Vec trained on the AFP portion of the English Gigaword corpus.[3]

Evaluation Due to the complex nature of our structured prediction task, we cannot expect model output to exactly match the gold standard. For instance, the numbering of the referents may be different, but nevertheless valid, or the order of the children of a tree node (e.g., "DRS(india(x_1) say(e_1))" and "DRS(say(e_1) india(x_1))" are the same). We thus use F_1 instead of exact match accuracy. Specifically, we report D-match[4] a metric designed to evaluate scoped meaning representations and released as part of the distribution of the Parallel Meaning Bank corpus (Abzianidze et al., 2017). D-match is based on Smatch[5], a metric used to evaluate AMR graphs (Cai and Knight, 2013); it calculates F_1 on discourse representation graphs (DRGs), i.e., triples of nodes, arcs, and their referents, applying multiple restarts to obtain a good referent (node) mapping between graphs.

We converted DRSs (predicted and goldstandard) into DRGs following the top-down procedure described in Algorithm 1.[6] IsCondition returns *true* if the child is a condition (e.g., india(x_1)), where three arcs are created, one is connected to a parent node and the other two are connected to arg1 and arg2, respectively (lines 7–12). IsQuantifier returns *true* if the child is a quantifier (e.g., π_1, $\neg$ and $\Box$) and three arcs are created; one is connected to the parent node, one to the referent that is created if and only

[3]The models are trained on a single GPU without batches.

[4]https://github.com/RikVN/D-match

[5]https://github.com/snowblink14/smatch

[6]We refer the interested reader to the supplementary material for more details.

Algorithm 1 DRS to DRG Conversion

```
Input: $T$, tree-like DRS
Output: $G$, a set of edges
 1: $n_b \leftarrow 0$; $n_c \leftarrow 0$; $G \leftarrow \emptyset$
 2: $stack \leftarrow []$; $R \leftarrow \emptyset$
 3: procedure TRAVELDRS(parent)
 4:     $stack.append(b_{n_b})$; $n_b \leftarrow n_b + 1$
 5:     $node_p \leftarrow stack.top$
 6:     for $child$ in $parent$ do
 7:         if ISCONDITION($child$) then
 8:             $G \leftarrow G \cup \{node_p \xrightarrow{child.rel} c_{n_c}\}$
 9:             $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg1}} child.arg1\}$
10:             $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg2}} child.arg2\}$
11:             $n_c \leftarrow n_c + 1$
12:             ADDREFERENT($node_p, child$)
13:         else if ISQUANTIFIER($child$) then
14:             $G \leftarrow G \cup \{node_p \xrightarrow{child.class} c_{n_c}\}$
15:             $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg1}} child.arg1\}$
16:             $G \leftarrow G \cup \{c_{n_c} \xrightarrow{\text{arg1}} b_{n_b+1}\}$
17:             $n_c \leftarrow n_c + 1$
18:             if ISPROPSEG($child$) then
19:                 ADDREFERENT($node_p, child$)
20:             end if
21:             TRAVELDRS($child.nextDRS$)
22:         end if
23:     end for
24:     $stack.pop()$
25: end procedure
26: procedure ADDREFERENT($node_p, child$)
27:     if $child.arg1$ not in $R$ then
28:         $G \leftarrow G \cup \{node_p \xrightarrow{\text{ref}} child.arg1\}$
29:         $R \leftarrow R \cup child.arg1$
30:     end if
31:     if $child.arg2$ not in $R$ then
32:         $G \leftarrow G \cup \{node_p \xrightarrow{\text{ref}} child.arg2\}$
33:         $R \leftarrow R \cup child.arg2$
34:     end if
35: end procedure
36: TRAVELDRS($T$)
37: return $G$
```

if the child is a proposition or segment (e.g., π_1 and k_1), and one is connected to the next DRS or SDRS nodes (lines 13–20). The algorithm will recursively travel all DRS or SDRS nodes (line 21). Furthermore, arcs are introduced to connect DRS or SDRS nodes to the referents that first appear in a condition (lines 26–35).

When comparing two DRGs, we calculate the F_1 over their arcs. For example consider the two DRGs (a) and (b) shown in Figure 4. Let $\{b_0 : b_0, x_1 : x_2, x_2 : x_3, c_0 : c_0, c_1 : c_2, c_2 : c_3\}$ denote the node alignment between them. The number of matching arcs is eight, the number of arcs in the gold DRG is nine, and the number of arcs in the predicted DRG is 12. So recall is 8/9, precision is 8/12, and F_1 is 76.19.

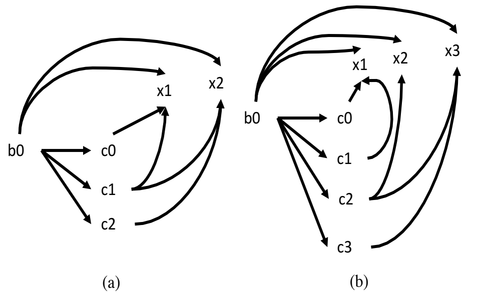


Figure 4: (a) is the gold DRS and (b) is the predicted DRS (condition names are not shown).

6 Results

Table 3 compares our three models on the development set. As can be seen, the shallow structured decoder performs better than the baseline decoder, and the proposed deep structure decoder outperforms both of them. Ablation experiments show that without pre-trained word embeddings or word lemma embeddings, the model generally performs worse. Compared to lemma embeddings, pretrained word embeddings contribute more.

Table 4 shows our results on the test set. To assess the degree to which the various decoders contribute to DRS parsing, we report results when predicting the full DRS structure (second block), when ignoring referents (third block), and when ignoring both referents and conditions (fourth block). Overall, we observe that the shallow structure model improves precision over the baseline with a slight loss in recall, while the deep structure model performs best by a large margin. When referents are not taken into account (compare the second and third blocks in Table 4), performance improves across the board. When conditions are additionally omitted, we observe further performance gains. This is hardly surprising, since errors propagate from one stage to the next when predicting full DRS structures. Further analysis revealed that the parser performs slightly better on (copy) conditions which correspond to natural language tokens compared to (insert) conditions (e.g., Topic, Agent) which are generated from global semantic content (83.22 vs 80.63 F_1). The parser is also better on sentences which do not represent SDRSs (79.12 vs 68.36 F_1) which is expected given that they usually correspond to more elaborate structures. We also found that rhetorical relations (linking segments) are predicted fairly accurately, especially if they are frequently attested (e.g., Continuation, Parallel), while the parser has difficulty with relations denoting contrast.

Model	P (%)	R (%)	F_1 (%)
baseline	51.35	63.85	56.92
shallow	67.88	63.53	65.63
deep	79.01	75.65	77.29
deep (–pre)	78.47	73.43	75.87
deep (–pre & lem)	78.21	72.82	75.42

Table 3: GMB development set.

Model	DRG			DRG w/o refs			DRG w/o refs & conds		
	P	R	F_1	P	R	F_1	P	R	F_1
baseline	52.21	64.46	57.69	47.20	58.93	52.42	52.89	71.80	60.91
shallow	66.61	63.92	65.24	66.05	62.93	64.45	83.30	62.91	71.68
deep	79.27	75.88	77.54	82.87	79.40	81.10	93.91	88.51	91.13

Table 4: GMB test set.

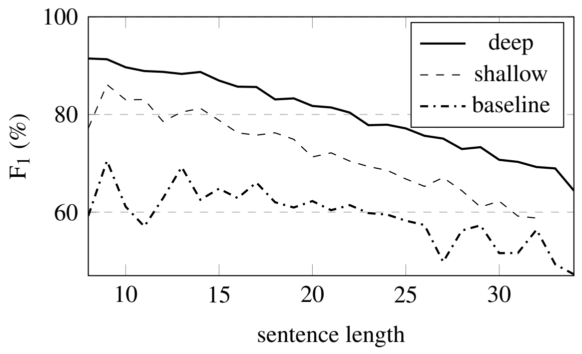


Figure 5: F_1 score as a function of sentence length.

Figure 5 shows F_1 performance for the three parsers on sentences of different length. We observe a similar trend for all models: as sentence length increases, model performance decreases. The baseline and shallow models do not perform well on short sentences which despite containing fewer words, can still represent complex meaning which is challenging to capture sequentially. On the other hand, the performance of the deep model is relatively stable. LSTMs in this case function relatively well, as they are faced with the easier task of predicting meaning in different stages (starting with a tree skeleton which is progressively refined). We provide examples of model output in the supplementary material.

7 Related Work

Tree-structured Decoding A few recent approaches develop structured decoders which make use of the syntax of meaning representations. Dong and Lapata (2016) and Alvarez-Melis and Jaakkola (2017) generate trees in a top-down fashion, while in other work (Xiao et al., 2016; Krishnamurthy et al., 2017) the decoder generates from a grammar that guarantees that predicted logical forms are well-typed. In a similar vein, Yin and Neubig (2017) generate abstract syntax trees (ASTs) based on the application of production rules defined by the grammar. Rabinovich et al. (2017) introduce a modular decoder whose various components are dynamically composed according to the generated tree structure. In comparison, our model does not use grammar information explicitly. We first decode the structure of the DRS, and then fill in details pertaining to its semantic content. Our model is not strictly speaking top-down, we generate partial trees sequentially, and then expand non-terminal nodes, ensuring that when we generate the children of a node, we have already obtained the structure of the entire tree.

Wide-coverage Semantic Parsing Our model is trained on the GMB (Bos et al., 2017), a richly annotated resource in the style of DRT which provides a unique opportunity for bootstrapping wide-coverage semantic parsers. Boxer (Bos, 2008) was a precursor to the GMB, the first semantic parser of this kind, which deterministically maps CCG derivations onto formal meaning representations. Le and Zuidema (2012) were the first to train a semantic parser on an early release of the GMB (2,000 documents; Basile et al. 2012), however, they abandon lambda calculus in favor of a graph based representation. The latter is closely related to AMR, a general-purpose meaning representation language for broad-coverage text. In AMR the meaning of a sentence is represented as a rooted, directed, edge-labeled and leaf-labeled graph. AMRs do not resemble classical meaning representations and do not have a model-theoretic interpretation. However, see Bos (2016) and Artzi et al. (2015) for translations to first-order logic.

8 Conclusions

We introduced a new end-to-end model for open-domain semantic parsing. Experimental results on the GMB show that our decoder is able to recover discourse representation structures to a good degree (77.54 F_1), albeit with some simplifications. In the future, we plan to model document-level representations which are more in line with DRT and the GMB annotations.

Acknowledgments We thank the anonymous reviewers for their feedback and Johan Bos for answering several questions relating to the GMB. We gratefully acknowledge the support of the European Research Council (Lapata, Liu; award number 681760) and the EU H2020 project SUMMA (Cohen, Liu; grant agreement 688139).

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representation (ICLR)*, Toulon, France.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Diego, California.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286.

Johan Bos. 2015. Open-domain semantic parsing with Boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304. Linköping University Electronic Press, Sweden.

Johan Bos. 2016. Expressive power of abstract meaning representations. *Computational Linguistics*, 42(3):527–535.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, volume 2, pages 463–496. Springer.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55, Vancouver, Canada.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Ann Copestake, Dan Flickinger, Carl Pollar, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 2–3(3):281–332.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, Christine Pao David Pallett, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: the atis-3 corpus. In *Proceedings of the workshop on ARPA Human Language Technology*, pages 43–48, Plainsboro, New Jersey.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany.

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

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany.

Hans Kamp and Uwe Reyle. 1993. From discourse to logic; an introduction to modeltheoretic semantics of natural language, formal logic and DRT.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1062–1068, Pittsburgh, PA.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, Banff, Canada.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark.

Phong Le and Willem Zuidema. 2012. Learning compositional semantics for open domain semantic parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1535–1552, Mumbai, India.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540, Vancouver, Canada.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Noortje J. Venhuizen, Johan Bos, and Harm Brouwer. 2013. Parsimonious semantic representations with projection pointers. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 252–263, Potsdam, Germany.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446, New York City, USA.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1050–1055, Portland, Oregon.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *PProceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland.

Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms

Dinghan Shen[1], Guoyin Wang[1], Wenlin Wang[1], Martin Renqiang Min[2]
Qinliang Su[3], Yizhe Zhang[4], Chunyuan Li[1], Ricardo Henao[1], Lawrence Carin[1]
[1] Duke University [2] NEC Laboratories America [3] Sun Yat-sen University [4] Microsoft Research
dinghan.shen@duke.edu

Abstract

Many deep learning architectures have been proposed to model the *compositionality* in text sequences, requiring a substantial number of parameters and expensive computations. However, there has not been a rigorous evaluation regarding the added value of sophisticated compositional functions. In this paper, we conduct a point-by-point comparative study between Simple Word-Embedding-based Models (SWEMs), consisting of parameter-free pooling operations, relative to word-embedding-based RNN/CNN models. Surprisingly, SWEMs exhibit comparable or even superior performance in the majority of cases considered. Based upon this understanding, we propose two additional pooling strategies over learned word embeddings: (i) a max-pooling operation for improved interpretability; and (ii) a hierarchical pooling operation, which preserves spatial (n-gram) information within text sequences. We present experiments on 17 datasets encompassing three tasks: (i) (long) document classification; (ii) text sequence matching; and (iii) short text tasks, including classification and tagging.

1 Introduction

Word embeddings, learned from massive unstructured text data, are widely-adopted building blocks for Natural Language Processing (NLP). By representing each word as a fixed-length vector, these embeddings can group semantically similar words, while implicitly encoding rich linguistic regularities and patterns (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014). Leveraging the word-embedding construct, many deep architectures have been proposed to model the *compositionality* in variable-length text sequences. These methods range from simple operations like addition (Mitchell and Lapata, 2010; Iyyer et al., 2015), to more sophisticated compositional functions such as Recurrent Neural Networks (RNNs) (Tai et al., 2015; Sutskever et al., 2014), Convolutional Neural Networks (CNNs) (Kalchbrenner et al., 2014; Kim, 2014; Zhang et al., 2017a) and Recursive Neural Networks (Socher et al., 2011a).

Models with more expressive compositional functions, *e.g.*, RNNs or CNNs, have demonstrated impressive results; however, they are typically computationally expensive, due to the need to estimate hundreds of thousands, if not millions, of parameters (Parikh et al., 2016). In contrast, models with simple compositional functions often compute a sentence or document embedding by simply adding, or averaging, over the word embedding of each sequence element obtained via, *e.g.*, *word2vec* (Mikolov et al., 2013), or *GloVe* (Pennington et al., 2014). Generally, such a Simple Word-Embedding-based Model (SWEM) does not explicitly account for spatial, word-order information within a text sequence. However, they possess the desirable property of having significantly fewer parameters, enjoying much faster training, relative to RNN- or CNN-based models. Hence, there is a computation-*vs.*-expressiveness tradeoff regarding how to model the compositionality of a text sequence.

In this paper, we conduct an extensive experimental investigation to understand when, and why, simple pooling strategies, operated over word embeddings alone, already carry sufficient information for natural language understanding. To account for the distinct nature of various NLP tasks that may require different semantic features, we compare SWEM-based models with existing recurrent and convolutional networks in a point-by-point manner. Specifically, we consider 17 datasets, including three distinct NLP tasks: *document classification* (Yahoo news, Yelp reviews, *etc.*), *natural language sequence matching* (SNLI, WikiQA, *etc.*) and *(short) sentence classification/tagging* (Stanford sentiment treebank, TREC, *etc.*). Surprisingly, SWEMs exhibit comparable or even superior performance in the majority of cases considered.

In order to validate our experimental findings, we conduct additional investigations to understand to what extent *the word-order information* is utilized/required to make predictions on different tasks. We observe that in text representation tasks, many words (*e.g.*, stop words, or words that are not related to sentiment or topic) do not meaningfully contribute to the final predictions (*e.g.*, sentiment label). Based upon this understanding, we propose to leverage a *max-pooling* operation directly over the word embedding matrix of a given sequence, to select its most *salient* features. This strategy is demonstrated to extract complementary features relative to the standard averaging operation, while resulting in a more interpretable model. Inspired by a case study on sentiment analysis tasks, we further propose a *hierarchical pooling* strategy to abstract and preserve the spatial information in the final representations. This strategy is demonstrated to exhibit comparable empirical results to LSTM and CNN on tasks that are sensitive to word-order features, while maintaining the favorable properties of not having compositional parameters, thus fast training.

Our work presents a simple yet strong baseline for text representation learning that is widely ignored in benchmarks, and highlights the general computation-*vs.*-expressiveness tradeoff associated with appropriately selecting compositional functions for distinct NLP problems. Furthermore, we quantitatively show that the word-embedding-based text classification tasks can have the similar level of difficulty regardless of the employed models, using the subspace training (Li et al., 2018) to constrain the trainable parameters. Thus, according to Occam's razor, simple models are preferred.

2 Related Work

A fundamental goal in NLP is to develop expressive, yet computationally efficient compositional functions that can capture the linguistic structure of natural language sequences. Recently, several studies have suggested that on certain NLP applications, much simpler word-embedding-based architectures exhibit comparable or even superior performance, compared with more-sophisticated models using recurrence or convolutions (Parikh et al., 2016; Vaswani et al., 2017). Although complex compositional functions are avoided in these models, additional modules, such as attention layers, are employed on top of the word embedding layer. As a result, the specific role that the word embedding plays in these models is not emphasized (or explicit), which distracts from understanding how important the word embeddings alone are to the observed superior performance. Moreover, several recent studies have shown empirically that the advantages of distinct compositional functions are highly dependent on the specific task (Mitchell and Lapata, 2010; Iyyer et al., 2015; Zhang et al., 2015a; Wieting et al., 2015; Arora et al., 2016). Therefore, it is of interest to study the practical value of the additional expressiveness, on a wide variety of NLP problems.

SWEMs bear close resemblance to Deep Averaging Network (DAN) (Iyyer et al., 2015) or fastText (Joulin et al., 2016), where they show that average pooling achieves promising results on certain NLP tasks. However, there exist several key differences that make our work unique. First, we explore a series of pooling operations, rather than only average-pooling. Specifically, a *hierarchical* pooling operation is introduced to incorporate spatial information, which demonstrates superior results on sentiment analysis, relative to average pooling. Second, our work not only explores when simple pooling operations are enough, but also investigates the underlying reasons, *i.e.*, what semantic features are required for distinct NLP problems. Third, DAN and fastText only focused on one or two problems at a time, thus a comprehensive study regarding the effectiveness of various compositional functions on distinct NLP tasks, *e.g.*, categorizing short sentence/long documents, matching natural language sentences, has heretofore been absent. In response, our work seeks to perform a comprehensive comparison with respect to simple-*vs.*-complex compositional functions, across a wide range of NLP problems, and reveals some general rules for rationally selecting models to tackle different tasks.

3 Models & training

Consider a text sequence represented as X (either a sentence or a document), composed of a sequence of words: $\{w_1, w_2, ..., w_L\}$, where L is the number of tokens, *i.e.*, the sentence/document length. Let $\{v_1, v_2, ..., v_L\}$ denote the respective word embeddings for each token, where $v_l \in \mathbb{R}^K$. The compositional function, $X \rightarrow z$, aims to combine word embeddings into a fixed-length sentence/document representation z. These representations are then used to make predictions about sequence X. Below, we describe different types of functions considered in this work.

3.1 Recurrent Sequence Encoder

A widely adopted compositional function is defined in a recurrent manner: the model successively takes word vector v_t at position t, along with the hidden unit h_{t-1} from the last position $t-1$, to update the current hidden unit via $h_t = f(v_t, h_{t-1})$, where $f(\cdot)$ is the transition function.

To address the issue of learning long-term dependencies, $f(\cdot)$ is often defined as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), which employs *gates* to control the flow of information abstracted from a sequence. We omit the details of the LSTM and refer the interested readers to the work by Graves et al. (2013) for further explanation. Intuitively, the LSTM encodes a text sequence considering its word-order information, but yields additional compositional parameters that must be learned.

3.2 Convolutional Sequence Encoder

The Convolutional Neural Network (CNN) architecture (Kim, 2014; Collobert et al., 2011; Gan et al., 2017; Zhang et al., 2017b; Shen et al., 2018) is another strategy extensively employed as the compositional function to encode text sequences. The convolution operation considers windows of n consecutive words within the sequence, where a set of filters (to be learned) are applied to these word windows to generate corresponding *feature maps*. Subsequently, an aggregation operation (such as max-pooling) is used on top of the feature maps to abstract the most salient semantic features, resulting in the final representation. For most experiments, we consider a single-layer CNN text model. However, Deep CNN text models have also been developed (Conneau et al., 2016), and are considered in a few of our experiments.

3.3 Simple Word-Embedding Model (SWEM)

To investigate the raw modeling capacity of word embeddings, we consider a class of models with no additional compositional parameters to encode natural language sequences, termed SWEMs. Among them, the simplest strategy is to compute the element-wise average over word vectors for a given sequence (Wieting et al., 2015; Adi et al., 2016):

$$z = \frac{1}{L} \sum_{i=1}^{L} v_i \, . \tag{1}$$

The model in (1) can be seen as an average pooling operation, which takes the mean over each of the K dimensions for all word embeddings, resulting in a representation z with the same dimension as the embedding itself, termed here SWEM-*aver*. Intuitively, z takes the information of every sequence element into account via the addition operation.

Max Pooling Motivated by the observation that, in general, only a small number of key words contribute to final predictions, we propose another SWEM variant, that extracts the most salient features from every word-embedding dimension, by taking the maximum value along each dimension of the word vectors. This strategy is similar to the max-over-time pooling operation in convolutional neural networks (Collobert et al., 2011):

$$z = \text{Max-pooling}(v_1, v_2, ..., v_L) \, . \tag{2}$$

We denote this model variant as SWEM-*max*. Here the j-th component of z is the maximum element in the set $\{v_{1j}, \ldots, v_{Lj}\}$, where v_{1j} is, for example, the j-th component of v_1. With this pooling operation, those words that are unimportant or unrelated to the corresponding tasks will be ignored in the encoding process (as the components of the embedding vectors will have small amplitude), unlike SWEM-*aver* where every word contributes equally to the representation.

Considering that SWEM-*aver* and SWEM-*max* are complementary, in the sense of accounting for different types of information from text sequences, we also propose a third SWEM variant, where the two abstracted features are concatenated together to form the sentence embeddings, denoted here as SWEM-*concat*. For all SWEM variants, there are no additional compositional parameters to be

Model	Parameters	Complexity	Sequential Ops
CNN	$n \cdot K \cdot d$	$\mathcal{O}(n \cdot L \cdot K \cdot d)$	$\mathcal{O}(1)$
LSTM	$4 \cdot d \cdot (K + d)$	$\mathcal{O}(L \cdot d^2 + L \cdot K \cdot d)$	$\mathcal{O}(L)$
SWEM	0	$\mathcal{O}(L \cdot K)$	$\mathcal{O}(1)$

Table 1: Comparisons of CNN, LSTM and SWEM architectures. Columns correspond to the number of *compositional* parameters, computational complexity and sequential operations, respectively.

learned. As a result, the models only exploit intrinsic word embedding information for predictions.

Hierarchical Pooling Both SWEM-*aver* and SWEM-*max* do not take word-order or spatial information into consideration, which could be useful for certain NLP applications. So motivated, we further propose a *hierarchical* pooling layer. Let $v_{i:i+n-1}$ refer to the *local* window consisting of n consecutive words words, $v_i, v_{i+1}, ..., v_{i+n-1}$. First, an average-pooling is performed on each local window, $v_{i:i+n-1}$. The extracted features from all windows are further down-sampled with a *global* max-pooling operation on top of the representations for every window. We call this approach SWEM-*hier* due to its layered pooling.

This strategy preserves the local spatial information of a text sequence in the sense that it keeps track of how the sentence/document is constructed from individual word windows, *i.e.*, n-grams. This formulation is related to bag-of-n-grams method (Zhang et al., 2015b). However, SWEM-*hier* learns fixed-length representations for the n-grams that appear in the corpus, rather than just capturing their occurrences via count features, which may potentially advantageous for prediction purposes.

3.4 Parameters & Computation Comparison

We compare CNN, LSTM and SWEM wrt their parameters and computational speed. K denotes the dimension of word embeddings, as above. For the CNN, we use n to denote the filter width (assumed constant for all filters, for simplicity of analysis, but in practice variable n is commonly used). We define d as the dimension of the final sequence representation. Specifically, d represents the dimension of hidden units or the number of filters in LSTM or CNN, respectively.

We first examine the number of *compositional parameters* for each model. As shown in Table 1, both the CNN and LSTM have a large number of parameters, to model the semantic compositionality of text sequences, whereas SWEM has no such parameters. Similar to Vaswani et al. (2017), we then consider the computational complexity and the minimum number of sequential operations required for each model. SWEM tends to be more efficient than CNN and LSTM in terms of computation complexity. For example, considering the case where $K = d$, SWEM is faster than CNN or LSTM by a factor of nd or d, respectively. Further, the computations in SWEM are highly parallelizable, unlike LSTM that requires $\mathcal{O}(L)$ sequential steps.

4 Experiments

We evaluate different compositional functions on a wide variety of supervised tasks, including document categorization, text sequence matching (given a sentence pair, X_1, X_2, predict their relationship, y) as well as (short) sentence classification. We experiment on 17 datasets concerning natural language understanding, with corresponding data statistics summarized in the Supplementary Material. Our code will be released to encourage future research.

We use GloVe word embeddings with $K = 300$ (Pennington et al., 2014) as initialization for all our models. Out-Of-Vocabulary (OOV) words are initialized from a uniform distribution with range $[-0.01, 0.01]$. The GloVe embeddings are employed in two ways to learn refined word embeddings: (i) directly updating each word embedding during training; and (ii) training a 300-dimensional Multilayer Perceptron (MLP) layer with ReLU activation, with GloVe embeddings as input to the MLP and with output defining the refined word embeddings. The latter approach corresponds to learning an MLP model that adapts GloVe embeddings to the dataset and task of interest. The advantages of these two methods differ from dataset to dataset. We choose the better strategy based on their corresponding performances on the validation set. The final classifier is implemented as an MLP layer with dimension selected from the set $[100, 300, 500, 1000]$, followed by a sigmoid or softmax function, depending on the specific task.

Adam (Kingma and Ba, 2014) is used to optimize all models, with learning rate selected from the set $[1 \times 10^{-3}, 3 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-5}]$ (with cross-validation used to select the appropriate parameter for a given dataset and task). Dropout regularization (Srivastava et al., 2014) is

Model	Yahoo! Ans.	AG News	Yelp P.	Yelp F.	DBpedia
Bag-of-means*	60.55	83.09	87.33	53.54	90.45
Small word CNN*	69.98	89.13	94.46	58.59	98.15
Large word CNN*	70.94	91.45	95.11	59.48	98.28
LSTM*	70.84	86.06	94.74	58.17	98.55
Deep CNN (29 layer)†	73.43	91.27	**95.72**	**64.26**	**98.71**
fastText ‡	72.0	91.5	93.8	60.4	98.1
fastText (bigram)‡	72.3	92.5	95.7	63.9	98.6
SWEM-*aver*	73.14	91.71	93.59	60.66	98.42
SWEM-*max*	72.66	91.79	93.25	59.63	98.24
SWEM-*concat*	**73.53**	**92.66**	93.76	61.11	**98.57**
SWEM-*hier*	73.48	92.48	**95.81**	**63.79**	98.54

Table 2: Test accuracy on (long) document classification tasks, in percentage. Results marked with $*$ are reported in Zhang et al. (2015b), with † are reported in Conneau et al. (2016), and with ‡ are reported in Joulin et al. (2016).

Politics	Science	Computer	Sports	Chemistry	Finance	Geoscience
philipdru	coulomb	system32	billups	sio2 (SiO_2)	proprietorship	fossil
justices	differentiable	cobol	midfield	nonmetal	ameritrade	zoos
impeached	paranormal	agp	sportblogs	pka	retailing	farming
impeachment	converge	dhcp	mickelson	chemistry	mlm	volcanic
neocons	antimatter	win98	juventus	quarks	budgeting	ecosystem

Table 3: Top five words with the largest values in a given word-embedding dimension (each column corresponds to a dimension). The first row shows the (manually assigned) topic for words in each column.

employed on the word embedding layer and final MLP layer, with dropout rate selected from the set $[0.2, 0.5, 0.7]$. The batch size is selected from $[2, 8, 32, 128, 512]$.

4.1 Document Categorization

We begin with the task of categorizing documents (with approximately 100 words in average per document). We follow the data split in Zhang et al. (2015b) for comparability. These datasets can be generally categorized into three types: *topic categorization* (represented by Yahoo! Answer and AG news), *sentiment analysis* (represented by Yelp Polarity and Yelp Full) and *ontology classification* (represented by DBpedia). Results are shown in Table 2. Surprisingly, on topic prediction tasks, our SWEM model exhibits stronger performances, relative to both LSTM and CNN compositional architectures, this by leveraging both the average and max-pooling features from word embeddings. Specifically, our SWEM-*concat* model even outperforms a 29-layer deep CNN model (Conneau et al., 2016), when predicting topics. On the ontology classification problem (DBpedia dataset), we observe the same trend, that SWEM exhibits comparable or even superior results, relative to CNN or LSTM models.

Since there are no compositional parameters in SWEM, our models have an order of magnitude fewer parameters (excluding embeddings) than LSTM or CNN, and are considerably more computationally efficient. As illustrated in Table 4, SWEM-*concat* achieves better results on Yahoo! Answer than CNN/LSTM, with only 61K parameters (one-tenth the number of LSTM parameters, or one-third the number of CNN parameters), while taking a fraction of the training time relative to the CNN or LSTM.

Model	Parameters	Speed
CNN	541K	171s
LSTM	1.8M	598s
SWEM	**61K**	**63s**

Table 4: Speed & Parameters on Yahoo! Answer dataset.

Interestingly, for the sentiment analysis tasks, both CNN and LSTM compositional functions perform better than SWEM, suggesting that word-order information may be required for analyzing sentiment orientations. This finding is consistent with Pang et al. (2002), where they hypothesize that the positional information of a word in text sequences may be beneficial to predict sentiment. This is intuitively reasonable since, for instance, the phrase "not really good" and "really not good" convey different levels of negative sentiment, while being different only by their word orderings. Contrary to SWEM, CNN and

LSTM models can both capture this type of information via convolutional filters or recurrent transition functions. However, as suggested above, such word-order patterns may be much less useful for predicting the topic of a document. This may be attributed to the fact that word embeddings alone already provide sufficient topic information of a document, at least when the text sequences considered are relatively long.

4.1.1 Interpreting model predictions

Although the proposed SWEM-*max* variant generally performs a slightly worse than SWEM-*aver*, it extracts complementary features from SWEM-*aver*, and hence in most cases SWEM-*concat* exhibits the best performance among all SWEM variants. More importantly, we found that the word embeddings learned from SWEM-*max* tend to be sparse. We trained our SWEM-*max* model on the Yahoo datasets (randomly initialized). With the learned embeddings, we plot the values for each of the word embedding dimensions, for the entire vocabulary. As shown in Figure 1, most of the values are highly concentrated around zero, indicating that the word embeddings learned are very sparse. On the contrary, the GloVe word embeddings, for the same vocabulary, are considerably denser than the embeddings learned from SWEM-*max*. This suggests that the model may only depend on a few key words, among the entire vocabulary, for predictions (since most words do not contribute to the max-pooling operation in SWEM-*max*). Through the embedding, the model learns the important words for a given task (those words with non-zero embedding components).

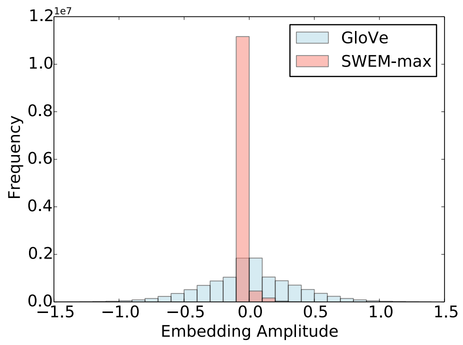


Figure 1: Histograms for learned word embeddings (randomly initialized) of SWEM-*max* and GloVe embeddings for the same vocabulary, trained on the Yahoo! Answer dataset.

In this regard, the nature of max-pooling process gives rise to a more interpretable model. For a document, only the word with largest value in each embedding dimension is employed for the final representation. Thus, we suspect that semantically similar words may have large values in some shared dimensions. So motivated, after training the SWEM-*max* model on the Yahoo dataset, we selected five words with the largest values, among the entire vocabulary, for each word embedding dimension (these words are selected preferentially in the corresponding dimension, by the max operation). As shown in Table 3, the words chosen wrt each embedding dimension are indeed highly relevant and correspond to a common topic (the topics are inferred from words). For example, the words in the first column of Table 3 are all political terms, which could be assigned to the *Politics & Government* topic. Note that our model can even learn locally interpretable structure that is not explicitly indicated by the label information. For instance, all words in the fifth column are *Chemistry*-related. However, we do not have a chemistry label in the dataset, and regardless they should belong to the *Science* topic.

4.2 Text Sequence Matching

To gain a deeper understanding regarding the modeling capacity of word embeddings, we further investigate the problem of sentence matching, including natural language inference, answer sentence selection and paraphrase identification. The corresponding performance metrics are shown in Table 5. Surprisingly, on most of the datasets considered (except WikiQA), SWEM demonstrates the best results compared with those with CNN or the LSTM encoder. Notably, on SNLI dataset, we observe that SWEM-*max* performs the best among all SWEM variants, consistent with the findings in Nie and Bansal (2017); Conneau et al. (2017), that *max-pooling* over BiLSTM hidden units outperforms average pooling operation on SNLI dataset. As a result, with only 120K parameters, our SWEM-*max* achieves a test accuracy of 83.8%, which is very competitive among state-of-the-art sentence encoding-based models (in terms of both performance and number of parameters)[1].

The strong results of the SWEM approach on these tasks may stem from the fact that when matching natural language sentences, it is sufficient in most cases to simply model the word-level

[1]See leaderboard at `https://nlp.stanford.edu/projects/snli/` for details.

Model	SNLI	MultiNLI Matched	MultiNLI Mismatched	WikiQA		Quora	MSRP	
	Acc.	*Acc.*	*Acc.*	*MAP*	*MRR*	*Acc.*	*Acc.*	*F1*
CNN	82.1	65.0	65.3	0.6752	0.6890	79.60	69.9	80.9
LSTM	80.6	66.9*	66.9*	**0.6820**	**0.6988**	82.58	70.6	80.5
SWEM-*aver*	82.3	66.5	66.2	**0.6808**	**0.6922**	82.68	71.0	81.1
SWEM-*max*	**83.8**	**68.2**	**67.7**	0.6613	0.6717	82.20	70.6	80.8
SWEM-*concat*	83.3	67.9	67.6	0.6788	0.6908	**83.03**	**71.5**	**81.3**

Table 5: Performance of different models on matching natural language sentences. Results with * are for Bidirectional LSTM, reported in Williams et al. (2017). Our reported results on MultiNLI are only trained MultiNLI training set (without training data from SNLI). For MSRP dataset, we follow the setup in Hu et al. (2014) and do not use any additional features.

alignments between two sequences (Parikh et al., 2016). From this perspective, word-order information becomes much less useful for predicting relationship between sentences. Moreover, considering the simpler model architecture of SWEM, they could be much easier to be optimized than LSTM or CNN-based models, and thus give rise to better empirical results.

4.2.1 Importance of word-order information

One possible disadvantage of SWEM is that it ignores the word-order information within a text sequence, which could be potentially captured by CNN- or LSTM-based models. However, we empirically found that except for sentiment analysis, SWEM exhibits similar or even superior performance as the CNN or LSTM on a variety of tasks. In this regard, one natural question would be: how important are word-order features for these tasks? To this end, we randomly shuffle the words for every sentence in the training set, while keeping the original word order for samples in the test set. The motivation here is to remove the word-order features from the training set and examine how sensitive the performance on different tasks are to word-order information. We use LSTM as the model for this purpose since it can captures word-order information from the original training set.

Datasets	Yahoo	Yelp P.	SNLI
Original	72.78	95.11	78.02
Shuffled	72.89	93.49	77.68

Table 6: Test accuracy for LSTM model trained on original/shuffled training set.

The results on three distinct tasks are shown in Table 6. Somewhat surprisingly, for Yahoo and SNLI datasets, the LSTM model trained on shuffled training set shows comparable accuracies to those trained on the original dataset, indicating that word-order information does not contribute significantly on these two problems, *i.e.*, topic categorization and textual entailment. However, on the Yelp polarity dataset, the results drop noticeably, further suggesting that word-order does matter for sentiment analysis (as indicated above from a different perspective).

Negative:	Friendly staff and nice selection of vegetarian options. Food **is just okay**, **not great**. **Makes me wonder why everyone likes** food fight so much.
Positive:	The store is small, but it carries specialties that are difficult to find in Pittsburgh. I **was particularly excited** to find middle eastern chili sauce and chocolate covered turkish delights.

Table 7: Test samples from Yelp Polarity dataset for which LSTM gives wrong predictions with shuffled training data, but predicts correctly with the original training set.

Notably, the performance of LSTM on the Yelp dataset with a shuffled training set is very close to our results with SWEM, indicating that the main difference between LSTM and SWEM may be due to the ability of the former to capture word-order features. Both observations are in consistent with our experimental results in the previous section.

Case Study To understand what type of sentences are sensitive to word-order information, we further show those samples that are wrongly predicted because of the shuffling of training data in Table 7. Taking the first sentence as an example, several words in the review are generally positive, *i.e. friendly*, *nice*, *okay*, *great* and *likes*. However, the most vital features for predicting the sentiment of this sentence could be the phrase/sentence *'is just okay'*, *'not great'* or *'makes me wonder why everyone likes'*, which cannot be captured without

Model	MR	SST-1	SST-2	Subj	TREC
RAE (Socher et al., 2011b)	77.7	43.2	82.4	–	–
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–
LSTM (Tai et al., 2015)	–	46.4	84.9	–	–
RNN (Zhao et al., 2015)	77.2	–	–	**93.7**	90.2
Constituency Tree-LSTM (Tai et al., 2015)	-	**51.0**	88.0	-	-
Dynamic CNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0
CNN (Kim, 2014)	**81.5**	48.0	**88.1**	93.4	**93.6**
DAN-ROOT (Iyyer et al., 2015)	-	46.9	85.7	-	-
SWEM-*aver*	77.6	45.2	83.9	92.5	**92.2**
SWEM-*max*	76.9	44.1	83.6	91.2	89.0
SWEM-*concat*	**78.2**	**46.1**	**84.3**	**93.0**	91.8

Table 8: Test accuracies with different compositional functions on (short) sentence classifications.

considering word-order features. It is worth noting the hints for predictions in this case are actually n-gram phrases from the input document.

4.3 SWEM-*hier* for sentiment analysis

As demonstrated in Section 4.2.1, word-order information plays a vital role for sentiment analysis tasks. However, according to the case study above, the most important features for sentiment prediction may be some key n-gram phrase/words from the input document. We hypothesize that incorporating information about the local word-order, *i.e.*, n-gram features, is likely to largely mitigate the limitations of the above three SWEM variants. Inspired by this observation, we propose using another simple pooling operation termed as hierarchical (SWEM-*hier*), as detailed in Section 3.3. We evaluate this method on the two document-level sentiment analysis tasks and the results are shown in the last row of Table 2.

SWEM-*hier* greatly outperforms the other three SWEM variants, and the corresponding accuracies are comparable to the results of CNN or LSTM (Table 2). This indicates that the proposed hierarchical pooling operation manages to abstract spatial (word-order) information from the input sequence, which is beneficial for performance in sentiment analysis tasks.

4.4 Short Sentence Processing

We now consider sentence-classification tasks (with approximately 20 words on average). We experiment on three sentiment classification datasets, *i.e.*, MR, SST-1, SST-2, as well as subjectivity classification (Subj) and question classification (TREC). The corresponding results are shown in Table 8. Compared with CNN/LSTM compositional functions, SWEM yields inferior accuracies on sentiment analysis datasets, consistent with our observation in the case of document categorization. However, SWEM exhibits comparable performance on the other two tasks, again with much less parameters and faster training. Further, we investigate two sequence tagging tasks: the standard CoNLL2000 chunking and CoNLL2003 NER datasets. Results are shown in the Supplementary Material, where LSTM and CNN again perform better than SWEMs. Generally, SWEM is less effective at extracting representations from *short* sentences than from *long* documents. This may be due to the fact that for a shorter text sequence, word-order features tend to be more important since the semantic information provided by word embeddings alone is relatively limited.

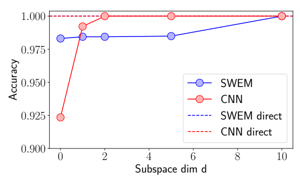
Moreover, we note that the results on these relatively small datasets are highly sensitive to model regularization techniques due to the overfitting issues. In this regard, one interesting future direction may be to develop specific regularization strategies for the SWEM framework, and thus make them work better on small sentence classification datasets.

5 Discussion

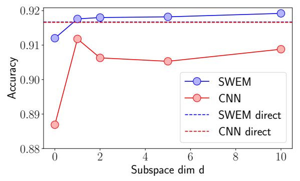
5.1 Comparison via subspace training

We use *subspace training* (Li et al., 2018) to measure the model complexity in text classification problems. It constrains the optimization of the trainable parameters in a subspace of low dimension d, the intrinsic dimension d_{int} defines the minimum d that yield a good solution. Two models are studied: the SWEM-*max* variant, and the CNN model including a convolutional layer followed by a FC layer. We consider two settings:

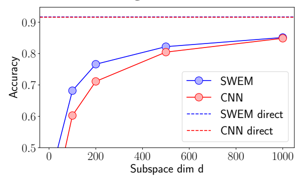
(1) The word embeddings are randomly intialized, and optimized jointly with the model parameters. We show the performance of direct and subspace training on AG News dataset in Figure 2 (a)(b). The two models trained via direct method share almost identical perfomrnace on training and



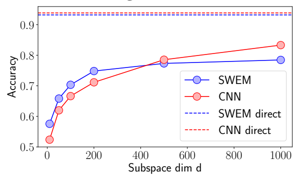
(a) Training on AG News



(b) Testing on AG News



(c) Testing on AG News



(d)Testing on Yelp P.

Figure 2: Performance of subspace training. Word embeddings are optimized in (a)(b), and frozen in (c)(d).

testing. The subspace training yields similar accuracy with direct training for very small d, even when model parameters are not trained at all ($d = 0$). This is because the word embeddings have the full degrees of freedom to adjust to achieve good solutions, regardless of the employed models. SWEM seems to have an easier loss landscape than CNN for word embeddings to find the best solutions. According to Occam's razor, simple models are preferred, if all else are the same.

(2) The pre-trained GloVe are frozen for the word embeddings, and only the model parameters are optimized. The results on testing datasets of AG News and Yelp P. are shown in Figure 2 (c)(d), respectively. SWEM shows significantly higher accuracy than CNN for a large range of low subspace dimension, indicating that SWEM is more parameter-efficient to get a decent solution. In Figure 2(c), if we set the performance threshold as 80% testing accuracy, SWEM exhibits a lower d_{int} than CNN on AG News dataset. However, in Figure 2(d), CNN can leverage more trainable parameters to achieve higher accuracy when d is large.

5.2 Linear classifiers

To further investigate the quality of representations learned from SWEMs, we employ a linear classifier on top of the representations for prediction, instead of a non-linear MLP layer as in the previous section. It turned out that utilizing a linear classifier only leads to a very small performance drop for both Yahoo! Ans. (from 73.53% to 73.18%) and Yelp P. datasets (from 93.76% to 93.66%) . This observation highlights that SWEMs are able to extract robust and informative sentence representations despite their simplicity.

5.3 Extension to other languages

We have also tried our SWEM-concat and SWEM-hier models on Sogou news corpus (with the same experimental setup as (Zhang et al., 2015b)), which is a *Chinese* dataset represented by Pinyin (a phonetic romanization of Chinese). SWEM-concat yields an accuracy of 91.3%, while SWEM-hier (with a local window size of 5) obtains an accuracy of 96.2% on the test set. Notably, the performance of SWEM-hier is comparable to the best accuracies of CNN (95.6%) and LSTM (95.2%), as reported in (Zhang et al., 2015b). This indicates that hierarchical pooling is more suitable than average/max pooling for Chinese text classification, by taking spatial information into account. It also implies that Chinese is more sensitive to local word-order features than English.

6 Conclusions

We have performed a comparative study between SWEM (with parameter-free pooling operations) and CNN or LSTM-based models, to represent text sequences on 17 NLP datasets. We further validated our experimental findings through additional exploration, and revealed some general rules for rationally selecting compositional functions for distinct problems. Our findings regarding when (and why) simple pooling operations are enough for text sequence representations are summarized as follows:

• Simple pooling operations are surprisingly effective at representing longer documents (with hundreds of words), while recurrent/convolutional compositional functions are most effective when constructing representations for short sentences.

• Sentiment analysis tasks are more sensitive to word-order features than topic categorization tasks. However, a simple *hierarchical pooling layer* proposed here achieves comparable results to LSTM/CNN on sentiment analysis tasks.

• To match natural language sentences, *e.g.*, textual entailment, answer sentence selection, *etc.*, simple pooling operations already exhibit similar or even superior results, compared to CNN and LSTM.

• In SWEM with max-pooling operation, each *individual dimension* of the word embeddings contains interpretable semantic patterns, and groups together words with a common theme or *topic*.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *ICLR*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR*, 3(Feb):1137–1155.

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

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *EMNLP*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *EMNLP*, pages 2380–2390.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, volume 1, pages 16 81–1691.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *EMNLP*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86. ACL.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. 2017. Adaptive convolutional filter generation for natural language understanding. *arXiv preprint arXiv:1709.08294*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2018. Deconvolutional latent-variable model for text sequence matching. *AAAI*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015a. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 495–500.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017a. Adversarial feature matching for text generation. In *ICML*.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017b. Deconvolutional paragraph representation learning. *NIPS*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*, pages 4069–4076.

ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations

John Wieting[1] **Kevin Gimpel**[2]
[1]Carnegie Mellon University, Pittsburgh, PA, 15213, USA
[2]Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
jwieting@cs.cmu.edu, kgimpel@ttic.edu

Abstract

We describe ParaNMT-50M, a dataset of more than 50 million English-English sentential paraphrase pairs. We generated the pairs automatically by using neural machine translation to translate the non-English side of a large parallel corpus, following Wieting et al. (2017). Our hope is that ParaNMT-50M can be a valuable resource for paraphrase generation and can provide a rich source of semantic knowledge to improve downstream natural language understanding tasks. To show its utility, we use ParaNMT-50M to train paraphrastic sentence embeddings that outperform all supervised systems on every SemEval semantic textual similarity competition, in addition to showing how it can be used for paraphrase generation.[1]

1 Introduction

While many approaches have been developed for generating or finding paraphrases (Barzilay and McKeown, 2001; Lin and Pantel, 2001; Dolan et al., 2004), there do not exist any freely-available datasets with millions of sentential paraphrase pairs. The closest such resource is the Paraphrase Database (PPDB; Ganitkevitch et al., 2013), which was created automatically from bilingual text by pivoting over the non-English language (Bannard and Callison-Burch, 2005). PPDB has been used to improve word embeddings (Faruqui et al., 2015; Mrkšić et al., 2016). However, PPDB is less useful for learning *sentence* embeddings (Wieting and Gimpel, 2017).

In this paper, we describe the creation of a dataset containing more than 50 million sentential paraphrase pairs. We create it automatically by scaling up the approach of Wieting et al. (2017). We use neural machine translation (NMT) to translate the Czech side of a large Czech-English parallel corpus. We pair the English translations with the English references to form paraphrase pairs. We call this dataset ParaNMT-50M. It contains examples illustrating a broad range of paraphrase phenomena; we show examples in Section 3. ParaNMT-50M has the potential to be useful for many tasks, from linguistically controlled paraphrase generation, style transfer, and sentence simplification to core NLP problems like machine translation.

We show the utility of ParaNMT-50M by using it to train paraphrastic sentence embeddings using the learning framework of Wieting et al. (2016b). We primarily evaluate our sentence embeddings on the SemEval semantic textual similarity (STS) competitions from 2012-2016. Since so many domains are covered in these datasets, they form a demanding evaluation for a general purpose sentence embedding model.

Our sentence embeddings learned from ParaNMT-50M outperform all systems in every STS competition from 2012 to 2016. These tasks have drawn substantial participation; in 2016, for example, the competition attracted 43 teams and had 119 submissions. Most STS systems use curated lexical resources, the provided supervised training data with manually-annotated similarities, and joint modeling of the sentence pair. We use none of these, simply encoding each sentence independently using our models and computing cosine similarity between their embeddings.

We experiment with several compositional architectures and find them all to work well. We find benefit from making a simple change to learning ("mega-batching") to better leverage the large training set, namely, increasing the search space

[1] Dataset, code, and embeddings are available at `https://www.cs.cmu.edu/~jwieting`.

of negative examples. In the supplementary, we evaluate on general-purpose sentence embedding tasks used in past work (Kiros et al., 2015; Conneau et al., 2017), finding our embeddings to perform competitively.

Finally, in Section 6, we briefly report results showing how PARANMT-50M can be used for paraphrase generation. A standard encoder-decoder model trained on PARANMT-50M can generate paraphrases that show effects of "canonicalizing" the input sentence. In other work, fully described by Iyyer et al. (2018), we used PARANMT-50M to generate paraphrases that have a specific syntactic structure (represented as the top two levels of a linearized parse tree).

We release the PARANMT-50M dataset, our trained sentence embeddings, and our code. PARANMT-50M is the largest collection of sentential paraphrases released to date. We hope it can motivate new research directions and be used to create powerful NLP models, while adding a robustness to existing ones by incorporating paraphrase knowledge. Our paraphrastic sentence embeddings are state-of-the-art by a significant margin, and we hope they can be useful for many applications both as a sentence representation function and as a general similarity metric.

2 Related Work

We discuss work in automatically building paraphrase corpora, learning general-purpose sentence embeddings, and using parallel text for learning embeddings and similarity functions.

Paraphrase discovery and generation. Many methods have been developed for generating or finding paraphrases, including using multiple translations of the same source material (Barzilay and McKeown, 2001), using distributional similarity to find similar dependency paths (Lin and Pantel, 2001), using comparable articles from multiple news sources (Dolan et al., 2004; Dolan and Brockett, 2005; Quirk et al., 2004), aligning sentences between standard and Simple English Wikipedia (Coster and Kauchak, 2011), crowdsourcing (Xu et al., 2014, 2015; Jiang et al., 2017), using diverse MT systems to translate a single source sentence (Suzuki et al., 2017), and using tweets with matching URLs (Lan et al., 2017).

The most relevant prior work uses bilingual corpora. Bannard and Callison-Burch (2005) used methods from statistical machine translation to find lexical and phrasal paraphrases in parallel text. Ganitkevitch et al. (2013) scaled up these techniques to produce the Paraphrase Database (PPDB). Our goals are similar to those of PPDB, which has likewise been generated for many languages (Ganitkevitch and Callison-Burch, 2014) since it only needs parallel text. In particular, we follow the approach of Wieting et al. (2017), who used NMT to translate the non-English side of parallel text to get English-English paraphrase pairs. We scale up the method to a larger dataset, produce state-of-the-art paraphrastic sentence embeddings, and release all of our resources.

Sentence embeddings. Our learning and evaluation setting is the same as that of our recent work that seeks to learn paraphrastic sentence embeddings that can be used for downstream tasks (Wieting et al., 2016b,a; Wieting and Gimpel, 2017; Wieting et al., 2017). We trained models on noisy paraphrase pairs and evaluated them primarily on semantic textual similarity (STS) tasks. Prior work in learning general sentence embeddings has used autoencoders (Socher et al., 2011; Hill et al., 2016), encoder-decoder architectures (Kiros et al., 2015; Gan et al., 2017), and other sources of supervision and learning frameworks (Le and Mikolov, 2014; Pham et al., 2015; Arora et al., 2017; Pagliardini et al., 2017; Conneau et al., 2017).

Parallel text for learning embeddings. Prior work has shown that parallel text, and resources built from parallel text like NMT systems and PPDB, can be used for learning embeddings for words and sentences. Several have used PPDB as a knowledge resource for training or improving embeddings (Faruqui et al., 2015; Wieting et al., 2015; Mrkšić et al., 2016). NMT architectures and training settings have been used to obtain better embeddings for words (Hill et al., 2014a,b) and words-in-context (McCann et al., 2017). Hill et al. (2016) evaluated the encoders of English-to-X NMT systems as sentence representations. Mallinson et al. (2017) adapted trained NMT models to produce sentence similarity scores in semantic evaluations.

3 The PARANMT-50M Dataset

To create our dataset, we used back-translation of bitext (Wieting et al., 2017). We used a Czech-English NMT system to translate Czech sentences

Dataset	Avg. Length	Avg. IDF	Avg. Para. Score	Vocab. Entropy	Parse Entropy	Total Size
Common Crawl	24.0$\pm$34.7	7.7$\pm$1.1	0.83$\pm$0.16	7.2	3.5	0.16M
CzEng 1.6	13.3$\pm$19.3	7.4$\pm$1.2	0.84$\pm$0.16	6.8	4.1	51.4M
Europarl	26.1$\pm$15.4	7.1$\pm$0.6	0.95$\pm$0.05	6.4	3.0	0.65M
News Commentary	25.2$\pm$13.9	7.5$\pm$1.1	0.92$\pm$0.12	7.0	3.4	0.19M

Table 1: Statistics of 100K-samples of Czech-English parallel corpora; standard deviations are shown for averages.

Reference Translation	Machine Translation
so, what's half an hour?	half an hour won't kill you.
well, don't worry. i've taken out tons and tons of guys. lots of guys.	don't worry, i've done it to dozens of men.
it's gonna be ...... classic.	yeah, sure. it's gonna be great.
greetings, all!	hello everyone!
but she doesn't have much of a case.	but as far as the case goes, she doesn't have much.
it was good in spite of the taste.	despite the flavor, it felt good.

Table 2: Example paraphrase pairs from PARANMT-50M, where each consists of an English reference translation and the machine translation of the Czech source sentence (not shown).

from the training data into English. We paired the translations with the English references to form English-English paraphrase pairs.

We used the pretrained Czech-English model from the NMT system of Sennrich et al. (2017). Its training data includes four sources: Common Crawl, CzEng 1.6 (Bojar et al., 2016), Europarl, and News Commentary. We did not choose Czech due to any particular linguistic properties. Wieting et al. (2017) found little difference among Czech, German, and French as source languages for back-translation. There were much larger differences due to data domain, so we focus on the question of domain in this section. We leave the question of investigating properties of back-translation of different languages to future work.

3.1 Choosing a Data Source

To assess characteristics that yield useful data, we randomly sampled 100K English reference translations from each data source and computed statistics. Table 1 shows the average sentence length, the average inverse document frequency (IDF) where IDFs are computed using Wikipedia sentences, and the average paraphrase score for the two sentences. The paraphrase score is calculated by averaging PARAGRAM-PHRASE embeddings (Wieting et al., 2016b) for the two sentences in each pair and then computing their cosine similarity. The table also shows the entropies of the vocabularies and constituent parses obtained using the Stanford Parser (Manning et al., 2014).[2]

Europarl exhibits the least diversity in terms of rare word usage, vocabulary entropy, and parse entropy. This is unsurprising given its formulaic and repetitive nature. CzEng has shorter sentences than the other corpora and more diverse sentence structures, as shown by its high parse entropy. In terms of vocabulary use, CzEng is not particularly more diverse than Common Crawl and News Commentary, though this could be due to the prevalence of named entities in the latter two.

In Section 5.3, we empirically compare these data sources as training data for sentence embeddings. The CzEng corpus yields the strongest performance when controlling for training data size. Since its sentences are short, we suspect this helps ensure high-quality back-translations. A large portion of it is movie subtitles which tend to use a wide vocabulary and have a diversity of sentence structures; however, other domains are included as well. It is also the largest corpus, containing over 51 million sentence pairs. In addition to providing a large number of training examples for downstream tasks, this means that the NMT system should be able to produce quality translations for this subset of its training data.

For all of these reasons, we chose the CzEng corpus to create PARANMT-50M. When doing so, we used beam search with a beam size of 12 and selected the highest scoring translation from the beam. It took over 10,000 GPU hours to back-translate the CzEng corpus. We show illustrative examples in Table 2.

3.2 Manual Evaluation

We conducted a manual analysis of our dataset in order to quantify its noise level and assess how the

[2]To mitigate sparsity in the parse entropy, we used only the top two levels of each parse tree.

Para. Score Range	# (M)	Avg. Tri. Overlap	Paraphrase 1	2	3	Fluency 1	2	3
(-0.1, 0.2]	4.0	0.00±0.0	92	6	2	1	5	94
(0.2, 0.4]	3.8	0.02±0.1	53	32	15	1	12	87
(0.4, 0.6]	6.9	0.07±0.1	22	45	33	2	9	89
(0.6, 0.8]	14.4	0.17±0.2	1	43	56	11	0	89
(0.8, 1.0]	18.0	0.35±0.2	1	13	86	3	0	97

Table 3: Manual evaluation of PARANMT-50M. 100-pair samples were drawn from five ranges of the automatic paraphrase score (first column). Paraphrase strength and fluency were judged on a 1-3 scale and counts of each rating are shown.

noise can be ameliorated with filtering. Two native English speakers annotated a sample of 100 examples from each of five ranges of the Paraphrase Score.[3] We obtained annotations for both the strength of the paraphrase relationship and the fluency of the translations.

To annotate paraphrase strength, we adopted the annotation guidelines used by Agirre et al. (2012). The original guidelines specify six classes, which we reduced to three for simplicity. We combined the top two into one category, left the next, and combined the bottom three into the lowest category. Therefore, for a sentence pair to have a rating of 3, the sentences must have the same meaning, but some unimportant details can differ. To have a rating of 2, the sentences are roughly equivalent, with some important information missing or that differs slightly. For a rating of 1, the sentences are not equivalent, even if they share minor details.

For fluency of the back-translation, we use the following: A rating of 3 means it has no grammatical errors, 2 means it has one to two errors, and 1 means it has more than two grammatical errors or is not a natural English sentence.

Table 3 summarizes the annotations. For each score range, we report the number of pairs, the mean trigram overlap score, and the number of times each paraphrase/fluency label was present in the sample of 100 pairs. There is noise but it is largely confined to the bottom two ranges which together comprise only 16% of the entire dataset. In the highest paraphrase score range, 86% of the pairs possess a strong paraphrase relationship. The annotations suggest that PARANMT-50M contains approximately 30 million strong paraphrase pairs, and that the paraphrase score is a good indicator of quality. At the low ranges, we inspected the data and found there to be many errors in the sentence alignment in the original bitext. With regards to fluency, approximately 90% of the back-translations are fluent, even at the low end of the paraphrase score range. We do see an outlier at the second-highest range of the paraphrase score, but this may be due to the small number of annotated examples.

[3]Even though the similarity score lies in $[-1, 1]$, most observed scores were positive, so we chose the five ranges shown in Table 3.

4 Learning Sentence Embeddings

To show the usefulness of the PARANMT-50M dataset, we will use it to train sentence embeddings. We adopt the learning framework from Wieting et al. (2016b), which was developed to train sentence embeddings from pairs in PPDB. We first describe the compositional sentence embedding models we will experiment with, then discuss training and our modification ("mega-batching").

Models. We want to embed a word sequence s into a fixed-length vector. We denote the tth word in s as s_t, and we denote its word embedding by x_t. We focus on three model families, though we also experiment with combining them in various ways. The first, which we call WORD, simply averages the embeddings x_t of all words in s. This model was found by Wieting et al. (2016b) to perform strongly for semantic similarity tasks.

The second is similar to WORD, but instead of word embeddings, we average character trigram embeddings (Huang et al., 2013). We call this TRIGRAM. Wieting et al. (2016a) found this to work well for sentence embeddings compared to other n-gram orders and to word averaging.

The third family includes long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). We average the hidden states to produce the final sentence embedding. For regularization during training, we scramble words with a small probability (Wieting and Gimpel, 2017). We also experiment with bidirectional LSTMs (BLSTM), averaging the forward and backward hidden states with no concatenation.[4]

Training. The training data is a set S of paraphrase pairs $\langle s, s' \rangle$ and we minimize a margin-

[4]Unlike Conneau et al. (2017), we found this to outperform max-pooling for both semantic similarity and general sentence embedding tasks.

based loss $\ell(s, s') =$

$$\max(0, \delta - \cos(g(s), g(s')) + \cos(g(s), g(t)))$$

where g is the model (WORD, TRIGRAM, etc.), δ is the margin, and t is a "negative example" taken from a mini-batch during optimization. The intuition is that we want the two texts to be more similar to each other than to their negative examples. To select t we choose the most similar sentence in some set. For simplicity we use the mini-batch for this set, i.e.,

$$t = \operatorname*{argmax}_{t' : \langle t', \cdot \rangle \in S_b \setminus \{\langle s, s' \rangle\}} \cos(g(s), g(t'))$$

where $S_b \subseteq S$ is the current mini-batch.

Modification: mega-batching. By using the mini-batch to select negative examples, we may be limiting the learning procedure. That is, if all potential negative examples in the mini-batch are highly dissimilar from s, the loss will be too easy to minimize. Stronger negative examples can be obtained by using larger mini-batches, but large mini-batches are sub-optimal for optimization.

Therefore, we propose a procedure we call "mega-batching." We aggregate M mini-batches to create one mega-batch and select negative examples from the mega-batch. Once each pair in the mega-batch has a negative example, the mega-batch is split back up into M mini-batches and training proceeds. We found that this provides more challenging negative examples during learning as shown in Section 5.5. Table 6 shows results for different values of M, showing consistently higher correlations with larger M values.

5 Experiments

We now investigate how best to use our generated paraphrase data for training paraphrastic sentence embeddings.

5.1 Evaluation

We evaluate sentence embeddings using the SemEval semantic textual similarity (STS) tasks from 2012 to 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016) and the STS Benchmark (Cer et al., 2017). Given two sentences, the aim of the STS tasks is to predict their similarity on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 means they are completely equivalent. As our test set, we report the average Pearson's r over each year of the STS tasks from 2012-2016. We use the small (250-example) English dataset from SemEval 2017 (Cer et al., 2017) as a development set, which we call STS2017 below.

Training Corpus	WORD	TRIGRAM	LSTM
Common Crawl	80.9	80.2	79.1
CzEng 1.6	**83.6**	**81.5**	**82.5**
Europarl	78.9	78.0	80.4
News Commentary	80.2	78.2	80.5

Table 4: Pearson's $r \times 100$ on STS2017 when training on 100k pairs from each back-translated parallel corpus. CzEng works best for all models.

The supplementary material contains a description of a method to obtain a paraphrase lexicon from PARANMT-50M that is on par with that provided by PPDB 2.0. We also evaluate our sentence embeddings on a range of additional tasks that have previously been used for evaluating sentence representations (Kiros et al., 2015).

5.2 Experimental Setup

For training sentence embeddings on PARANMT-50M, we follow the experimental procedure of Wieting et al. (2016b). We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix for all models that use word embeddings. We fix the mini-batch size to 100 and the margin δ to 0.4. We train all models for 5 epochs. For optimization we use Adam (Kingma and Ba, 2014) with a learning rate of 0.001. For the LSTM and BLSTM, we fixed the scrambling rate to 0.3.[5]

5.3 Dataset Comparison

We first compare parallel data sources. We evaluate the quality of a data source by using its back-translations paired with its English references as training data for paraphrastic sentence embeddings. We compare the four data sources described in Section 3. We use 100K samples from each corpus and trained 3 different models on each: WORD, TRIGRAM, and LSTM. Table 4 shows that CzEng provides the best training data for all models, so we used it to create PARANMT-50M and for all remaining experiments.

[5]As in our prior work (Wieting and Gimpel, 2017), we found that scrambling significantly improves results, even with our much larger training set. But while we previously used a scrambling rate of 0.5, we found that a smaller rate of 0.3 worked better when training on PARANMT-50M, presumably due to the larger training set.

Filtering Method	Model Avg.
Translation Score	83.2
Trigram Overlap	83.1
Paraphrase Score	**83.3**

Table 5: Pearson's $r \times 100$ on STS2017 for the best training fold across the average of WORD, TRIGRAM, and LSTM models for each filtering method.

CzEng is diverse in terms of both vocabulary and sentence structure. It has significantly shorter sentences than the other corpora, and has much more training data, so its translations are expected to be better than those in the other corpora. Wieting et al. (2017) found that sentence length was the most important factor in filtering quality training data, presumably due to how NMT quality deteriorates with longer sentences. We suspect that better translations yield better data for training sentence embeddings.

5.4 Data Filtering

Since the PARANMT-50M dataset is so large, it is computationally demanding to train sentence embeddings on it in its entirety. So, we filter the data to create a training set for sentence embeddings.

We experiment with three simple methods: (1) the length-normalized translation score from decoding, (2) trigram overlap (Wieting et al., 2017), and (3) the paraphrase score from Section 3. Trigram overlap is calculated by counting trigrams in the reference and translation, then dividing the number of shared trigrams by the total number in the reference or translation, whichever has fewer.

We filtered the back-translated CzEng data using these three strategies. We ranked all 51M+ paraphrase pairs in the dataset by the filtering measure under consideration and then split the data into tenths (so the first tenth contains the bottom 10% under the filtering criterion, the second contains those in the bottom 10-20%, etc.).

We trained WORD, TRIGRAM, and LSTM models for a single epoch on 1M examples sampled from each of the ten folds for each filtering criterion. We averaged the correlation on the STS2017 data across models for each fold. Table 5 shows the results of the filtering methods. Filtering based on the paraphrase score produces the best data for training sentence embeddings.

We randomly selected 5M examples from the top two scoring folds using paraphrase score filtering, ensuring that we only selected examples in which both sentences have a maximum length of 30 tokens.[6] These resulting 5M examples form the training data for the rest of our experiments. Note that many more than 5M pairs from the dataset are useful, as suggested by our human evaluations in Section 3.2. We have experimented with doubling the training data when training our best sentence similarity model and found the correlation increased by more than half a percentage point on average across all datasets.

M	WORD	TRIGRAM	LSTM
1	82.3	81.5	81.5
20	84.0	83.1	84.6
40	**84.1**	**83.4**	**85.0**

Table 6: Pearson's $r \times 100$ on STS2017 with different mega-batch sizes M.

original	sir, i'm just trying to protect.
negative examples:	
$M = 1$	i mean, colonel...
$M = 20$	i only ask that the baby be safe.
$M = 40$	just trying to survive. on instinct.
original	i'm looking at him, you know?
$M = 1$	they know that i've been looking for her.
$M = 20$	i'm keeping him.
$M = 40$	i looked at him with wonder.
original	i'il let it go a couple of rounds.
$M = 1$	sometimes the ball doesn't go down.
$M = 20$	i'll take two.
$M = 40$	i want you to sit out a couple of rounds, all right?

Table 7: Negative examples for various mega-batch sizes M with the BLSTM model.

5.5 Effect of Mega-Batching

Table 6 shows the impact of varying the mega-batch size M when training for 5 epochs on our 5M-example training set. For all models, larger mega-batches improve performance. There is a smaller gain when moving from 20 to 40, but all models show clear gains over $M = 1$.

Table 7 shows negative examples with different mega-batch sizes M. We use the BLSTM model and show the negative examples (nearest neighbors from the mega-batch excluding the current training example) for three sentences. Using larger mega-batches improves performance, presumably by producing more compelling negative examples for the learning procedure. This is likely more important when training on sentences than

[6] Wieting et al. (2017) found that sentence length cutoffs were effective for filtering back-translated parallel text.

	Training Data	Model	Dim.	2012	2013	2014	2015	2016
Our Work	ParaNMT	Word	300	66.2	61.8	76.2	79.3	77.5
		Trigram	300	67.2	60.3	76.1	79.7	**78.3**
		LSTM	300	67.0	62.3	76.3	78.5	76.0
		LSTM	900	**68.0**	60.4	76.3	78.8	75.9
		BLSTM	900	67.4	60.2	76.1	79.5	76.5
		Word + Trigram (addition)	300	67.3	**62.8**	**77.5**	80.1	78.2
		Word + Trigram + LSTM (addition)	300	67.1	**62.8**	76.8	79.2	77.0
		Word, Trigram (concatenation)	600	67.8	62.7	77.4	**80.3**	78.1
		Word, Trigram, LSTM (concatenation)	900	67.7	**62.8**	76.9	79.8	76.8
	SimpWiki	Word, Trigram (concatenation)	600	61.8	58.4	74.4	77.0	74.0
STS Competitions		1st Place System	-	64.8	62.0	74.3	79.0	77.7
		2nd Place System	-	63.4	59.1	74.2	78.0	75.7
		3rd Place System	-	64.1	58.3	74.3	77.8	75.7
Related Work		InferSent (AllSNLI) (Conneau et al., 2017)	4096	58.6	51.5	67.8	68.3	67.2
		InferSent (SNLI) (Conneau et al., 2017)	4096	57.1	50.4	66.2	65.2	63.5
		FastSent (Hill et al., 2016)	100	-	-	63	-	-
		DictRep (Hill et al., 2016)	500	-	-	67	-	-
		SkipThought (Kiros et al., 2015)	4800	-	-	29	-	-
		CPHRASE (Pham et al., 2015)	-	-	-	65	-	-
		CBOW (from Hill et al., 2016)	500	-	-	64	-	-
		BLEU (Papineni et al., 2002)	-	39.2	29.5	42.8	49.8	47.4
		METEOR (Denkowski and Lavie, 2014)	-	53.4	47.6	63.7	68.8	61.8

Table 8: Pearson's $r \times 100$ on the STS tasks of our models and those from related work. We compare to the top performing systems from each SemEval STS competition. Note that we are reporting the mean correlations over domains for each year rather than weighted means as used in the competitions. Our best performing overall model (Word, Trigram) is in bold.

	Dim.	Corr.
Our Work (Unsupervised)		
Word	300	79.2
Trigram	300	79.1
LSTM	300	78.4
Word + Trigram (addition)	300	79.9
Word + Trigram + LSTM (addition)	300	79.6
Word, Trigram (concatenation)	600	79.9
Word, Trigram, LSTM (concatenation)	900	79.2
Related Work (Unsupervised)		
InferSent (AllSNLI) (Conneau et al., 2017)	4096	70.6
C-PHRASE (Pham et al., 2015)		63.9
GloVe (Pennington et al., 2014)	300	40.6
word2vec (Mikolov et al., 2013)	300	56.5
sent2vec (Pagliardini et al., 2017)	700	75.5
Related Work (Supervised)		
Dep. Tree LSTM (Tai et al., 2015)		71.2
Const. Tree LSTM (Tai et al., 2015)		71.9
CNN (Shao, 2017)		78.4

Table 9: Results on STS Benchmark test set.

prior work on learning from text snippets (Wieting et al., 2015, 2016b; Pham et al., 2015).

5.6 Model Comparison

Table 8 shows results on the 2012-2016 STS tasks and Table 9 shows results on the STS Benchmark.[7] Our best models outperform all STS competition systems and all related work of which we are aware on the 2012-2016 STS datasets. Note that the large improvement over BLEU and METEOR suggests that our embeddings could be useful for evaluating machine translation output.

[7]Baseline results are from http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark, except for the unsupervised InferSent result which we computed.

Models	Mean Pearson Abs. Diff.
Word / Trigram	2.75
Word / LSTM	2.17
Trigram / LSTM	2.89

Table 10: The means (over all 25 STS competition datasets) of the absolute differences in Pearson's r between each pair of models.

Overall, our individual models (Word, Trigram, LSTM) perform similarly. Using 300 dimensions appears to be sufficient; increasing dimensionality does not necessarily improve correlation. When examining particular STS tasks, we found that our individual models showed marked differences on certain tasks. Table 10 shows the mean absolute difference in Pearson's r over all 25 datasets. The Trigram model shows the largest differences from the other two, both of which use word embeddings. This suggests that Trigram may be able to complement the other two by providing information about words that are unknown to models that rely on word embeddings.

We experiment with two ways of combining models. The first is to define additive architectures

Target Syntax	Paraphrase
original	with the help of captain picard, the borg will be prepared for everything.
`(SBARQ(ADVP)(,)(S)(,)(SQ))`	now, the borg will be prepared by picard, will it?
`(S(NP)(ADVP)(VP))`	the borg here will be prepared for everything.
original	you seem to be an excellent burglar when the time comes.
`(S(SBAR)(,)(NP)(VP))`	when the time comes, you'll be a great thief.
``(S(``)(UCP)('')(NP)(VP))``	"you seem to be a great burglar, when the time comes." you said.

Table 11: Syntactically controlled paraphrases generated by the SCPN trained on PARANMT-50M.

that form the embedding for a sentence by adding the embeddings computed by two (or more) individual models. All parameters are trained jointly just like when we train individual models; that is, we do not first train two simple models and add their embeddings. The second way is to define concatenative architectures that form a sentence embedding by concatenating the embeddings computed by individual models, and again to train all parameters jointly.

In Table 8 and Table 9, these combinations show consistent improvement over the individual models as well as the larger LSTM and BLSTM. Concatenating WORD and TRIGRAM results in the best performance on average across STS tasks, outperforming the best supervised systems from each year. We have released the pretrained model for these "WORD, TRIGRAM" embeddings. In addition to providing a strong baseline for future STS tasks, these embeddings offer the advantages of being extremely efficient to compute and being robust to unknown words.

We show the usefulness of PARANMT by also reporting the results of training the "WORD, TRIGRAM" model on SimpWiki, a dataset of aligned sentences from Simple English and standard English Wikipedia (Coster and Kauchak, 2011). It has been shown useful for training sentence embeddings in past work (Wieting and Gimpel, 2017). However, Table 8 shows that training on PARANMT leads to gains in correlation of 3 to 6 points compared to SimpWiki.

6 Paraphrase Generation

In addition to powering state-of-the-art paraphrastic sentence embeddings, our dataset is useful for paraphrase generation. We briefly describe two efforts in paraphrase generation here.

We have found that training an encoder-decoder model on PARANMT-50M can produce a paraphrase generation model that canonicalizes text. For this experiment, we used a bidirectional LSTM encoder and a two-layer LSTM decoder with soft attention over the encoded states (Bahdanau et al., 2015). The attention computation consists of a bilinear product with a learned parameter matrix. Table 12 shows examples of output generated by this model, showing how the model is able to standardize the text and correct grammatical errors. This model would be interesting to evaluate for automatic grammar correction as it does so without any direct supervision. Future work could also use this canonicalization to improve performance of models by standardizing inputs and removing noise from data.

original	overall, i that it's a decent buy, and am happy that i own it.
paraphrase	it's a good buy, and i'm happy to own it.
original	oh, that's a handsome women, that is.
paraphrase	that's a beautiful woman.

Table 12: Examples from our paraphrase generation model that show the ability to canonicalize text and correct grammatical errors.

PARANMT-50M has also been used for syntactically-controlled paraphrase generation; this work is described in detail by Iyyer et al. (2018). A syntactically controlled paraphrase network (SCPN) is trained to generate a paraphrase of a sentence whose constituent structure follows a provided parse template. A parse template contains the top two levels of a linearized parse tree. Table 11 shows example outputs using the SCPN. The paraphrases mostly preserve the semantics of the input sentences while changing their syntax to fit the target syntactic templates. The SCPN was used for augmenting training data and finding adversarial examples.

We believe that PARANMT-50M and future datasets like it can be used to generate rich paraphrases that improve the performance and robustness of models on a multitude of NLP tasks.

7 Discussion

One way to view PARANMT-50M is as a way to represent the learned translation model in a monolingual generated dataset. This raises the question of whether we could learn an effective sentence embedding model from the original parallel text used to train the NMT system, rather than requiring the intermediate step of generating a paraphrase training set.

However, while Hill et al. (2016) and Mallinson et al. (2017) used trained NMT models to produce sentence similarity scores, their correlations are considerably lower than ours (by 10% to 35% absolute in terms of Pearson). It appears that NMT encoders form representations that do not necessarily encode the semantics of the sentence in a way conducive to STS evaluations. They must instead create representations suitable for a decoder to generate a translation. These two goals of representing sentential semantics and producing a translation, while likely correlated, evidently have some significant differences.

Our use of an intermediate dataset leads to the best results, but this may be due to our efforts in optimizing learning for this setting (Wieting et al., 2016b; Wieting and Gimpel, 2017). Future work will be needed to develop learning frameworks that can leverage parallel text directly to reach the same or improved correlations on STS tasks.

8 Conclusion

We described the creation of ParaNMT-50M, a dataset of more than 50M English sentential paraphrase pairs. We showed how to use ParaNMT-50M to train paraphrastic sentence embeddings that outperform supervised systems on STS tasks, as well as how it can be used for generating paraphrases for purposes of data augmentation, robustness, and even grammar correction.

The key advantage of our approach is that it only requires parallel text. There are hundreds of millions of parallel sentence pairs, and more are being generated continually. Our procedure is immediately applicable to the wide range of languages for which we have parallel text.

We release ParaNMT-50M, our code, and pretrained sentence embeddings, which also exhibit strong performance as general-purpose representations for a multitude of tasks. We hope that ParaNMT-50M, along with our embeddings, can impart a notion of meaning equivalence to improve NLP systems for a variety of tasks. We are actively investigating ways to apply these two new resources to downstream applications, including machine translation, question answering, and additional paraphrase generation tasks.

Acknowledgments

We thank the anonymous reviewers, the developers of Theano (Theano Development Team, 2016), the developers of PyTorch (Paszke et al., 2017), and NVIDIA Corporation for donating GPUs used in this research.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of* *the International Conference on Learning Representations*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting on Association for Computational Linguistics*, pages 50–57.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Computer Science, pages 231–238, Cham / Heidelberg / New York / Dordrecht / London. Masaryk University, Springer International Publishing.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark.

William Coster and David Kauchak. 2011. Simple English Wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 665–669.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2380–2390, Copenhagen, Denmark.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014a. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*.

Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014b. Not all neural embeddings are born equal. *arXiv preprint arXiv:1410.0718*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding task design trade-offs in crowdsourced paraphrase collection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 103–109, Vancouver, Canada.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):342–360.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP 2004*, pages 142–149, Barcelona, Spain.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain.

Yang Shao. 2017. HCTI at SemEval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 130–133.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Yui Suzuki, Tomoyuki Kajiwara, and Mamoru Komachi. 2017. Building a non-trivial paraphrase corpus using multiple machine translation systems. In *Proceedings of ACL 2017, Student Research Workshop*, pages 36–42, Vancouver, Canada. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics* *and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088, Vancouver, Canada.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 274–285, Copenhagen, Denmark.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Event2Mind: Commonsense Inference on Events, Intents, and Reactions

Hannah Rashkin[†*] **Maarten Sap**[†*] **Emily Allaway**[†] **Noah A. Smith**[†] **Yejin Choi**[†‡]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Allen Institute for Artificial Intelligence
{hrashkin,msap,eallaway,nasmith,yejin}@cs.washington.edu

Abstract

We investigate a new commonsense inference task: given an event described in a short free-form text ("X drinks coffee in the morning"), a system reasons about the likely intents ("X wants to stay awake") and reactions ("X feels alert") of the event's participants. To support this study, we construct a new crowdsourced corpus of 25,000 event phrases covering a diverse range of everyday events and situations. We report baseline performance on this task, demonstrating that neural encoder-decoder models can successfully compose embedding representations of previously unseen events and reason about the likely intents and reactions of the event participants. In addition, we demonstrate how commonsense inference on people's intents and reactions can help unveil the implicit gender inequality prevalent in modern movie scripts.

1 Introduction

Understanding a narrative requires commonsense reasoning about the mental states of people in relation to events. For example, if "Alex is dragging his feet at work", pragmatic implications about Alex's *intent* are that "Alex wants to avoid doing things" (Figure 1). We can also infer that Alex's *emotional reaction* might be feeling "lazy" or "bored". Furthermore, while not explicitly mentioned, we can infer that people other than Alex are affected by the situation, and these people are likely to feel "frustrated" or "impatient".

This type of pragmatic inference can potentially be useful for a wide range of NLP applications

[*]These two authors contributed equally.

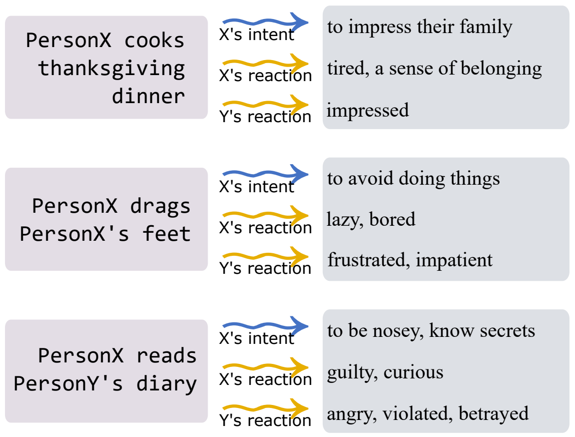


Figure 1: Examples of commonsense inference on mental states of event participants. In the third example event, common sense tells us that Y is likely to feel betrayed as a result of X reading their diary.

that require accurate anticipation of people's intents and emotional reactions, even when they are not explicitly mentioned. For example, an ideal dialogue system should react in empathetic ways by reasoning about the human user's mental state based on the events the user has experienced, without the user explicitly stating how they are feeling. Similarly, advertisement systems on social media should be able to reason about the emotional reactions of people after events such as mass shootings and remove ads for guns which might increase social distress (Goel and Isaac, 2016). Also, pragmatic inference is a necessary step toward automatic narrative understanding and generation (Tomai and Forbus, 2010; Ding and Riloff, 2016; Ding et al., 2017). However, this type of social commonsense reasoning goes far beyond the widely studied entailment tasks (Bowman et al., 2015; Dagan et al., 2006) and thus falls outside the scope of existing benchmarks.

In this paper, we introduce a new task, corpus,

PersonX's Intent	Event Phrase	PersonX's Reaction	Others' Reactions
to express anger to vent their frustration to get PersonY's full attention	**PersonX starts to yell at PersonY**	mad frustrated annoyed	shocked humiliated mad at PersonX
to communicate something without being rude to let the other person think for themselves to be subtle	**PersonX drops a hint**	sly secretive frustrated	oblivious surprised grateful
to catch the criminal to be civilized justice	**PersonX reports ___ to the police**	anxious worried nervous	sad angry regret
to wake up to feel more energized	**PersonX drinks a cup of coffee**	alert awake refreshed	NONE
to be feared to be taken seriously to exact revenge	**PersonX carries out PersonX's threat**	angry dangerous satisfied	sad afraid angry
NONE	**It starts snowing**	NONE	calm peaceful cold

Table 1: Example annotations of intent and reactions for 6 event phrases. Each annotator could fill in up to three free-responses for each mental state.

and model, supporting commonsense inference on events with a specific focus on modeling stereotypical intents and reactions of people, described in short free-form text. Our study is in a similar spirit to recent efforts of Ding and Riloff (2016) and Zhang et al. (2017), in that we aim to model aspects of commonsense inference via natural language descriptions. Our new contributions are: (1) a new corpus that supports commonsense inference about people's intents and reactions over a diverse range of everyday events and situations, (2) inference about even those people who are not directly mentioned by the event phrase, and (3) a task formulation that aims to *generate* the textual descriptions of intents and reactions, instead of classifying their polarities or classifying the inference relations between two given textual descriptions.

Our work establishes baseline performance on this new task, demonstrating that, given the phrase-level inference dataset, neural encoder-decoder models can successfully compose phrasal embeddings for previously unseen events and reason about the mental states of their participants.

Furthermore, in order to showcase the practical implications of commonsense inference on events and people's mental states, we apply our model to modern movie scripts, which provide a new insight into the gender bias in modern films beyond what previous studies have offered (England et al., 2011; Agarwal et al., 2015; Ramakrishna et al., 2017; Sap et al., 2017). The resulting corpus includes around 25,000 event phrases, which combine automatically extracted phrases from stories and blogs with all idiomatic verb phrases listed in the Wiktionary. Our corpus is publicly available.[1]

2 Dataset

One goal of our investigation is to probe whether it is feasible to build computational models that can perform limited, but well-scoped commonsense inference on short free-form text, which we refer to as *event phrases*. While there has been much prior research on phrase-level paraphrases (Pavlick et al., 2015) and phrase-level entailment (Dagan et al., 2006), relatively little prior work focused on phrase-level inference that requires prag-

[1]https://tinyurl.com/event2mind

matic or commonsense interpretation. We scope our study to two distinct types of inference: given a phrase that describes an event, we want to reason about the likely intents and emotional reactions of people who caused or affected by the event. This complements prior work on more general commonsense inference (Speer and Havasi, 2012; Li et al., 2016; Zhang et al., 2017), by focusing on the causal relations between events and people's mental states, which are not well covered by most existing resources.

We collect a wide range of phrasal event descriptions from stories, blogs, and Wiktionary idioms. Compared to prior work on phrasal embeddings (Wieting et al., 2015; Pavlick et al., 2015), our work generalizes the phrases by introducing (typed) variables. In particular, we replace words that correspond to entity mentions or pronouns with typed variables such as `PersonX` or `PersonY`, as shown in examples in Table 1. More formally, the phrases we extract are a combination of a verb predicate with partially instantiated arguments. We keep specific arguments together with the predicate, if they appear frequently enough (e.g., `PersonX eats pasta for dinner`). Otherwise, the arguments are replaced with an untyped blank (e.g., `PersonX eats ___ for dinner`). In our work, only person mentions are replaced with typed variables, leaving other types to future research.

Inference types The first type of pragmatic inference is about *intent*. We define intent as an explanation of why the agent causes a volitional event to occur (or "none" if the event phrase was unintentional). The intent can be considered a mental pre-condition of an action or an event. For example, if the event phrase is `PersonX takes a stab at ___`, the annotated intent might be that "PersonX wants to solve a problem".

The second type of pragmatic inference is about *emotional reaction*. We define reaction as an explanation of how the mental states of the agent and other people involved in the event would change as a result. The reaction can be considered a mental post-condition of an action or an event. For example, if the event phrase is that `PersonX gives PersonY ___ as a gift`, PersonX might "feel good about themselves" as a result, and PersonY might "feel grateful" or "feel thankful".

Source	# Unique Events	# Unique Verbs	Average κ
ROC Story	13,627	639	0.57
G. N-grams	7,066	789	0.39
Spinn3r	2,130	388	0.41
Idioms	1,916	442	0.42
Total	**24,716**	**1,333**	**0.45**

Table 2: Data and annotation agreement statistics for our new phrasal inference corpus. Each event is annotated by three crowdworkers.

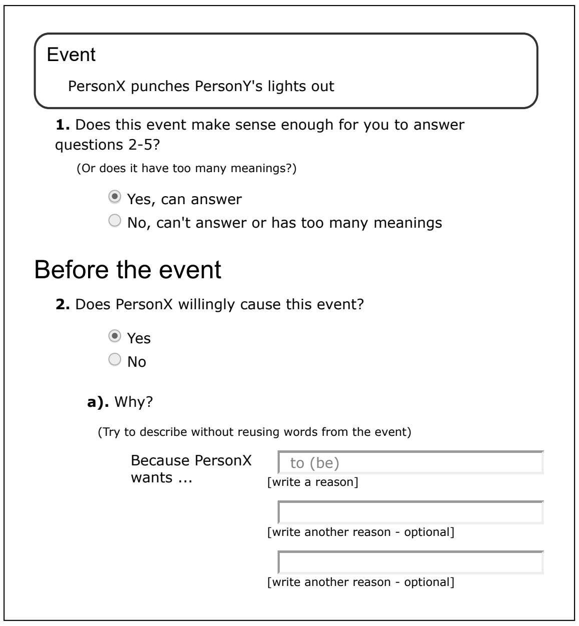
2.1 Event Extraction

We extract phrasal events from three different corpora for broad coverage: the ROC Story training set (Mostafazadeh et al., 2016), the Google Syntactic N-grams (Goldberg and Orwant, 2013), and the Spinn3r corpus (Gordon and Swanson, 2008). We derive events from the set of verb phrases in our corpora, based on syntactic parses (Klein and Manning, 2003). We then replace the predicate subject and other entities with the typed variables (e.g., `PersonX`, `PersonY`), and selectively substitute verb arguments with blanks (___). We use frequency thresholds to select events to annotate (for details, see Appendix A.1). Additionally, we supplement the list of events with all 2,000 verb idioms found in Wiktionary, in order to cover events that are less compositional.[2] Our final annotation corpus contains nearly 25,000 event phrases, spanning over 1,300 unique verb predicates (Table 2).

2.2 Crowdsourcing

We design an Amazon Mechanical Turk task to annotate the mental pre- and post-conditions of event phrases. A snippet of our MTurk HIT design is shown in Figure 2. For each phrase, we ask three annotators whether the agent of the event, PersonX, intentionally causes the event, and if so, to provide up to three possible textual descriptions of their intents. We then ask annotators to provide up to three possible reactions that PersonX might experience as a result. We also ask annotators to provide up to three possible reactions of *other people*, when applicable. These other people can be either explicitly mentioned (e.g., "PersonY" in `PersonX punches PersonY's lights out`), or only implied

[2]We compiled the list of idiomatic verb phrases by cross-referencing between Wiktionary's English idioms category and the Wiktionary English verbs categories.


Event

PersonX punches PersonY's lights out

1. Does this event make sense enough for you to answer questions 2-5?

(Or does it have too many meanings?)

[✓ Yes, can answer]
No, can't answer or has too many meanings

Before the event

2. Does PersonX willingly cause this event?

[✓ Yes]
No

a). Why?

(Try to describe without reusing words from the event)

Because PersonX wants ... to (be)
[write a reason]
[write another reason - optional]
[write another reason - optional]

Figure 2: *Intent* portion of our annotation task. We allow annotators to label events as invalid if the phrase is unintelligible. The full annotation setup is shown in Figure 8 in the appendix.

(e.g., given the event description `PersonX yells at the classroom`, we can infer that other people such as "students" in the classroom may be affected by the act of PersonX). For quality control, we periodically removed workers with high disagreement rates, at our discretion.

Coreference among `Person` variables With the typed `Person` variable setup, events involving multiple people can have multiple meanings depending on coreference interpretation (e.g., `PersonX eats PersonY's lunch` has very different mental state implications from `PersonX eats PersonX's lunch`). To prune the set of events that will be annotated for intent and reaction, we ran a preliminary annotation to filter out candidate events that have implausible coreferences. In this preliminary task, annotators were shown a combinatorial list of coreferences for an event (e.g., `PersonX punches PersonX's lights out`, `PersonX punches PersonY's lights out`) and were asked to select only the plausible ones (e.g., `PersonX punches PersonY's lights out`). Each set of coreferences was annotated by 3 workers, yielding an overall agreement of κ =0.4. This annotation excluded 8,406 events with implausible coreference from our set (out of 17,806 events).

2.3 Mental State Descriptions

Our dataset contains nearly 25,000 event phrases, with annotators rating 91% of our extracted events as "valid" (i.e., the event makes sense). Of those events, annotations for the multiple choice portions of the task (whether or not there exists intent/reaction) agree moderately, with an average Cohen's $\kappa = 0.45$ (Table 2). The individual κ scores generally indicate that turkers disagree half as often as if they were randomly selecting answers.

Importantly, this level of agreement is acceptable in our task formulation for two reasons. First, unlike linguistic annotations on syntax or semantics where experts in the corresponding theory would generally agree on a single correct label, pragmatic interpretations may better be defined as distributions over multiple correct labels (e.g., after `PersonX takes a test`, PersonX might feel relieved and/or stressed; de Marneffe et al., 2012). Second, because we formulate our task as a conditional language modeling problem, where a distribution over the textual descriptions of intents and reactions is conditioned on the event description, this variation in the labels is only as expected.

A majority of our events are annotated as willingly caused by the agent (86%, Cohen's $\kappa = 0.48$), and 26% involve other people ($\kappa = 0.41$). Most event patterns in our data are fully instantiated, with only 22% containing blanks (___). In our corpus, the intent annotations are slightly longer (3.4 words on average) than the reaction annotations (1.5 words).

3 Models

Given an event phrase, our models aim to generate three entity-specific pragmatic inferences: PersonX's intent, PersonX's reaction, and others' reactions. The general outline of our model architecture is illustrated in Figure 3.

The input to our model is an event pattern described through free-form text with typed variables such as `PersonX gives PersonY ___ as a gift`. For notation purposes, we describe each event pattern E as a sequence of word embeddings $\langle e_1, e_2, \ldots, e_n \rangle \in \mathbb{R}^{n \times D}$. This input is encoded as a vector $h_E \in \mathbb{R}^H$ that will be used for predicting output. The output of the model is its hypotheses about PersonX's intent, PersonX's reaction, and others' reactions (v_i,v_x, and v_o, respectively). We experiment with representing the

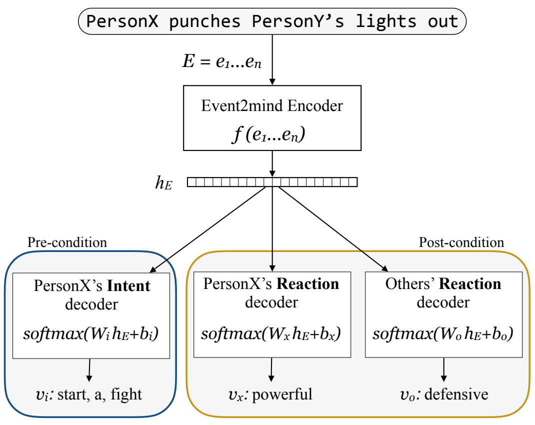


Figure 3: Overview of the model architecture. From an encoded event, our model predicts intents and reactions in a multitask setting.

output in two decoding set-ups: three vectors interpretable as discrete distributions over words and phrases (n-gram reranking) or three sequences of words (sequence decoding).

Encoding events The input event phrase E is compressed into an H-dimensional embedding h_E via an encoding function $f : \mathbb{R}^{n \times D} \rightarrow \mathbb{R}^H$:

$$h_E = f(e_1, \dots, e_n)$$

We experiment with several ways for defining f, inspired by standard techniques in sentence and phrase classification (Kim, 2014). First, we experiment with max-pooling and mean-pooling over the word vectors $\{e_i\}_{i=1}^n$. We also consider a convolutional neural network (ConvNet; LeCun et al., 1998) taking the last layer of the network as the encoded version of the event. Lastly, we encode the event phrase with a bi-directional RNN (specifically, a GRU; Cho et al., 2014), concatenating the final hidden states of the forward and backward cells as the encoding: $h_E = [\overrightarrow{h_n}; \overleftarrow{h_1}]$. For hyperparameters and other details, we refer the reader to Appendix B.

Though the event sequences are typically rather short (4.6 tokens on average), our model still benefits from the ConvNet and BiRNN's ability to compose words.

Pragmatic inference decoding We use three decoding modules that take the event phrase embedding h_E and output distributions of possible PersonX's intent (v_i), PersonX's reactions (v_x), and others' reactions (v_o). We experiment with two different decoder set-ups.

First, we experiment with *n-gram re-ranking*, considering the $|V|$ most frequent $\{1, 2, 3\}$-grams in our annotations. Each decoder projects the event phrase embedding h_E into a $|V|$-dimensional vector, which is then passed through a softmax function. For instance, the distribution over descriptions of PersonX's intent is given by:

$$v_i = \text{softmax}(W_i h_E + b_i)$$

Second, we experiment with *sequence generation*, using RNN decoders to generate the textual description. The event phrase embedding h_E is set as the initial state h_{dec} of three decoder RNNs (using GRU cells), which then output the intent/reactions one word at a time (using beam-search at test time). For example, an event's intent sequence ($v_i = v_i^{(0)} v_i^{(1)} \dots$) is computed as follows:

$$v_i^{(t+1)} = \text{softmax}(W_i \, \text{RNN}(v_i^{(t)}, h_{i,dec}^{(t)}) + b_i)$$

Training objective We minimize the cross-entropy between the predicted distribution over words and phrases, against the one actually observed in our dataset. Further, we employ multitask learning, simultaneously minimizing the loss for all three decoders at each iteration.

Training details We fix our input embeddings, using 300-dimensional skip-gram word embeddings trained on Google News (Mikolov et al., 2013). For decoding, we consider a vocabulary of size $|V|$ = 14,034 in the n-gram re-ranking setup. For the sequence decoding setup, we only consider the unigrams in V, yielding an output space of 7,110 at each time step.

We randomly divided our set of 24,716 unique events (57,094 annotations) into a training/dev./test set using an 80/10/10% split. Some annotations have multiple responses (i.e., a crowdworker gave multiple possible intents and reactions), in which case we take each of the combinations of their responses as a separate training example.

4 Empirical Results

Table 3 summarizes the performance of different encoding models on the dev and test set in terms of cross-entropy and recall at 10 predicted intents and reactions. As expected, we see a moderate improvement in recall and cross-entropy when using the more compositional encoder models (ConvNet and BiRNN; both n-gram and sequence de-

Encoding Function	Decoder	Development Average Cross-Ent	Development Recall @10 (%) Intent	XReact	OReact	Test Average Cross-Ent	Test Recall @10 (%) Intent	XReact	OReact
max-pool	n-gram	5.75	31	35	68	5.14	31	37	67
mean-pool	n-gram	4.82	35	39	69	4.94	34	40	68
ConvNet	n-gram	4.85	36	42	69	4.81	37	44	69
BiRNN 300d	n-gram	4.78	36	42	68	4.74	36	43	69
BiRNN 100d	n-gram	4.76	36	41	68	4.73	37	43	68
mean-pool	sequence	4.59	39	36	67	4.54	40	38	66
ConvNet	sequence	4.44	42	39	68	4.40	43	40	67
BiRNN 100d	sequence	4.25	39	38	67	4.22	40	40	67

Table 3: Average cross-entropy (lower is better) and recall @10 (percentage of times the gold falls within the top 10 decoded; higher is better) on development and test sets for different modeling variations. We show recall values for PersonX's intent, PersonX's reaction and others' reaction (denoted as "Intent", "XReact", and "OReact"). Note that because of two different decoding setups, cross-entropy between n-gram and sequence decoding are not directly comparable.

coding setups). Additionally, BiRNN models outperform ConvNets on cross-entropy in both decoding setups. Looking at the recall split across intent vs. reaction labels ("Intent", "XReact" and "OReact" columns), we see that much of the improvement in using these two models is within the prediction of PersonX's intents. Note that recall for "OReact" is much higher, since a majority of events do not involve other people.

Human evaluation To further assess the quality of our models, we randomly select 100 events from our test set and ask crowd-workers to rate generated intents and reactions. We present 5 workers with an event's top 10 most likely intents and reactions according to our model and ask them to select all those that make sense to them. We evaluate each model's precision @10 by computing the average number of generated responses that make sense to annotators.

Figure 4 summarizes the results of this evaluation. In most cases, the performance is higher for the sequential decoder than the corresponding n-gram decoder. The biggest gain from using sequence decoders is in intent prediction, possibly because intent explanations are more likely to be longer. The BiRNN and ConvNet encoders consistently have higher precision than the mean-pooling with the BiRNN-seq setup slightly outperforming other models. Unless otherwise specified, this is the model we employ in further sections.

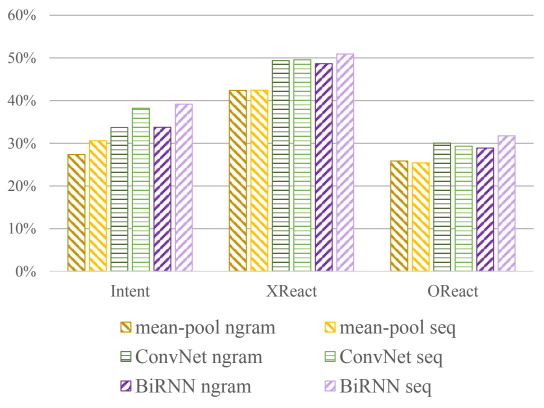
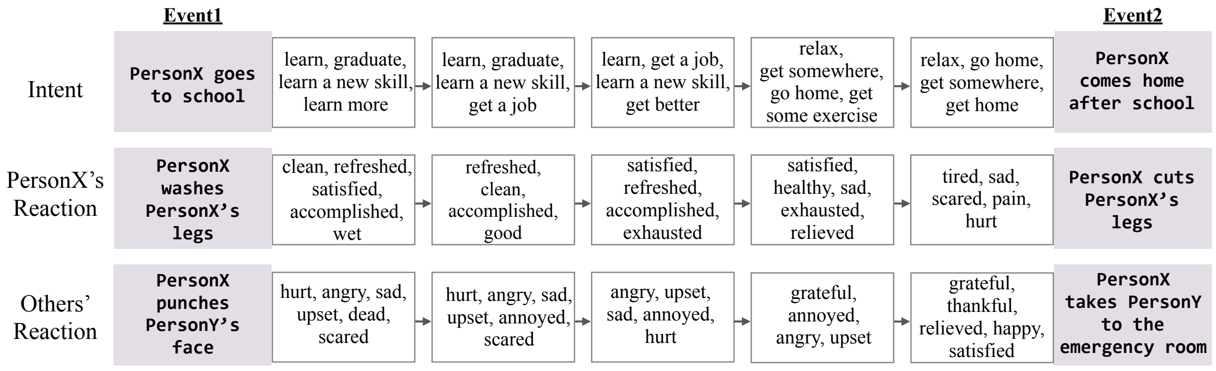


Figure 4: Average precision @10 of each model's top ten responses in the human evaluation. We show results for various encoder functions (mean-pool, ConvNet, BiRNN-100d) combined with two decoding setups (n-gram re-ranking, sequence generation).

Error analyses We test whether certain types of events are easier for predicting commonsense inference. In Figure 6, we show the difference in cross-entropy of the BiRNN 100d model on predicting different portions of the development set including: `Blank` events (events containing non-instantiated arguments), `2+ People` events (events containing multiple different Person variables), and `Idiom` events (events coming from the Wiktionary idiom list). Our results show that, while intent prediction performance remains sim-



	Event1						Event2
Intent	PersonX goes to school	learn, graduate, learn a new skill, learn more	learn, graduate, learn a new skill, get a job	learn, get a job, learn a new skill, get better	relax, get somewhere, go home, get some exercise	relax, go home, get somewhere, get home	PersonX comes home after school
PersonX's Reaction	PersonX washes PersonX's legs	clean, refreshed, satisfied, accomplished, wet	refreshed, clean, accomplished, good	satisfied, refreshed, accomplished, exhausted	satisfied, healthy, sad, exhausted, relieved	tired, sad, scared, pain, hurt	PersonX cuts PersonX's legs
Others' Reaction	PersonX punches PersonY's face	hurt, angry, sad, upset, dead, scared	hurt, angry, sad, upset, annoyed, scared	angry, upset, sad, annoyed, hurt	grateful, annoyed, angry, upset	grateful, thankful, relieved, happy, satisfied	PersonX takes PersonY to the emergency room

Figure 5: Sample predictions from homotopic embeddings (gradual interpolation between Event1 and Event2), selected from the top 10 beam elements decoded in the sequence generation setup. Examples highlight differences captured when ideas are similar (*going to* and *coming from* school), when only a single word differs (*washes* versus *cuts*), and when two events are unrelated.

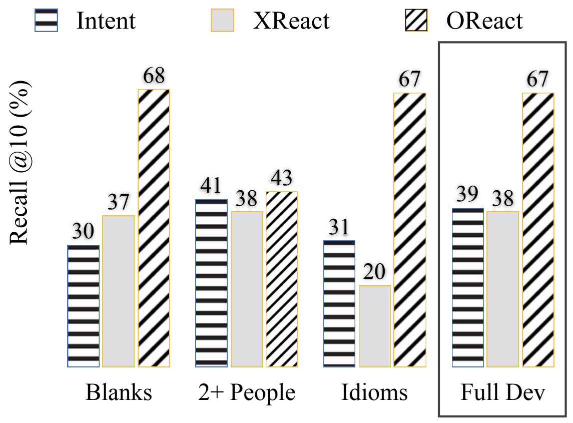


Figure 6: Recall @ 10 (%) on different subsets of the development set for intents, PersonX's reactions, and other people's reactions, using the BiRNN 100d model. "Full dev" represents the recall on the entire development dataset.

ilar for all three sets of events, it is 10% behind intent prediction on the full development set. Additionally, predicting other people's reactions is more difficult for the model when other people are explicitly mentioned. Unsurprisingly, idioms are particularly difficult for commonsense inference, perhaps due to the difficulty in composing meaning over nonliteral or noncompositional event descriptions.

To further evaluate the geometry of the embedding space, we analyze interpolations between pairs of event phrases (from outside the train set), similar to the homotopic analysis of Bowman et al. (2016). For a handful of event pairs, we decode intents, reactions for PersonX, and reactions for other people from points sampled at equal intervals on the interpolated line between two event phrases. We show examples in Figure 5. The embedding space distinguishes changes from generally positive to generally negative words and is also able to capture small differences between event phrases (such as "washes" versus "cuts").

5 Analyzing Bias via Event2Mind Inference

Through Event2Mind inference, we can attempt to bring to the surface what is implied about people's behavior and mental states. We employ this inference to analyze implicit bias in modern films. As shown in Figure 7, our model is able to analyze character portrayal beyond what is explicit in text, by performing pragmatic inference on character actions to explain aspects of a character's mental state. In this section, we use our model's inference to shed light on gender differences in intents behind and reactions to characters' actions.

5.1 Processing of Movie Scripts

For our portrayal analyses, we use scene descriptions from 772 movie scripts released by Gorinski and Lapata (2015), assigned to over 21,000 characters as done by Sap et al. (2017). We extract events from the scene descriptions, and generate their 10 most probable intent and reaction sequences using our BiRNN sequence model (as in Figure 7).

We then categorize generated intents and reactions into groups based on LIWC category scores of the generated output (Tausczik and Pennebaker, 2016).[3] The intent and reaction categories are then

[3]We only consider content word categories: 'Core Drives

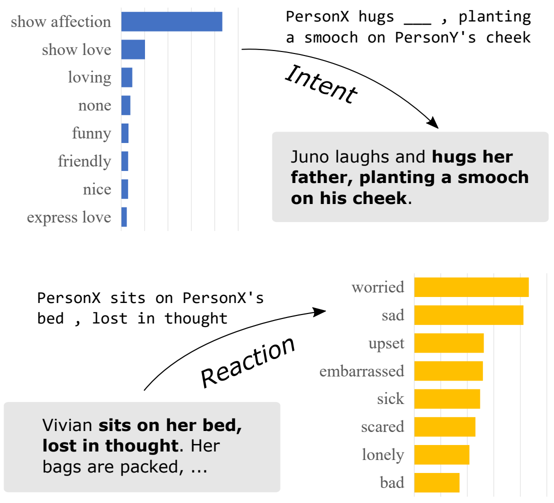


Figure 7: Two scene description snippets from *Juno* (2007, top) and *Pretty Woman* (1990, bottom), augmented with Event2mind inferences on the characters' intents and reactions. E.g., our model infers that the event `PersonX sits on PersonX's bed, lost in thought` implies that the agent, Vivian, is sad or worried.

aggregated for each character, and standardized (zero-mean and unit variance).

We compute correlations with gender for each category of intent or reaction using a logistic regression model, testing significance while using Holm's correction for multiple comparisons (Holm, 1979).[4] To account for the gender skew in scene presence (29.4% of scenes have women), we statistically control for the total number of words in a character's scene descriptions. Note that the original event phrases are all gender agnostic, as their participants have been replaced by variables (e.g., `PersonX`). We also find that the types of gender biases uncovered remain similar when we run these analyses on the human annotations or the generated words and phrases from the BiRNN with n-gram re-ranking decoding setup.

5.2 Revealing Implicit Bias via Explicit Intents and Reactions

Female: intents
AFFILIATION, FRIEND, FAMILY
BODY, SEXUAL, INGEST
SEE, INSIGHT, DISCREP
Male: intents
DEATH, HEALTH, ANGER, NEGEMO
RISK, POWER, ACHIEVE, REWARD, WORK
CAUSE, TENTATIVE‡
Female: reactions
POSEMO, AFFILIATION, FRIEND, REWARD
INGEST, SEXUAL‡, BODY‡
Male: reactions
WORK, ACHIEVE, POWER, HEALTH†
Female: others' reactions
POSEMO, AFFILIATION, FRIEND
INGEST, SEE, INSIGHT
Male: others' reactions
ACHIEVE, RISK†
SAD, NEGEMO‡, ANGER†

Table 4: Select LIWC categories correlated with gender. All results are significant when corrected for multiple comparisons at $p < 0.001$, except $^\dagger p < 0.05$ and $^\ddagger p < 0.01$.

Our Event2Mind inferences automate portrayal analyses that previously required manual annotations (Behm-Morawitz and Mastro, 2008; Prentice and Carranza, 2002; England et al., 2011). Shown in Table 4, our results indicate a gender bias in the behavior ascribed to characters, consistent with psychology and gender studies literature (Collins, 2011). Specifically, events with female semantic agents are intended to be helpful to other people (intents involving FRIEND, FAMILY, and AFFILIATION), particularly relating to eating and making food for themselves and others (INGEST, BODY). Events with male agents on the other hand are motivated by and resulting in achievements (ACHIEVE, MONEY, REWARDS, POWER).

Women's looks and sexuality are also emphasized, as their actions' intents and reactions are sexual, seen, or felt (SEXUAL, SEE, PERCEPT). Men's actions, on the other hand, are motivated by violence or fighting (DEATH, ANGER, RISK), with strong negative reactions (SAD, ANGER, NEGATIVE EMOTION).

Our approach decodes nuanced implications

and Needs', 'Personal Concerns', 'Biological Processes', 'Cognitive Processes', 'Social Words', 'Affect Words', 'Perceptual Processes'. We refer the reader to Tausczik and Pennebaker (2016) or http://liwc.wpengine.com/compare-dictionaries/ for a complete list of category descriptions.

[4] Given the data limitation, we represent gender as a binary, but acknowledge that gender is a more complex social construct.

into more explicit statements, helping to identify and explain gender bias that is prevalent in modern literature and media. Specifically, our results indicate that modern movies have the bias to portray female characters as having pro-social attitudes, whereas male characters are portrayed as being competitive or pro-achievement. This is consistent with gender stereotypes that have been studied in movies in both NLP and psychology literature (Agarwal et al., 2015; Madaan et al., 2017; Prentice and Carranza, 2002; England et al., 2011).

6 Related Work

Prior work has sought formal frameworks for inferring roles and other attributes in relation to events (Baker et al., 1998; Das et al., 2014; Schuler et al., 2009; Hartshorne et al., 2013, *inter alia*), implicitly connoted by events (Reisinger et al., 2015; White et al., 2016; Greene, 2007; Rashkin et al., 2016), or sentiment polarities of events (Ding and Riloff, 2016; Choi and Wiebe, 2014; Russo et al., 2015; Ding and Riloff, 2018). In addition, recent work has studied the patterns which evoke certain polarities (Reed et al., 2017), the desires which make events affective (Ding et al., 2017), the emotions caused by events (Vu et al., 2014), or, conversely, identifying events or reasoning behind particular emotions (Gui et al., 2017). Compared to this prior literature, our work uniquely learns to model intents and reactions over a diverse set of events, includes inference over event participants not explicitly mentioned in text, and formulates the task as predicting the textual descriptions of the implied commonsense instead of classifying various event attributes.

Previous work in natural language inference has focused on linguistic entailment (Bowman et al., 2015; Bos and Markert, 2005) while ours focuses on commonsense-based inference. There also has been inference or entailment work that is more generation focused: generating, e.g., entailed statements (Zhang et al., 2017; Blouw and Eliasmith, 2018), explanations of causality (Kang et al., 2017), or paraphrases (Dong et al., 2017). Our work also aims at generating inferences from sentences; however, our models infer implicit information about mental states and causality, which has not been studied by most previous systems.

Also related are commonsense knowledge bases (Espinosa and Lieberman, 2005; Speer and Havasi, 2012). Our work complements these existing resources by providing commonsense relations that are relatively less populated in previous work. For instance, ConceptNet contains only 25% of our events, and only 12% have relations that resemble intent and reaction. We present a more detailed comparison with ConceptNet in Appendix C.

7 Conclusion

We introduced a new corpus, task, and model for performing commonsense inference on textually-described everyday events, focusing on stereotypical intents and reactions of people involved in the events. Our corpus supports learning representations over a diverse range of events and reasoning about the likely intents and reactions of previously unseen events. We also demonstrate that such inference can help reveal implicit gender bias in movie scripts.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We also thank xlab members at the University of Washington, Martha Palmer, Tim O'Gorman, Susan Windisch Brown, Ghazaleh Kazeminejad as well as other members at the University of Colorado at Boulder for many helpful comments for our development of the annotation pipeline. This work was supported in part by National Science Foundation Graduate Research Fellowship Program under grant DGE-1256082, NSF grant IIS-1714566, and the DARPA CwC program through ARO (W911NF-15-1-0543).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Apoorv Agarwal, Jiehan Zheng, Shruti Kamath, Sriramkumar Balasubramanian, and Shirin Ann Dey. 2015. Key female characters in film have more to talk about besides men: Automating the bechdel test. In *NAACL*, pages 830–840, Denver, Colorado. Association for Computational Linguistics.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *COLING-ACL*.

Elizabeth Behm-Morawitz and Dana E Mastro. 2008. Mean girls? The influence of gender portrayals in teen movies on emerging adults' gender-based attitudes and beliefs. *Journalism & Mass Communication Quarterly*, 85(1):131–146.

Peter Blouw and Chris Eliasmith. 2018. Using neural networks to generate inferential roles for natural language. *Frontiers in Psychology*, 8:2335.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with robust logical inference. In *MLCW*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*.

Yoonjung Choi and Janyce Wiebe. 2014. +/-effectwordnet: Sense-level lexicon acquisition for opinion inference. In *EMNLP*.

Rebecca L Collins. 2011. Content analysis of gender roles in media: Where are we now and where should we go? *Sex Roles*, 64(3-4):290–298.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, pages 177–190. Springer.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Haibo Ding, Tianyu Jiang, and Ellen Riloff. 2017. Why is an event affective? Classifying affective events based on human needs. In *AAAI Workshop on Affective Content Analysis*.

Haibo Ding and Ellen Riloff. 2016. Acquiring knowledge of affective events from blogs using label propagation. In *AAAI*.

Haibo Ding and Ellen Riloff. 2018. Weakly supervised induction of affective events by optimizing semantic consistency. In *AAAI*.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *EMNLP*.

Dawn Elizabeth England, Lara Descartes, and Melissa A Collier-Meek. 2011. Gender role portrayal and the Disney princesses. *Sex roles*, 64(7-8):555–567.

José H. Espinosa and Henry Lieberman. 2005. Eventnet: Inferring temporal relations between commonsense events. In *MICAI*.

Vindu Goel and Mike Isaac. 2016. Facebook Moves to Ban Private Gun Sales on its Site and Instagram. `https://www.nytimes.com/2016/01/30/technology/facebook-gun-sales-ban.html`. Accessed: 2018-02-19.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *SEM2013*.

Andrew S Gordon and Reid Swanson. 2008. StoryUpgrade: finding stories in internet weblogs. In *ICWSM*.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *NAACL*, pages 1066–1076.

Stephan Charles Greene. 2007. Spin: Lexical semantics, transitivity, and the identification of implicit sentiment.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu, and Jiachen Du. 2017. A question answering approach for emotion cause extraction. In *EMNLP*.

Joshua K. Hartshorne, Claire Bonial, and Martha Palmer. 2013. The verbcorner project: Toward an empirically-based semantic decomposition of verbs. In *EMNLP*.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard H. Hovy. 2017. Detecting and explaining causes from text for a time series event. In *EMNLP*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *ACL*.

Nishtha Madaan, Sameep Mehta, Taneea S. Agrawaal, Vrinda Malhotra, Aditi Aggarwal, and Mayank Saxena. 2017. Analyzing gender stereotyping in bollywood movies. *CoRR*, abs/1710.04117.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? the pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38:301–333.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*.

Deborah A Prentice and Erica Carranza. 2002. What women and men should be, shouldn't be, are allowed to be, and don't have to be: The contents of prescriptive gender stereotypes. *Psychology of women quarterly*, 26(4):269–281.

Anil Ramakrishna, Victor R Martínez, Nikolaos Malandrakis, Karan Singla, and Shrikanth Narayanan. 2017. Linguistic analysis of differences in portrayal of movie characters. In *ACL*, pages 1669–1678, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation frames: A data-driven investigation. In *ACL*.

Lena Reed, JiaQi Wu, Shereen Oraby, Pranav Anand, and Marilyn A. Walker. 2017. Learning lexico-functional patterns for first-person affect. In *ACL*.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *TACL*, 3:475–488.

Irene Russo, Tommaso Caselli, and Carlo Strapparava. 2015. Semeval-2015 task 9: Clipeval implicit polarity of events. In *SemEval@NAACL-HLT*.

Maarten Sap, Marcella Cindy Prasetio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *EMNLP*, pages 2329–2334.

Karin Kipper Schuler, Anna Korhonen, and Susan Windisch Brown. 2009. Verbnet overview, extensions, mappings and applications. In *NAACL*.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*.

Yla R Tausczik and James W Pennebaker. 2016. The psychological meaning of words: LIWC and computerized text analysis methods. *J. Lang. Soc. Psychol.*

Emmett Tomai and Ken Forbus. 2010. Using narrative functions as a heuristic for relevance in story understanding. In *Proceedings of the Intelligent Narrative Technologies III Workshop*, page 9. ACM.

Hoa Trong Vu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Acquiring a dictionary of emotion-provoking events. In *EACL*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *EMNLP*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*, 3:345–358.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *TACL*, 5:379–395.

Neural Adversarial Training for Semi-supervised Japanese Predicate-argument Structure Analysis

Shuhei Kurita[†‡] **Daisuke Kawahara**[†‡] **Sadao Kurohashi**[†‡]
[†]Graduate School of Informatics, Kyoto University
[‡]CREST, JST
{kurita, dk, kuro}@nlp.ist.i.kyoto-u.ac.jp

Abstract

Japanese predicate-argument structure (PAS) analysis involves zero anaphora resolution, which is notoriously difficult. To improve the performance of Japanese PAS analysis, it is straightforward to increase the size of corpora annotated with PAS. However, since it is prohibitively expensive, it is promising to take advantage of a large amount of raw corpora. In this paper, we propose a novel Japanese PAS analysis model based on semi-supervised adversarial training with a raw corpus. In our experiments, our model outperforms existing state-of-the-art models for Japanese PAS analysis.

1 Introduction

In pro-drop languages, such as Japanese and Chinese, pronouns are frequently omitted when they are inferable from their contexts and background knowledge. The natural language processing (NLP) task for detecting such omitted pronouns and searching for their antecedents is called zero anaphora resolution. This task is essential for downstream NLP tasks, such as information extraction and summarization.

For Japanese, zero anaphora resolution is usually conducted within predicate-argument structure (PAS) analysis as a task of finding an omitted argument for a predicate. PAS analysis is a task to find an argument for each case of a predicate. For Japanese PAS analysis, the *ga* (nominative, NOM), *wo* (accusative, ACC) and *ni* (dative, DAT) cases are generally handled. To develop models for Japanese PAS analysis, supervised learning methods using annotated corpora have been applied on the basis of morpho-syntactic clues. However, omitted pronouns have few clues and thus these models try to learn relations between a predicate and its (omitted) argument from the annotated corpora. The annotated corpora consist of several tens of thousands sentences, and it is difficult to learn predicate-argument relations or selectional preferences from such small-scale corpora. The state-of-the-art models for Japanese PAS analysis achieve an accuracy of around 50% for zero pronouns (Ouchi et al., 2015; Shibata et al., 2016; Iida et al., 2016; Ouchi et al., 2017; Matsubayashi and Inui, 2017).

A promising way to solve this data scarcity problem is enhancing models with a large amount of raw corpora. There are two major approaches to using raw corpora: extracting knowledge from raw corpora beforehand (Sasano and Kurohashi, 2011; Shibata et al., 2016) and using raw corpora for data augmentation (Liu et al., 2017b).

In traditional studies on Japanese PAS analysis, selectional preferences are extracted from raw corpora beforehand and are used in PAS analysis models. For example, Sasano and Kurohashi (2011) propose a supervised model for Japanese PAS analysis based on case frames, which are automatically acquired from a raw corpus by clustering predicate-argument structures. However, case frames are not based on distributed representations of words and have a data sparseness problem even if a large raw corpus is employed. Some recent approaches to Japanese PAS analysis combines neural network models with knowledge extraction from raw corpora. Shibata et al. (2016) extract selectional preferences by an unsupervised method that is similar to negative sampling (Mikolov et al., 2013). They then use the pre-extracted selectional preferences as one of the features to their PAS analysis model. The PAS analysis model is trained by a supervised method and the selectional preference representations are fixed during training. Us-

	Predicate	NOM	ACC	DAT
(1) タクシーが$_{\mathrm{NOM}}$ 客を$_{\mathrm{ACC}}$ 駅に$_{\mathrm{DAT}}$ 送った 。 takushi-ga kyaku-wo eki-ni okutta . A taxi carried passengers to the station.	送った okutta sent/carried	タクシー takushi taxi	客 kyaku passenger	駅 eki station
(2) その 列車は 荷物を$_{\mathrm{ACC}}$ 運んだ 。 sono ressha-wa nimotsu-wo hakonda. The train also carried baggages.	運んだ hakonda carried	[列車 ressha train]	荷物 nimotsu baggage	`NULL`
(3) タクシーが$_{\mathrm{NOM}}$ 客を$_{\mathrm{ACC}}$ 乗せた とき 事故に$_{\mathrm{DAT}}$ 巻き込まれた 。 takushi-ga kyaku-wo noseta toki jiko-ni makikomareta. When the taxi picked up passengers, it was involved in the accident.	乗せた noseta picked up	タクシー takushi taxi	客 kyaku passenger	`NULL`
	巻き込まれた makikomareta was involved	[タクシー takushi taxi]	`NULL`	事故 jiko accident
(4) この 列車に は 乗れません 。 kono ressha-ni-wa noremasen. You can not take this train.	乗れません noremasen can not take	[あなた anata you]	`NULL`	列車 ressha train

Table 1: Examples of Japanese sentences and their PAS analysis. In sentence (1), case markers (が(ga), を(wo), and に(ni)) correspond to NOM, ACC, and DAT. In example (2), the correct case marker is hidden by the topic marker は (wa). In sentence (3), the NOM argument of the second predicate 巻き込まれた (was involved), is dropped. `NULL` indicates that the predicate does not have the corresponding case argument or that the case argument is not written in the sentence.

ing pre-trained external knowledge in the form of word embeddings has also been ubiquitous. However, such external knowledge is overwritten in the task-specific training.

The other approach to using raw corpora for PAS analysis is data augmentation. Liu et al. (2017b) generate pseudo training data from a raw corpus and use them for their zero pronoun resolution model. They generate the pseudo training data by dropping certain words or pronouns in a raw corpus and assuming them as correct antecedents. After generating the pseudo training data, they rely on ordinary supervised training based on neural networks.

In this paper, we propose a neural semi-supervised model for Japanese PAS analysis. We adopt neural adversarial training to directly exploit the advantage of using a raw corpus. Our model consists of two neural network models: a generator model of Japanese PAS analysis and a so-called "validator" model of the generator prediction. The generator neural network is a model that predicts probabilities of candidate arguments of each predicate using RNN-based features and a head-selection model (Zhang et al., 2017). The validator neural network gets inputs from the generator and scores them. This validator can score the generator prediction even when PAS gold labels are not available. We apply supervised learning to the generator and unsupervised learning to the entire network using a raw corpus.

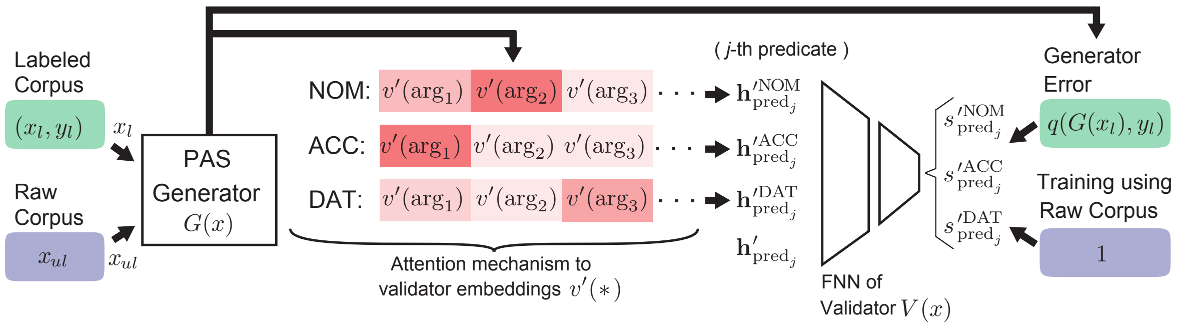
Our contributions are summarized as follows: (1) a novel adversarial training model for PAS analysis; (2) learning from a raw corpus as a source of external knowledge; and (3) as a result, we achieve state-of-the-art performance on Japanese PAS analysis.

2 Task Description

Japanese PAS analysis determines essential case roles of words for each predicate: *who* did *what* to *whom*. In many languages, such as English, case roles are mainly determined by word order. However, in Japanese, word order is highly flexible. In Japanese, major case roles are the nominative case (NOM), the accusative case (ACC) and the dative case (DAT), which roughly correspond to Japanese surface case markers: が(ga), を(wo), and に(ni). These case markers are often hidden by topic markers, and case arguments are also often omitted.

We explain two detailed tasks of PAS analysis: case analysis and zero anaphora resolution. In Table 1, we show four example Japanese sentences and their PAS labels. PAS labels are attached to nominative, accusative and dative cases of each predicate. Sentence (1) has surface case markers that correspond to argument cases.

Sentence (2) is an example sentence for case analysis. Case analysis is a task to find hidden case markers of arguments that have direct depen-



Labeled Corpus
(x_l, y_l)
x_l
Raw Corpus
x_{ul}
x_{ul}
PAS Generator $G(x)$
NOM: $v'(\mathrm{arg}_1)$ $v'(\mathrm{arg}_2)$ $v'(\mathrm{arg}_3)$ $\cdots$ $\mathbf{h}'^{\mathrm{NOM}}_{\mathrm{pred}_j}$
ACC: $v'(\mathrm{arg}_1)$ $v'(\mathrm{arg}_2)$ $v'(\mathrm{arg}_3)$ $\cdots$ $\mathbf{h}'^{\mathrm{ACC}}_{\mathrm{pred}_j}$
DAT: $v'(\mathrm{arg}_1)$ $v'(\mathrm{arg}_2)$ $v'(\mathrm{arg}_3)$ $\cdots$ $\mathbf{h}'^{\mathrm{DAT}}_{\mathrm{pred}_j}$
$\mathbf{h}'_{\mathrm{pred}_j}$
(j-th predicate)
Attention mechanism to validator embeddings $v'(*)$
FNN of Validator $V(x)$
$s'^{\mathrm{NOM}}_{\mathrm{pred}_j}$
$s'^{\mathrm{ACC}}_{\mathrm{pred}_j}$
$s'^{\mathrm{DAT}}_{\mathrm{pred}_j}$
Generator Error
$q(G(x_l), y_l)$
Training using Raw Corpus
1

Figure 1: The overall model of adversarial training with a raw corpus. The PAS generator $G(x)$ and validator $V(x)$. The validator takes inputs from the generator as a form of the attention mechanism. The validator itself is a simple feed-forward network with inputs of j-th predicate and its argument representations: $\{h'_{\mathrm{pred}_j}, h'^{\mathrm{case}_k}_{\mathrm{pred}_j}\}$. The validator returns scores for three cases and they are used for both the supervised training of the validator and the unsupervised training of the generator. The supervised training of the generator is not included in this figure.

dencies to their predicates. Sentence (2) does not have the nominative case marker が(ga). It is hidden by the topic case marker は(wa). Therefore, a case analysis model has to find the correct NOM case argument 列車(train).

Sentence (3) is an example sentence for zero anaphora resolution. Zero anaphora resolution is a task to find arguments that do not have direct dependencies to their predicates. At the second predicate "巻き込まれた"(was involved), the correct nominative argument is "タクシー"(taxi), while this does not have direct dependencies to the second predicate. A zero anaphora resolution model has to find "タクシー"(taxi) from the sentence, and assign it to the NOM case of the second predicate.

In the zero anaphora resolution task, some correct arguments are not specified in the article. This is called as *exophora*. We consider "author" and "reader" arguments as exophora (Hangyo et al., 2013). They are frequently dropped from Japanese natural sentences. Sentence (4) is an example of dropped nominative arguments. In this sentence, the nominative argument is "あなた" (you), but "あなた" (you) does not appear in the sentence. This is also included in zero anaphora resolution. Except these special arguments of exophora, we focus on intra-sentential anaphora resolution in the same way as (Shibata et al., 2016; Iida et al., 2016; Ouchi et al., 2017; Matsubayashi and Inui, 2017). We also attach `NULL` labels to cases that predicates do not have.

3 Model

3.1 Generative Adversarial Networks

Generative adversarial networks are originally proposed in image generation tasks (Goodfellow et al., 2014; Salimans et al., 2016; Springenberg, 2015). In the original model in Goodfellow et al. (2014), they propose a generator G and a discriminator D. The discriminator D is trained to devide the real data distribution $p_{data}(\mathbf{x})$ and images generated from the noise samples $\mathbf{z}^{(i)} \in \mathcal{D}_\mathbf{z}$ from noise prior $p(\mathbf{z})$. The discriminator loss is

$$\mathcal{L}_D = -\big(\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] \\ +\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]\big) \ , \quad (1)$$

and they train the discriminator by minimizing this loss while fixing the generator G. Similarly, the generator G is trained through minimizing

$$\mathcal{L}_G = \frac{1}{|\mathcal{D}_\mathbf{z}|}\sum_i \Big[\log\Big(1 - D(G(\mathbf{z}^{(i)}))\Big)\Big] \ , \quad (2)$$

while fixing the discriminator D. By doing this, the discriminator tries to descriminate the generated images from real images, while the generator tries to generate images that can deceive the adversarial discriminator. This training scheme is applied for many generative tasks including sentence generation (Subramanian et al., 2017), machine translation (Britz et al., 2017), dialog generation (Li et al., 2017), and text classification (Liu et al., 2017a).

3.2 Proposed Adversarial Training Using Raw Corpus

Japanese PAS analysis and many other syntactic analyses in NLP are not purely generative, and we can make use of a raw corpus instead of the numerical noise distribution $p(\mathbf{z})$. In this work, we use an adversarial training method using a raw corpus, combined with ordinary supervised learning using an annotated corpus. Let $\mathbf{x}_l \in \mathcal{D}_l$ indicate labeled data and $p(\mathbf{x}_l)$ indicate their label distribution. We also use unlabeled data $\mathbf{x}_{ul} \in \mathcal{D}_{ul}$ later. Our generator G can be trained by the cross entropy loss with labeled data:

$$\mathcal{L}_{G/SL} = -\mathbb{E}_{\mathbf{x}_l, y \sim p(\mathbf{x}_l)} \left[\log G(\mathbf{x}_l) \right] \quad . \qquad (3)$$

Supervised training of the generator works by minimizing this loss. Note that we follow the notations of Subramanian et al. (2017) in this subsection.

In addition, we train a so-called *validator* against the generator errors. We use the term "validator" instead of "discriminator" for our adversarial training. Unlike the discriminator that is used for dividing generated images and real images, our validator is used to score the generator results. Assume that $\mathbf{y}_l$ is the true labels and $G(\mathbf{x}_l)$ is the predicted label distribution of data $\mathbf{x}_l$ from the generator. We define the labels of the generator errors as:

$$q(G(\mathbf{x}_l), \mathbf{y}_l) = \delta_{\arg\max[G(\mathbf{x}_l)],\, \mathbf{y}_l} \quad , \qquad (4)$$

where $\delta_{i,j} = 1$ only if $i = j$, otherwise $\delta_{i,j} = 0$. This means that q is equal to 1 if the argument that the generator predicts is correct, otherwise 0. We use this generator error for training labels of the following validator. The inputs of the validator are both the generator outputs $G(\mathbf{x})$ and data $\mathbf{x} \in \mathcal{D}$. The validator can be written as $V(G(\mathbf{x}))$. The validator V is trained with labeled data $\mathbf{x}_l$ by

$$\mathcal{L}_{V/SL} = -\mathbb{E}_{\mathbf{x}_l, y \sim q(G(\mathbf{x}_l), \mathbf{y}_l)} \left[\log V(G(\mathbf{x}_l)) \right] \quad , \qquad (5)$$

while fixing the generator G. This equation means that the validator is trained with labels of the generator error $q(G(\mathbf{x}_l), \mathbf{y}_l)$.

Once the validator is trained, we train the generator with an unsupervised method. The generator G is trained with unlabeled data $\mathbf{x}_{ul} \in \mathcal{D}_{ul}$ by minimizing the loss

$$\mathcal{L}_{G/UL} = -\frac{1}{|\mathcal{D}_{ul}|} \sum_i \left[\log V(G(\mathbf{x}_{ul}^{(i)})) \right] \quad , \qquad (6)$$

while fixing the validator V. This generator training loss using the validator can be explained as follows. The generator tries to increase the validator scores to 1, while the validator is fixed. If the validator is well-trained, it returns scores close to 1 for correct PAS labels that the generator outputs, and 0 for wrong labels. Therefore, in Equation (6), the generator tries to predict correct labels in order to increase the scores of fixed validator. Note that the validator has a sigmoid function for the output of scores. Therefore output scores of the validator are in $[0, 1]$.

We first conduct the supervised training of generator network with Equation (3). After this, following Goodfellow et al. (2014), we use k-steps of the validator training and one-step of the generator training. We also alternately conduct l-steps of supervised training of the generator. The entire loss function of this adversarial training is

$$\mathcal{L} = \mathcal{L}_{G/SL} + \mathcal{L}_{V/SL} + \mathcal{L}_{G/UL} \quad . \qquad (7)$$

Our contribution is that we propose the validator and train it against the generator errors, instead of discriminating generated data from real data. Salimans et al. (2016) explore the semi-supervised learning using adversarial training for K-classes image classification tasks. They add a new class of images that are generated by the generator and classify them.

Miyato et al. (2016) propose virtual adversarial training for semi-supervised learning. They exploit unlabeled data for continuous smoothing of data distributions based on the adversarial perturbation of Goodfellow et al. (2015). These studies, however, do not use the counterpart neural networks for learning structures of unlabeled data.

In our Japanese PAS analysis model, the generator corresponds to the head-selection-based neural network for Japanese anaphora resolution. Figure 1 shows the entire model. The labeled data correspond to the annotated corpora and the labels correspond to the PAS argument labels. The unlabeled data correspond to raw corpora. We explain the details of the generator and the validator neural networks in Sec.3.3 and Sec.3.4 in turn.

3.3 Generator of PAS Analysis

The generator predicts the probabilities of arguments for each of the NOM, ACC and DAT cases of a predicate. As shown in Figure 2, the generator consists of a sentence encoder and an argument selection model. In the sentence encoder, we

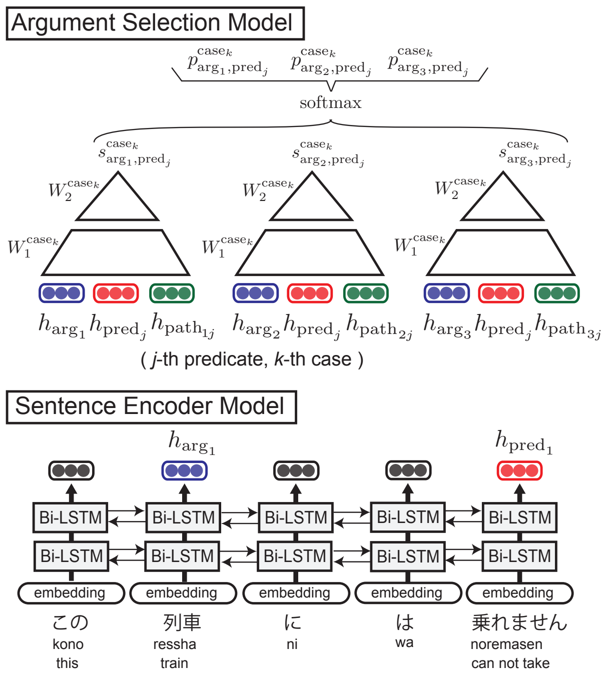


Figure 2: The generator of PAS. The sentence encoder is a three-layer bi-LSTM to compute the distributed representations of a predicate and its arguments: h_{pred_i} and h_{arg_i}. The argument selection model is two-layer feedforward neural networks to compute the scores, $s^{\text{case}_k}_{\text{arg}_i,\text{pred}_j}$, of candidate arguments for each case of a predicate.

use a three-layer bidirectional-LSTM (bi-LSTM) to read the whole sentence and extract both global and local features as distributed representations. The argument selection model consists of a two-layer feedforward neural network (FNN) and a softmax function.

For the sentence encoder, inputs are given as a sequence of embeddings $v(x)$, each of which consist of word x, its inflection from, POS and detailed POS. They are concatenated and fed into the bi-LSTM layers. The bi-LSTM layers read these embeddings in forward and backward order and outputs the distributed representations of a predicate and a candidate argument: h_{pred_j} and h_{arg_i}. Note that we also use the exophora entities, i.e., an author and a reader, as argument candidates. Therefore, we use specific embeddings for them. These embeddings are not generated by the bi-LSTM layers but are directly used in the argument selection model.

We also use path embeddings to capture a dependency relation between a predicate and its candidate argument as used in Roth and Lapata (2016). Although Roth and Lapata (2016) use a one-way LSTM layer to represent the dependency path from a predicate to its potential argument, we use a bi-LSTM layer for this purpose. We feed the embeddings of words and POS tags to the bi-LSTM layer. In this way, the resulting path embedding represents both predicate-to-argument and argument-to-predicate paths. We concatenate the bidirectional path embeddings to generate $h_{\text{path}_{ij}}$, which represents the dependency relation between the predicate j and its candidate argument i.

For the argument selection model, we apply the argument selection model (Zhang et al., 2017) to evaluate the relation between a predicate and its potential argument for each argument case. In the argument selection model, a single FNN is repeatedly used to calculate scores for a child word and its head candidate word, and then a softmax function calculates normalized probabilities of candidate heads. We use three different FNNs that correspond to the NOM, ACC and DAT cases. These three FNNs have the same inputs of the distributed representations of j-th predicate h_{pred_j}, i-th candidate argument h_{arg_i} and path embedding $h_{\text{path}_{ij}}$ between the predicate j and candidate argument i. The FNNs for NOM, ACC and DAT compute the argument scores $s^{\text{case}_k}_{\text{arg}_i,\text{pred}_j}$, where $\text{case}_k \in \{\text{NOM}, \text{ACC}, \text{DAT}\}$. Finally, the softmax function computes the probability $p(\text{arg}_i|\text{pred}_j, \text{case}_k)$ of candidate argument i for case k of j-th predicate as:

$$p(\text{arg}_i|\text{pred}_j,\text{case}_k) = \frac{\exp\left(s^{\text{case}_k}_{\text{arg}_i,\text{pred}_j}\right)}{\sum_{\text{arg}_i} \exp\left(s^{\text{case}_k}_{\text{arg}_i,\text{pred}_j}\right)}. \quad (8)$$

Our argument selection model is similar to the neural network structure of Matsubayashi and Inui (2017). However, Matsubayashi and Inui (2017) does not use RNNs to read the whole sentence. Their model is also designed to choose a case label for a pair of a predicate and its argument candidate. In other words, their model can assign the same case label to multiple arguments by itself, while our model does not. Since case arguments are almost unique for each case of a predicate in Japanese, Matsubayashi and Inui (2017) select the argument that has the highest probability for each case, even though probabilities of case arguments are not normalized over argument candidates. The model of Ouchi et al. (2017) has the same problem.

3.4 Validator

We exploit a validator to train the generator using a raw corpus. It consists of a two-layer FNN to which embeddings of a predicate and its arguments are fed. For predicate j, the input of the FNN is the representations of the predicate h'_{pred_j} and three arguments $\left\{h'^{\text{NOM}}_{\text{pred}_j}, h'^{\text{ACC}}_{\text{pred}_j}, h'^{\text{DAT}}_{\text{pred}_j}\right\}$ that are inferred by the generator. The two-layer FNN outputs three values, and then three sigmoid functions compute the scores of scalar values in a range of $[0, 1]$ for the NOM, ACC and DAT cases: $\left\{s'^{\text{NOM}}_{\text{pred}_j}, s'^{\text{ACC}}_{\text{pred}_j}, s'^{\text{DAT}}_{\text{pred}_j}\right\}$. These scores are the outputs of the validator $D(x)$. We use dropout of 0.5 at the FNN input and hidden layer.

The generator and validator networks are coupled by the attention mechanism, or the weighted sum of the validator embeddings. As shown in Equation (8), we compute a probability distribution of candidate arguments. We use the weighted sum of embeddings $v'(x)$ of candidate arguments to compute the input representations of the validator:

$$\begin{aligned} h'^{\text{case}_k}_{\text{pred}_j} &= E_{\mathbf{x}\sim p(\text{arg}_i)}[v'(\mathbf{x})] \\ &= \sum_{\text{arg}_i} p(\text{arg}_i|\text{pred}_j,\text{case}_k)v'(\text{arg}_i). \end{aligned}$$

This summation is taken over candidate arguments in the sentence and the exophora entities. Note that we use embeddings $v'(x)$ for the validator that are different from the embeddings $v(x)$ for the generator, in order to separate the computation graphs of the generator and the validator neural networks except the joint part. We use this weighted sum by the softmax outputs instead of the argmax function. This allows the backpropagation through this joint. We also feed the embedding of a predicate to the validator:

$$h'_{\text{pred}_j} = v'(\text{pred}_j). \tag{9}$$

Note that the validator is a simple neural network compared with the generator. The validator has limited inputs of predicates and arguments and no inputs of other words in sentences. This allows the generator to overwhelm the validator during the adversarial training.

Type	Value
Size of hidden layers of FNNs	1,000
Size of Bi-LSTMs	256
Dim. of word embedding	100
Dim. of POS, detailed POS, inflection form tags	10, 10, 9
Minibatch size for the generator and validator	16, 1

Table 2: Parameters for neural network structure and training.

KWDLC	# snt	# of dep	# of zero
Train	11,558	9,227	8,216
Dev.	1,585	1,253	821
Test	2,195	1,780	1,669

Table 3: KWDLC data statistics.

3.5 Implementation Details

The neural networks are trained using backpropagation. The backpropagation has been done to the word and POS tags. We use Adam (Kingma and Ba, 2015) at the initial training of the generator network for the gradient learning rule. In adversarial learning, Adagrad (Duchi et al., 2010) is suitable because of the stability of learning. We use pre-trained word embeddings from 100M sentences from Japanese web corpus by word2vec (Mikolov et al., 2013). Other embeddings and hidden weights of neural networks are randomly initialized.

For adversarial training, we first train the generator for two epochs by the supervised method, and train the validator while fixing the generator for another epoch. This is because the validator training preceding the generator training makes the validator result worse. After this, we alternately do the unsupervised training of the generator ($L_{G/UL}$), k-times of supervised training of the validator ($L_{V/SL}$) and l-times of supervised training of the generator ($L_{G/SL}$).

We use the $N(L_{G/UL})/N(L_{G/SL}) = 1/4$ and $N(L_{V/SL})/N(L_{G/SL}) = 1/4$, where $N(\cdot)$ indicates the number of sentences used for training. Also we use minibatch of 16 sentences for both supervised and unsupervised training of the generator, while we do not use minibatch for validator training. Therefore, we use $k = 16$ and $l = 4$. Other parameters are summarized in Table 2.

KWDLC	NOM	ACC	DAT
# of dep	7,224	1,555	448
# of zero	6,453	515	1,248

Table 4: KWDLC training data statistics for each case.

	Case	Zero
Ouchi+ 2015	76.5	42.1
Shibata+ 2016	89.3	53.4
Gen	91.5	56.2
Gen+Adv	**92.0**‡	**58.4**‡

Table 5: The results of case analysis (Case) and zero anaphora resolution (Zero). We use F-measure as an evaluation measure. ‡ denotes that the improvement is statistically significant at $p < 0.05$, compared with Gen using paired t-test.

4 Experiments

4.1 Experimental Settings

Following Shibata et al. (2016), we use the KWDLC (Kyoto University Web Document Leads Corpus) corpus (Hangyo et al., 2012) for our experiments.[1] This corpus contains various Web documents, such as news articles, personal blogs, and commerce sites. In KWDLC, lead three sentences of each document are annotated with PAS structures including zero pronouns. For a raw corpus, we use a Japanese web corpus created by Hangyo et al. (2012), which has no duplicated sentences with KWDLC. This raw corpus is automatically parsed by the Japanese dependency parser KNP.

We focus on intra-sentential anaphora resolution, and so we apply a preprocess to KWDLC. We regard the anaphors whose antecedents are in the preceding sentences as `NULL` in the same way as Ouchi et al. (2015); Shibata et al. (2016). Tables 3 and 4 list the statistics of KWDLC.

We use the exophora entities, i.e., an author and a reader, following the annotations in KWDLC. We also assign author/reader labels to the following expressions in the same way as Hangyo et al. (2013); Shibata et al. (2016):

author "私" (I), "僕" (I), "我々" (we), "弊社" (our company)

reader "あなた" (you), "君" (you), "客" (customer), "皆様" (you all)

Following Ouchi et al. (2015) and Shibata et al. (2016), we conduct two kinds of analysis: (1) case analysis and (2) zero anaphora resolution. Case analysis is the task to determine the correct case labels when predicates and their arguments have direct dependencies but their case markers are hidden by surface markers, such as topic markers. Zero anaphora resolution is a task to find certain case arguments that do not have direct dependencies to their predicates in the sentence.

Following Shibata et al. (2016), we exclude predicates that the same arguments are filled in multiple cases of a predicate. This is relatively uncommon and 1.5 % of the whole corpus are excluded. Predicates are marked in the gold dependency parses. Candidate arguments are just other tokens than predicates. This setting is also the same as Shibata et al. (2016).

All performances are evaluated with micro-averaged F-measure (Shibata et al., 2016).

4.2 Experimental Results

We compare two models: the supervised generator model (Gen) and the proposed semi-supervised model with adversarial training (Gen+Adv). We also compare our models with two previous models: Ouchi et al. (2015) and Shibata et al. (2016), whose performance on the KWDLC corpus is reported.

Table 5 lists the experimental results. Our models (Gen and Gen+Adv) outperformed the previous models. Furthermore, the proposed model with adversarial training (Gen+Adv) was significantly better than the supervised model (Gen).

4.3 Comparison with Data Augmentation Model

We also compare our GAN-based approach with data augmentation techniques. A data augmentation approach is used in Liu et al. (2017b). They automatically process raw corpora and make drops of words with some rules. However, it is difficult to directly apply their approach to Japanese PAS analysis because Japanese zero-pronoun depends on dependency trees. If we make some drops of arguments of predicates in sentences, this can cause lacks of nodes in dependency trees. If we prune some branches of dependency trees of the sentence, this cause the data bias problem.

[1] The KWDLC corpus is available at http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KWDLC

	Case analysis			Zero anaphora resolution		
Model	NOM	ACC	DAT	NOM	ACC	DAT
Ouchi+ 2015	87.4	40.2	27.6	48.8	0.0	10.7
Shibata+ 2016	94.1	75.6	30.0	57.7	17.3	37.8
Gen	**95.3**	83.6	39.7	60.7	30.4	41.2
Gen+Adv	**95.3**	**85.4**	**51.5**	**62.3**	**31.1**	**44.6**

Table 6: The detailed results of case analysis and zero anaphora resolution for the NOM, ACC and DAT cases. Our models outperform the existing models in all cases. All values are evaluated with F-measure.

	Case	Zero
Gen	91.5	56.2
Gen+Aug	91.2	57.0
Gen+Adv	**92.0**‡	**58.4**‡

Table 7: The comparisons of Gen+Adv with Gen and the data augmentation model (Gen+Aug). ‡ denotes that the improvement is statistically significant at $p < 0.05$, compared with Gen+Aug.

Therefore we use existing training corpora and word embeddings for the data augmentation. First we randomly choose an argument word w in the training corpus and then swap it with another word w' with the probability of $p(w, w')$. We choose top-20 nearest words to the original word w in the pre-trained word embedding as candidates of swapped words. The probability is defined as $p(w, w') \propto [v(w)^\top v(w')]^r$, where $r = 10$. This probability is normalized by top-20 nearest words. We then merge this pseudo data and the original training corpus and train the model in the same way with the Gen model. We conducted several experiments and found that the model trained with the same amount of the pseudo data as the training corpus achieved the best result.

Table 7 shows the results of the data augmentation model and the GAN-based model. Our Gen+Adv model performs better than the data augmented model. Note that our data augmentation model does not use raw corpora directly.

4.4 Discussion

4.4.1 Result Analysis

We report the detailed performance for each case in Table 6. Among the three cases, zero anaphora resolution of the ACC and DAT cases is notoriously difficult. This is attributed to the fact that these ACC and DAT cases are fewer than the NOM case in the corpus as shown in Table 4. However, we can see that our proposed model, Gen+Adv, performs much better than the previous models especially for the ACC and DAT cases. Although the number of training instances of ACC and DAT is much smaller than that of NOM, our semi-supervised model can learn PAS for all three cases using a raw corpus. This indicates that our model can work well in resource-poor cases.

We analyzed the results of Gen+Adv by comparing with Gen and the model of Shibata et al. (2016). Here, we focus on the ACC and DAT cases because their improvements are notable.

- “パックは 洗って、 分別して リサイクルに 出さなきゃいけないので 手間がかかる。“
  It is bothersome to wash, classify and recycle spent packs.

In this sentence, the predicates “洗って” (wash), “分別して” (classify), “(リサイクルに) 出す” (recycle) takes the same ACC argument, “パック” (pack). This is not so easy for Japanese PAS analysis because the actual ACC case marker “を” (wo) of “パック” (pack) is hidden by the topic marker “は” (wa). The Gen+Adv model can detect the correct argument while the model of Shibata et al. (2016) fails. In the Gen+Adv model, each predicate gives a high probability to “パック” (pack) as an ACC argument and finally chooses this. We found many examples similar to this and speculate that our model captures a kind of selectional preferences.

The next example is an error of the DAT case by the Gen+Adv model.

- “各専門分野も お任せ下さい。”
  please leave every professional field (to ϕ)

The gold label of this DAT case (to ϕ) is NULL because this argument is not written in the sentence.

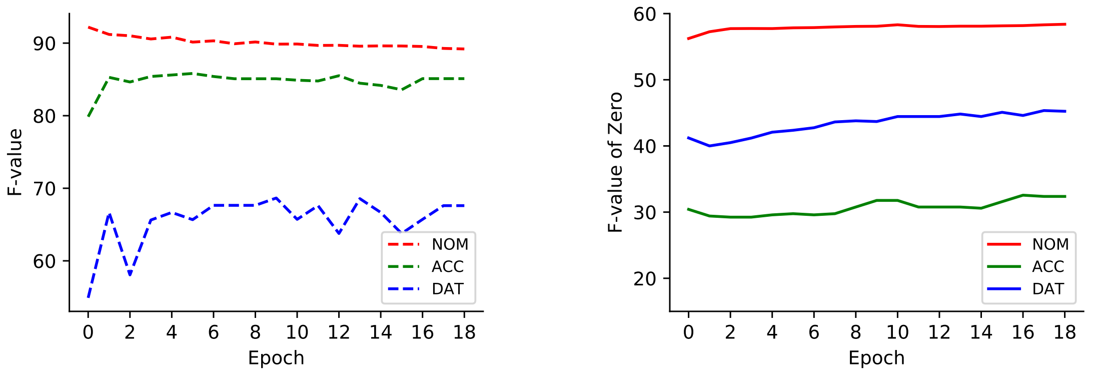


Figure 3: Left: validator scores with the development set during adversarial training epochs. Right: generator scores for Zero with the development set during adversarial training epochs.

However, the Gen+Adv model judged the DAT argument as "author". Although we cannot specify ϕ as "author" only from this sentence, "author" is a possible argument depending on the context.

4.4.2 Validator Analysis

We also evaluate the performance of the validator during the adversarial training with raw corpora. Figure 3 shows the validator performance and the generator performance of Zero on the development set. The validator score is evaluated with the outputs of generator.

We notice that the NOM case and the other two cases have different curves in both graphs. This can be explained by the speciality of the NOM case. The NOM case has much more author/reader expressions than the other cases. The prediction of author/reader expressions depends not only on selectional preferences of predicates and arguments but on the whole of sentences. Therefore the validator that relies only on predicate and argument representations cannot predict author/reader expressions well.

In the ACC and DAT cases, the scores of the generator and validator increase in the first epochs. This suggests that the validator learns the weakness of the generator and vice versa. However, in later epochs, the scores of the generator increase with fluctuation, while the scores of the validator saturates. This suggests that the generator gradually becomes stronger than the validator.

5 Related Work

Shibata et al. (2016) proposed a neural network-based PAS analysis model using local and global features. This model is based on the non-neural model of Ouchi et al. (2015). They achieved state-of-the-art results on case analysis and zero anaphora resolution using the KWDLC corpus. They use an external resource to extract selectional preferences. Since our model uses an external resource, we compare our model with the models of Shibata et al. (2016) and Ouchi et al. (2015).

Ouchi et al. (2017) proposed a semantic role labeling-based PAS analysis model using Grid-RNNs. Matsubayashi and Inui (2017) proposed a case label selection model with feature-based neural networks. They conducted their experiments on NAIST Text Corpus (NTC) (Iida et al., 2007, 2016). NTC consists of newspaper articles, and does not include the annotations of author/reader expressions that are common in Japanese natural sentences.

6 Conclusion

We proposed a novel Japanese PAS analysis model that exploits a semi-supervised adversarial training. The generator neural network learns Japanese PAS and selectional preferences, while the validator is trained against the generator errors. This validator enables the generator to be trained from raw corpora and enhance it with external knowledge. In the future, we will apply this semi-supervised training method to other NLP tasks.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR1301, Japan and JST ACT-I Grant Number JPMJPR17U8, Japan.

References

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark, pages 118–126. http://www.aclweb.org/anthology/W17-4712.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. UCB/EECS-2010-24.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 2672–2680.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*. Faculty of Computer Science, Universitas Indonesia, Bali,Indonesia, pages 535–544.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 924–934.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop, LAW@ACL 2007, Prague, Czech Republic, June 28-29, 2007*. pages 132–139.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1244–1254. https://aclweb.org/anthology/D16-1132.

D. P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2157–2169.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1–10.

Ting Liu, Yiming Cui, Qingyu Yin, Wei-Nan Zhang, Shijin Wang, and Guoping Hu. 2017b. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 102–111.

Yuichiroh Matsubayashi and Kentaro Inui. 2017. Revisiting the design issues of local models for japanese predicate-argument structure analysis. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 128–133.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. volume abs/1301.3781. http://arxiv.org/abs/1301.3781.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing by virtual adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for japanese predicate argument structure analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 961–970.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Neural modeling of multi-predicate interactions for japanese predicate argument structure analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1591–1600.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers). Association for Computational Linguistics, Berlin, Germany, pages 1192–1202.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. 2016. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pages 2234–2242.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 758–766.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural network-based model for japanese predicate argument structure analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1235–1244.

Jost Tobias Springenberg. 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks.

Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. 2017. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Vancouver, Canada, pages 241–251.

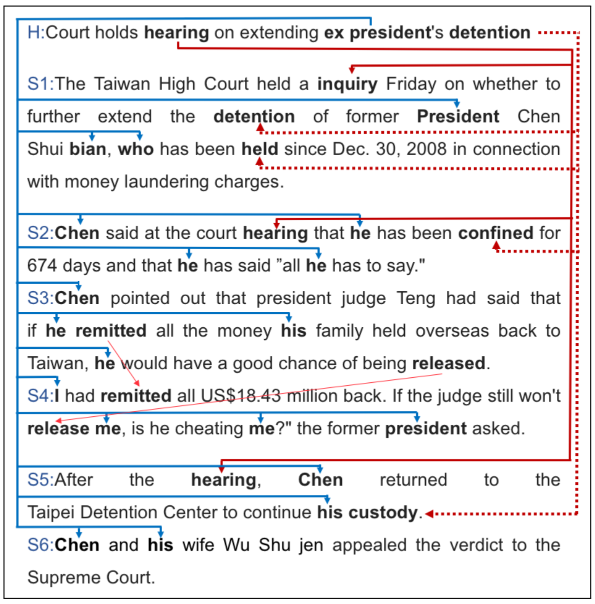
Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 665–676.

Improving Event Coreference Resolution by Modeling Correlations between Event Coreference Chains and Document Topic Structures

Prafulla Kumar Choubey and Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
`(prafulla.choubey, huangrh)@tamu.edu`

Abstract

This paper proposes a novel approach for event coreference resolution that models correlations between event coreference chains and document topical structures through an Integer Linear Programming formulation. We explicitly model correlations between the main event chains of a document with topic transition sentences, inter-coreference chain correlations, event mention distributional characteristics and sub-event structure, and use them with scores obtained from a local coreference relation classifier for jointly resolving multiple event chains in a document. Our experiments across KBP 2016 and 2017 datasets suggest that each of the structures contribute to improving event coreference resolution performance.


H:Court holds **hearing** on extending **ex president**'s **detention**

S1:The Taiwan High Court held a **inquiry** Friday on whether to further extend the **detention** of former **President** Chen Shui **bian**, **who** has been **held** since Dec. 30, 2008 in connection with money laundering charges.

S2:**Chen** said at the court **hearing** that **he** has been **confined** for 674 days and that **he** has said "all **he** has to say."

S3:**Chen** pointed out that president judge Teng had said that if **he remitted** all the money **his** family held overseas back to Taiwan, **he** would have a good chance of being **released**.

S4:**I** had **remitted** all US$18.43 million back. If the judge still won't **release me**, is he cheating **me**?" the former **president** asked.

S5:After the **hearing**, **Chen** returned to the Taipei Detention Center to continue **his custody**.

S6:**Chen** and **his** wife Wu Shu jen appealed the verdict to the Supreme Court.

Figure 1: An example document to illustrate the characteristics of event (red) and entity (blue) coreference chains.

1 Introduction

Event coreference resolution aims to identify and link event mentions in a document that refer to the same real-world event, which is vital for identifying the skeleton of a story and text understanding and is beneficial to numerous other NLP applications such as question answering and summarization. In spite of its importance, compared to considerable research for resolving coreferential entity mentions, far less attention has been devoted to event coreference resolution. Event coreference resolution thus remained a challenging task and the best performance remained low.

Event coreference resolution presents unique challenges. Compared to entities, coreferential event mentions are fewer in a document and much more sparsely scattered across sentences. Figure 1 shows a typical news article. Here, the main entity, "President Chen", appears frequently in every sentence, while the main event "hearing" and its accompanying event "detention" are mentioned much less frequently. If we look more closely, referring back to the same entity serves a different purpose than referring to the same event. The protagonist entity of a story is involved in many events and relations; thus, the entity is referred back each time such an event or relation is described. In this example, the entity was mentioned when describing various events he participated or was involved in, including "detention", "said", "pointed out", "remitted", "have a chance", "release", "cheating", "asked" and "returned", as well as when describing several relations involving him, including "former president", "his family" and "his wife". In contrast, most events only appear once in a text, and there is less motivation to repeat them: a story is mainly formed by a se-

Dataset	Type	0	1	2	3	4	> 4
richERE	event	11	**34**	**20**	**9**	**7**	**19**
	entity	**34**	33	14	6	3	10
ACE-05	event	5	**33**	**19**	**10**	**9**	**24**
	entity	**37**	28	12	7	4	13
KBP 2015	event	15	**34**	**12**	**9**	**6**	**24**
KBP 2016	event	8	**43**	**15**	**7**	**6**	**21**
KBP 2017	event	12	**49**	**13**	**7**	**4**	**15**

Table 1: Percentages of adjacent (event vs. entity) mention pairs based on the number of sentences between two mentions.

ries of related but different events. Essentially, (1) the same event is referred back only when a new aspect or further information of the event has to be described, and (2) repetitions of the same events are mainly used for content organization purposes and, consequently, correlate well with topic structures.

Table 1 further shows the comparisons of positional patterns between event coreference and entity coreference chains, based on two benchmark datasets, ERE (Song et al., 2015) and ACE05 (Walker et al., 2006), where we paired each event (entity) mention with its nearest antecedent event (entity) mention and calculated the percentage of (event vs. entity) coreferent mention pairs based on the number of sentences between two mentions. Indeed, for entity coreference resolution, centering and nearness are striking properties (Grosz et al., 1995), and the nearest antecedent of an entity mention is mostly in the same sentence or in the immediately preceding sentence (70%). This is especially true for nominals and pronouns, two common types of entity mentions, where the nearest preceding mention that is also compatible in basic properties (e.g., gender, person and number) is likely to co-refer with the current mention. In contrast, coreferential event mentions are rarely from the same sentence (10%) and are often sentences apart. The sparse distribution of coreferent event mentions also applies to the three KBP corpora used in this work.

To address severe sparsity of event coreference relations in a document, we propose a holistic approach to identify coreference relations between event mentions by considering their correlations with document topic structures. Our key observation is that event mentions make the backbone of a document and coreferent mentions of the same event play a key role in achieving a coherent content structure. For example, in figure 1, the events "hearing" and "detention" were mentioned in the headline (H), in the first sentence (S1) as a story overview, in the second sentence (S2) for transitioning to the body section of the story describing what happened during the hearing, and then in the fifth sentence (S5) for transitioning to the ending section of the story describing what happened after the hearing. By attaching individual event mentions to a coherent story and its topic structures, our approach recognizes event coreference relations that are otherwise not easily seen due to a mismatch of two event mentions' local contexts or long distances between event mentions.

We model several aspects of correlations between event coreference chains and document level topic structures, in an Integer Linear Programming (ILP) joint inference framework. Experimental results on the benchmark event coreference resolution dataset KBP-2016 (Ellis et al., 2016) and KBP 2017 (Getman et al., 2017) show that the ILP system greatly improves event coreference resolution performance by modeling different aspects of correlations between event coreferences and document topic structures, which outperforms the previous best system on the same dataset consistently across several event coreference evaluation metrics.

2 Correlations between Event Coreference Chains and Document Topic Structures

We model four aspects of correlations.

Correlations between Main Event Chains and Topic Transition Sentences: the main events of a document, e.g., "hearing" and "detention" in this example 1, usually have multiple coreferent event mentions that span over a large portion of the document and align well with the document topic layout structure (Choubey et al., 2018). While fine-grained topic segmentation is a difficult task in its own right, we find that topic transition sentences often overlap in content (for reminding purposes) and can be identified by calculating sentence similarities. For example, sentences S1, S2 and S5 in Figure 1 all mentioned the two main events and the main entity "President Chen". We, therefore, encourage coreference links between event mentions that appear in topic transition sentences by designing constraints in ILP and modifying the objective function. In addition, to avoid fragmented partial event chains and recover complete chains for the main events, we also encourage associating more coreferent event mentions to a chain that has a large stretch (the number of sentences between the first and the last event mention based on their textual positions).

Correlations across Semantically Associated Event Chains: semantically associated events often co-occur in the same sentence. For example, mentions of the two main events "hearing" and "detention" co-occur across the document in sentences H, S1, S2 and S5. The correlation across event chains is not specific to global main events, for example, the local events "remitted" and "release" have their mentions co-occur in sentences S3 and S4 as well. In ILP, we leverage this observation and encourage creating coreference links between event mentions in sentences that contain other already known coreferent event mentions.

Genre-specific Distributional Patterns: we model document level distributional patterns of coreferent event mentions that may be specific to a genre in ILP. Specifically, news article often begins with a summary of the overall story and then introduces the main events and their closely associated events. In subsequent paragraphs, detailed information of events may be introduced to provide supportive evidence to the main story. Thereby, a majority of event coreference chains tend to be initiated in the early sections of the document. Event mentions in the later paragraphs may exist as coreferent mentions of an established coreference chain or as singleton event mentions which, however, are less likely to initiate a new coreference chain. Inspired by this observation, we simply modify the objective function of ILP to encourage more event coreference links in early sections of a document.

Subevents: subevents exist mainly to provide details and evidence for the parent event, therefore, the relation between subevents and their parent event presents another aspect of correlations between event relations and hierarchical document topic structures. Subevents may share the same lexical form as the parent event and cause spurious event coreference links (Araki et al., 2014). We observe that subevents referring to specific actions were seldomly referred back in a document and are often singleton events. Following the approach proposed by (Badgett and Huang, 2016), we identify such specific action events and improve event coreference resolution by specifying constraints in ILP to discourage coreference links between a specific action event and other event mentions.

3 Related Work

Compared to entity coreference resolution (Lee et al., 2017; Clark and Manning, 2016a,b; Martschat and Strube, 2015; Lee et al., 2013), far less research was conducted for event coreference resolution. Most existing methods (Ahn, 2006; Chen et al., 2009; Cybulska and Vossen, 2015a,b) heavily rely on surface features, mainly event arguments (i.e., entities such as event participants, time, location, etc.) that were extracted from local contexts of two events, and determine that two events are coreferential if their arguments match. Often, a clustering algorithm, hierarchical Bayesian (Bejan and Harabagiu, 2010, 2014; Yang et al., 2015) or spectral clustering algorithms (Chen and Ji, 2009), is applied on top of a pairwise surface feature based classifier for inducing event clusters. However, identifying potential arguments, linking arguments to a proper event mention, and recognizing compatibilities between arguments are all error-prone (Lu et al., 2016). Joint event and entity coreference resolution (Lee et al., 2012), joint inferences of event detection and event coreference resolution (Lu and Ng, 2017), and iterative information propagation (Liu et al., 2014; Choubey and Huang, 2017a) have been proposed to mitigate argument mismatch issues.

However, such methods are incapable of handling more complex and subtle cases, such as partial event coreference with incompatible arguments (Choubey and Huang, 2017a) and cases lacking informative local contexts. Consequently, many event coreference links were missing and the resulted event chains are fragmented. The low performance of event coreference resolution limited its uses in downstream applications. (**?**) shows that instead of human annotated event coreference relations, using system predicted relations resulted in a significant performance reduction in identifying the central event of a document. Moreover, the recent research by Moosavi and Strube (2017) found that the extensive use of lexical and surface features biases entity coreference resolvers towards seen mentions and do not generalize to unseen domains, and the finding can perfectly apply to event coreference resolution. Therefore, we propose to improve event coreference resolution by modeling correlations between event coreferences and the overall topic structures of a document, which is more likely to yield robust and generalizable event coreference resolvers.

4 Modeling Event Coreference Chain - Topic Structure Correlations Using Integer Linear Programming

We model discourse level event-topic correlation structures by formulating the event coreference resolution task as an Integer Linear Programming (ILP) problem. Our baseline ILP system is defined over pairwise scores between event mentions obtained from a pairwise neural network-based coreference resolution classifier.

4.1 The Local Pairwise Coreference Resolution Classifier

Our local pairwise coreference classifier uses a neural network model based on features defined for an event mention pair. It includes a common layer with 347 neurons shared between two event mentions to generate embeddings corresponding to word lemmas (300) and parts-of-speech (POS) tags (47). The common layer aims to enrich event word embeddings with the POS tags using the shared weight parameters. It also includes a second layer with 380 neurons to embed suffix[1] and prefix [2] of event words, distances (euclidean, absolute and cosine) between word embeddings of two event lemmas and common arguments between two event mentions. The output from the second layer is concatenated and fed into the third neural layer with 10 neurons. The output embedding from the third layer is finally fed into an output layer with 1 neuron that generates a score indicating the confidence of assigning the given event pair to the same coreference cluster. All three layers and the output layer use the sigmoid activation function.

4.2 The Basic ILP for Event Coreference Resolution

Let λ represents the set of all event mentions in a document, Λ denotes the set of all event mention pairs i.e. $\Lambda = \{< i,j > \mid < i,j > \in \lambda \times \lambda \; and \; i < j\}$ and $p_{ij} = p_{cls}(coref|i,j)$ represents the cost of assigning event mentions i and j to the same coreferent cluster, we can formulate the baseline objective function that minimizes equation 1. Further we add constraints (equation 2) over each triplets of mentions to enforce transitivity (Denis et al., 2007; Finkel and Manning, 2008). This guarantees legal clustering by ensuring that $x_{ij} = x_{jk} = 1$ implies x_{ik} = 1.

$$\Theta_B = \sum_{i,j \in \Lambda} -log(p_{ij})x_{ij} - log(1-p_{ij})(\neg x_{ij}) \quad s.t. \; x_{ij} \in \{0,1\} \tag{1}$$

$$\neg x_{ij} + \neg x_{jk} \geq \neg x_{ik} \tag{2}$$

We then add constituent objective functions and constraints to the baseline ILP formulation to induce correlations between coreference chains and topical structures (Θ_T), discourage fragmented chains (Θ_G), encourage semantic associations among chains (Θ_C), model genre-specific distributional patterns (Θ_D) and discourage subevents from having coreferent mentions (Θ_S). They are described in the following subsections.

4.2.1 Modeling the Correlation between Main Event Chains and Topic Transition Sentences

As shown in the example Figure 1, main events are likely to have mentions appear in topic transition sentences. Therefore, We add the following objective function (equation 3) to the basic objective function and add the new constraint 4 in order to encourage coreferent event mentions to occur in topic transition sentences.

$$\Theta_T = \sum_{m,n \in \Omega} -log(s_{mn})w_{mn} - log(1-s_{mn})(\neg w_{mn}) \quad s.t. \; w_{mn} \in \{0,1\}, \; (n-m) \geq |S|/\theta_s \tag{3}$$

$$\sum_{i' \in \xi_m, j' \in \xi_n} x_{i'j'} \geq w_{mn} \tag{4}$$

Specifically, let ω represents the set of sentences in a document and Ω denotes the set of sentence pairs i.e. $\Omega = \{< m,n > \mid < m,n > \in \omega \times \omega$ and $m < n\}$. Then, let $s_{ij} = p_{sim}(simscore|m,n)$, which represents the similarity score between sentences m and n and |S| equals to the number of sentences in a given document. Here, the indicator variable w_{mn} indicates if the two sentences m and n are topic transition sentences. Essentially, when two sentences have a high similarity score (> 0.5) and are not near (with $|S|/\theta_s$or more sentences

[1] te, tor, or, ing, cy, id, ed, en, er, ee, pt, de, on, ion, tion, ation, ction, de, ve, ive, ce, se, ty, al, ar, ge, nd, ize, ze, it, lt

[2] re, in, at, tr, op

apart, in our experiments we set θ_s to 5), this objective function Θ_T tries to set the corresponding indicator variable w_{mn} to 1. Then, we add constraint 4 to encourage coreferent event mentions to occur in topic transition sentences. Note that ξ_m refers to all the event mentions in sentence m, and x_{ij} is the indicator variable which is set to 1 if event mentions defined by index i and j are coreferent. Thus, the above constraint ensures that two topic transition sentences contain at least one coreferent event pair.

Identifying Topic Transition Sentences Using Sentence Similarities: First, we use the unsupervised method based on weighted word embedding average proposed by Arora et al. (2016) to obtain sentence embeddings. We first compute the weighted average of words' embeddings in a sentence, where the weight of a word w is given by $a/(a+p(w))$. Here, p(w) represents the estimated word frequency obtained from English Wikipedia and a is a small constant (1e-5). We then compute the first principal component of averaged word embeddings corresponding to sentences in a document and remove the projection on the first principal component from each averaged word embedding for each sentence.

Then using the resulted averaged word embedding as the sentence embedding, we compute the similarity between two sentences as cosine similarity between their embeddings. We particularly choose this simple unsupervised model to reduce the reliance on any additional corpus for training a new model for calculating sentence similarities. This model was found to perform comparably to supervised RNN-LSTM based models for the semantic textual similarity task.

Constraints for Avoiding Fragmented Partial Event Chains: The above equations (3-4) consider a pair of sentences and encourage two coreferent event mentions to appear in a pair of topic transition sentences. But the local nature of these constraints can lead to fragmented main event chains. Therefore, we further model the distributional characteristics of global event chains and encourage the main event chains to have a large number of coreferential mentions and a long stretch (the number of sentences that are present in between the first and last event mention of a chain), to avoid creating partial chains. Specifically, we add the following objective function (equation 5) and the new constraints (equation 6 and 7):

$$\Theta_G = -\sum_{i,j\in\mu} \gamma_{ij} \tag{5}$$

$$\begin{gathered}\sigma_{ij} = \sum_{k<i} \neg x_{ki} \wedge \sum_{j<l} \neg x_{jl} \wedge x_{ij} \\ \sigma_{ij} \in \{0,1\}\end{gathered} \tag{6}$$

$$\begin{gathered}\Gamma_i = \sum_{k,i\in\Lambda} x_{ki} + \sum_{i,j\in\Lambda} x_{ij} \\ M(1-y_{ij}) \geq (\varphi[j]-\varphi[i]).\sigma_{ij} - \lceil 0.75\,(|S|)\rceil \\ \gamma_{ij} - \Gamma_i - \Gamma_j \geq M.y_{ij} \\ \Gamma_i, \Gamma_j, \gamma_{ij} \in Z;\ \Gamma_i, \Gamma_j, \gamma_{ij} \geq 0;\ y_{ij} \in \{0,1\}\end{gathered} \tag{7}$$

First, we define an indicator variable σ_{ij} by equation 6 [3], corresponding to each event mention pair, that takes value 1 if (1) the event mentions at index i and j are coreferent; (2) the event mention at index i doesn't corefer to any of the mentions preceding it; and (3) mention at index j doesn't corefer to any event mention following it. Essentially, setting σ_{ij} to 1 defines an event chain that starts from the event mention i and ends at the event mention j.

Then with equation 7, variable σ_{ij} is used to identify main event chains as those chains which are extended to at least 75% of the document. When a chain is identified as a global chain, we encourage it to have more coreferential mentions. Here, Γ_i (Γ_j) equals the sum of indicator variables x corresponding to event pairs that include the event mention at index i (j) i.e. the number of mentions that are coreferent to i (j), $\varphi[i]$ ($\varphi[j]$) represents the sentence number of event mention i (j), M is a large positive number and y_{ij} represents a slack variable that takes the value 0 if the event chain represented by σ_{ij} is a global chain. Given $\sigma_{i,j}$ is identified as a global chain, variable γ_{ij} equals the sum of variables Γ_i and Γ_j and is used in the objective function Θ_G (equation 5) to encourage more coreferential mentions.

[3] Equation 6 can be implemented as

$$n_p + n_s \leq \sum_{k<i} x_{ki} + \sum_{j<l} x_{jl} - x_{ij} + (n_p + n_s + 1).\sigma_{ij}$$
$$\sum_{k<i} x_{ki} + \sum_{j<l} x_{jl} - x_{ij} + (n_p + n_s + 1).\sigma_{ij} \geq 0$$

where n_p, n_s represent the number of event mentions preceding event mention i and the number of event mentions following event mention j respectively.

4.2.2 Cross-chain Inferences

As illustrated through Figure 1, semantically related events tend to have their mentions co-occur within the same sentence. So, we define the objective function (equation 8) and constraints (9) to favor a sentence with a mention from one event chain to also contain a mention from another event chain, if the two event chains are known to have event mentions co-occur in several other sentences.

$$\Theta_C = -\sum_{m,n\in\Omega} \Phi_{mn} \tag{8}$$

$$\Phi_{mn} = \sum_{i\in\xi_m, j\in\xi_n} x_{ij}$$
$$|\xi_m| > 1;\ |\xi_n| > 1;\ \Phi_{mn} \in Z;\ \Phi_{mn} \geq 0 \tag{9}$$

To do so, we first define a variable ϕ_{mn} that equals the number of coreferent event pairs in a sentence pair, with each sentence having more than one event mention. We then define Θ_C to minimize the negative sum of ϕ_{mn}. Following the previous notations, ξ_m in the above equation represents the event mentions in sentence m.

4.2.3 Modeling Segment-wise Distributional Patterns

The position of an event mention in a document has a direct influence on event coreference chains. Event mentions that occur in the first few paragraphs are more likely to initiate an event chain. On the other hand, event mentions in later parts of a document may be coreferential with a previously seen event mention but are extremely unlikely to begin a new coreference chain. This distributional association is even stronger in the journalistic style of writing. We model this through a simple objective function and constraints (equation 10).

$$\begin{aligned}\Theta_D = -\sum_{i\in\xi_m, j\in\xi_n} x_{ij} + \sum_{k\in\xi_p, l\in\xi_q} x_{kl}\\ s.t.\ m,n < \lfloor\alpha|S|\rfloor;\ p,q > \lceil\beta|S|\rceil\\ \alpha\in[0,1];\ \beta\in[0,1]\end{aligned} \tag{10}$$

Specifically, for the event pairs that belong to the first α (or the last β) sentences in a document, we add the negative (positive) sum of their indicator variables (x) in objective function Θ_D.

The equation 10 is meant to inhibit coreference links between event mentions that exist within the latter half of document. They do not influence the links within event chains that start early and extend till the later segments of the document. It is also important to understand that position-based features used in entity coreference resolution (Haghighi and Klein, 2007) are usually defined for an entity pair. However, we model the distributional patterns of an event chain in a document.

4.2.4 Restraining Subevents from Being Included in Coreference Chains

Subevents are known to be a major source of false coreference links due to their high surface similarity with their parent events. Therefore, we discourage subevents from being included in coreference chains in our model and modify the global optimization goal by adding a new objective function (equation 11).

$$\Theta_S = \sum_{s\in\mathbb{S}} \Gamma_s \tag{11}$$

where $\mathbb{S}$ represents the set of subevents in a document. We define the objective function Θ_S as the sum of Γ_s, where Γ_s equals the number of mentions that are coreferent to s. Then our goal is to minimize Θ_S and restrict the subevents from being included in coreference chains.

We identify probable subevents by using surface syntactic cues corresponding to identifying a sequence of events in a sentence (Badgett and Huang, 2016). In particular, a sequence of two or more verb event mentions in a conjunction structure are extracted as subevents.

4.3 The full ILP Model and the Parameters

The equations 3-11 model correlations between non-local structures within or across event chains and document topical structures. We perform ILP inference for coreference resolution by optimizing a global objective function(Θ), defined in equation 12, that incorporates prior knowledge by means of hard or soft constraints.

$$\Theta = \kappa_B\Theta_B + \kappa_T\Theta_T + \kappa_G\Theta_G + \kappa_C\Theta_C + \kappa_D\Theta_D + \kappa_S\Theta_S \tag{12}$$

Here, all the κ parameters are floating point constants. For the sake of simplicity, we set κ_B and κ_T to 1.0 and $\kappa_G = \kappa_C$. Then we estimate the parameters $\kappa_G(\kappa_C)$ and κ_D through 2-d grid search in range [0, 5.0] at the interval of 0.5 on a held out training data. We found that the best performance was obtained for $\kappa_C = \kappa_G = 0.5$ and $\kappa_D = 2.5$. Since, Θ_S aims to inhibit subevents from being included in coreference chains, we set a high value for κ_S and found that, indeed, the performance remained same for all the values of κ_S in range [5.0,15.0]. In our final model, we keep κ_S = 10.0. Also, we found that the performance is roughly invariant to the parameters κ_G and κ_C if they are set to values between 0.5 and 2.5.

In our experiments, we process each document to define a distinct ILP problem which is solved using the PuLP library (Mitchell et al., 2011).

5 Evaluation

5.1 Experimental Setup

We trained our ILP system on the KBP 2015 (Ellis et al., 2015) English dataset and evaluated the system on KBP 2016 and KBP 2017 English datasets[4]. All the KBP corpora include documents from both discussion forum[5] and news articles. But as the goal of this study is to leverage discourse level topic structure in a document for improving event coreference resolution performance, we only evaluate the ILP system using regular documents (news articles) in the KBP corpora. Specifically, we train our event extraction system and local coreference resolution classifier on 310 documents from the KBP 2015 corpus that consists of both discussion forum documents and news articles, tune the hyper-parameters corresponding to ILP using 50 news articles[6] from the KBP 2015 corpus and evaluate our system on news articles from the official KBP 2016 and 2017 evaluation corpora[7] respectively. For direct comparisons, the results reported for the baselines, including the previous state-of-the-art model, were based on news articles in the test datasets as well.

We report the event coreference resolution results based on the version 1.8 of the official KBP 2017 scorer. The scorer employs four coreference scoring measures, namely B^3 (Bagga and Baldwin, 1998), $CEAF_e$ (Luo, 2005), MUC (Vilain et al., 1995) and BLANC (Recasens and Hovy, 2011) and the unweighted average of their F1 scores (AVG_{F1}).

5.2 Event Mention Identification

	Lu and Ng (2017)		Ours	
Corpus	Untyped	Typed	Untyped	Typed
KBP 2016	60.13	49.00	60.03	45.45
KBP 2017	-	-	62.89	49.34

Table 2: F1 scores for event mention extraction on the KBP 2016 and 2017 corpus

We use an ensemble of multi-layer feed forward neural network classifiers to identify event mentions (Choubey and Huang, 2017b). All basic classifiers are trained on features derived from the local context of words. The features include the embedding of word lemma, absolute difference between embeddings of word and its lemma, prefix and suffix of word and pos-tag and dependency relation of its context words, modifiers and governor.

We trained 10 classifiers on same feature sets with slightly different neural network architectures and different training parameters including dropout rate, optimizer, learning rate, epochs and network initialization. All the classifiers use relu, tanh and softmax activations in the input, hidden and output layers respectively. We use GloVe vectors (Pennington et al., 2014) for word embeddings and one-hot vectors for pos-tag and dependency relations in each individual model. Pos-tagging, dependency parsing, named entity recognition and entity coreference resolution are performed using Stanford CoreNLP (Manning et al., 2014)

Table 2 shows the event mention identification results. We report the F1 score for event mention identification based on the KBP scorer, which considers a mention correct if its span, type and sub-

[4]The ECB+ (Cybulska and Vossen, 2014) corpus is another commonly used dataset for evaluating event coreference resolution performance. But we determined that this corpus is not appropriate for evaluating our ILP model that explicitly focuses on using discourse level topic structures for event coreference resolution. Particularly, the ECB+ corpus was created to facilitate both cross-document and in-document event coreference resolution research. Thus, the documents in the corpus were grouped based on several common topics and in each document, event mentions and coreference relations were only annotated selectively in sentences that are on a common topic. When the annotated sentences in each document are stitched together, they do not well reveal the original document structure, which makes the ECB+ corpus a bad choice for evaluating our approach. In addition, due to the selective annotation issue, in-document event coreference resolution with the ECB+ corpus is somewhat easier than with the KBP corpus, which partly explained the significant differences of published in-document event coreference resolution results on the two corpora.

[5]Each discussion forum document consists of a series of posts in an online discussion thread, which lacks coherent discourse structures as a regular document. Therefore, only news articles in the KBP corpora are appropriate for evaluating our approach.

[6]KBP 2015 dataset consists of 181 and 179 documents from discussion forum and news articles respectively. We randomly picked 50 documents from news articles for tuning ILP hyper-parameters and remaining 310 documents for training classifiers.

[7]There are 85 and 83 news articles in KBP 2016 and 2017 corpora respectively.

	KBP 2016					KBP 2017				
Model	B^3	$CEAF_e$	MUC	BLANC	AVG	B^3	$CEAF_e$	MUC	BLANC	AVG
Local classifier	51.47	47.96	26.29	30.82	39.13	50.24	48.47	30.81	29.94	39.87
Clustering	46.97	41.95	18.79	26.88	33.65	46.51	40.21	23.10	25.08	33.72
Basic ILP	51.44	47.77	26.65	30.95	39.19	50.4	48.49	31.33	30.58	40.2
+Topic structure	51.44	47.94	28.86	31.87	40.03	50.39	48.23	33.08	31.26	40.74
+Cross-chain	51.09	47.53	31.27	33.07	40.74	50.39	47.67	35.15	31.88	41.27
+Distribution	51.06	48.28	33.53	33.63	41.62	50.42	48.67	37.52	32.08	42.17
+Subevent	51.67	49.1	34.08	34.08	42.23	50.35	48.61	37.24	31.94	42.04
Joint learning	50.16	48.59	32.41	32.72	40.97	-	-	-	-	-

Table 3: Results for event coreference resolution systems on the KBP 2016 and 2017 corpus. Joint learning results correspond to the actual result files evaluated in (Lu and Ng, 2017). The file was obtained from the authors.

type are the same as the gold mention and assigns a partial score if span partially overlaps with the gold mention. We also report the event mention identification F1 score that only considers mention spans and ignores mention types. We can see that compared to the recent system by (Lu and Ng, 2017) which conducts joint inferences of both event mention detection and event coreference resolution, detecting types for event mentions is a major bottleneck to our event extraction system.

Note that the official KBP 2017 event coreference resolution scorer considers a mention pair coreferent if they strictly match on the event type and subtype, which has been discussed recently to be too conservative (Mitamura et al., 2017). But since improving event mention type detection is not our main goal, we therefore relax the constraints and do not consider event mention type match while evaluating event coreference resolution systems. This allows us to directly interpret the influences of document structures in the event coreference resolution task by overlooking any bias from upstream tasks.

5.3 Baseline Systems

We compare our document-structure guided event coreference resolution model with three baselines. Local classifier performs greedy merging of event mentions using scores predicted by the local pairwise coreference resolution classifier. An event mention is merged to its best matching antecedent event mention if the predicted score between the two event mentions is highest and greater than 0.5. Clustering performs spectral graph clustering (Pedregosa et al., 2011), which represents commonly used clustering algorithms for event coreference resolution. We used the relation between the size of event mentions and the number of coreference clusters in training data for pre-specifying the number of clusters. Its low performance is partially accounted to the difficulty of determining the number of coreference clusters.

Joint learning uses a structured conditional random field model that operates at the document level to jointly model event mention extraction, event coreference resolution and an auxiliary task of event anaphoricity determination. This model has achieved the best event coreference resolution performance to date on the KBP 2016 corpus (Lu and Ng, 2017).

5.4 Our Systems

We gradually augment the ILP baseline with additional objective functions and constraints described in sub-sections 4.2.1, 4.2.2, 4.2.3 and 4.2.4. In all the systems below, we combine objective functions with their corresponding coefficients (as described in sub-section 4.3).

The Basic ILP System formulates event coreference resolution as an ILP optimization task. It uses scores produced by the local pairwise classifier as weights on variables that represent ILP assignments for event coreference relations. (Equations 1, 2).

+Topic structure incorporates the topical structure and the characteristics of main event chains in baseline ILP system (Equations 1-5).

+Cross-chain adds constraints and objective function defined for cross-chain inference to the Topical structure system (Equations 1-8).

+Distribution further adds distributional patterns to the Cross-chain system (Equations 1-10).

+Subevent (Full) optimizes the objective function defined in equation 12 by considering all the constraints defined in 1-11, including constraints for modeling subevent structures.

5.5 Results and Analysis

Table 3 shows performance comparisons of our ILP systems with other event coreference resolution approaches including the recent joint learning approach (Lu and Ng, 2017) which is the best performing model on the KBP 2016 corpus. For both datasets, the full discourse structure augmented model achieved superior performance compared to the local classifier based system. The improvement is observed across all metrics with average F1 gain of 3.1 for KBP 2016 and 2.17 for KBP 2017. Most interestingly, we see over 28% improvement in MUC F1 score which directly evaluates the pairwise coreference link predictions. This implies that the document level structures, indeed, helps in linking more coreferent event mentions, which otherwise are difficult with the local classifier trained on lexical and surface features. Our ILP based system also outperforms the previous best model on the KBP 2016 corpus (Lu and Ng, 2017) consistently using all the evaluation metrics, with an overall improvement of 1.21 based on the average F1 scores.

In Table 3, we also report the F1 scores when we increasingly add each type of structure in the ILP baseline. Among different scoring metrics, all structures positively contributed to the MUC and BLANC scores for KBP 2016 corpus. However, subevent based constraints slightly reduced the F1 scores on KBP 2017 corpus. Based on our preliminary analysis, this can be accounted to the simple method applied for subevent extraction. We only extracted 31 subevents in KBP 2017 corpus compared to 211 in KBP 2016 corpus.

5.6 Discussions on Generalizability

The correlations between event coreference chains and document topic structures are not specific to news articles and widely exist. Several main distributional characteristics of coreferent event mentions, including 1) main event coreference chains often have extended presence and have mentions scattered across segments, and 2) semantically correlated events often have their respective event mentions co-occur in a sentence, directly apply to other sources of texts such as clinical notes. But certain distributional characteristics are genre specific. For instance, while it is common to observe more coreferent event mentions early on in a news article, coreference chains in a clinical note often align well with pre-defined segments like the history of present illness, description of a visit and treatment plan. Thus, the objective functions and constraints defined in equations 1-8 can be directly applied for other domains as well, while other structures like segment-wise distributional patterns may require alteration based on domain-specific knowledge.

6 Conclusions and the Future Work

We have presented an ILP based joint inference system for event coreference resolution that utilizes scores predicted by a pairwise event coreference resolution classifier, and models several aspects of correlations between event coreference chains and document level topic structures, including the correlation between the main event chains and topic transition sentences, interdependencies among event coreference chains, genre-specific coreferent mention distributions and subevents. We have shown that these structures are generalizable by conducting experiments on both the KBP 2016 and KBP 2017 datasets. Our model outperformed the previous state-of-the-art model across all coreference scoring metrics. In the future, we will explore the use of additional discourse structures that correlate highly with event coreference chains. Moreover, we will extend this work to other domains such as biomedical domains.

Acknowledgments

This work was partially supported by the National Science Foundation via NSF Award IIS-1755943. Disclaimer: the views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*. Association for Computational Linguistics, Stroudsburg, PA, USA, ARTE '06, pages 1–8. http://dl.acm.org/citation.cfm?id=1629235.1629236.

Jun Araki, Zhengzhong Liu, Eduard H Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *LREC*. pages 4553–4558.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics* 4:385–399.

Allison Badgett and Ruihong Huang. 2016. Extracting subevents via an effective two-phase approach. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 906–911.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*. Granada, volume 1, pages 563–566.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1412–1422.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics* 40(2):311–347.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics, pages 54–57.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the workshop on events in emerging text types*. Association for Computational Linguistics, pages 17–22.

Prafulla Kumar Choubey and Ruihong Huang. 2017a. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2124–2133.

Prafulla Kumar Choubey and Ruihong Huang. 2017b. Tamu at kbp 2017: Event nugget detection and coreference resolution. In *Proceedings of TAC KBP 2017 Workshop, National Institute of Standards and Technology*.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the most dominant event in a news article by mining event coreference relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. volume 2, pages 340–345.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 2256–2262.

Kevin Clark and Christopher D Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 643–653.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *LREC*. pages 4545–4552.

Agata Cybulska and Piek Vossen. 2015a. Translating granularity of event slots into features for event coreference resolution. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. pages 1–10.

A.K. Cybulska and P.T.J.M. Vossen. 2015b. Bag of events approach to event coreference resolution. supervised classification of event templates. *Lecture Notes in Computer Science* (9042). 978-3-319-18117-2.

Pascal Denis, Jason Baldridge, et al. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT-NAACL*. pages 236–243.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the tac kbp 2015 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2015 Workshop, National Institute of Standards and Technology*. pages 16–17.

Joe Ellis, Jeremy Getman, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2016. Overview of linguistic resources for the tac kbp 2016 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2016 Workshop, National Institute of Standards and Technology*.

Jenny Rose Finkel and Christopher D Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*. Association for Computational Linguistics, pages 45–48.

Jeremy Getman, Joe Ellis, Zhiyi Song, Jennifer Tracey, and Stephanie Strassel. 2017. Overview of linguistic resources for the tac kbp 2017 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2017 Workshop, National Institute of Standards and Technology*.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics* 21(2):203–225.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th annual meeting of the association of computational linguistics*. pages 848–855.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics* 39(4):885–916.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 489–500.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 188–197.

Zhengzhong Liu, Jun Araki, Eduard H Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *LREC*. pages 4539–4544.

Jing Lu and Vincent Ng. 2017. Joint learning for event coreference resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 90–101.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 3264–3275.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, pages 25–32.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association of Computational Linguistics* 3(1):405–418.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2017. Events detection, coreference and sequencing: Whats next? overview of the tac kbp 2017 event track. In *Proceedings of TAC KBP 2017 Workshop, National Institute of Standards and Technology*.

Stuart Mitchell, Michael OSullivan, and Iain Dunning. 2011. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand, http://www. optimization-online. org/DB_FILE/2011/09/3178. pdf* .

Nafise Sadat Moosavi and Michael Strube. 2017. Lexical features in coreference resolution: To be used with caution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 14–19.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Marta Recasens and Eduard Hovy. 2011. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering* 17(4):485–510.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. pages 89–98.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*. Association for Computational Linguistics, pages 45–52.

Christopher Walker, Medero Strassel, Maeda Julie, and Kazuaki. 2006. Ace 2005 multilingual training corpus. In *Linguistic Data Consortium, LDC Catalog No.: LDC2006T06.*.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent bayesian model for event coreference resolution. *Transactions of the Association of Computational Linguistics* 3(1):517–528.

DSGAN: Generative Adversarial Training for Distant Supervision Relation Extraction

Pengda Qin♯, Weiran Xu♯, William Yang Wang♭
♯Beijing University of Posts and Telecommunications, China
♭University of California, Santa Barbara, USA
{qinpengda, xuweiran}@bupt.edu.cn
{william}@cs.ucsb.edu

Abstract

Distant supervision can effectively label data for relation extraction, but suffers from the noise labeling problem. Recent works mainly perform soft bag-level noise reduction strategies to find the relatively better samples in a sentence bag, which is suboptimal compared with making a hard decision of false positive samples in sentence level. In this paper, we introduce an adversarial learning framework, which we named DSGAN, to learn a sentence-level true-positive generator. Inspired by Generative Adversarial Networks, we regard the positive samples generated by the generator as the negative samples to train the discriminator. The optimal generator is obtained until the discrimination ability of the discriminator has the greatest decline. We adopt the generator to filter distant supervision training dataset and redistribute the false positive instances into the negative set, in which way to provide a cleaned dataset for relation classification. The experimental results show that the proposed strategy significantly improves the performance of distant supervision relation extraction comparing to state-of-the-art systems.

1 Introduction

Relation extraction is a crucial task in the field of natural language processing (NLP). It has a wide range of applications including information retrieval, question answering, and knowledge base completion. The goal of relation extraction system is to predict relation between entity pair in a sentence (Zelenko et al., 2003; Bunescu and Mooney, 2005; GuoDong et al., 2005). For example, given a sentence "The $[owl]_{e1}$ held the mouse in its $[claw]_{e2}$.", a relation classifier should figure out the relation **Component-Whole** between entity owl and $claw$.

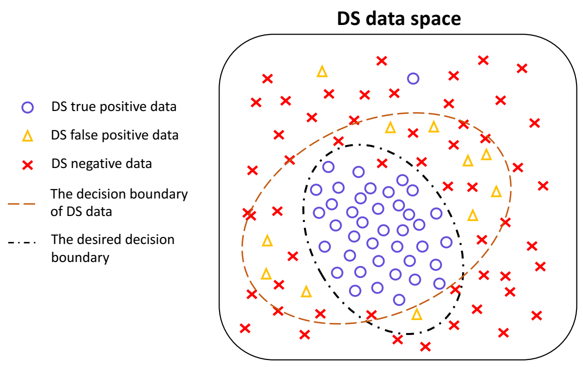


Figure 1: Illustration of the distant supervision training data distribution for one relation type.

With the infinite amount of facts in real world, it is extremely expensive, and almost impossible for human annotators to annotate training dataset to meet the needs of all walks of life. This problem has received increasingly attention. Few-shot learning and Zero-shot Learning (Xian et al., 2017) try to predict the unseen classes with few labeled data or even without labeled data. Differently, distant supervision (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012) is to efficiently generate relational data from plain text for unseen relations with distant supervision (DS). However, it naturally brings with some defects: the resulted distantly-supervised training samples are often very noisy (shown in Figure 1), which is the main problem of impeding the performance (Roth et al., 2013). Most of the current state-of-the-art methods (Zeng et al., 2015; Lin et al., 2016) make the denoising operation in the sentence bag of entity pair, and integrate this process into the distant supervision relation extraction. Indeed, these methods can filter a substantial number of noise samples; However, they overlook the case that all sentences of an entity pair are false positive, which is also the common phenomenon in distant supervision datasets. Under this consideration, an independent and accurate **sentence-level** noise reduction strategy is the better choice.

In this paper, we design an adversarial learning process (Goodfellow et al., 2014; Radford et al., 2015) to obtain a sentence-level generator that can recognize the true positive samples from the noisy distant supervision dataset without any supervised information. In Figure 1, the existence of false positive samples makes the DS decision boundary suboptimal, therefore hinders the performance of relation extraction. However, in terms of quantity, the true positive samples still occupy most of the proportion; this is the prerequisite of our method. Given the discriminator that possesses the decision boundary of DS dataset (the brown decision boundary in Figure 1), the generator tries to generate true positive samples from DS positive dataset; Then, we assign the generated samples with negative label and the rest samples with positive label to challenge the discriminator. Under this adversarial setting, if the generated sample set includes more true positive samples and more false positive samples are left in the rest set, the classification ability of the discriminator will drop faster. Empirically, we show that our method has brought consistent performance gains in various deep-neural-network-based models, achieving strong performances on the widely used New York Times dataset (Riedel et al., 2010). Our contributions are three-fold:

- We are the first to consider adversarial learning to denoise the distant supervision relation extraction dataset.
- Our method is sentence-level and model-agnostic, so it can be used as a plug-and-play technique for any relation extractors.
- We show that our method can generate a cleaned dataset without any supervised information, in which way to boost the performance of recently proposed neural relation extractors.

In Section 2, we outline some related works on distant supervision relation extraction. Next, we describe our adversarial learning strategy in Section 3. In Section 4, we show the stability analyses of DSGAN and the empirical evaluation results. And finally, we conclude in Section 5.

2 Related Work

To address the above-mentioned data sparsity issue, Mintz et al. (2009) first align unlabeled text corpus with Freebase by distant supervision. However, distant supervision inevitably suffers from the wrong labeling problem. Instead of explicitly removing noisy instances, the early works intend to suppress the noise. Riedel et al. (2010) adopt multi-instance single-label learning in relation extraction; Hoffmann et al. (2011) and Surdeanu et al. (2012) model distant supervision relation extraction as a multi-instance multi-label problem.

Recently, some deep-learning-based models (Zeng et al., 2014; Shen and Huang, 2016) have been proposed to solve relation extraction. Naturally, some works try to alleviate the wrong labeling problem with deep learning technique, and their denoising process is integrated into relation extraction. Zeng et al. (2015) select one most plausible sentence to represent the relation between entity pairs, which inevitably misses some valuable information. Lin et al. (2016) calculate a series of soft attention weights for all sentences of one entity pair and the incorrect sentences can be down-weighted; Base on the same idea, Ji et al. (2017) bring the useful entity information into the calculation of the attention weights. However, compared to these soft attention weight assignment strategies, recognizing the true positive samples from distant supervision dataset before relation extraction is a better choice. Takamatsu et al. (2012) build a noise-filtering strategy based on the linguistic features extracted from many NLP tools, including NER and dependency tree, which inevitably suffers the error propagation problem; while we just utilize word embedding as the input information. In this work, we learn a true-positive identifier (the generator) which is independent of the relation prediction of entity pairs, so it can be directly applied on top of any existing relation extraction classifiers. Then, we redistribute the false positive samples into the negative set, in which way to make full use of the distantly labeled resources.

3 Adversarial Learning for Distant Supervision

In this section, we introduce an adversarial learning pipeline to obtain a robust generator which can automatically discover the true positive samples from the noisy distantly-supervised dataset without any supervised information. The overview of our adversarial learning process is shown in Figure 2. Given a set of distantly-labeled sentences, the generator tries to generate true positive samples from it; But, these generated samples are regarded as negative samples to train the discriminator. Thus, when finishing scanning the DS positive dataset one time, the more true positive samples that the generator discovers, the sharper drop of performance the discriminator obtains. After adversarial training, we hope to obtain a robust generator that is capable of forcing discriminator into maximumly losing its classification ability.

In the following section, we describe the adversarial training pipeline between the generator and the discriminator, including the pre-training strategy, objective functions and gradient calculation. Because the generator involves a discrete sampling step, we introduce a policy gradient method to calculate gradients for the generator.

3.1 Pre-Training Strategy

Both the generator and the discriminator require the pre-training process, which is the common setting for GANs (Cai and Wang, 2017; Wang et al., 2017). With the better initial parameters, the adversarial learning is prone to convergence. As presented in Figure 2, the discriminator is pre-trained with DS positive dataset P (label 1) and DS negative set N^D (label 0). After our adversarial learning process, we desire a strong generator that can, to the maximum extent, collapse the discriminator. Therefore, the more robust generator can be obtained via competing with the more robust discriminator. So we pre-train the discriminator until the accuracy reaches 90% or more. The pre-training of generator is similar to the discriminator; however, for the negative dataset, we use another completely different dataset N^G, which makes sure the robustness of the experiment. Specially, we let the generator overfits the DS positive dataset P. The reason of this setting is that we hope the generator wrongly give high probabilities to all of the noisy DS positive samples at the beginning of the training process. Then, along with our adversarial learning, the generator learns to gradually decrease the probabilities of the false positive samples.

3.2 Generative Adversarial Training for Distant Supervision Relation Extraction

The generator and the discriminator of DSGAN are both modeled by simple CNN, because CNN performs well in understanding sentence (Zeng et al., 2014), and it has less parameters than RNN-based networks. For relation extraction, the input information consists of the sentences and entity pairs; thus, as the common setting (Zeng et al., 2014; Nguyen and Grishman, 2015), we use both word embedding and position embedding to convert input instances into continuous real-valued vectors.

What we desire the generator to do is to accurately recognize true positive samples. Unlike the generator applied in computer vision field (Im et al., 2016) that generates new image from the input noise, our generator just needs to discover true positive samples from the noisy DS positive dataset. Thus, it is to realize the "sampling from a probability distribution" process of the discrete GANs (Figure 2). For a input sentence s_j, we define the probability of being true positive sample by generator as $p_G(s_j)$. Similarly, for discriminator, the probability of being true positive sample is represented as $p_D(s_j)$. We define that one epoch means that one time scanning of the entire DS positive dataset. In order to obtain more feedbacks and make the training process more efficient, we split the DS positive dataset $P = \{s_1, s_2, ..., s_j, ...\}$ into N bags $B = \{B^1, B^2, ...B^N\}$, and the network parameters θ_G, θ_D are updated when finishing processing one bag B_i[1]. Based on the notion of adversarial learning, we define the objectives of the generator and the discriminator as follow, and they are alternatively trained towards their respective objectives.

Generator Suppose that the generator produces a set of probability distribution $\{p_G(s_j)\}_{j=1...|B_i|}$ for a sentence bag B_i. Based on these probabilities, a set of sentence are sampled and we denote this set as T.

$$T = \{s_j\}, s_j \sim p_G(s_j), j = 1, 2, ..., |B_i| \quad (1)$$

[1]The *bag* here has the different definition from the sentence *bag* of an entity pair mentioned in the Section 1.

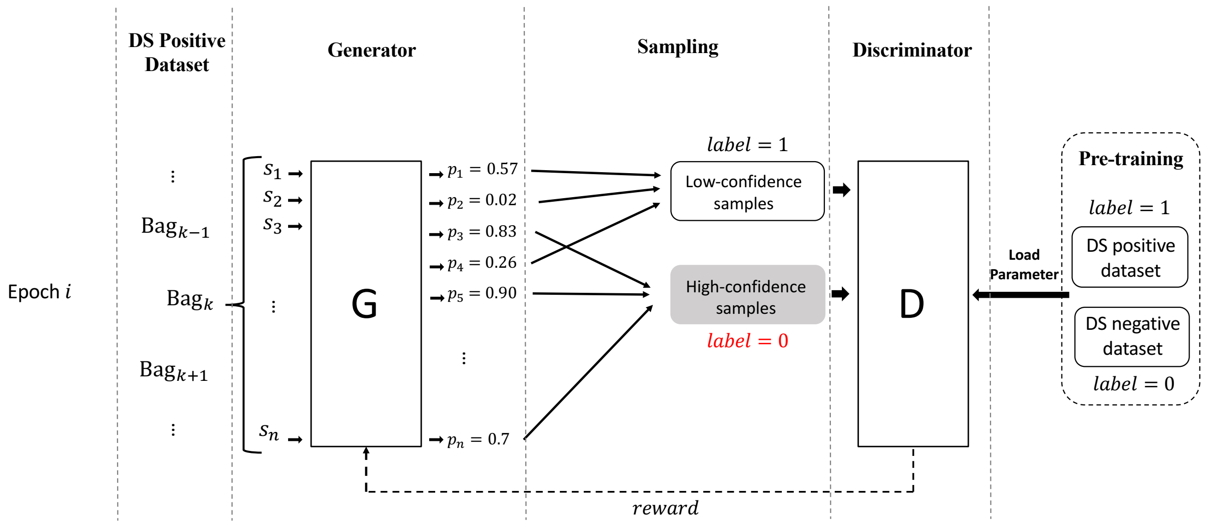


Figure 2: An overview of the DSGAN training pipeline. The generator (denoted by **G**) calculates the probability distribution over a bag of DS positive samples, and then samples according to this probability distribution. The high-confidence samples generated by G are regarded as true positive samples. The discriminator (denoted by **D**) receives these high-confidence samples but regards them as negative samples; conversely, the low-confidence samples are still treated as positive samples. For the generated samples, **G maximizes** the probability of being true positive; on the contrary, **D minimizes** this probability.

This generated dataset T consists of the high-confidence sentences, and is regard as true positive samples by the current generator; however, it will be treated as the negative samples to train the discriminator. In order to challenge the discriminator, the objective of the generator can be formulated as **maximizing** the following probabilities of the generated dataset T:

$$L_G = \sum_{s_j \in T} \log p_D(s_j) \tag{2}$$

Because L_G involves a discrete sampling step, so it cannot be directly optimized by gradient-based algorithm. We adopt a common approach: the policy-gradient-based reinforcement learning. The following section will give the detailed introduction of the setting of reinforcement learning. The parameters of the generator are continually updated until reaching the convergence condition.

Discriminator After the generator has generated the sample subset T , the discriminator treats them as the negative samples; conversely, the rest part $F = B_i - T$ is treated as positive samples. So, the objective of the discriminator can be formulated as **minimizing** the following cross-entropy loss function:

$$\begin{aligned} L_D = -(&\sum_{s_j \in (B_i - T)} \log p_D(s_j) \\ &+ \sum_{s_j \in T} \log(1 - p_D(s_j))) \end{aligned} \tag{3}$$

The update of discriminator is identical to the common binary classification problem. Naturally, it can be simply optimized by any gradient-based algorithm.

What needs to be explained is that, unlike the common setting of discriminator in previous works, our discriminator *loads the same pre-trained parameter set at the beginning of each epoch* as shown in Figure 2. There are two reasons. First, at the end of our adversarial training, what we need is a robust generator rather than a discriminator. Second, our generator is to sample data rather than generate new data from scratch; Therefore, the discriminator is relatively easy to be collapsed. So we design this new adversarial strategy: the robustest generator is yielded when the discriminator has the largest drop of performance in one epoch. In order to create the equal condition, the bag set B for each epoch is identical, including the sequence and the sentences in each

Algorithm 1 The DSGAN algorithm.

```
Data: DS positive set $P$, DS negative set $N^G$ for generator G, DS negative set $N^D$ for discriminator D
Input: Pre-trained G with parameters $\theta_G$ on dataset ($P$, $N^G$); Pre-trained D with parameters $\theta_D$ on
    dataset ($P$, $N^D$)
Output: Adversarially trained generator G
 1: Load parameters $\theta_G$ for G
 2: Split $P$ into the bag sequence $P = \{B^1, B^2, ..., B^i, ..., B^N\}$
 3: repeat
 4:     Load parameters $\theta_D$ for D
 5:     $G_G \leftarrow 0, G_D \leftarrow 0$
 6:     for $B_i \in P, i = 1 \text{ to } N$ do
 7:         Compute the probability $p_G(s_j)$ for each sentence $s_j$ in $B_i$
 8:         Obtain the generated part $T$ by sampling according to $\{p_G(s_j)\}_{j=1...|B|}$ and the rest set $F =
    B_i - T$
 9:         $G_D \leftarrow -\frac{1}{|P|}\{\nabla_{\theta_D} \sum^T \log(1 - p_D(s_j)) + \nabla_{\theta_D} \sum^F \log p_D(s_j)\}$
10:         $\theta_D \leftarrow \theta_D - \alpha_D G_D$
11:         Calculate the reward $r$
12:         $G_G \leftarrow \frac{1}{|T|} \sum^T r \nabla_{\theta_G} \log p_G(s_j)$
13:         $\theta_G \leftarrow \theta_G + \alpha_G G_G$
14:     end for
15:     Compute the accuracy $ACC_D$ on $N^D$ with the current $\theta_D$
16: until $ACC_D$ no longer drops
17: Save $\theta_G$
```

bag B_i.

Optimizing Generator The objective of the generator is similar to the objective of the one-step reinforcement learning problem: Maximizing the expectation of a given function of samples from a parametrized probability distribution. Therefore, we use a policy gradient strategy to update the generator. Corresponding to the terminology of reinforcement learning, s_j is the *state* and $P_G(s_j)$ is the *policy*. In order to better reflect the quality of the generator, we define the reward r from two angles:

- As the common setting in adversarial learning, for the generated sample set, we hope the confidence of being positive samples by the discriminator becomes higher. Therefore, the first component of our reward is formulated as below:

$$r_1 = \frac{1}{|T|} \sum_{s_j \in T} p_D(s_j) - b_1 \tag{4}$$

  the function of b_1 is to reduce variance during reinforcement learning.

- The second component is from the average prediction probability of N^D,

$$\tilde{p} = \frac{1}{|N^D|} \sum_{s_j \in N^D} p_D(s_j) \tag{5}$$

  N^D participates the pre-training process of the discriminator, but not the adversarial training process. When the classification capacity of discriminator declines, the accuracy of being predicted as negative sample on N^D gradually drops; thus, $\tilde{p}$ increases. In other words, the generator becomes better. Therefore, for epoch k, after processing the bag B_i, reward r_2 is calculated as below,

$$\begin{gathered} r_2 = \eta(\tilde{p}_i^k - b_2) \\ where\ b_2 = \max\{\tilde{p}_i^m\}, m = 1..., k-1 \end{gathered} \tag{6}$$

  b_2 has the same function as b_1.

The gradient of L_G can be formulated as below:

$$\begin{aligned} \nabla_{\theta_D} L_G &= \sum_{s_j \in B_i} \mathbb{E}_{s_j \sim p_G(s_j)} r \nabla_{\theta_G} \log p_G(s_j) \\ &= \frac{1}{|T|} \sum_{s_j \in T} r \nabla_{\theta_G} \log p_G(s_j) \end{aligned} \tag{7}$$

3.3 Cleaning Noisy Dataset with Generator

After our adversarial learning process, we obtain one generator for one relation type; These generators possess the capability of generating true positive samples for the corresponding relation type. Thus, we can adopt the generator to filter the noise samples from distant supervision dataset. Simply and clearly, we utilize the generator as a binary classifier. In order to reach the maximum utilization of data, we develop a strategy: for an entity pair with a set of annotated sentences, if all of these sentences are determined as false negative by our generator, this entity pair will be redistributed into the negative set. Under this strategy, the scale of distant supervision training set keeps unchanged.

4 Experiments

This paper proposes an adversarial learning strategy to detect true positive samples from the noisy distant supervision dataset. Due to the absence of supervised information, we define a generator to heuristically learn to recognize true positive samples through competing with a discriminator. Therefore, our experiments are intended to demonstrate that our DSGAN method possess this capability. To this end, we first briefly introduce the dataset and the evaluation metrics. Empirically, the adversarial learning process, to some extent, has instability; Therefore, we next illustrate the convergence of our adversarial training process. Finally, we demonstrate the efficiency of our generator from two angles: the quality of the generated samples and the performance on the widely-used distant supervision relation extraction task.

4.1 Evaluation and Implementation Details

The Reidel dataset[2] (Riedel et al., 2010) is a commonly-used distant supervision relation extraction dataset. Freebase is a huge knowledge base including billions of triples: the entity pair and the specific relationship between them. Given these triples, the sentences of each entity pair are selected from the New York Times corpus(NYT). Entity mentions of NYT corpus are recognized by the Stanford named entity recognizer (Finkel et al., 2005). There are 52 actual relationships and a special relation NA which indicates there is no relation between head and tail entities. Entity pairs of NA are defined as the entity pairs that appear in the same sentence but are not related according to Freebase.

Due to the absence of the corresponding labeled dataset, there is not a ground-truth test dataset to evaluate the performance of distant supervision relation extraction system. Under this circumstance, the previous work adopt the held-out evaluation to evaluate their systems, which can provide an approximate measure of precision without requiring costly human evaluation. It builds a test set where entity pairs are also extracted from Freebase. Similarly, relation facts that discovered from test articles are automatically compared with those in Freebase. CNN is widely used in relation classification (Santos et al., 2015; Qin et al., 2017), thus the generator and the discriminator are both modeled as a simple CNN with the window size c_w and the kernel size c_k. Word embedding is directly from the released word embedding matrix by Lin et al. (2016)[3]. Position embedding has the same setting with the previous works: the maximum distance of -30 and 30. Some detailed hyperparameter settings are displayed in Table 1.

Hyperparameter	**Value**
CNN Window c_w, kernel size c_k	3, 100
Word embedding d_e, $\|V\|$	50, 114042
Position embedding d_p	5
Learning rate of G, D	1e-5, 1e-4

Table 1: Hyperparameter settings of the generator and the discriminator.

4.2 Training Process of DSGAN

Because adversarial learning is widely regarded as an effective but unstable technique, here we illustrate some property changes during the training process, in which way to indicate the learning trend of our proposed approach. We use 3 relation types as the examples: */business/person/company*, */people/person/place_lived* and */location/neighborhood/neighborhood_of*. Because they are from three major classes (*bussiness*, *people*, *location*) of Reidel dataset and they all have enough distant-supervised instances. The first row in Figure 3 shows the classification ability change of the discriminator during training. The accuracy is calculated from the negative set N^D. At the beginning of adversarial learning, the

[2]http://iesl.cs.umass.edu/riedel/ecml/

[3]https://github.com/thunlp/NRE

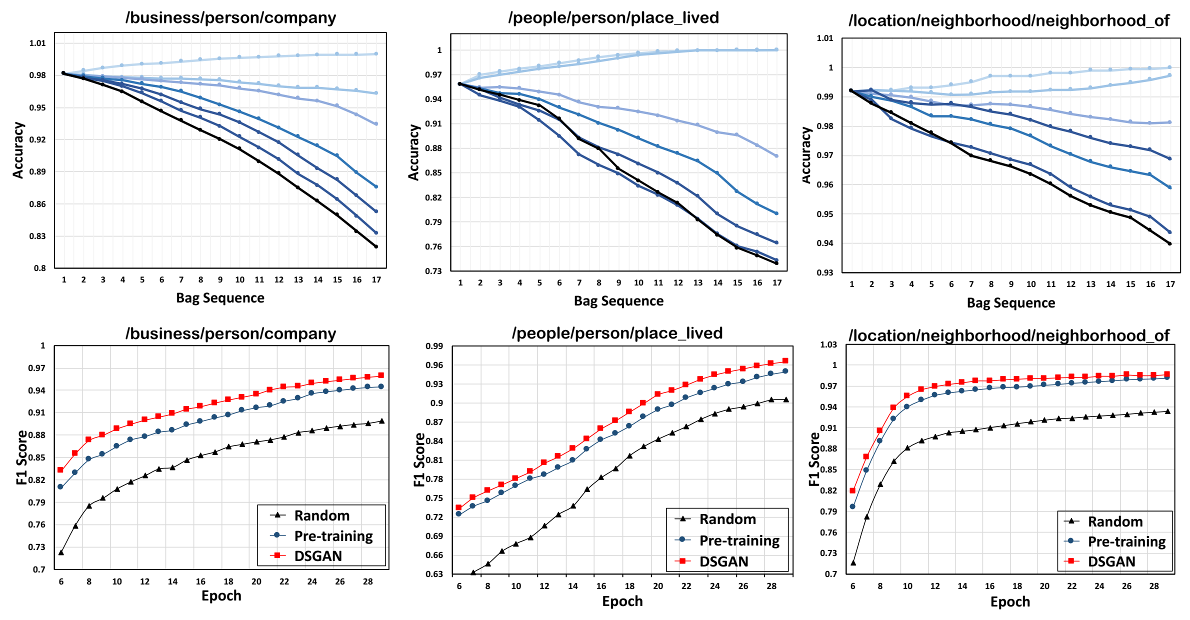


Figure 3: The convergence of the DSGAN training process for 3 relation types and the performance of their corresponding generators. The figures in the first row present the performance change on N^D in some specific epochs during processing the $B = \{B^1, B^2, ...B^N\}$. Each curve stands for one epoch; The color of curves become darker as long as the epoch goes on. Because the discriminator reloads the pre-trained parameters at the beginning of each epoch, all curves start from the same point for each relation type; Along with the adversarial training, the generator gradually collapses the discriminator. The figures in the second row reflect the performance of generators from the view of the difficulty level of training with the positive datasets that are generated by different strategies. Based on the noisy DS positive dataset P, *DSGAN* represents that the cleaned positive dataset is generated by our DSGAN generator; *Random* means that the positive set is randomly selected from P; *Pre-training* denotes that the dataset is selected according to the prediction probability of the pre-trained generator. These three new positive datasets are in the same size.

discriminator performs well on N^D; moreover, N^D is not used during adversarial training. Therefore, the accuracy on N^D is the criterion to reflect the performance of the discriminator. In the early epochs, the generated samples from the generator increases the accuracy, because it has not possessed the ability of challenging the discriminator; however, as the training epoch increases, this accuracy gradually decreases, which means the discriminator becomes weaker. It is because the generator gradually learn to generate more accurate true positive samples in each bag. After the proposed adversarial learning process, the generator is strong enough to collapse the discriminator. Figure 4 gives more intuitive display of the trend of accuracy. Note that there is a critical point of the decline of accuracy for each presented relation types. It is because that the chance we give the generator to challenge the discriminator is just one time scanning of the noisy dataset; this critical point is yielded when the generator has already been robust enough. Thus, we stop the training process when the model reaches this critical point. To sum up, the capability of our generator can steadily increases, which indicates that DSGAN is a robust adversarial learning strategy.

4.3 Quality of Generator

Due to the absence of supervised information, we validate the quality of the generator from another angle. Combining with Figure 1, for one relation type, the true positive samples must have evidently higher relevance (the cluster of purple circles). Therefore, a positive set with more true positive samples is easier to be trained; In other words, the convergence speed is faster and the fitting degree on training set is higher. Based on this , we present the comparison tests in the second row of Figure 3. We build three positive

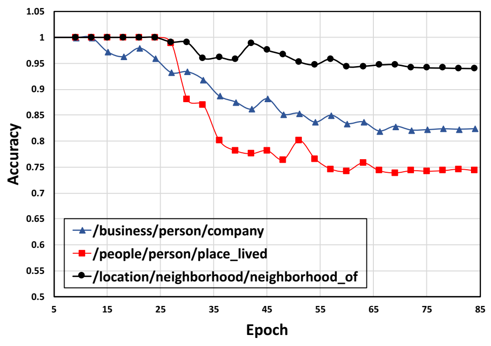


Figure 4: The performance change of the discriminator on N^D during the training process. Each point in the curves records the prediction accuracy on N^D when finishing each epoch. We stop the training process when this accuracy no longer decreases.

datasets from the noisy distant supervision dataset P: the randomly-selected positive set, the positive set base on the pre-trained generator and the positive set base on the DSGAN generator. For the pre-trained generator, the positive set is selected according to the probability of being positive from high to low. These three sets have the same size and are accompanied by the same negative set. Obviously, the positive set from the DSGAN generator yields the best performance, which indicates that our adversarial learning process is able to produce a robust true-positive generator. In addition, the pre-trained generator also has a good performance; however, compared with the DSGAN generator, it cannot provide the boundary between the false positives and the true positives.

4.4 Performance on Distant Supervision Relation Extraction

Based on the proposed adversarial learning process, we obtain a generator that can recognize the true positive samples from the noisy distant supervision dataset. Naturally, the improvement of distant supervision relation extraction can provide a intuitive evaluation of our generator. We adopt the strategy mentioned in Section 3.3 to relocate the dataset. After obtaining this redistributed dataset, we apply it to train the recent state-of-the-art models and observe whether it brings further improvement for these systems. Zeng et al. (2015) and Lin et al. (2016) are both the robust models to solve wrong labeling problem of distant supervision relation extraction. According to the comparison displayed in Figure 5 and Figure 6, all four models (*CNN+ONE*, *CNN+ATT*, *PCNN+ONE* and *PCNN+ATT*) achieve further improvement.

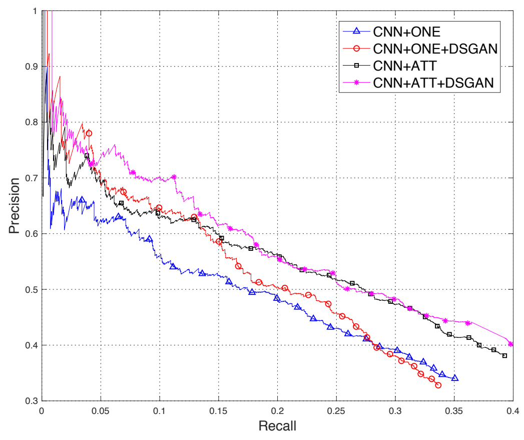


Figure 5: Aggregate PR curves of CNN˙based model.

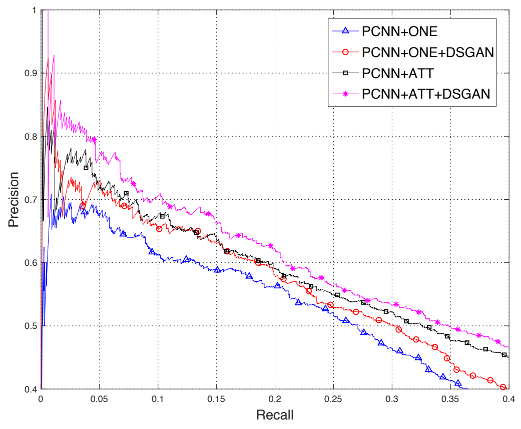


Figure 6: Aggregate PR curves of PCNN˙based model.

Even though Zeng et al. (2015) and Lin et al. (2016) are designed to alleviate the influence of false positive samples, both of them merely focus on the noise filtering in the sentence bag of entity pairs. Zeng et al. (2015) combine at-least-one multi-instance learning with deep neural network to extract only one active sentence to represent the target entity pair; Lin et al. (2016) assign soft attention weights to the representations of all sentences of one entity pair, then employ the weighted sum of these representations to predict the relation between the target entity pair. However, from our manual inspection of Riedel dataset (Riedel et al., 2010), we found another false positive case that all the sentences of a specific entity pair are wrong; but the aforementioned methods overlook

Model	-	+DSGAN	p-value
CNN+ONE	0.177	**0.189**	4.37e-04
CNN+ATT	0.219	**0.226**	8.36e-03
PCNN+ONE	0.206	**0.221**	2.89e-06
PCNN+ATT	0.253	**0.264**	2.34e-03

Table 2: Comparison of AUC values between previous studies and our DSGAN method. The *p-value* stands for the result of t-test evaluation.

this case, while the proposed method can solve this problem. Our DSGAN pipeline is independent of the relation prediction of entity pairs, so we can adopt our generator as the true-positive indicator to filter the noisy distant supervision dataset before relation extraction, which explains the origin of these further improvements in Figure 5 and Figure 6. In order to give more intuitive comparison, in Table 2, we present the AUC value of each PR curve, which reflects the area size under these curves. The larger value of AUC reflects the better performance. Also, as can be seen from the result of t-test evaluation, all the p-values are less than 5e-02, so the improvements are obvious.

5 Conclusion

Distant supervision has become a standard method in relation extraction. However, while it brings the convenience, it also introduces noise in distantly labeled sentences. In this work, we propose the first generative adversarial training method for robust distant supervision relation extraction. More specifically, our framework has two components: a generator that generates true positives, and a discriminator that tries to classify positive and negative data samples. With adversarial training, our goal is to gradually decrease the performance of the discriminator, while the generator improves the performance for predicting true positives when reaching equilibrium. Our approach is model-agnostic, and thus can be applied to any distant supervision model. Empirically, we show that our method can significantly improve the performances of many competitive baselines on the widely used New York Time dataset.

Acknowledge

This work was supported by National Natural Science Foundation of China (61702047), Beijing Natural Science Foundation (4174098), the Fundamental Research Funds for the Central Universities (2017RC02) and National Natural Science Foundation of China (61703234)

References

Razvan Bunescu and Raymond J Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS*, pages 171–178.

Liwei Cai and William Yang Wang. 2017. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. 2016. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL (1)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL (2)*, pages 365–371.

Pengda Qin, Weiran Xu, and Jun Guo. 2017. Designing an adaptive attention mechanism for relation classification. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 4356–4362. IEEE.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 73–78. ACM.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 515–524. ACM.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. *arXiv preprint arXiv:1703.04394*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 17–21.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism

Xiangrong Zeng[12], **Daojian Zeng**[3], **Shizhu He**[1], **Kang Liu**[12], **Jun Zhao**[12]
[1]National Laboratory of Pattern Recognition (NLPR), Institute of Automation
Chinese Academy of Sciences, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
[3]Changsha University of Science & Technology, Changsha, 410114, China
{xiangrong.zeng, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn
zengdj@csust.edu.cn

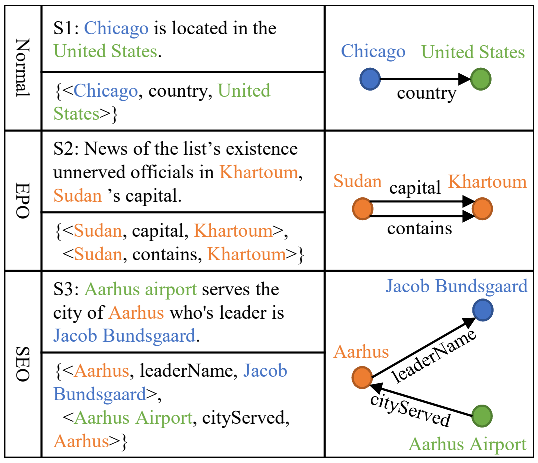
Abstract

The relational facts in sentences are often complicated. Different relational triplets may have overlaps in a sentence. We divided the sentences into three types according to triplet overlap degree, including *Normal*, *EntityPairOverlap* and *SingleEntiyOverlap*. Existing methods mainly focus on *Normal* class and fail to extract relational triplets precisely. In this paper, we propose an end-to-end model based on sequence-to-sequence learning with copy mechanism, which can jointly extract relational facts from sentences of any of these classes. We adopt two different strategies in decoding process: employing only one united decoder or applying multiple separated decoders. We test our models in two public datasets and our model outperform the baseline method significantly.

1 Introduction

Recently, to build large structural knowledge bases (KB), great efforts have been made on extracting relational facts from natural language texts. A relational fact is often represented as a triplet which consists of two entities (an entity pair) and a semantic relation between them, such as $< Chicago, country, UnitedStates >$.

So far, most previous methods mainly focused on the task of relation extraction or classification which identifies the semantic relations between two pre-assigned entities. Although great progresses have been made (Hendrickx et al., 2010; Zeng et al., 2014; Xu et al., 2015a,b), they all assume that the entities are identified beforehand and neglect the extraction of entities. To extract both of entities and relations, early works(Zelenko et al., 2003; Chan and Roth, 2011) adopted a pipeline



Normal	S1: Chicago is located in the United States.	
	{<Chicago, country, United States>}	
EPO	S2: News of the list's existence unnerved officials in Khartoum, Sudan 's capital.	
	{<Sudan, capital, Khartoum>, <Sudan, contains, Khartoum>}	
SEO	S3: Aarhus airport serves the city of Aarhus who's leader is Jacob Bundsgaard.	
	{<Aarhus, leaderName, Jacob Bundsgaard>, <Aarhus Airport, cityServed, Aarhus>}	

Figure 1: Examples of *Normal*, *EntityPairOverlap* (*EPO*) and *SingleEntityOverlap* (*SEO*) classes. The overlapped entities are marked in yellow. S1 belongs to *Normal* class because none of its triplets have overlapped entities; S2 belongs to *EntityPairOverlap* class since the entity pair $< Sudan, Khartoum >$ of it's two triplets are overlapped; And S3 belongs to *SingleEntityOverlap* class because the entity *Aarhus* of it's two triplets are overlapped and these two triplets have no overlapped entity pair.

manner, where they first conduct entity recognition and then predict relations between extracted entities. However, the pipeline framework ignores the relevance of entity identification and relation prediction (Li and Ji, 2014). Recent works attempted to extract entities and relations jointly. Yu and Lam (2010); Li and Ji (2014); Miwa and Sasaki (2014) designed several elaborate features to construct the bridge between these two subtasks. Similar to other natural language processing (NLP) tasks, they need complicated feature engineering and heavily rely on pre-existing NLP tools for feature extraction.

Recently, with the success of deep learning on many NLP tasks, it is also applied on relational facts extraction. Zeng et al. (2014); Xu et al. (2015a,b) employed CNN or RNN on relation classification. Miwa and Bansal (2016); Gupta et al. (2016); Zhang et al. (2017) treated relation extraction task as an end-to-end (end2end) table-filling problem. Zheng et al. (2017) proposed a novel tagging schema and employed a Recurrent Neural Networks (RNN) based sequence labeling model to jointly extract entities and relations.

Nevertheless, the relational facts in sentences are often complicated. Different relational triplets may have overlaps in a sentence. Such phenomenon makes aforementioned methods, whatever deep learning based models and traditional feature engineering based joint models, always fail to extract relational triplets precisely. Generally, according to our observation, we divide the sentences into three types according to triplet overlap degree, including *Normal*, *EntityPairOverlap* (*EPO*) and *SingleEntityOverlap* (*SEO*). As shown in Figure 1, a sentence belongs to *Normal* class if none of its triplets have overlapped entities. A sentence belongs to *EntityPairOverlap* class if some of its triplets have overlapped entity pair. And a sentence belongs to *SingleEntityOverlap* class if some of its triplets have an overlapped entity and these triplets don't have overlapped entity pair. In our knowledge, most previous methods focused on *Normal* type and seldom consider other types. Even the joint models based on neural network (Zheng et al., 2017), it only assigns a single tag to a word, which means one word can only participate in at most one triplet. As a result, the triplet overlap issue is not actually addressed.

To address the aforementioned challenge, we aim to design a model that could extract triplets, including entities and relations, from sentences of *Normal*, *EntityPairOverlap* and *SingleEntityOverlap* classes. To handle the problem of triplet overlap, one entity must be allowed to freely participate in multiple triplets. Different from previous neural methods, we propose an end2end model based on sequence-to-sequence (Seq2Seq) learning with copy mechanism, which can jointly extract relational facts from sentences of any of these classes. Specially, the main component of this model includes two parts: encoder and decoder. The encoder converts a natural language sentence (the source sentence) into a fixed length semantic vector. Then, the decoder reads in this vector and generates triplets directly. To generate a triplet, firstly, the decoder generates the relation. Secondly, by adopting the copy mechanism, the decoder copies the first entity (head entity) from the source sentence. Lastly, the decoder copies the second entity (tail entity) from the source sentence. In this way, multiple triplets can be extracted (In detail, we adopt two different strategies in decoding process: employing only one unified decoder (OneDecoder) to generate all triplets or applying multiple separated decoders (MultiDecoder) and each of them generating one triplet). In our model, one entity is allowed to be copied several times when it needs to participate in different triplets. Therefore, our model could handle the triplet overlap issue and deal with both of *EntityPairOverlap* and *SingleEntityOverlap* sentence types. Moreover, since extracting entities and relations in a single end2end neural network, our model could extract entities and relations jointly.

The main contributions of our work are as follows:

- We propose an end2end neural model based on sequence-to-sequence learning with copy mechanism to extract relational facts from sentences, where the entities and relations could be jointly extracted.
- Our model could consider the relational triplet overlap problem through copy mechanism. In our knowledge, the relational triplet overlap problem has never been addressed before.
- We conduct experiments on two public datasets. Experimental results show that we outperforms the state-of-the-arts with 39.8% and 31.1% improvements respectively.

2 Related Work

By giving a sentence with annotated entities, Hendrickx et al. (2010); Zeng et al. (2014); Xu et al. (2015a,b) treat identifying relations in sentences as a multi-class classification problem. Zeng et al. (2014) among the first to introduce CNN into relation classification. Xu et al. (2015a) and Xu et al. (2015b) learned relation representations from shortest dependency paths through a CNN or RNN. Despite their success, these models ignore the extraction of the entities from sentences and could not truly extract relational facts.

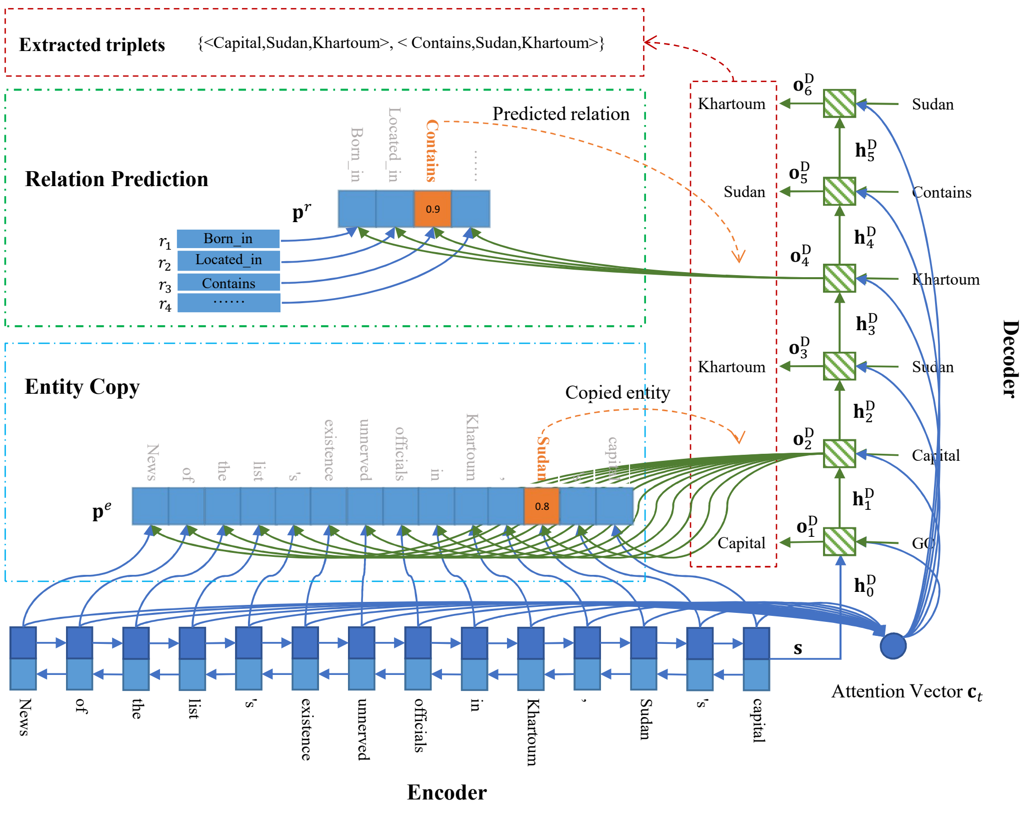


Figure 2: The overall structure of OneDecoder model. A bi-directional RNN is used to encode the source sentence and then a decoder is used to generate triples directly. The relation is predicted and the entity is copied from source sentence.

By giving a sentence without any annotated entities, researchers proposed several methods to extract both entities and relations. Pipeline based methods, like Zelenko et al. (2003) and Chan and Roth (2011), neglected the relevance of entity extraction and relation prediction. To resolve this problem, several joint models have been proposed. Early works (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014) need complicated process of feature engineering and heavily depends on NLP tools for feature extraction. Recent models, like Miwa and Bansal (2016); Gupta et al. (2016); Zhang et al. (2017); Zheng et al. (2017), jointly extract the entities and relations based on neural networks. These models are based on tagging framework, which assigns a relational tag to a word or a word pair. Despite their success, none of these models can fully handle the triplet overlap problem mentioned in the first section. The reason is in their hypothesis, that is, a word (or a word pair) can only be assigned with just one relational tag.

This work is based on sequence-to-sequence learning with copy mechanism, which have been adopted for some NLP tasks. Dong and Lapata (2016) presented a method based on an attention-enhanced and encoder-decoder model, which encodes input utterances and generates their logical forms. Gu et al. (2016); He et al. (2017) applied copy mechanism to sentence generation. They copy a segment from the source sequence to the target sequence.

3 Our Model

In this section, we introduce a differentiable neural model based on Seq2Seq learning with copy mechanism, which is able to extract multiple relational facts in an end2end fashion.

Our neural model encodes a variable-length sentence into a fixed-length vector representation first and then decodes this vector into the corresponding relational facts (triplets). When decoding, we can either decode all triplets with one unified decoder or decode every triplet with a separated decoder. We denote them as **OneDecoder** model and **MultiDecoder** model separately.

3.1 OneDecoder Model

The overall structure of OneDecoder model is shown in Figure 2.

3.1.1 Encoder

To encode a sentence $s = [w_1, .., w_n]$, where w_t represent the t-th word and n is the source sentence length, we first turn it into a matrix $X = [x_1, \cdots, x_n]$, where x_t is the embedding of t-th word.

The canonical RNN encoder reads this matrix X sequentially and generates output $\mathbf{o}_t^E$ and hidden state $\mathbf{h}_t^E$ in time step $t (1 \leq t \leq n)$ by

$$\mathbf{o}_t^E, \mathbf{h}_t^E = f(x_t, \mathbf{h}_{t-1}^E) \quad (1)$$

where $f(\cdot)$ represents the encoder function.

Following (Gu et al., 2016), our encoder uses a bi-directional RNN (Chung et al., 2014) to encode the input sentence. The forward and backward RNN obtain output sequence $\{\overrightarrow{\mathbf{o}_1^E}, \cdots, \overrightarrow{\mathbf{o}_n^E}\}$ and $\{\overleftarrow{\mathbf{o}_n^E}, \cdots, \overleftarrow{\mathbf{o}_1^E}\}$, respectively. We then concatenate $\overrightarrow{\mathbf{o}_t^E}$ and $\overleftarrow{\mathbf{o}_{n-t+1}^E}$ to represent the t-th word. We use $\mathbf{O}^E = [\mathbf{o}_1^E, ..., \mathbf{o}_n^E]$, where $\mathbf{o}_t^E = [\overrightarrow{\mathbf{o}_t^E}; \overleftarrow{\mathbf{o}_{n-t+1}^E}]$, to represent the concatenate result. Similarly, the concatenation of forward and backward RNN hidden states are used as the representation of sentence, that is $\mathbf{s} = [\overrightarrow{\mathbf{h}_n^E}; \overleftarrow{\mathbf{h}_n^E}]$

3.1.2 Decoder

The decoder is used to generate triplets directly. Firstly, the decoder generates a relation for the triplet. Secondly, the decoder copies an entity from the source sentence as the first entity of the triplet. Lastly, the decoder copies the second entity from the source sentence. Repeat this process, the decoder could generate multiple triplets. Once all valid triplets are generated, the decoder will generate NA triplets, which means "stopping" and is similar to the "*eos*" symbol in neural sentence generation. Note that, a NA triplet is composed of an NA-relation and an NA-entity pair.

As shown in Figure 3 (a), in time step t $(1 \leq t)$, we calculate the decoder output $\mathbf{o}_t^D$ and hidden state $\mathbf{h}_t^D$ as follows:

$$\mathbf{o}_t^D, \mathbf{h}_t^D = g(\mathbf{u}_t, \mathbf{h}_{t-1}^D) \quad (2)$$

where $g(\cdot)$ is the decoder function and $\mathbf{h}_{t-1}^D$ is the hidden state of time step $t-1$. We initialize $\mathbf{h}_0^D$ with the representation of source sentence $\mathbf{s}$. $\mathbf{u}_t$ is the decoder input in time step t and we calculate it as:

$$\mathbf{u}_t = [\mathbf{v}_t; \mathbf{c}_t] \cdot \mathbf{W}^u \quad (3)$$

where $\mathbf{c}_t$ is the attention vector and $\mathbf{v}_t$ is the embedding of copied entity or predicted relation in time step $t-1$. $\mathbf{W}^u$ is a weight matrix.

Attention Vector. The attention vector $\mathbf{c}_t$ is calculated as follows:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_i \times \mathbf{o}_i^E \quad (4)$$

$$\boldsymbol{\alpha} = softmax(\boldsymbol{\beta}) \quad (5)$$

$$\beta_i = selu([\mathbf{h}_{t-1}^D; \mathbf{o}_i^E] \cdot \mathbf{w}^c) \quad (6)$$

where $\mathbf{o}_i^E$ is the output of encoder in time step i, $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]$ and $\boldsymbol{\beta} = [\beta_1, ..., \beta_n]$ are vectors, $\mathbf{w}^c$ is a weight vector. $selu(\cdot)$ is activation function (Klambauer et al., 2017).

After we get decoder output $\mathbf{o}_t^D$ in time step t $(1 \leq t)$, if $t\%3 = 1$ (that is $t = 1, 4, 7, ...$), we use $\mathbf{o}_t^D$ to predict a relation, which means we are decoding a new triplet. Otherwise, if $t\%3 = 2$ (that is $t = 2, 5, 8, ...$), we use $\mathbf{o}_t^D$ to copy the first entity from the source sentence, and if $t\%3 = 0$ (that is $t = 3, 6, 9, ...$), we copy the second entity.

Predict Relation. Suppose there are m valid relations in total. We use a fully connected layer to calculate the confidence vector $\mathbf{q}^r = [q_1^r, ..., q_m^r]$ of all valid relations:

$$\mathbf{q}^r = selu(\mathbf{o}_t^D \cdot \mathbf{W}^r + \mathbf{b}^r) \quad (7)$$

where $\mathbf{W}^r$ is the weight matrix and $\mathbf{b}^r$ is the bias. When predict the relation, it is possible to predict the NA-relation when the model try to generate NA-triplet. To take this into consideration, we calculate the confidence value of NA-relation as:

$$\mathbf{q}^{NA} = selu(\mathbf{o}_t^D \cdot \mathbf{W}^{NA} + \mathbf{b}^{NA}) \quad (8)$$

where $\mathbf{W}^{NA}$ is the weight matrix and $\mathbf{b}^{NA}$ is the bias. We then concatenate $\mathbf{q}^r$ and $\mathbf{q}^{NA}$ to form the confidence vector of all relations (including the NA-relation) and apply softmax to obtain the probability distribution $\mathbf{p}^r = [p_1^r, ..., p_{m+1}^r]$ as:

$$\mathbf{p}^r = softmax([\mathbf{q}^r; \mathbf{q}^{NA}]) \quad (9)$$

We select the relation with the highest probability as the predict relation and use it's embedding as the next time step input $\mathbf{v}_{t+1}$.

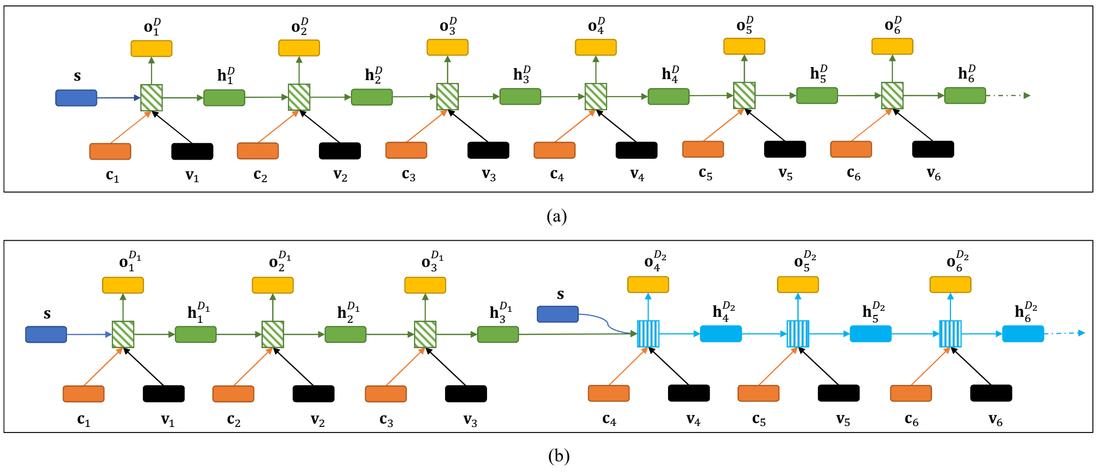


Figure 3: The inputs and outputs of the decoder(s) of OneDecoder model and MultiDecoder model. (a) is the decoder of OneDecoder model. As we can see, only one decoder (the green rectangle with shadows) is used and this encoder is initialized with the sentence representation **s**. (b) is the decoders of MultiDecoder model. There are two decoders (the green rectangle and blue rectangle with shadows). The first decoder is initialized with **s**; Other decoder(s) are initialized with **s** and previous decoder's state.

Copy the First Entity. To copy the first entity, we calculate the confidence vector $\mathbf{q}^e = [q_1^e, ..., q_n^e]$ of all words in source sentence as:

$$q_i^e = selu([\mathbf{o}_t^D; \mathbf{o}_i^E] \cdot \mathbf{w}^e) \tag{10}$$

where $\mathbf{w}^e$ is the weight vector. Similar with the relation prediction, we concatenate $\mathbf{q}^e$ and $\mathbf{q}^{NA}$ to form the confidence vector and apply softmax to obtain the probability distribution $\mathbf{p}^e = [p_1^e, ..., p_{n+1}^e]$:

$$\mathbf{p}^e = softmax([\mathbf{q}^e; \mathbf{q}^{NA}]) \tag{11}$$

Similarly, We select the word with the highest probability as the predict the word and use it's embedding as the next time step input $\mathbf{v}_{t+1}$.

Copy the Second Entity. Copy the second entity is almost the same as copy the first entity. The only difference is when copying the second entity, we cannot copy the first entity again. This is because in a valid triplet, two entities must be different. Suppose the first copied entity is the k-th word in the source sentence, we introduce a mask vector M with n (n is the length of source sentence) elements, where:

$$M_i = \begin{cases} 1, & i \neq k \\ 0, & i = k \end{cases} \tag{12}$$

then we calculate the probability distribution $\mathbf{p}^e$ as:

$$\mathbf{p}^e = softmax([M \otimes \mathbf{q}^e; \mathbf{q}^{NA}]) \tag{13}$$

where $\otimes$ is element-wise multiplication. Just like copy the first entity, We select the word with the highest probability as the predict word and use it's embedding as the next time step input $\mathbf{v}_{t+1}$.

3.2 MultiDecoder Model

MultiDecoder model is an extension of the proposed OneDecoder model. The main difference is when decoding triplets, MultiDecoder model decode triplets with several separated decoders. Figure 3 (b) shows the inputs and outputs of decoders of MultiDecoder model. There are two decoders (the green and blue rectangle with shadows). Decoders work in a sequential order: the first decoder generate the first triplet and then the second decoder generate the second triplet.

Similar with Eq 2, we calculate the hidden state $\mathbf{h}_t^{D_i}$ and output $\mathbf{o}_t^{D_i}$ of i-th ($1 \leq i$) decoder in time step t as follows:

$$\mathbf{o}_t^{D_i}, \mathbf{h}_t^{D_i} = \begin{cases} g^{D_i}(\mathbf{u}_t, \mathbf{h}_{t-1}^{D_i}), & t\%3 = 2, 0 \\ g^{D_i}(\mathbf{u}_t, \hat{\mathbf{h}}_{t-1}^{D_i}), & t\%3 = 1 \end{cases} \tag{14}$$

$g^{D_i}(\cdot)$ is the decoder function of decoder i. $\mathbf{u}_t$ is the decoder input in time step t and we calculated it as Eq 3. $\mathbf{h}_{t-1}^{D_i}$ is the hidden state of i-th decoder in time step $t-1$. $\hat{\mathbf{h}}_{t-1}^{D_i}$ is the initial hidden state of i-th decoder, which is calculated as follows:

$$\hat{\mathbf{h}}_{t-1}^{D_i} = \begin{cases} \mathbf{s}, & i = 1 \\ \frac{1}{2}(\mathbf{s} + \mathbf{h}_{t-1}^{D_{i-1}}), & i > 1 \end{cases} \tag{15}$$

Class	NYT		WebNLG	
	Train	Test	Train	Test
Normal	37013	3266	1596	246
EPO	9782	978	227	26
SEO	14735	1297	3406	457
ALL	56195	5000	5019	703

Table 1: The number of sentences of *Normal*, *EntityPairOverlap* (*EPO*) and *SingleEntityOverlap* (*SEO*) classes. It's worthy noting that a sentence can belongs to both *EPO* class and *SEO* class.

3.3 Training

Both OneDecoder and MultiDecoder models are trained with the negative log-likelihood loss function. Given a batch of data with B sentences $S = \{s_1, ..., s_B\}$ with the target results $Y = \{\mathbf{y}_1, ..., \mathbf{y}_B\}$, where $\mathbf{y}_i = [y_i^1, ..., y_i^T]$ is the target result of s_i, the loss function is defined as follows:

$$\mathbf{L} = \frac{1}{B \times T} \sum_{i=1}^{B} \sum_{t=1}^{T} -log(p(y_i^t | y_i^{<t}, s_i, \theta)) \quad (16)$$

T is the maximum time step of decoder. $p(x|y)$ is the conditional probability of x given y. θ denotes parameters of the entire model.

4 Experiments

4.1 Dataset

To evaluate the performance of our methods, we conduct experiments on two widely used datasets.

The first is New York Times (NYT) dataset, which is produced by distant supervision method (Riedel et al., 2010). This dataset consists of 1.18M sentences sampled from 294k 1987-2007 New York Times news articles. There are 24 valid relations in total. In this paper, we treat this dataset as supervised data as the same as Zheng et al. (2017). We filter the sentences with more than 100 words and the sentences containing no positive triplets, and 66195 sentences are left. We randomly select 5000 sentences from it as the test set, 5000 sentences as the validation set and the rest 56195 sentences are used as train set.

The second is WebNLG dataset (Gardent et al., 2017). It is originally created for Natural Language Generation (NLG) task. This dataset contains 246 valid relations. In this dataset, a instance including a group of triplets and several standard sentences (written by human). Every standard sentence contains all triplets of this instance. We only use the first standard sentence in our experiments and we filter out the instances if all entities of triplets are not found in this standard sentence. The origin WebNLG dataset contains train set and development set. In our experiments, we treat the origin development set as test set and randomly split the origin train set into validation set and train set. After filtering and splitting, the train set contains 5019 instances, the test set contains 703 instances and the validation set contains 500 instances.

The number of sentences of every class in NYT and WebNLG dataset are shown in Table 1. It's worthy noting that a sentence can belongs to both *EntityPairOverlap* class and *SingleEntityOverlap* class.

4.2 Settings

In our experiments, for both dataset, we use LSTM (Hochreiter and Schmidhuber, 1997) as the model cell; The cell unit number is set to 1000; The embedding dimension is set to 100; The batch size is 100 and the learning rate is 0.001; The maximum time steps T is 15, which means we predict at most 5 triplets for each sentence (therefore, there are 5 decoders in MultiDecoder model). These hyperparameters are tuned on the validation set. We use Adam (Kingma and Ba, 2015) to optimize parameters and we stop the training when we find the best result in the validation set.

4.3 Baseline and Evaluation Metrics

We compare our models with NovelTagging model (Zheng et al., 2017), which conduct the best performance on relational facts extraction. We directly run the code released by Zheng et al. (2017) to acquire the results.

Following Zheng et al. (2017), we use the standard micro Precision, Recall and F1 score to evaluate the results. Triplets are regarded as correct when it's relation and entities are both correct. When copying the entity, we only copy the last word of it. A triplet is regarded as NA-triplet when and only when it's relation is NA-relation and it has an NA-entity pair. The predicted NA-triplets will be excluded.

4.4 Results

Table 2 shows the Precision, Recall and F1 value of NovelTagging model (Zheng et al., 2017) and our OneDecoder and MultiDecoder models.

Model	NYT			WebNLG		
	Precision	Recall	F1	Precision	Recall	F1
NovelTagging	**0.624**	0.317	0.420	**0.525**	0.193	0.283
OneDecoder	0.594	0.531	0.560	0.322	0.289	0.305
MultiDecoder	0.610	**0.566**	**0.587**	0.377	**0.364**	**0.371**

Table 2: Results of different models in NYT dataset and WebNLG dataset.

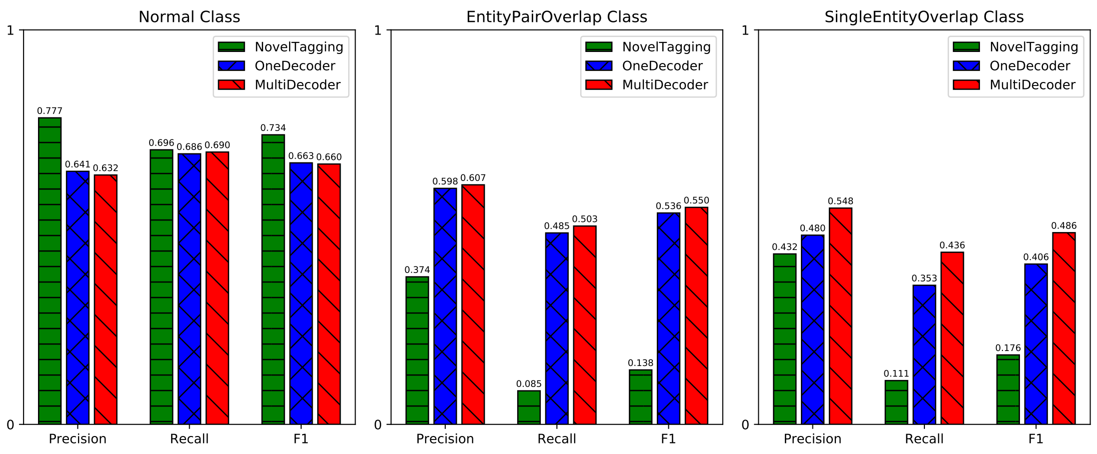


Figure 4: Results of NovelTagging, OneDecoder, and MultiDecoder model in *Normal*, *EntityPairOverlap* and *SingleEntityOverlap* classes in NYT dataset.

As we can see, in NYT dataset, our MultiDecoder model achieves the best F1 score, which is 0.587. There is 39.8% improvement compared with the NovelTagging model, which is 0.420. Besides, our OneDecoder model also outperforms the NovelTagging model. In the WebNLG dataset, MultiDecoder model achieves the highest F1 score (0.371). MultiDecoder and OneDecoder models outperform the NovelTagging model with 31.1% and 7.8% improvements, respectively. These observations verify the effectiveness of our models.

We can also observe that, in both NYT and WebNLG dataset, the NovelTagging model achieves the highest precision value and lowest recall value. By contrast, our models are much more balanced. We think that the reason is in the structure of the proposed models. The NovelTagging method finds triplets through tagging the words. However, they assume that only one tag could be assigned to just one word. As a result, one word can participate at most one triplet. Therefore, the NovelTagging model can only recall a small number of triplets, which harms the recall performance. Different from the NovelTagging model, our models apply copy mechanism to find entities for a triplet, and a word can be copied many times when this word needs to participate in multiple different triplets. Not surprisingly, our models recall more triplets and achieve higher recall value. Further experiments verified this.

4.5 Detailed Results on Different Sentence Types

To verify the ability of our models in handling the overlapping problem, we conduct further experiments on NYT dataset.

Figure 4 shows the results of NovelTagging, OneDecoder and MultiDecoder model in *Normal*, *EntityPairOverlap* and *SingleEntityOverlap* classes. As we can see, our proposed models perform much better than NovelTagging model in *EntityPairOverlap* class and *SingleEntityOverlap* classes. Specifically, our models achieve much higher performance on all metrics. Another observation is that NovelTagging model achieves the best performance in *Normal* class. This is because the NovelTagging model is designed more suitable for *Normal* class. However, our proposed models are more suitable for the triplet overlap issues. Furthermore, it is still difficult for our models to judge how many triplets are needed for the input sentence. As a result, there is a loss in our models for *Normal* class. Nevertheless, the overall perfor-

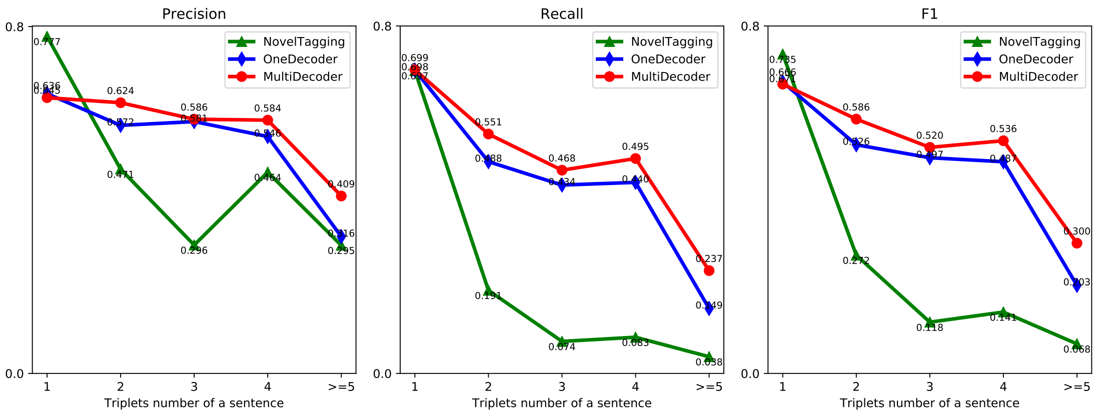


Figure 5: Relation Extraction from sentences that contains different number of triplets. We divide the sentences of NYT test set into 5 subclasses. Each class contains sentences that have 1,2,3,4 or $>=5$ triplets.

Model	NYT	WebNLG
OneDecoder	0.858	0.745
MultiDecoder	0.862	0.821

Table 3: F1 values of entity generation.

Model	NYT	WebNLG
OneDecoder	0.874	0.759
MultiDecoder	0.870	0.751

Table 4: F1 values of relation generation.

mance of the proposed models still outperforms NoverTagging. Moreover, we notice that the whole extracted performance of *EntityPairOverlap* and *SingleEntityOverlap* class is lower than that in *Normal* class. It proves that extracting relational facts from *EntityPairOverlap* and *SingleEntity-Overlap* classes are much more challenging than from *Normal* class.

We also compare the model's ability of extracting relations from sentences that contains different number of triplets. We divide the sentences in NYT test set into 5 subclasses. Each class contains sentences that has 1,2,3,4 or $>=5$ triplets. The results are shown in Figure 5. When extracting relation from sentences that contains 1 triplets, NovelTagging model achieve the best performance. However, when the number of triplets increases, the performance of NovelTagging model decreases significantly. We can also observe the huge decrease of recall value of NovelTagging model. These experimental results demonstrate the ability of our model in handling multiple relation extraction.

4.6 OneDecoder vs. MultiDecoder

As shown in the previous experiments (Table 2, Figure 4 and Figure 5), our MultiDecoder model performs better then OneDecoder model and NovelTagging model. To find out why MultiDecoder model performs better than OneDecoder model, we analyzed their ability of entity generation and relation generation. The experiment results are shown in Table 3 and Table 4. We can observe that on both NYT and WebNLG datasets, these two models have comparable abilities on relation generation. However, MultiDecoder performs better than OneDecoder model when generating entities. We think that it is because MultiDecoder model utilizes different decoder to generate different triplets so that the entity generation results could be more diverse.

Conclusions and Future Work

In this paper, we proposed an end2end neural model based on Seq2Seq learning framework with copy mechanism for relational facts extraction. Our model can jointly extract relation and entity from sentences, especially when triplets in the sentences are overlapped. Moreover, we analyze the different overlap types and adopt two strategies for this issue, including one unified decoder and multiple separated decoders. We conduct experiments on two public datasets to evaluate the effectiveness of our models. The experiment results show that our models outperform the baseline method significantly and our models can extract relational facts from all three classes.

This challenging task is far from being solved. Our future work will concentrate on how to improve the performance further. Another future work is test our model in other NLP tasks like event extraction.

Acknowledgments

The authors thank Yi Chang from Huawei Tech. Ltm for his helpful discussions. This work is supported by the Natural Science Foundation of China (No.61533018 and No.61702512). This work is also supported in part by Beijing Unisound Information Technology Co., Ltd, and Huawei Innovation Research Program of Huawei Tech. Ltm.

References

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of ACL*, pages 551–560.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*, pages 33–43.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of ACL*, pages 179–188.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, pages 1631–1640.

Pankaj Gupta, Hinrich Schtze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING*, pages 2537–2547.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of ACL*, pages 199–208.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

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

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. In *Proceedings of ICLR*, pages 1–15.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in NIPS*, pages 971–980.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*, pages 402–412.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of EMNLP*, pages 1858–1869.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML PKDD*, pages 148–163.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of EMNLP*, pages 1785–1794.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of COLING*, pages 1399–1407.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of EMNLP*, pages 1730–1740.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of ACL*, pages 1227–1236.

Self-regulation: Employing a Generative Adversarial Network to Improve Event Detection

Yu Hong Wenxuan Zhou Jingli Zhang Qiaoming Zhu Guodong Zhou[*]
Institute of Artificial Intelligence, Soochow University
School of Computer Science and Technology, Soochow University
No.1, Shizi ST, Suzhou, China, 215006
{tianxianer, wxchow024, jlzhang05}@gmail.com
{qmzhu, gdzhou}@suda.edu.cn

Abstract

Due to the ability of encoding and mapping semantic information into a high-dimensional latent feature space, neural networks have been successfully used for detecting events to a certain extent. However, such a feature space can be easily contaminated by spurious features inherent in event detection. In this paper, we propose a self-regulated learning approach by utilizing a generative adversarial network to generate spurious features. On the basis, we employ a recurrent network to eliminate the fakes. Detailed experiments on the ACE 2005 and TAC-KBP 2015 corpora show that our proposed method is highly effective and adaptable.

1 Introduction

Event detection aims to locate the event triggers of specified types in text. Normally, triggers are words or nuggets that evoke the events of interest.

Detecting events in an automatic way is challenging, not only because an event can be expressed in different words, but also because a word may express a variety of events in different contexts. In particular, the frequent utilization of common words, ambiguous words and pronouns in event mentions makes them harder to detect:

1) **Generality** – *<u>taken</u> home* <`Transport`>
Ambiguity 1 – *<u>campaign</u> in Iraq* <`Attack`>
Ambiguity 2 – *political <u>campaign</u>* <`Elect`>
Coreference – *Either <u>its</u> bad or good* <`Marry`>

A promising solution to this challenge is through semantic understanding. Recently, neural networks have been widely used in this direction (Nguyen and Grishman, 2016; Ghaeini et al., 2016; Feng et al., 2016; Liu et al., 2017b; Chen et al., 2017), which allows semantics of event mentions (trigger plus context) to be encoded in a high-dimensional latent feature space. This facilitates the learning of deep-level semantics. Besides, the use of neural networks not only strengthens current supervised classification of events but alleviates the complexity of feature engineering.

However, compared to the earlier study (Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013), in which the features are carefully designed by experts, the neural network based methods suffer more from spurious features. Here, spurious feature is specified as the latent information which looks like the semantically related information to an event, but actually not (Liu et al., 2017a). For example, in the following sample, the semantic information of the word "*prison*" most probably enables spurious features to come into being, because the word often co-occurs with the trigger "*taken*" to evoke an `Arrest-jail` event instead of the ground-truth event `Transport`:

2) *<u>Prison</u> authorities have given the nod for Anwar to be <u>taken</u> home later in the afternoon.*
Trigger: *taken*. **Event Type**: `Transport`

It is certain that spurious features often result from the semantically pseudo-related context, and during training, a neural network may mistakenly and unconsciously preserve the memory to produce the fakes. However, it is difficult to determine which words are pseudo-related in a specific case, and when they will "jump out" to mislead the generation of latent features during testing.

To address the challenge, we suggest to regulate the learning process with a two-channel self-regulated learning strategy. In the self-regulation process, on one hand, a generative adversarial network is trained to produce the most spurious features, while on the other hand, a neural network

[*] Corresponding author

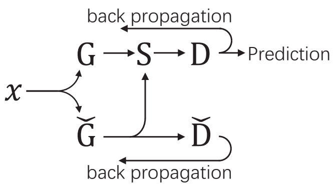


Figure 1: Self-regulated learning scheme

is equipped with a memory suppressor to eliminate the fakes. Detailed experiments on event detection show that our proposed method achieves a substantial performance gain, and is capable of robust domain adaptation.

2 Task Definition

The task of event detection is to determine whether there is one or more event triggers in a sentence. Trigger is defined as a token or nugget that best signals the occurrence of an event. If successfully identified, a trigger is required to be assigned a tag to indicate the event type:

Input: *Either its bad or good*

Output: *its* <trigger>; Marry <type>

We formalize the event detection problem as a multi-class classification problem. Given a sentence, we classify every token of the sentence into one of the predefined event classes (Doddington et al., 2004) or non-trigger class.

3 Self-Regulated Learning (SELF)

SELF is a double-channel model (Figure 1), consisted of a cooperative network (Islam et al., 2003) and a generative adversarial net (GAN) (Goodfellow et al., 2014). A memory suppressor *S* is used to regulate communication between the channels.

3.1 Cooperative Network

In channel 1, the generator G is specified as a multilayer perceptron. It plays a role of a "diligent student". By a differentiable function $G(x, \theta_g)$ with parameters θ_g, the generator learns to produce a vector of latent features o_g that may best characterize the token x, i.e., $o_g = G(x, \theta_g)$.

The discriminator D (called "a lucky professor") is a single-layer perceptron, implemented as a differentiable function $D(o_g, \theta_d)$ with parameters θ_d. Relying on the feature vector o_g, it attempts to accurately predict the probability of the token x triggering an event for all event classes, i.e., $\hat{y} = D(o_g, \theta_d)$, and assigns x to the most probable class c (*iff* $\hat{y}_c > \forall \hat{y}_{\bar{c}}, \bar{c} \neq c$).

Therefore, G and D cooperate with each other during training, developing the parameters θ_g and θ_d with the same goal – to minimize the performance loss $\mathcal{L}(\hat{y}, y)$ in the detection task:

$$\begin{bmatrix}\theta_g \\ \theta_d\end{bmatrix} = argmin\ \mathcal{L}(\hat{y}, y) \qquad (1)$$

where, y denotes the ground-truth probability distribution over event classes, and $\mathcal{L}$ indicates the deviation of the prediction from the ground truth.

3.2 Generative Adversarial Network

In channel 2, the generator $\check{G}$ and discriminator $\check{D}$ have the same perceptual structures as G and D. They also perform learning by differentiable functions, respectively $\check{G}(x, \theta_{\check{g}})$ and $\check{D}(o_{\check{g}}, \theta_{\check{d}})$. A major difference, however, is that they are caught into a cycle of highly adversarial competition.

The generator $\check{G}$ is a "trouble maker". It learns to produce spurious features, and utilizes them to contaminate the feature vector $o_{\check{g}}$ of the token x. Thus $\check{G}$ changes a real sample x into a fake z – sometimes successfully, sometimes less so. Using the fakes, $\check{G}$ repeatedly instigates the discriminator $\check{D}$ to make mistakes. On the other side, $\check{D}$ ("a hapless professor") has to avoid being deceived, and struggles to correctly detect events no matter whether it encounters x or z.

In order to outsmart the adversary, $\check{G}$ develops the parameters $\theta_{\check{g}}$ during training to maximize the performance loss, but on the contrary, $\check{D}$ develops the parameters $\theta_{\check{d}}$ to minimize the loss:

$$\theta_{\check{g}} = argmax\ \mathcal{L}(\hat{y}, y) \qquad (2)$$

$$\theta_{\check{d}} = argmin\ \mathcal{L}(\hat{y}, y) \qquad (3)$$

Numerous studies have confirmed that the two-player minmax game enables both $\check{G}$ and $\check{D}$ to improve their methods (Goodfellow et al., 2014; Liu and Tuzel, 2016; Huang et al., 2017).

3.3 Regulation with Memory Suppressor

Using a memory suppressor, we try to optimize the diligent student G. The goal is to enable G to be as dissimilar as possible to the troublemaker $\check{G}$.

The suppressor uses the output $o_{\check{g}}$ of $\check{G}$ as a reference resource which should be full of spurious features. On the basis, it looks over the output o_g of G, so as to verify whether the features in o_g are different to those in $o_{\check{g}}$. If very different, the suppressor allows G to preserve the memory (viz., θ_g in $G(x, \theta_g)$), otherwise update. In other word, for G, the suppressor forcibly erases the memory which may result in the generation of spurious features. We call this the self-regulation.

Self-regulation is performed for the whole sentence which is fed into G and $\check{G}$. Assume that O_g is a matrix, constituted with a series of feature vectors, i.e., the vectors generated by G for all the tokens in an input sentence ($o_g \in O_g$), while $O_{\check{g}}$ is another feature matrix, generated by $\check{G}$ for the tokens ($o_{\check{g}} \in O_{\check{g}}$). Thus, we utilize the matrix approximation between O_g and $O_{\check{g}}$ for measuring the loss of self-regulation learning $\mathcal{L}_{diff}$. The higher the similarity, the greater the loss. During training, the generator G is required to develop the parameters θ_g to minimize the loss:

$$\theta_g = argmin\ \mathcal{L}_{diff}(o_g, o_{\check{g}}) \tag{4}$$

We present in detail the matrix approximate calculation in section 4.4, where the squared Frobenius norm (Bousmalis et al., 2016) is used.

3.4 Learning to Predict

We incorporate the cooperative network with the GAN, and enhance their learning by joint training.

In the 4-member incorporation, i.e., $\{G, \check{G}, D$ and $\check{D}\}$, the primary beneficiary is the lucky professor D. It can benefit from both the cooperation in channel 1 and the competition in channel 2. The latent features it uses are well-produced by G, and decontaminated by eliminating possible fakes like those made by $\check{G}$. Therefore, in experiments, we choose to output the prediction results of D.

In this paper, we use two recurrent neural networks (RNN) (Sutskever et al., 2014; Chung et al., 2014) of the same structure as the generators. And both the discriminators are implemented as a fully-connected layer followed by a softmax layer.

4 Recurrent Models for SELF

RNN with long short-term memory (abbr., LSTM) is adopted due to the superior performance in a variety of NLP tasks (Liu et al., 2016a; Lin et al., 2017; Liu et al., 2017a). Furthermore, the bidirectional LSTM (Bi-LSTM) architecture (Schuster and Paliwal, 1997; Ghaeini et al., 2016; Feng et al., 2016) is strictly followed. This architecture enables modeling of the semantics of a token with both the preceding and following contexts.

4.1 LSTM based Generator

Given a sentence, we follow Chen et al (2015) to take all the tokens of the whole sentence as the input. Before feeding the tokens into the network, we transform each of them into a real-valued vector $x \in \mathbb{R}^e$. The vector is formed by concatenating a word embedding with an entity type embedding.

- **Word Embedding**: It is a fixed-dimensional real-valued vector which represents the hidden semantic properties of a token (Collobert and Weston, 2008; Turian et al., 2010).

- **Entity Type Embedding**: It is specially used to characterize the entity type associated with a token. The BIO2 tagging scheme (Wang and Manning, 2013; Huang et al., 2015) is employed for assigning a type label to each token in the sentence.

For the input token x_t at the current time step t, the LSTM generates the latent feature vector $o_t \in \mathbb{R}^d$ by the previous memory. Meanwhile, the token is used to update the current memory.

The LSTM possesses a long-term memory unit $c_t \in \mathbb{R}^d$ and short-term $\widetilde{c}_t \in \mathbb{R}^d$. In addition, it is equipped with the input gate i_t, forgetting gate f_t and a hidden state h_t, which are assembled together to promote the use of memory, as well as dynamic memory updating. Similarly, they are defined as a d-dimensional vector in $\mathbb{R}^d$. Thus LSTM works in the following way:

$$\begin{bmatrix} o_t \\ \widetilde{c}_t \\ i_t \\ f_t \end{bmatrix} = \begin{bmatrix} \sigma \\ tanh \\ \sigma \\ \sigma \end{bmatrix} \left(\mathrm{W} \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + \mathrm{b} \right) \tag{5}$$

$$h_t = o_t \odot tanh(c_t) \tag{6}$$

$$c_t = \widetilde{c}_t \odot i_t + c_{t-1} \odot \mathrm{f}_t \tag{7}$$

where $\mathrm{W} \in \mathbb{R}^{4d \times (d+e)}$ and $\mathrm{b} \in \mathbb{R}^{4d}$ are parameters of affine transformation; σ refers to the logistic sigmoid function and $\odot$ denotes element-wise multiplication.

The output functions of both the generators in SELF, i.e., G and $\check{G}$, can be boiled down to the output gate $o_t \in \mathbb{R}^d$ of the LSTM cell:

$$o_t = LSTM(x_t; \theta) \tag{8}$$

where, the function LSTM (·;·) is a shorthand for Eq. (5-7) and θ represents all the parameters of LSTM. For both G and $\check{G}$, θ are initialized with the same values in experiments. But due to the distinct training goals of G and $\check{G}$ (diligence or making-trouble), the values of the parameters in the two cases will change to be very different after training. Therefore, we have $o_{g,t} = LSTM(x_t, \theta_{g,t})$ and $o_{\breve{g},t} = LSTM(x_t, \theta_{\breve{g},t})$.

4.2 Fully-connected Layer for Discrimination

Depending on the feature vectors $o_{g,t}$ and $o_{\breve{g},t}$, the two discriminators D and $\breve{D}$ predict the probability of the token x_t triggering an event for all event classes. As usual, they compute the probability distribution over classes using a fully connected layer followed by a softmax layer:

$$\hat{y} = softmax(\hat{W} \cdot o_t + \hat{b}) \tag{9}$$

where $\breve{y}$ is a C-dimensional vector, in which each dimension indicates the prediction for a class; C is the class number; $\hat{W} \in \mathbb{R}^d$ is the weight which needs to be learned; $\hat{b}$ is a bias term.

It is noteworthy that the discriminator D and $\breve{D}$ don't share the weight and the bias. It means that, for the same token x_t, they may make markedly different predictions: $\hat{y}_{g,t} = softmax(\hat{W}_g \cdot o_{g,t} + \hat{b}_g)$ and $\hat{y}_{\breve{g},t} = softmax(\hat{W}_{\breve{g}} \cdot o_{\breve{g},t} + \hat{b}_{\breve{g}})$.

4.3 Classification Loss

We specify the loss as the cross-entropy between the predicted and ground-truth probability distributions over classes. Given a batch of training data that includes N samples (x_i, y_i), we calculate the losses the discriminators cause as below:

$$\mathcal{L}(\hat{y}_g, y) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j log(\hat{y}_{g,i}^j) \tag{10}$$

$$\mathcal{L}(\hat{y}_{\breve{g}}, y) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j log(\hat{y}_{\breve{g},i}^j) \tag{11}$$

where y_i is a C-dimensional one-hot vector. The value of its j-th dimension is set to be 1 only if the token x_i triggers an event of the j-th class, otherwise 0. Both $\hat{y}_{g,i}$ and $\hat{y}_{\breve{g},i}$ are the predicted probability distributions over the C classes for x_i.

4.4 Loss of Self-regulated Learning

Assume that O_g is a matrix, consisted of the feature vectors output by G for all the tokens in a sentence, i.e., $o_{g,t} \in O_g$, and $O_{\breve{g}}$ is that provided by $\breve{G}$, i.e., $o_{\breve{g},t} \in O_{\breve{g}}$, thus we compute the similarity between O_g and $O_{\breve{g}}$ and use it as the measure of self-regulation loss $\mathcal{L}_{diff}(O_g, O_{\breve{g}})$:

$$\mathcal{L}_{diff}(O_g, O_{\breve{g}}) = \|O_g O_{\breve{g}}^{\top}\|_F^2 \tag{12}$$

where, $\|\cdot\|_F^2$ denotes the squared Frobenius norm (Bousmalis et al., 2016), which is used to calculate the similarity between matrices.

It is noteworthy that the feature vectors a generator outputs are required to serve as the rows in the matrix, deployed in a top-down manner and arranged in the order in which they are generated. For example, the feature vector $o_{g,t}$ the generator G outputs at the time t needs to be placed in the t-th row of the matrix O_g.

At the very beginning of the measurement, the similarity between every feature vector in O_g and that in $O_{\breve{G}}$ is first calculated by the matrix-matrix multiplication $O_g O_{\breve{g}}^{\top}$:

$$\begin{pmatrix} o_{g,1}o_{\breve{g},1}^{\top} & \dots & o_{g,1}o_{\breve{g},t}^{\top} & \dots & o_{g,1}o_{\breve{g},l}^{\top} \\ \dots & \dots & \dots & \dots & \dots \\ o_{g,1}o_{\breve{g},t}^{\top} & \dots & o_{g,t}o_{\breve{g},t}^{\top} & \dots & o_{g,t}o_{\breve{g},l}^{\top} \\ \dots & \dots & \dots & \dots & \dots \\ o_{g,1}o_{\breve{g},l}^{\top} & \dots & o_{g,l}o_{\breve{g},t}^{\top} & \dots & o_{g,l}o_{\breve{g},l}^{\top} \end{pmatrix}$$

where, the symbol $\top$ denotes the transpose operation; l is the sentence length which is defined to be uniform for all sentences (l=80), and if it is larger than the real ones, padding is used; $o_{g,i}o_{\breve{g},j}$ denotes the scalar product between the feature vectors $o_{g,i}$ and $o_{\breve{g},j}$.

Let $A_{m\times n}$ be a matrix, the squared Frobenius norm of $A_{m\times n}$ (i.e., $\|A_{m\times n}\|_F^2$) is defined as:

$$\|A_{m\times n}\|_F^2 = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\right)^{\frac{1}{2}} \tag{13}$$

where, a_{ij} denotes the j-th element in the i-th row of $A_{m\times n}$. Thus, if we let $A_{m\times n}$ be the matrix produced by the matrix-matrix multiplication $O_g O_{\breve{g}}^{\top}$, the self-regulation loss $\mathcal{L}_{diff}(O_g, O_{\breve{g}})$ can be eventually obtained by:

$$\mathcal{L}_{diff}(O_g, O_{\breve{g}}) = \left(\sum_{i=1}^{l}\sum_{j=1}^{l} |o_{g,i}o_{\breve{g},j}|^2\right)^{\frac{1}{2}} \tag{14}$$

For a batch of training data that includes N' sentences, the global self-regulation loss is specified as the sum of the losses for all the sentences: $\mathcal{L}_{SELF} = \sum_{i=1}^{N'} \mathcal{L}_{diff}(O_g, O_{\breve{g}})$.

4.5 Training

We train the cooperative network in SELF to minimize the classification loss $\mathcal{L}(\hat{y}_g, y)$ and the loss of self-regulated learning $\mathcal{L}_{SELF}$:

$$\theta_g = argmin\ (\mathcal{L}\hat{y}_g, y) \quad (15)$$

$$\theta_d = argmin\ (\mathcal{L}(\hat{y}_g, y) + \lambda \cdot \mathcal{L}_{SELF}) \quad (16)$$

where λ is a hyper-parameter, which is used to harmonize the two losses.

The min-max game is utilized for training the adversarial net in SELF: $\theta_{\breve{g}} = argmax\ \mathcal{L}(\hat{y}_{\breve{g}}, y)$; $\theta_{\breve{d}} = argmin\ \mathcal{L}(\hat{y}_{\breve{g}}, y)$.

All the networks in SELF are trained jointly using the same batches of samples. They are trained via stochastic gradient descent (Nguyen and Grishman, 2015) with shuffled mini-batches and the AdaDelta update rule (Zeiler, 2012). The gradients are computed using back propagation. And regularization is implemented by a dropout (Hinton et al., 2012).

5 Experimentation

5.1 Resource and Experimental Datasets

We test the presented model on the ACE 2005 corpus. The corpus is annotated with single-token event triggers and has 33 predefined event types (Doddington et al., 2004; Ahn, 2006), along with one class "*None*" for the non-trigger tokens, constitutes a 34-class classification problem.

For comparison purpose, we use the corpus in the traditional way, randomly selecting 30 articles in English from different genres as the development set, and utilizing a separate set of 40 English newswire articles as the test set. The remaining 529 English articles are used as the training set.

5.2 Hyperparameter Settings

The word embeddings are initialized with the 300-dimensional real-valued vectors. We follow Chen et al (2015) and Feng et al (2016) to pre-train the embeddings over NYT corpus using Mikolov et al (2013)'s skip-gram tool. The entity type embeddings, as usual (Nguyen et al., 2016; Feng et al., 2016; Liu et al., 2017b), are specified as the 50-dimensional real-valued vectors. They are initialized with the 32-bit floating-point values, which are all randomly sampled from the uniformly distributed values in [-1, 1][1]. We initialize other adjustable parameters of the back-propagation algorithm by randomly sampling in [-0.1, 0.1].

We follow Feng et al (2016) to set the dropout rate as 0.2 and the mini-batch size as 10. We tune the initialized parameters mentioned above, harmonic coefficient λ, learning rate and the L2 norm on the development set. Grid search (Liu et al., 2017a) is used to seek for the optimal parameters. Eventually, we take the coefficient λ of 0.1^{+3}, learning rate of 0.3 and L2 norm of 0.

The source code of SELF[2] to reproduce the experiments has been made publicly available.

5.3 Compared Systems

The state-of-the-art models proposed in the past decade are compared with ours. By taking learning framework as the criterion, we divide the models into three classes:

Minimally supervised approach: is Peng et al (2016)'s **MSEP-EMD**.

Feature based approaches: primarily including Liao and Grishman (2010)'s **Cross-Event** inference model, which is based on the max-entropy classification and embeds the document-level confident information in the feature space; Hong et al (2011)'s **Cross-Entity** inference model, in which existential backgrounds of name entities are employed as the additional discriminant features; and Li et al (2013)'s **Joint** model, a sophisticated predictor frequently ranked among the top 3 in recent TAC-KBP evaluations for nugget and coreference detection (Hong et al., 2014, 2015; Yu et al., 2016). It is based on structured perceptron and combines the local and global features.

Neural network based approaches: including the convolutional neural network (**CNN**) (Nguyen and Grishman, 2015), the non-consecutive *N*-grams based CNN (**NC-CNN**) (Nguyen and Grishman, 2016) and the CNN that is assembled with a dynamic multi-pooling layer (**DM-CNN**) (Chen et al., 2015). Others include Ghaeini et al (2016)'s forward-backward recurrent neural network (**FB-RNN**) which is developed using gated recurrent units (GRU), Nguyen et al (2016)'s bidirectional RNN (**Bi-RNN**) and Feng et al (2016)'s **Hybrid** networks that consist of a Bi-LSTM and a CNN.

Besides, we compare our model with Liu et al (2016b)'s artificial neural networks (**ANNs**), Liu et al (2017b)'s attention-based ANN (**ANN-S2**) and Chen et al (2017)'s **DM-CNN***. The models recently have become popular because, although simple in structure, they are very analytic by learning from richer event examples, such as those in

[1]https://www.tensorflow.org/api_docs/python/tf/random_uniform

[2]https://github.com/JoeZhouWenxuan/Self-regulation-Employing-a-Generative-Adversarial-Network-to-Improve-Event-Detection/tree/master

Method	P (%)	R (%)	F (%)
Joint (Local+Global)	76.9	65.0	70.4
MSEP-EMD	75.6	69.8	72.6
DM-CNN	80.4	67.7	73.5
DM-CNN*	79.7	69.6	74.3
Bi-RNN	68.5	75.7	71.9
Hybrid: Bi-LSTM+CNN	80.8	71.5	75.9
SELF: Bi-LSTM+GAN	**75.3**	**78.8**	**77.0**

Table 1: Trigger identification performance

FrameNet (**FN**) and Wikipeida (**Wiki**).

5.4 Experimental Results

We evaluate our model using Precision (P), Recall (R) and F-score (F). To facilitate the comparison, we review the best performance of the competitors, which has been evaluated using the same metrics, and publicly reported earlier.

Trigger identification

Table 1 shows the trigger identification performance. It can be observed that SELF outperforms other models, with a performance gain of no less than 1.1% F-score.

Frankly, the performance mainly benefits from the higher recall (78.8%). But in fact the relatively comparable precision (75.3%) to the recall reinforces the advantages. By contrast, although most of the compared models achieve much higher precision over SELF, they suffer greatly from the substantial gaps between precision and recall. The advantage is offset by the greater loss of recall.

GAN plays an important role in optimizing Bi-RNN. This is proven by the fact that SELF (Bi-LSTM+GAN) outperforms Nguyen et al (2016)'s Bi-RNN. To be honest, the models use two different kinds of recurrent units. Bi-RNN uses GRUs, but SELF uses the units that possess LSTM. Nevertheless, GRU has been experimentally proven to be comparable in performance to LSTM (Chung et al., 2014; Jozefowicz et al., 2015). This allows a fair comparison between Bi-RNN and SELF.

Event classification

Table 2 shows the performance of multi-class classification. SELF achieves nearly the same F-score as Feng et al (2016)'s Hybrid, and outperforms the others. More importantly, SELF is the only one which obtains a performance higher than 70% for both precision and recall.

Besides, by analyzing the experimental results, we have identified the following regularities:

Methods	P (%)	R (%)	F (%)
MSEP-EMD	70.4	65.0	67.6
Cross-Event	68.8	68.9	68.8
Cross-Entity	72.9	64.3	68.3
Joint (Local+Global)	73.7	62.3	67.5
CNN	71.8	66.4	69.0
DM-CNN	75.6	63.6	69.1
NC-CNN	-	-	71.3
FB-RNN (GRU)	66.8	68.0	67.4
Bi-RNN (GRU)	66.0	73.0	69.3
ANNs (ACE+FN)	77.6	65.2	70.7
DM-CNN*(ACE+Wiki)	75.7	66.0	70.5
ANN-S2 (ACE+FN)	76.8	67.5	71.9
Hybrid: Bi-LSTM+**CNN**	**84.6**	64.9	**73.4**
SELF: Bi-LSTM+**GAN**	71.3	**74.7**	**73.0**

Table 2: Detection performance (trigger identification plus multi-class classification)

- Similar to the pattern classifiers that are based on hand-designed features, the CNN models enable higher precision to be obtained. However the recall is lower.
- The RNN models contribute to achieving a higher recall. However the precision is lower.
- Expansion of the training data set helps to increase the precision.

Let us turn to the structurally more complicated models, SELF and Hybrid.

SELF inherits the merits of the RNN models, classifying the events with higher recall. Besides, by the utilization of GAN, SELF has evolved from the traditional learning strategies, being capable of learning from GAN and getting rid of the mistakenly generated spurious features. So that it outperforms other RNNs, with improvements of no less than 4.5% precision and 1.7% recall.

Hybrid is elaborately established by assembling a RNN with a CNN. It models an event from two perspectives: language generation and pragmatics. The former is deeply learned by using the continuous states hidden in the recurrent units, while the later the convolutional features. Multi-angled cognition enables Hybrid to be more precise. However it is built using a single-channel architecture, concatenating the RNN and the CNN. This results in twofold accumulation of feature information, causing a serious overfitting problem. Therefore, Hybrid is localized to much higher precision but substantially lower recall.

Overfitting results in enlargement of the gap between precision and recall when the task changes to be more difficult. For Hybrid, as illustrated in

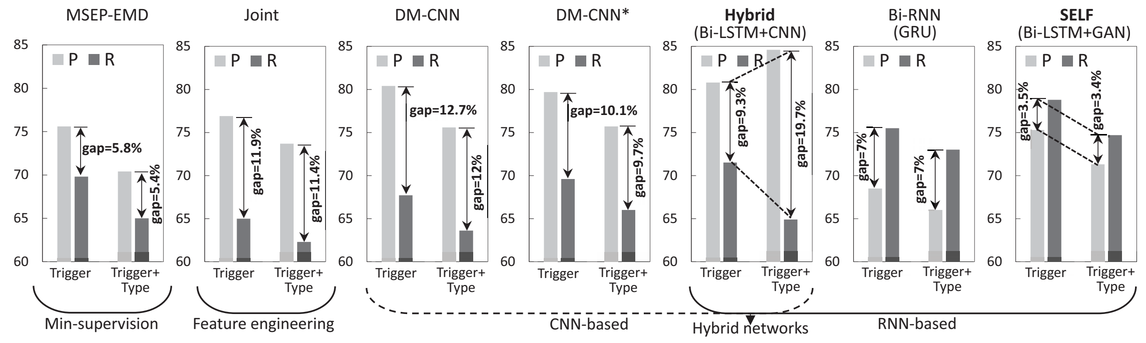


Figure 2: Gaps between precision and recall in the tasks of trigger identification and event classification

Methods	Embedding Types	Training Data
ANNs	word	ACE+FN
ANN-S2	word, NE-type	ACE+FN
DM-CNN*	word, PSN	ACE+Wiki
CNN	word, NE-type, PSN	ACE
NC-CNN	word, NE-type, PSN	ACE
Bi-RNN	word, NE-type, DEP	ACE
Hybrid	word, NE-type, PSN	ACE
DM-CNN	word, PSN	ACE
FB-RNN	word, branch	ACE
SELF	**word, NE-type**	**ACE**

Table 3: Embedding types and training data (DEP: Dependency grammar; PSN: Position)

Figure 2, the gap becomes much wider (from 9% to 19.7%) when the binary classification task (trigger identification) is shifted to multi-class classification (event detection). By contrast, other work shows a nearly constant gap. In particular, SELF yields a minimum gap in each task, which changes negligibly from 3.5% to 3.4%.

It may be added that, similar to DM-CNN and FB-RNN, SELF is cost-effective. Compared to other models (Table 3), it either uses less training data, or is only required to learn two kinds of embeddings, such as that of words and entity types.

5.5 Discussion: Adaptation, Robustness and Effectiveness

Domain adaptation is a key criteria for evaluating the utility of a model in practical application. A model can be thought of being adaptable only if it works well for the unlabeled data in the target domain when trained on the source domain (Blitzer et al., 2006; Plank and Moschitti, 2013).

We perform two groups of domain adaptation experiments, respectively, using the ACE 2005 corpus and the corpus for TAC-KBP 2015 event nugget track (Ellis et al., 2015).

The ACE corpus consists of 6 domains: broadcast conversation (`bc`), broadcast news (`bn`), telephone conversation (`cts`), newswire (`nw`), usenet (`un`) and web blogs (`wl`). Following the common practice of adaptation research on this data (Nguyen and Grishman, 2014, 2015; Plank and Moschitti, 2013), we take the union of `bn` and `nw` as the source domain and `bc`, `cts` and `wl` as three different target domains. We randomly select half of the instances from `bc` to constitute the development set. The TAC-KBP corpus consists of 2 domains: newswire (`NW`) and discussion forum (`DF`). We follow Peng et al (2016) to use one of `NW` and `DF` in alternation as the source domain, while the other the target domain. We randomly select a proportion (20%) of the instances from the target domain to constitute the development set.

We compare with **Joint**, **CNN**, **MSEP-EMD**, **SSED** (Sammons et al., 2015) and **Hybrid**. All the models except Hybrid have been reported for the performance assessment of domain adaptation. In this section, we only cite the best performance they obtained. We reproduce Hybrid by using the source code given by authors. To ensure a fair comparison, we perform 3 runs, in each of which, both Hybrid and SELF were redeveloped on a new development set. What we report herein is the average performance they obtained over the 3 runs.

Adaptation Performance

We show the adaptation performance on the ACE corpus in Tables 4 and that on TAC-KBP in Table 5. It can be observed that SELF outperforms other models in the out-of-domain scenarios.

Besides, when testing is performed on the out-of-domain ACE corpus, the performance degradation of SELF is not much larger than that of CNN and Hybrid. When the out-of-domain TAC-KBP corpus is used, the performance of SELF is impaired much less severely than SSED and Hybrid.

Methods	In-domain (`bn+nw`)			Out-of-domain (`bc`)				Out-of-domain (`cts`)				Out-of-domain (`wl`)			
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	Loss	P(%)	R(%)	F(%)	Loss	P(%)	R(%)	F(%)	Loss
Joint	72.9	63.2	67.7	68.8	57.5	62.6	↓5.1	64.5	52.3	57.7	↓10.0	56.4	38.5	45.7	↓22.0
CNN	69.2	67.0	68.0	70.2	65.2	67.6	↓0.4	68.3	58.2	62.8	↓5.2	54.8	42.0	47.5	↓20.5
Hybrid	68.8	54.8	61.0	64.7	58.8	61.6	↑0.6	59.9	50.6	54.9	↓6.1	54.0	37.9	44.5	↓16.5
SELF	**73.8**	**65.7**	**69.5**	**70.0**	**67.2**	**68.9**	**↓0.6**	**68.3**	**60.2**	**63.3**	**↓6.2**	**58.0**	**44.0**	**50.0**	**↓19.5**

Table 4: Experimental results of domain adaptation on the ACE 2005 corpus

Methods	In-domain (`NW`)			Out-of-domain (`DF`)				In-domain (`DF`)			Out-of-domain (`NW`)			
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	Loss	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	Loss
MSEP-EMD	NA	NA	58.5	NA	NA	52.8	↓5.7	NA	NA	57.9	NA	NA	55.1	↓2.8
SSED	NA	NA	63.7	NA	NA	52.3	↓11.4	NA	NA	62.6	NA	NA	54.8	↓7.8
Hybrid	72.6	55.4	62.9	62.3	39.2	48.1	↓14.8	66.0	42.6	51.8	59.1	48.4	53.3	↑1.5
SELF	**67.6**	**60.6**	**63.9**	**69.0**	**58.7**	**56.7**	**↓7.2**	**70.5**	**48.3**	**57.3**	**69.3**	**51.7**	**59.2**	**↑1.9**

Table 5: Experimental results of domain adaptation on the TAC-KBP 2015 corpus (NA: not released)

More importantly, the adaptability of SELF is relatively close to that of MSEP-EMD. Considering that MSEP-EMD is stable due to using minimal supervision (Peng et al., 2016), we suggest the fully trained networks in SELF may not appear to be extremely inflexible, but on the contrary, they should be transferable for use (Ge et al., 2016).

Robustness in Resource-Poor Settings

There are two resource-poor conditions discussed in this section, including lack of in-domain training data and that of out-domain. Hybrid and SELF are brought into the discussion.

For the former (**in-domain**) case, we went over the numbers of samples used for training in the adaptation experiments, which are shown in Table 6. It can be observed that there is a minimum number of training samples (triggers plus tokens) contained in the domain of `NW`. By contrast, the domain of `bn+nw` contains the smallest number of positive samples (triggers) though an overwhelming number of negative samples (general tokens).

Under such conditions, Hybrid performs better in the domain of `NW` compared to `bn+nw` and `DF` in the three in-domain adaptation experiments (see the column labelled as "In-domain `bn+nw`" in Table 4 as well as "In-domain `NW`" and "In-domain `DF`" in Table 5). It illustrates that Hybrid unnecessarily relies on a tremendous number of training samples to ensure the robustness. But SELF does. It needs far more negative samples than Hybrid because of the following reasons:

- It relies on the use of spurious features to implement self-regulation during training.

Domain	**Training**		**Testing**	
	trigger	token	trigger	token
`bn+nw`	1,721	74,179	343	16,336
`NW`	2,098	31,014	2,813	55,459
`DF`	4,106	10,9275	1,773	43,877

Table 6: Data distribution in the source domains

- For a positive sample, the concerned spurious features (if have) most probably hide in some negative samples.
- It's impossible to be aware of such negative samples. Therefore, taking into consideration as many negative samples as possible may help to increase the probability that the spurious features will be discovered.

This is demonstrated by the fact that SELF obtains better performance in the domain of `bn+nw` but not `NW` (see the column labeled as "Training" in Table 6 and "In-domain" in Table 4 and 5). It may be added that SELF performs worse in `DF` although there are more negative samples used for training (see Table 6). Taking a glance at the number of positive samples, one may find that it is approximately 2.4 times more than that in `bn+nw`. But the number of negative samples in `DF` is only 1.5 times more than that in `bn+nw`. It implies that, if there are more positive samples used for training, SELF needs to consume proportionally more negative samples for self-regulation. Otherwise, the performance will degrade.

For the **out-domain** case, ideally, both Hybrid and SELF encounter the problem that there is lack of target domain data available for training. In this case, SELF displays less performance degradation

Event mentions	Type
*And **<u>it</u>** still does*	Die
*We had no part in **<u>it</u>***	Arrest-Jail
*Nobody questions if **<u>this</u>** is right or ...*	Attack
*And **<u>that</u>** is what ha- what is happening*	End-Position
*Oh, yeah, **<u>it</u>** wasn't perfect*	Marry

Table 7: Examples of pronouns that act as a trigger

(7.2%) than Hybrid (14.8%) when `NW` is used for training. Considering that `NW` contains the minimum number of samples, we would like to believe that SELF is more robust than Hybrid for cross-domain event detection in a resource-poor setting.

Recall and Missing

SELF is able to accurately recall the events whose occurrence is triggered by ambiguous words, such as "*fine*", "*charge*", "*campaign*", etc. These ambiguous words easily causes confusion. For example, "*campaign*" may trigger an `Elect` event or `Attack` in the ACE corpus.

More importantly, SELF fishes out the common words which serve as a trigger, although they are not closely related to any kind of events, such as "*take*", "*try*", "*acquire*", "*become*", "*create*", etc. In general, it is very difficult to accurately recall such triggers because their meanings are not concrete enough, and the contexts may be full of kinds of noises (see example 2 in pg. 1). We observe that Bi-RNN and Hybrid seldom pick them up.

However, SELF fails to recall the pronouns that act as a trigger. This is because they occur in spoken language much more frequently than they occur in written language. The lack of narrative content makes it difficult to learn the relationship between the pronouns and the events. Some real examples collected from ACE are shown in Table 7.

6 Related Work

Event detection is an important subtask of event extraction (Doddington et al., 2004; Ahn, 2006).

The research can be traced back to the pattern based approach (Grishman et al., 2005). Encouraged by the high accuracy and the benefit of easy-to-use, researchers have made great efforts to extract discriminative patterns. Cao et al (2015a; 2015b) use dependency regularization and active leaning to generalize and expand the patterns.

In the earlier study, another trend is to explore the features that best characterize each event class, so as to facilitate supervised classification. A variety of strategies have emerged for converting classification clues into feature vectors (Ahn, 2006; Patwardhan and Riloff, 2009; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013, 2014; Wei et al., 2017). Benefiting from the general modeling framework, the methods enable the fusion of multiple features, and more importantly, they are flexible to use by feature selection. But considerable expertise is required for feature engineering.

Recently, the use of neural networks for event detection has become a promising line of research. The closely related work has been presented in section 5.3. The primary advantages of neural networks have been demonstrated in the work, such as performance enhancement, self-learning capability and robustness.

The generative adversarial network (Goodfellow et al., 2014) has emerged as an increasingly popular approach for text processing (Zhang et al., 2016; Lamb et al., 2016; Yu et al., 2017). Liu et al (2017a) use the adversarial multi-task learning for text classification. We follow the work to create spurious features, but use them to regulate the self-learning process in a single-task situation.

7 Conclusion

We use a self-regulated learning approach to improve event detection. In the learning process, the adversarial and cooperative models are utilized in decontaminating the latent feature space.

In this study, the performance of the discriminator in the adversarial network is left to be evaluated. Most probably, the discriminator also performs well because it is gradually enhanced by fierce competition. Considering this possibility, we suggest to drive the two discriminators in our self-regulation framework to cooperate with each other. Besides, the global features extracted in Li et al (2013)'s work are potentially useful for detecting the event instances referred by pronouns, although involve noises. Therefore, in the future, we will encode the global information by neural networks and use the self-regulation strategy to reduce the negative influence of noises.

Acknowledgments

We thank Xiaocheng Feng and his colleagues who shared the source code of Hybrid with us.

This work was supported by the national Natural Science Foundation of China (NSFC) via Grant Nos. 61525205, 61751206, 61672368.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events, Association for Computational Linguistics (ACL'06)*. Association for Computational Linguistics, pages 1–8. http://www.aclweb.org/anthology/W06-0901.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on Empirical Methods in Natural Language Processing (EMNLP'06)*. Association for Computational Linguistics, pages 120–128. http://www.aclweb.org/anthology/W06-1615.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

Kai Cao, Xiang Li, Miao Fan, and Ralph Grishman. 2015a. Improving event detection with active learning. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP'15)*. pages 72–77. http://www.aclweb.org/anthology/R15-1010.

Kai Cao, Xiang Li, and Ralph Grishman. 2015b. Improving event detection with dependency regularization. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP'15)*. pages 78–83. http://www.aclweb.org/anthology/R15-1011.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. volume 1, pages 409–419. https://doi.org/10.18653/v1/P17-1038.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, Jun Zhao, et al. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL'15)*. pages 167–176. https://doi.org/10.3115/v1/P15-1017.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML'08)*. ACM, pages 160–167.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *LREC*. volume 2, pages 1–4. http://www.aclweb.org/anthology/L04-1011.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the tac kbp 2015 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2015 Workshop, National Institute of Standards and Technology (TAC'15)*. pages 16–17.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. volume 2, pages 66–71. https://doi.org/10.18653/v1/P16-2011.

Tao Ge, Lei Cui, Baobao Chang, Zhifang Sui, and Ming Zhou. 2016. Event detection with burst information networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 3276–3286.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. volume 2, pages 369–373. https://doi.org/10.18653/v1/P16-2060.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's English ACE 2005 system description. *ACE'05* .

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. RPI BLENDER TAC-KBP2015 system description. In *Proceedings of Text Analysis Conference (TAC'15)*.

Yu Hong, Xiaobin Wang, Yadong Chen, Jian Wang, Tongtao Zhang, Jin Zheng, Dian Yu, Qi Li, Boliang Zhang, Han Wang, et al. 2014. RPI BLENDER TAC-KBP2014 knowledge base population system. In *Proceedings of Text Analysis Conference (TAC'14)*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11)*. Association for Computational Linguistics, pages 1127–1136. http://www.aclweb.org/anthology/P11-1113.

Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. 2017. Stacked generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. volume 2, page 4.

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

Md M Islam, Xin Yao, and Kazuyuki Murase. 2003. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on neural networks* 14(4):820–834.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. pages 2342–2350.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL'13)*. pages 73–82. http://www.aclweb.org/anthology/P13-1008.

Qi Li, Heng Ji, HONG Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. pages 1846–1851. https://doi.org/10.3115/v1/D14-1198.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*. Association for Computational Linguistics, pages 789–797. http://www.aclweb.org/anthology/P10-1081.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* .

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Advances in neural information processing systems*. pages 469–477.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. volume 1, pages 1034–1043. https://doi.org/10.18653/v1/P16-1098.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742* https://doi.org/10.18653/v1/P17-1001.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016b. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. https://doi.org/10.18653/v1/P16-1201.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017b. Exploiting argument information to improve event detection via supervised attention mechanisms 1:1789–1797. https://doi.org/10.18653/v1/P17-1164.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'13)*. volume 13, pages 746–751. http://www.aclweb.org/anthology/N13-1090.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16)*. pages 300–309. https://doi.org/10.18653/v1/N16-1034.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL'14)*. pages 68–74. https://doi.org/10.3115/v1/P14-2012.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL'15)*. pages 365–371. https://doi.org/10.3115/v1/P15-2060.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. pages 886–891. https://doi.org/10.18653/v1/D16-1085.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*. Association for Computational Linguistics, pages 151–160. http://www.aclweb.org/anthology/D09-1016.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. pages 392–402. https://doi.org/10.18653/v1/D16-1038.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL'13)*. pages 1498–1507. http://www.aclweb.org/anthology/P13-1147.

Mark Sammons, Haoruo Peng, Yangqiu Song, Shyam Upadhyay, Chen-Tse Tsai, Pavankumar Reddy, Subhro Roy, and Dan Roth. 2015. Illinois CCG TAC 2015 event nugget, entity discovery and linking, and slot filler validation systems. In *Proceedings of Text Analytics Conference (TAC'15)*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*. Association for Computational Linguistics, pages 384–394. https://doi.org/http://www.aclweb.org/anthology/P10-1040.

Mengqiu Wang and Christopher D Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP'13)*. pages 1285–1291. https://doi.org/http://www.aclweb.org/anthology/I13-1183.

Sam Wei, Igor Korostil, Joel Nothman, and Ben Hachey. 2017. English event detection with translated language features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. volume 2, pages 293–298. https://doi.org/10.18653/v1/P17-2046.

Dian Yu, Xiaoman Pan, Boliang Zhang, Lifu Huang, Di Lu, Spencer Whitehead, and Heng Ji. 2016. RPI BLENDER TAC-KBP2016 system description. In *Proceedings of Text Analysis Conference (TAC'16)*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'17)*. pages 2852–2858.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*. volume 21.

Context-Aware Neural Model for Temporal Information Extraction

Yuanliang Meng
Text Machine Lab for NLP
Department of Computer Science
University of Massachusetts Lowell
ymeng@cs.uml.edu

Anna Rumshisky
Text Machine Lab for NLP
Department of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

Abstract

We propose a context-aware neural network model for temporal information extraction, with a uniform architecture for event-event, event-timex and timex-timex pairs. A Global Context Layer (GCL), inspired by the Neural Turing Machine (NTM), stores processed temporal relations in the narrative order, and retrieves them for use when the relevant entities are encountered. Relations are then classified in this larger context. The GCL model uses long-term memory and attention mechanisms to resolve long-distance dependencies that regular RNNs cannot recognize. GCL does not use postprocessing to resolve timegraph conflicts, outperforming previous approaches that do so. To our knowledge, GCL is also the first model to use an NTM-like architecture to incorporate the information about global context into discourse-scale processing of natural text.

1 Introduction

Extracting information about the order and timing of events from text is crucial to any system that attempts an in-depth natural language understanding, whether related to question answering, temporal inference, or other related tasks. Earlier temporal information extraction (TemporalIE) systems tended to rely on traditional statistical learning with feature-engineered task-specific models, typically used in succession (Yoshikawa et al., 2009; Ling and Weld, 2010; Sun et al., 2013; Chambers et al., 2014; Mirza and Minard, 2015).

Recently, there have been some attempts to extract temporal relations with neural network models, particularly with recurrent neural networks (RNN) models (Meng et al., 2017; Cheng and Miyao, 2017; Tourille et al., 2017) and convolutional neural networks (CNN) (Lin et al., 2017). These models predominantly use token embeddings as input, avoiding handcrafted features for each task. Typically, neural network models outperform traditional statistical models. Some studies also try to combine neural network models with rule-based information retrieval methods (Fries, 2016). These systems require different models for different pair types, so several models must be combined to fully process text.

A common disadvantage of all these models is that they build relations from *isolated* pairs of entities (events or temporal expressions). This context-blind, pairwise classification often generates conflicts in the resulting timegraph. Common ways of ameliorating the conflicts is to apply some *ad hoc* constraints to account for basic properties of relations (e.g. transitivity), often without considering the content of the text *per se*. For example, Ling and Weld (2010) designed transitivity formulae, used with local features. Sun (2014) proposed a strategy that "prefers the edges that can be inferred by other edges in the graph and remove the ones that are least so". Another approach is to use the results from separate classifiers to rank results according to their general confidence (Mani et al., 2007; Chambers et al., 2014). High-ranking results overwrite low-ranking ones. Meng et al. (2017) used a greedy pruning algorithm to remove weak edges from the timegraph until it is coherent.

When humans read text, we certainly do not follow the procedure of interpreting interpret relations only locally first, and later come up with a compromise solution that involves all the entities. Instead, if local information is insufficient, we consider the relevant information from the wider context, and resolve the ambiguity as soon as possible. The resolved relations are stored in our memory as "context" for further processing. If the later evidence suggests our early interpretation was wrong, we can correct it.

This paper proposes a model to simulate such mechanisms. Our model introduces a Global Context Layer (GCL), inspired by the Neural Turing Machine (NTM) architecture (Graves et al., 2014), to store processed relations in narrative order, and retrieve them for use when related entities are encountered. The stored information can also be updated if necessary, allowing for self-correction.

This paper's contributions are as follows. To our knowledge, this is the first attempt to use neural network models with updateable external memory to incorporate global context information for discourse-level processing of natural text in general and for temporal relation extraction in particular. It gives a uniform treatment of *all* pairs of temporally relevant entities. We obtain state-of-the-art results on TimeBank-Dense, which is a standard benchmark for TemporalIE.

2 Dataset

We train and evaluate our model on TimeBank-Dense[1] (Chambers et al., 2014). There are 6 classes of relations: SIMULTANEOUS, BEFORE, AFTER, IS_INCLUDED, INCLUDES, and VAGUE TimeBank-Dense annotation aims to approximate a complete temporal relation graph by including all intra-sentential relations, all relations between adjacent sentences, and all relations with document creation time. TimeBank-Dense is one of the standard benchmarks for intrinsic evalution of TemporalIE systems. We follow the experimental setup in Chambers et al. (2014), which splits the corpus into training/validation/test sets of 22, 5, and 9 documents, respectively. Previous publications often use the micro-averaged F1 score, which is equivalent to accuracy in this case. We also rely on the micro-averaged F1 score for model selection and evaluation.

Following Meng et al. (2017), we augment the data by flipping all pairs, except for relations involving document creation time (DCT). In other words, if a pair (e_i, e_j) exists, we add (e_j, e_i) to the dataset with the opposite label (e.g. BEFORE becomes AFTER). The augmentation applies to the validation and test sets also. In the final evaluation, a double-checking technique picks one result from the two-way classification, based on output scores. The dataset is heavily imbalanced. The training set has as much as 44.1% VAGUE labels, whereas only 1.8% labels are SIMULTANEOUS. We did not do any up-sampling or down-sampling.

[1] https://www.usna.edu/Users/cs/nchamber/caevo/#corpus

3 System

Our system has two main components. The first one is a pairwise relation classifier, and the other is the Global Context Layer (GCL). The pairwise relation classifier follows the architecture designed by Meng et al. (2017), which used the dependency paths to the least common ancestor (LCA) from each entity as input. We train the first component first, and then assemble them in a combined neural network to continue training. Fig. 1 gives an overview of the system.

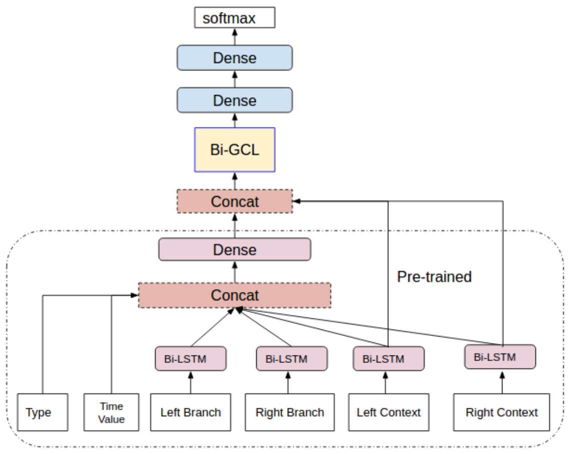


Figure 1: System overview. Originally, the pre-trained system has one more dense layer and an output layer, but they are truncated before combination. The max pooling layers on top of each Bi-LSTM layers are omitted here.

3.1 Global Context Layer

The Global Context Layer (GCL) we propose is inspired by the Neural Turing Machine (NTM) architecture, which is an extension of a recurrent neural network with external memory and an attention mechanism for reading and writing to that memory. NTM has been shown to perform basic tasks such as copying, sorting, and associative recall (Graves et al., 2014). The external memory not only enables a large (theoretically infinite) capacity for information storage, but also allows flexible access based on attention mechanisms.

Essentially, GCL is a specialized form of NTM, which eliminates some parameters to facilitate training, and specializes some functions to impose restrictions. While not as powerful as the canonical NTM, it is more suitable for the task of retaining and updating global context information.

3.1.1 Motivation

Vanilla RNNs struggle with capturing long-distance dependencies. Gated RNNs such as LSTM have trainable gates to address the "vanishing and exploding gradient" problem (Hochreiter and Schmidhuber, 1997). At each time step, it chooses what to memorize and forget, so patterns over arbitrary time intervals can be recognized. However, the memory in LSTM is still *short-term*. No matter how long the cell states keep certain information, once it is forgotten, it gets lost forever. Such a mechanism suffices for modeling contiguous sequences. For example, sentences are naturally fit units for such models, since a sentence starts only after the preceding sentence is finished, and LSTM may be an adequate tool to process sentences. However, when the sequences are not contiguous, as in temporal and other discourse-scale relations, LSTM models do not have the capability to look for input pieces across sequences.

When humans read text, discourse-level information is often distributed across the full scope of the text. To fully understand an article, we must be able to organize the processed information across sentences and paragraphs. In particular, to interpret temporal relations between entities in a sentence, sometimes we also look at relations with other entities elsewhere in the text. Such entities or relations form no regular sequences, and only a system with long-term memory as well as attention mechanisms can process them. An NTM-like architecture has an external memory with attention mechanisms, so it is an ideal candidate for such tasks. Furthermore, unlike the models that use attention over inputs (Vinyals et al., 2015; Kumar et al., 2016), NTM-like models are capable of updating previously stored representations. We describe below the GCL architecture that we use to store and update the global context information.

3.1.2 Reading

The input to the GCL layer is a concatenation of three layers from the pairwise neural network. Two of these are the entity context representation layers, encoded by the two LSTM branches. The other is the penultimate hidden layer before the output layer, which encodes the relation. We can write them as $[\mathbf{e_1}, \mathbf{e_2}, \mathbf{x}]$. The context representations are used as "keys" to uniquely identify the

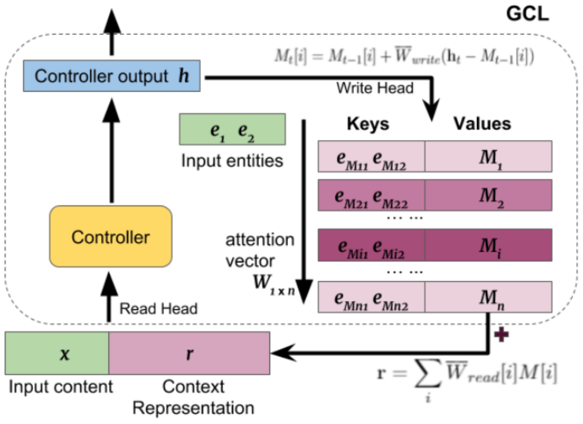


Figure 2: GCL computing attention weights. Input entity representations are compared to the Key section of GCL memory. Slots with the same or similar entities get more attention.

entities. Note that we use flat context embeddings, rather than dependency path embeddings, because dependency paths tend to be short and will also vary for the same entity, depending on the other entity in the pair. As such, they do not provide a unique way to represent an entity.

The original design of NTM has a complex addressing mechanism for reading, which also makes it difficult to train. An important difference in GCL is that we separate the "key" component from the "content" component of memory. Each memory slot $S[i]$ consists of $[K[i]; M[i]]$, where S is the whole memory with n slots, $i \leq n$ is the index, K is the key and M is the content. Addressing is only performed on the key component. The key component stores the representation of two entities, provided by the layers encoding the flat entity context.

$$K[i] = \mathbf{e_{M1}}[i] \oplus \mathbf{e_{M2}}[i] \tag{1}$$

Here $\oplus$ is the concatenation operator. In the GCL model, the read head computes a reading weight $W_{n\times 1}$ from the input entity representations $\mathbf{e_1}$, $\mathbf{e_2}$ and the entity representations $\mathbf{e_{M1}}$, $\mathbf{e_{M2}}$ in memory (i.e., the keys in each memory slot). The first step is to compute the distance between current input and the memory columns, as shown in Eq. 2. $D[i]$ is the Euclidean distance between the input key and the memory key of slot $M[i]$. $D'[i]$ is computed after flipping the two entities. We do so because the order of entities in a pair should not affect their relevance.

$$\begin{aligned} D[i] &= \frac{1}{Z}||\mathbf{e_1} \oplus \mathbf{e_2} - \mathbf{e_{M1}}[i] \oplus \mathbf{e_{M2}}[i]||_2^2 \\ D'[i] &= \frac{1}{Z'}||\mathbf{e_2} \oplus \mathbf{e_1} - \mathbf{e_{M1}}[i] \oplus \mathbf{e_{M2}}[i]||_2^2 \end{aligned} \tag{2}$$

where $Z = \sum_i D[i]$ is the normalization factor, and so is Z' for the flipped case. The reading weight is then calculated as in Eq. 3, where $\mathbf{1}_{n\times 1}$ is a vector of all 1's.

$$W[i] = \max(\mathrm{softmax}(\mathbf{1} - \mathbf{D})[i], \\ \mathrm{softmax}(\mathbf{1} - \mathbf{D}')[i]) \tag{3}$$

Every element of W represents the relevance of the corresponding memory slot (see Fig. 2). Often it is still too blurred and needs to be further sharpened as in Eq. 4. Here β is a positive number. W^β is a point-wise exponential function by power of β. A large β allows "winner takes all", so only the most relevant memory slots are read.

$$\overline{W}_{read} = \mathrm{softmax}(W^\beta) \tag{4}$$

Parameter β could be a constant, or could be trainable. Our model computes it from the current input $\mathbf{x}_t$ and the previous output h_{t-1}, and thus it varies in each time step. W_{sharp} and b_{sharp} are trainable weights and bias, c_β is a constant, and $ReLU$ is the rectified linear function.

$$\beta_t = ReLU(W_{sharp}[\mathbf{x}_t, \mathbf{h}_{t-1}] + b_{sharp}) + c_\beta \tag{5}$$

With the sharpened reading weight vector, we are able to obtain the read vector $\mathbf{r}_{1\times m}$ from M as a weighted sum, as in Eq. 6.

$$\mathbf{r} = \sum_i \overline{W}_{read}[i]M[i] \tag{6}$$

Generally speaking, the depth of memory M should be large enough to allow sparse encoding, so that crucial information is not lost after the summation. The read vector then contains contextual information relevant to current input. Both the read vector and the current input are fed to the controller, yielding GCL output. Unlike the canonical NTM, the CGL model does not have a trainable gate interpolating the $\overline{W}_t$ computed at time t, with $\overline{W}_{t-1}$ computed at previous time $t-1$. The weight vector is not passed to next time step, so the attention has no "inertia".

We tried two variants of the controller: (a) *state-tracking*, with an LSTM layer, and (b) *stateless*, with a dense layer. An LSTM controller has an internal state, and also has gates to select input and output. If the input data and/or the read vector from M have regular patterns with respect to time steps, an LSTM controller would be a better choice. For the specific task of temporal relation extraction, we saw no difference in performance.

3.1.3 Writing

The controller produces an output h_t, which is sent to the next layer and also used to update M. Similar to reading, the first step of writing procedure is to compute an attention weight vector over the slots of M. As described above, the reading procedure computes a weighted sum over slots of M. The writing procedure writes a weighted h_t to each slot. The attention mechanism here is *de facto* a soft addressing mechanism. The slots with a higher attention value will be the addresses which will get more of an update.

The same weight vector W computed as shown in Eq. 3 is used for writing. However, an additional operation is introduced for writing. Recall that the weights are computed from entity representations. If the input entities are $\mathbf{e_1}$ and $\mathbf{e_2}$, the weight vector should have high values in the slots corresponding to $\mathbf{e_1}$ and/or $\mathbf{e_2}$. But we may not always want relevant memory slots to be overwritten. Instead, additional information can be written to a different slot. Additionally, when M is relatively empty, as at the beginning, the addressing mechanism may treat all slots equally, and uniformly update all slots in the same way. In this case we want the weight vector to shift each time, so M can diversify fast.

Therefore we use a shift function similar to the canonical NTM. The idea is to compute a shifted weight vector $\widetilde{W}$ by convolving W with a shift kernel $\mathbf{s}$ which maps a shift distance to a probability value. For example, $\mathbf{s}(-1) = 0.2$, $\mathbf{s}(0) = 0.5$, $\mathbf{s}(1) = 0.3$ means the probabilities of shifting left, no shifting, and shifting right are 0.2, 0.5, 0.3, respectively. Generally speaking, we want $\mathbf{s}$ to give zeros for most shift distances, so the shifting operation is limited to a small range.

$$\widetilde{W}[i] = \sum_{j=0}^{n-1} W[j]\mathbf{s}[i-j] \tag{7}$$

At each time step, the shift kernel depends on current input and output. If the allowed shift range is [-s/2, +s/2], we train a weight W_s and bias b_s to calculate the shift weights $C_{s\times 1}$,

$$C_t = \mathrm{softmax}(W_s[\mathbf{x}_t, \mathbf{h}_t] + b_s) \tag{8}$$

Then the weights are mapped to a circulant kernel to perform the convolution in Eq. 7, the final output is $\widetilde{W}$.

Finally, the sharpening still needs to be applied. For the writing procedure, both addressing and shifting are "soft" in nature, and thus could yield a blurred outcome. Again, we train the weights to obtain a sharpening parameter γ each time, and perform softmax over $\widetilde{W}$.

$$\gamma_t = ReLU(W_{sharp}[\mathbf{x}_t, \mathbf{h}_t] + b_{sharp}) + c_\gamma \quad (9)$$

$$\overline{W}_{write} = \text{softmax}(\widetilde{W}^\gamma) \quad (10)$$

$\widetilde{W}^\gamma$ is the point-wise exponential function, over the shifted weight vector. c_γ is a positive constant.

The original NTM model has gates for interpolating $\widetilde{W}^\gamma$ at the current time with the one computed at the previous time step, but we omit this operation. We also omit the *erase vector* and the *add vector*, so $\overline{W}_{write}$ fully controls what to overwrite in M and what to retain. As a result, the writing operation can be expressed as:

$$M_t[i] = M_{t-1}[i] + \overline{W}_{write}[i](\mathbf{h}_t - M_{t-1}[i]) \quad (11)$$

The first term in Eq. 11 is the memory in the previous time step, and the second term is the update. We update the keys in the same way. As we can see, the keys come from entity representations, but are not exactly the same, due to $\overline{W}_{write}$.

$$K_t[i] = K_{t-1}[i] + \overline{W}_{write}[i](\mathbf{e_1} \oplus \mathbf{e_2} - K_{t-1}[i]) \quad (12)$$

3.1.4 GCL vs. Canonical NTM

We highlight below some major differences between the canonical NTM and the GCL model. Typically, NTM computes the keys from input and output for accessing different memory addresses. In GCL, the keys are simply the entity representations $[\mathbf{e_1}, \mathbf{e_2}]$ from input, in either order. The key function effectively involves slicing and flipping the input. Further discussion of the differences between the GCL addressing mechanism and some of the other NTM variations is provided in Sec. 5.

Another major difference is that we do not use any gates to interpolate the attention vector at the current time step with the one from the previous time step. Instead, the previous attention vector is totally ignored. Since we do not compute the erase vector or the add vector, this allows the attention vector to fully control memory updates.

In addition, we unified the trainable weights for calculating β and γ at each time step. We found these parameters not to be crucial, and setting them to be constant does not affect the results. We also do not shift attention for reading. A possible advantage of shifting attention is that neighboring slots of the focus can also be accessed, providing a way to simulate associative recall. This is based on the fact that the writing procedure tends to write similar memories close to each other. However, in this study we want the reading procedure to be restricted. Associative recall can be realized from attention vector itself, without shifting.

3.2 Pairwise Classification Model

The pairwise model classifies individual entity pairs, where entities are events and time expressions (timexes). In other words, for each pair, we only use the local context, and the relation of one pair does not affect the classification results for other pairs. We follow the architecture proposed in Meng et al. (2017), but with the following changes: (1) all three types of pairs are handled by the same neural network, rather than by three separately trained models; (2) the neighboring words (a flat context) of entity mentions are used to generate input, in addition to words on syntactic dependency paths; (3) all timex-timex pairs are included as well, not only event-timex and event-event pairs; (4) every pair is assigned a 3-dimensional "time value", to approximate the rule-based approach when possible.

3.2.1 Event Pairs and Event-Timex Pairs

TimeBank-Dense dataset labels three types of pairs: intra-sentence, cross-sentence and document creation time (DCT). For intra-sentence pairs and cross-sentence pairs, we follow Meng et al. (2017). The shortest dependency path between the two entities is identified, and the word embeddings from the path to the least common ancestor for each entity are processed by two LSTM branches, with a separate max pooling layer for each branch. Path to the root is used for cross-sentence relations. For relations with the DCT, we use a single word now as a placeholder for the DCT branch. Unlike Meng et al. (2017), we allow the model to accept all three pair types, with a "pair type" feature as a component of input, defined as an integer with the value 1, -1 or 0, respectively.

In addition to the shortest dependency path, our model also uses a flat local context window, that is, the words around each entity mention, regardless of syntactic structures. For an entity starting with word w_i, the local context window is 5 words to the left and 10 words to its right i.e. $w_{i-5}w_{i-4}...w_iw_{i+1}...w_{i+10}$. The windows are cut short at the edge of a sentence, or when the second entity in encountered. By using this context window, the words between two entities are often used twice by the system, and thus given more consideration. To inform the system of other entity mentions, we also add special input tokens at the locations where events and timexes are tagged. The embeddings of the special tokens are uniformly initialized, and automatically tuned during the training process.

3.2.2 Timex Pairs

The method described in Meng et al. (2017) classifies timex pairs by handcrafted rules and then adds them to the final results prior to postprocessing. Since timexes have concrete time values, a rule-based method would seem appropriate. However, since our model uses global context to help classify relations and timex-timex pairs enrich the global context representation, we design a way for a common classifier model to handle such pairs.

When DCT is not involved, timex pairs are created the same way as cross-sentence pairs, that is, path to the root is used for each entity. DCT is represented by the placeholder word *now*. In addition to the word-based representations, another input vector is used to simulate the rule-based approach, to be explained next.

3.2.3 Time Value Vectors

Every timex tag has a time value, following the ISO-8601 standard. Every value can be mapped to a 2D vector of real values $(start, end)$. For a pair we use the subtraction of the vectors to represent the difference. Suppose we have timexes in below:

```
THE HAGUE, Netherlands (AP)_ The World Court
<TIMEX3 tid="t21" type="DATE" value="1998-02-27"
temporalFunction="true" functionInDocument="NONE"
anchorTimeID="t0">Friday</TIMEX3> rejected U.S.
and British objections to a Libyan World Court
case that has  the trial of two Libyans suspected
of blowing up a Pan Am jumbo jet over Scotland in
<TIMEX3 tid="t22" type="DATE" value="1988"
temporalFunction="false"
functionInDocument="NONE">1988</TIMEX3>.
```

The first timex can be represented as (1998 + 1/12 + 26/365, 1998 + 1/12 + 26/365) = (1998.155, 1998.155), and the second one (1988, 1988 + 364/365) = (1988, 1988.997). The difference of the values are put in the sign function, to obtain the representation: (sign(1988 - 1998.155), sign(1988.997 - 1998.155)) = (-1, -1). Vector (-1, -1) clearly indicates the AFTER relation between $t21$ and $t22$. We set the minimum interval to be a day, which is generally sufficient for our data. The DURATION timexes are not considered, and word-based input vectors are used to represent them.

In order to make all the input data have the same shape, we assign the time value vector to all pairs, even if a timex is not involved. For non-timex pairs, a vector (-1, 0, 0) is used. The first element -1 to indicate a "pseudo" time value. Real timex pairs have the first value of 1, so the example we just discussed would be assigned a vector (1, -1, -1). The time value vectors allow the model to take advantage of rule-based information.

3.3 Combining Two Components

We tried training the two components in a combined system, but found it slow to converge. In our experiments, we trained the pairwise model first, froze it, and then combined it with the GCL layer to train the GCL. This method also helps us observe whether the GCL component alone improves results, given the same input.

We tried combining the systems in two ways. One is to connect the output layer of the pre-trained model to GCL, and the other is to slice the pre-trained model and connect its hidden layer to GCL. All the GCL layers are bi-directional, averaging forward and backward passes. By connecting the output layer, which has a softmax activation, we hand the final decisions made by the pairwise model to GCL. On the other hand, the hidden layer provides higher layers with cruder but richer information. We found that the latter performs better. It is also possible to train the two components together from scratch. In this case, the learning rate has to be set much lower to assure convergence, and the training requires more epochs.

4 Experiments

For all the experiments, hyperparameters including the number of epochs are tuned with the validation set only. Training data is segmented into chunks. Each chunk contains relation pairs in the narrative order. The size of chunks is randomly chosen from [40, 60, 80, 120, 160] at the beginning of each epoch of training. The GCL maintains a memory for each chunk, and clears it at the end of a chunk. The idea here is to train the model on short paragraphs to avoid overfitting.

To introduce further randomness, the chunks are rotated for each epoch. For a specific training file, if chunk i starts with pair n_i in epoch 1, in epoch

2, chunk i will start with pair $n_i + chunksize + 11$. 11 is a prime number we chose to assure each epoch observes different compositions of chunks. By doing the rotation, some pairs in the final chunk of epoch 1 will show up in the first chunk in epoch 2 as well. However, within each chunk, we do not randomize pairs, so narrative order is preserved at this level. We also do not shuffle the chunks, but only rotate them.

Evaluation on the test set uses only one chunk for each file (chunk size is the number of pairs). Each relation pair is only processed once, without "multiple rounds of reading". Thus, we essentially train the model to read shorter paragraphs (varied in length), but test it on long articles.

4.1 Pairwise Model

As described in Section 3.2, the pairwise classifier has the following input vectors: left and right shortest path branches, two flat context vectors, a pair type flag, and a time value vector. Word embeddings are initialized with glove.840B.300d word vectors[2], and set to be trainable. The Bi-LSTM layers are followed by max-pooling. The two hidden layers have size 512 and 128, respectively. We train this model for 40 epochs, using the RMSProp optimizer (Tieleman and Hinton, 2012). The learning rate is scheduled as $lr = 2 \times 10^{-3} \times 2^{-\frac{n}{5}}$, where n is the number of epochs.

The middle block of Table 1 shows the performance of the pairwise model after applying double-checking. Since all pairs are flipped, double-checking combines results from (e_i, e_j) and (e_j, e_i), picking the label with the higher probability score, which typically boosts performance. The results without double-checking show similar trends.

4.2 GCL model

After training the pairwise model, we combine it with GCL. Unless otherwise indicated, the results reported in this section use the model configuration that connects the hidden layer (rather than the output layer) of the pairwise model with a bidirectional GCL layer. The bidirectional GCL is realized as the average of a forward GCL and a backward GCL, each producing a sequence. Then two more hidden layers are put on top of it, followed by an output layer. All the layers in the pre-trained pairwise model are set to be untrainable. The two trainable hidden layers have sizes 512 and 128, respectively, with ReLU activtion and 0.3 dropout after each one. The GCLs have 128 memory slots. Learning rate is scheduled as $lr = 2 \times 10^{-4} \times 2^{-\frac{n}{2}}$. In the experiments, we found the models converge quite fast with respect to the number of epochs. It is not surprising because the lower layers are already well trained, and frozen (no updating). After the 5th epoch, the training accuracy typically reaches 0.95. We stop training after 10 epochs.

Model	Micro-F1	Macro-F1
CAEVO (not NN model)	.507	
CATENA (not NN model)	.511	
Cheng et al. 2017	.520[3]	
Meng et al. 2017		.519
pairwise	.535	.528
Two more hidden layers	.539	.532
GCL w/ state-tracking controller	.545	**.538**
GCL w/ stateless controller	**.546**	**.538**
GCL w/ pre-trained output layer	.541	.536

Table 1: Results on the test set. The GCL models use the same hyperparameters, if possible. The two models on the top do not use neural networks. The results in the two lower blocks all use double-check. "Two more hidden layers" means adding two dense layers on top of the pre-trained model without using GCL. The last row corresponds to connecting the output layer of a pre-trained model to GCL layers with stateless controller.

The bottom block of Table 1 presents the results, showing that all models from the present paper outperform existing models from the literature. One may argue the combined system adds more hidden layers over a pre-trained model, which contributes to the improvement in performance. We show a comparison to a baseline model which adds two dense layers on top of the pairwise model, without the GCL. The configuration of the two layers is the same as we used for the GCL models. The result shows that the performance is slightly higher than what we get from the pairwise model, but the difference is smaller than what we get from GCL models – suggesting that the performance improvement with GCL models is not just due to more parameters. We also tried adding an LSTM layer on top of the pre-trained model, and found the system cannot converge. It again confirms that GCL is more powerful than LSTM in handling irregular time series.

We found no difference in performance between the stateless controller and state-tracking controller. Connecting the output layer of the pre-

[2]https://nlp.stanford.edu/projects/glove/

[3]This result does not include timex-timex pairs, which is 3% of total test instances.

trained model to GCL seems to generate weaker results than connecting the hidden layer, although it also outperforms the pairwise model, and all previous models in literature.

We performed significance testing to compare the pairwise model and the GCL-enabled model. A paired one-tailed t-test shows the results from the GCL model are significantly higher than results from pairwise model (p-value 0.0015). While significant, the improvement is relatively small, we believe due in part to the small size of Timebank-Dense dataset.

4.3 Case Study

To illustrate the difference in performance of the pairwise model and the GCL model, we created a sample paragraph in which long-distance dependencies and references to DCT are needed to resolve some of the temporal relations:

> John *met* Mary in Massachusetts when they *attended* the same university. They are *getting married* in 2019, 2 years after their *graduation*. But this year, they have *relocated* to New Hampshire.

We created the gold standard annotation for this text with 5 events, 2 timexes, and 24 TLINKS (see appendix)[4]. We set the DCT to an arbitrary date "2018-04-01". There are no VAGUE or SIMULTANEOUS relations.

For this paragraph, the pairwise model yields an accuracy (i.e. micro-averaged F1) of 0.292, while the GCL-enabled model yields 0.417. Overall, the GCL-enabled model assigns 6 VAGUE labels while the pairwise assigns 11. It reflects the fact that GCL tries to infer relations from otherwise vague evidence. For example, it is difficult to infer the relation between *met* and *2019* from the local context (without DCT, particularly), so the pairwise model labels it as VAGUE, while the GCL-enabled model correctly assigns BEFORE.

Recall that the GCL is placed on top of a pretrained pairwise model, so the mistakes made by the pairwise model propagate to GCL. For example, the pairwise model incorrectly classifies *2019* as BEFORE *graduation* – perhaps, due to a somewhat unusual syntax. But the GCL-enabled system assigns it a VAGUE label, probably as a way to compromise. In the TimeBank-Dense test data, VAGUE cases dominate, which may have made it more difficult for GCL to assign proper labels. In the future, we believe it may be better to omit writing (and reading) the VAGUE relations to/from GCL.

[4]Note that in TimeBank-Dense, no TLINKS are associated DURATION timexes, so 2 years is not annotated

4.4 Error Analysis

Table 2 shows the overall performance for each relation using the GCL system with the stateless controller. Since we flip pairs and use double-checking to pick one result for each pair, BEFORE/AFTER and IS_INCLUDED/INCLUDES are actually treated in the same way, respectively. Here we map the results to original pairs, in order to compare to other systems.

	Predicted labels						
	SIMUL	**BEF**	**AFT**	**IS_INCL**	**INCL**	**VAG**	Total
SIMUL	10	0	9	2	1	17	39
BEF	0	327	27	15	5	215	589
AFT	1	26	208	4	5	184	428
IS_INCL	1	27	3	59	2	67	159
INCL	0	16	9	2	19	70	116
VAG	1	171	87	28	17	596	900

Table 2: Overall results per relation.

As the table shows, the VAGUE relation causes the most trouble. It is not only because VAGUE is the largest class, but also because it is often semantically ambiguous, so even human experts have low inter-annotator agreement. If we allow a relatively sparse labeling of data, and use other evaluation methods (e.g. question answering), the VAGUE class is not likely to have similar effects.

We also break down the results according to the types of pairs. Compared to other systems, our approach has a big advantage for event-event (E-E) pairs, which is by far the most common (64%) relation pairs for all data, and also requires more complex natural language understanding. Compared to CAEVO, our performance on event-DCT (E-D) and event-timex (E-T) pairs is not that great. CAEVO uses engineered features such as entity attributes, temporal signals, and semantic information from WordNet, which seems to work well in these two cases. We took a closer look at our E-D

Systems	**E-D**	**E-E**	**E-T**	**Overall**
Frequency	14%	64%	19%	97%
CAEVO	**.553**	.494	**.494**	.502
CATENA	.534	.519	.468	.512
Cheng et al. 2017	.546	.529	.471	.520
GCL	.489	**.570**	.487	**.542**

Table 3: Results on the E-D, E-D and E-T pairs. GCL stands for the GCL-enabled system with a stateless controller. Frequencies are percentages in the test set. T-T pairs are not shown here. CAEVO is from Chambers et al. (2014). CATENA is from Mirza and Tonelli (2016)

results, and found that the relatively low performance is mainly caused by misclassifying VAGUE as AFTER. As Table 4 shows, among the 72

	Predicted labels					
	SIMUL	BEF	AFT	IS_INCL	INCL	VAG
SIMUL	0	0	0	0	0	0
BEF	0	57	11	15	6	37
AFT	0	3	36	0	0	10
IS_INCL	0	11	1	31	1	12
INCL	0	0	2	1	3	2
VAG	0	4	20	9	14	25

Table 4: Test results from event and document creation time (E-D) pairs. The rows are true labels and the columns are predicted labels.

VAGUE relations in E-D pairs, 20 are labeled AFTER by our system. In a news article, most events occur before the DCT i.e. the time when the article was written. If the temporal relation is vague, our system tends to guess that the event occurs after the DCT. It is interesting because AFTER only accounts for 16% of all E-D pairs in test data (and about the same in training data), behind BEFORE (41%), VAGUE (21%), and IS_INCLUDED (18%). However, E-D is a relatively small category with only 311 instances in the test set, so it is difficult to draw any a substantive conclusion in this case.

Recall that our model has a uniform architecture for all input types and is trained on event-event, event-timex and event-DCT pairs simultaneously. As a result, its performance is not optimal for some lower-frequency pair types. Tuning the model for each pair type separately, as well as resampling to deal with class imbalance would, perhaps, improve performance. However, the point of these experiments was not to get the largest improvement, but to show that the GCL mechanism can replace heuristic-based timegraph conflict resolution, improving the performance of an otherwise very similar model.

5 Related Work

While the GCL model is inspired by NTM, other NTM variants have also been proposed recently. Zhang et al. (2015) proposed structured memory architectures for NTMs, and argue they could alleviate overfitting and increase predictive accuracy. Graves et al. (2016) proposed a memory access mechanism on top of NTM, which they call Differentiable Neural Computer (DNC). DNC can store the transitions between memory locations it accesses, and thus can model some structured data.

Gülçehre et al. (2016) proposed a Dynamic Neural Turing Machine (D-NTM) model, which allows discrete access to memory. Gülçehre et al. (2017) further simplified the addressing algorithm, so a single trainable matrix is used to get locations for read and write. Both models separate the address section from the content section of memory, as do we. We came up with the idea independently, noting that the content-based addressing in the canonical NTM model is difficult to train. A crucial difference between GCL and these models is that they use input "content" to compute keys. In GCL, the addressing mechanism fully depends on the entity representations, which are provided by the context encoding layers and not computed by the GCL controller. Addressing then involves matching the input entities and the entities in memory.

Other than NTM-based approaches, there are models that use an attention mechanism over either input or external memory. For instance, the Pointer Networks (Vinyals et al., 2015) uses attention over input timesteps. However, it has no power to rewrite information for later use, since they have no "memory" except for the RNN states. The Dynamic Memory Networks (Kumar et al., 2016) has an "episodic memory" module which can be updated at each timestep. However, the memory there is a vector ("episode") without internal structure, and the attention mechanism works on inputs, just as in Pointer Networks. Our GCL model and other NTM-based models have a memory with multiple slots, and the addressing function (attention) dictates writing and reading to/from certain slots in the memory.

6 Conclusion

We have proposed the first context-aware neural model for temporal information extraction using an external memory to represent global context. Our model introduces a Global Context Layer which is able to save and retrieve processed temporal relations, and then use this global context to infer new relations from new input. The memory can be updated, allowing self-correction. Experimental results show that the proposed model beats previous results without resorting to ad-hoc resolution of timegraph conflicts in postprocessing.

Acknowledgments

This project is funded in part by an NSF CAREER award to Anna Rumshisky (IIS-1652742).

References

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional lstm over dependency paths. In *ACL*.

Jason Alan Fries. 2016. Brundlefly at semeval-2016 task 12: Recurrent neural networks vs. joint inference for clinical temporal information extraction. *CoRR*, abs/1606.01433.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Çaglar Gülçehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *CoRR*, abs/1701.08718.

Çaglar Gülçehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2016. Dynamic neural turing machine with soft and hard addressing schemes. *CoRR*, abs/1607.00036.

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

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1378–1387, New York, New York, USA. PMLR.

Chen Lin, Timothy A. Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2017. Representations of time expressions for temporal relation extraction with convolutional neural networks. In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 322–327.

Xiao Ling and Daniel S. Weld. 2010. Temporal information extraction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Inderjeet Mani, Ben Wellner, Marc Verhagen, and James Pustejovsky. 2007. Three approaches to learning tlinks in timeml. *Technical Report CS-07–268, Computer Science Department*.

Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. In *Proc. of the conference on empirical methods in natural language processing (EMNLP)*.

P Mirza and S Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *The 26th International Conference on Computational Linguistics*, pages 64–75. Association for Computational Linguistics.

Paramita Mirza and Anne-Lyse Minard. 2015. Hlt-fbk: a complete temporal processing system for qa tempeval. In *Proc. of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 801–805. Association for Computational Linguistics.

Weiyi Sun. 2014. *Time Well Tell: Temporal Reasoning in Clinical Narratives*. PhD dissertation. Department of Informatics, University at Albany, SUNY.

Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813.

T Tieleman and G Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: A bi-lstm approach for detecting narrative containers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 224–230, Vancouver, Canada. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 405–413, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wei Zhang, Yang Yu, and Bowen Zhou. 2015. Structured memory for neural turing machines. *CoRR*, abs/1510.03931.

Temporal Event Knowledge Acquisition via Identifying Narratives

Wenlin Yao and Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
{wenlinyao, huangrh}@tamu.edu

Abstract

Inspired by the double temporality characteristic of narrative texts, we propose a novel approach for acquiring rich temporal "before/after" event knowledge across sentences in narrative stories. The double temporality states that a narrative story often describes a sequence of events following the chronological order and therefore, the temporal order of events matches with their textual order. We explored narratology principles and built a weakly supervised approach that identifies 287k narrative paragraphs from three large text corpora. We then extracted rich temporal event knowledge from these narrative paragraphs. Such event knowledge is shown useful to improve temporal relation classification and outperform several recent neural network models on the narrative cloze task.

Michael Kennedy graduated with a bachelor's degree from Harvard University in 1980. He married his wife, Victoria, in 1981 and attended law school at the University of Virginia. After receiving his law degree, he briefly worked for a private law firm before joining Citizens Energy Corp. He took over management of the corporation, a non-profit firm that delivered heating fuel to the poor, from his brother Joseph in 1988. Kennedy expanded the organization goals and increased fund raising.

Beth paid the taxi driver. She jumped out of the taxi and headed towards the door of her small cottage. She reached into her purse for keys. Beth entered her cottage and got undressed. Beth quickly showered deciding a bath would take too long. She changed into a pair of jeans, a tee shirt, and a sweater. Then, she grabbed her bag and left the cottage.

Figure 1: Two narrative examples

1 Introduction

Occurrences of events, referring to changes and actions, show regularities. Specifically, certain events often co-occur and in a particular temporal order. For example, people often go to *work* after *graduation* with a degree. Such "before/after" temporal event knowledge can be used to recognize temporal relations between events in a document even when their local contexts do not indicate any temporal relations. Temporal event knowledge is also useful to predict an event given several other events in the context. Improving event temporal relation identification and event prediction capabilities can benefit various NLP applications, including event timeline generation, text summarization and question answering.

While being in high demand, temporal event knowledge is lacking and difficult to obtain. Existing knowledge bases, such as Freebase (Bollacker et al., 2008) or Probase (Wu et al., 2012), often contain rich knowledge about entities, e.g., the birthplace of a person, but contain little event knowledge. Several approaches have been proposed to acquire temporal event knowledge from a text corpus, by either utilizing textual patterns (Chklovski and Pantel, 2004) or building a temporal relation identifier (Yao et al., 2017). However, most of these approaches are limited to identifying temporal relations within one sentence.

Inspired by the double temporality characteristic of narrative texts, we propose a novel approach for acquiring rich temporal "before/after" event knowledge across sentences via identifying narrative stories. The double temporality states that a narrative story often describes a sequence of events following the chronological order and therefore, the temporal order of events matches with their textual order (Walsh, 2001; Riedl and Young, 2010; Grabes, 2013). Therefore, we can easily distill temporal event knowledge if we have identified a large collection of narrative texts. Consider the two narrative examples in figure 1, where the top one is from a news article of New York Times and the bottom one is from a novel book. From the top one, we can easily extract one chronologically ordered event sequence {graduated, marry, attend, receive, work, take over, expand, increase}, with all events related to the main character Michael Kennedy. While some parts of the event sequence are specific to this story, the event sequence contains regular event temporal relations, e.g., people often {graduate} first and then get {married}, or {take over} a role first and then {expand} a goal. Similarly, from the bottom one, we can easily extract another event sequence {pay, jump out, head, reach into, enter, undress, shower, change, grab, leave} that contains routine actions when people take a shower and change clothes.

There has been recent research on narrative identification from blogs by building a text classifier in a supervised manner (Gordon and Swanson, 2009; Ceran et al., 2012). However, narrative texts are common in other genres as well, including news articles and novel books, where little annotated data is readily available. Therefore, in order to identify narrative texts from rich sources, we develop a weakly supervised method that can quickly adapt and identify narrative texts from different genres, by heavily exploring the principles that are used to characterize narrative structures in narratology studies. It is generally agreed in narratology (Forster, 1962; Mani, 2012; Pentland, 1999; Bal, 2009) that a narrative is a discourse presenting a sequence of events arranged in their time order (the plot) and involving specific characters (the characters). First, we derive specific grammatical and entity co-reference rules to identify narrative paragraphs that each contains a sequence of sentences sharing the same actantial syntax structure (i.e., *NP VP* describing *a character did something*) (Greimas, 1971) and mentioning the same character. Then, we train a classifier using the initially identified seed narrative texts and a collection of grammatical, co-reference and linguistic features that capture the two key principles and other textual devices of narratives. Next, the classifier is applied back to identify new narratives from raw texts. The newly identified narratives will be used to augment seed narratives and the bootstrapping learning process iterates until no enough new narratives can be found.

Then by leveraging the double temporality characteristic of narrative paragraphs, we distill general temporal event knowledge. Specifically, we extract event pairs as well as longer event sequences consisting of strongly associated events that often appear in a particular textual order in narrative paragraphs, by calculating Causal Potential (Beamer and Girju, 2009; Hu et al., 2013) between events.

Specifically, we obtained 19k event pairs and 25k event sequences with three to five events from the 287k narrative paragraphs we identified across three genres, news articles, novel books and blogs. Our evaluation shows that both the automatically identified narrative paragraphs and the extracted event knowledge are of high quality. Furthermore, the learned temporal event knowledge is shown to yield additional performance gains when used for temporal relation identification and the Narrative Cloze task. The acquired event temporal knowledge and the knowledge acquisition system are publicly available[1].

2 Related Work

Several previous works have focused on acquiring temporal event knowledge from texts. VerbOcean (Chklovski and Pantel, 2004) used predefined lexico-syntactic patterns (e.g., "X and then Y") to acquire event pairs with the temporal *happens_before* relation from the Web. Yao et al. (2017) simultaneously trained a temporal "before/after" relation classifier and acquired event pairs that are regularly in a temporal relation by exploring the observation that some event pairs tend to show the same temporal relation regardless of contexts. Note that these prior works are limited to identifying temporal relations within individual sentences. In contrast, our approach is designed to acquire temporal relations across sentences in a narrative paragraph. Interestingly, only 195 (1%) out of 19k event pairs acquired by our approach can be found in VerbOcean or regular event pairs learned by the previous two approaches.

Our design of the overall event knowledge acquisition also benefits from recent progress on narrative identification. Gordon and Swanson (2009) annotated a small set of paragraphs presenting stories in the ICWSM Spinn3r Blog corpus (Burton et al., 2009) and trained a classifier using bag-of-words features to identify more stories. (Ceran

[1]`http://nlp.cs.tamu.edu/resources.html`

et al., 2012) trained a narrative classifier using semantic triplet features on the CSC Islamic Extremist corpus. Our weakly supervised narrative identification method is closely related to Eisenberg and Finlayson (2017), which also explored the two key elements of narratives, the plot and the characters, in designing features with the goal of obtaining a generalizable story detector. But different from this work, our narrative identification method does not require any human annotations and can quickly adapt to new text sources.

Temporal event knowledge acquisition is related to script learning (Chambers and Jurafsky, 2008), where a script consists of a sequence of events that are often temporally ordered and represent a typical scenario. However, most of the existing approaches on script learning (Chambers and Jurafsky, 2009; Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016) were designed to identify clusters of closely related events, not to learn the temporal order between events though. For example, Chambers and Jurafsky (2008, 2009) learned event scripts by first identifying closely related events that share an argument and then recognizing their partial temporal orders by a separate temporal relation classifier trained on the small labeled dataset TimeBank (Pustejovsky et al., 2003). Using the same method to get training data, Jans et al. (2012); Granroth-Wilding and Clark (2016); Pichotta and Mooney (2016); Wang et al. (2017) applied neural networks to learn event embeddings and predict the following event in a context. Distinguished from the previous script learning works, we focus on acquiring event pairs or longer script-like event sequences with events arranged in a complete temporal order. In addition, recent works (Regneri et al., 2010; Modi et al., 2016) collected script knowledge by directly asking Amazon Mechanical Turk (AMT) to write down typical temporally ordered event sequences in a given scenario (e.g., shopping or cooking). Interestingly, our evaluation shows that our approach can yield temporal event knowledge that covers 48% of human-provided script knowledge.

3 Key Elements of Narratives

It is generally agreed in narratology (Forster, 1962; Mani, 2012; Pentland, 1999; Bal, 2009) that a narrative presents a sequence of events arranged in their time order (the plot) and involving specific characters (the characters).

Plot. The plot consists of a sequence of closely related events. According to (Bal, 2009), an event in a narrative often describes a “transition from one state to another state, caused or experienced by actors”. Moreover, as Mani (2012) illustrates, a narrative is often “an account of past events in someone’s life or in the development of something”. These prior studies suggest that sentences containing a plot event are likely to have the actantial syntax “NP VP”[2] (Greimas, 1971) with the main verb in the past tense.

Character. A narrative usually describes events caused or experienced by actors. Therefore, a narrative story often has one or two main characters, called protagonists, who are involved in multiple events and tie events together. The main character can be a person or an organization.

Other Textual Devices. A narrative may contain peripheral contents other than events and characters, including time, place, the emotional and psychological states of characters etc., which do not advance the plot but provide essential information to the interpretation of the events (Pentland, 1999). We use rich Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2015) features to capture a variety of textual devices used to describe such contents.

4 Phase One: Weakly Supervised Narrative Identification

In order to acquire rich temporal event knowledge, we first develop a weakly supervised approach that can quickly adapt to identify narrative paragraphs from various text sources.

4.1 System Overview

The weakly supervised method is designed to capture key elements of narratives in each of two stages. As shown in figure 2, in the first stage, we identify the initial batch of narrative paragraphs that satisfy strict rules and the key principles of narratives. Then in the second stage, we train a statistical classifier using the initially identified seed narrative texts and a collection of soft features for capturing the same key principles and other textual devices of narratives. Next, the classifier is applied to identify new narratives from raw texts again. The newly identified narratives will be used to augment seed narratives and the bootstrapping

[2]NP is Noun Phrase and VP is Verb Phrase.

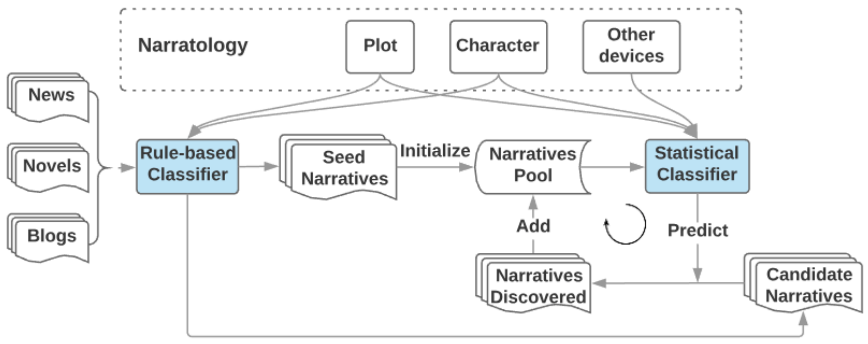


Figure 2: Overview of the Narrative Learning System

learning process iterates until no enough (specifically, less than 2,000) new narratives can be found. Here, in order to specialize the statistical classifier to each genre, we conduct the learning process on news, novels and blogs separately.

4.2 Rules for Identifying Seed Narratives

Grammar Rules for Identifying Plot Events. Guided by the prior narratology studies (Greimas, 1971; Mani, 2012) and our observations, we use context-free grammar production rules to identify sentences that describe an event in an actantial syntax structure. Specifically, we use three sets of grammar rules to specify the overall syntactic structure of a sentence. First, we require a sentence to have the basic active voiced structure "S → NP VP" or one of the more complex sentence structures that are derived from the basic structure considering Coordinating Conjunctions (CC), Adverbial Phrase (ADVP) or Prepositional Phrase (PP) attachments[3]. For example, in the narrative of Figure 1, the sentence *"Michael Kennedy earned a bachelor's degree from Harvard University in 1980."* has the basic sentence structure "S → NP VP", where the "NP" governs the character mention of 'Michael Kennedy' and the "VP" governs the rest of the sentence and describes a plot event.

In addition, considering that a narrative is usually "an account of past events in someone's life or in the development of something" (Mani, 2012; Dictionary, 2007), we require the headword of the VP to be in the past tense. Furthermore, the subject of the sentence is meant to represent a character. Therefore, we specify 12 grammar rules[4] to require the sentence subject noun phrase to have a simple structure and have a proper noun or pronoun as its head word.

For seed narratives, we consider paragraphs containing at least four sentences and we require 60% or more sentences to satisfy the sentence structure specified above. We also require a narrative paragraph to contain no more than 20% of sentences that are interrogative, exclamatory or dialogue, which normally do not contain any plot events. The specific parameter settings are mainly determined based on our observations and analysis of narrative samples. The threshold of 60% for "sentences with actantial structure" was set to reflect the observation that sentences in a narrative paragraph usually (over half) have an actantial structure. A small portion (20%) of interrogative, exclamatory or dialogue sentences is allowed to reflect the observation that many paragraphs are overall narratives even though they may contain 1 or 2 such sentences, so that we achieve a good coverage in narrative identification.

The Character Rule. A narrative usually has a protagonist character that appears in multiple sentences and ties a sequence of events, therefore, we also specify a rule requiring a narrative paragraph to have a protagonist character. Concretely, inspired by Eisenberg and Finlayson (2017), we applied the named entity recognizer (Finkel et al., 2005) and entity coreference resolver (Lee et al., 2013) from the CoreNLP toolkit (Manning et al., 2014) to identify the longest entity chain in a paragraph that has at least one mention recognized as a *Person* or *Organization*, or a gendered pronoun. Then we calculate the normalized length of this entity chain by dividing the number of entity mentions by the number of sentences in the paragraph. We require the normalized length of this longest

[3]We manually identified 14 top-level sentence production rules, for example, "S → NP ADVP VP", "S → PP , NP VP" and "S → S CC S". Appendix shows all the rules.

[4]The example NP rules include "NP → NNP", "NP → NP CC NP" and "NP → DT NNP".

entity chain to be ≥ 0.4, meaning that 40% or more sentences in a narrative mention a character [5].

4.3 The Statistical Classifier for Identifying New Narratives

Using the seed narrative paragraphs identified in the first stage as positive instances, we train a statistical classifier to continue to identify more narrative paragraphs that may not satisfy the specific rules. We also prepare negative instances to compete with positive narrative paragraphs in training. Negative instances are paragraphs that are not likely to be narratives and do not present a plot or protagonist character, but are similar to seed narratives in others aspects. Specifically, similar to seed narratives, we require a non-narrative paragraph to contain at least four sentences with no more than 20% of sentences being interrogative, exclamatory or dialogue; but in contrast to seed narratives, a non-narrative paragraph should contain 30% of or fewer sentences that have the actantial sentence structure, where the longest character entity chain should not span over 20% of sentences. We randomly sample such non-narrative paragraphs that are five times of narrative paragraphs [6].

In addition, since it is infeasible to apply the trained classifier to all the paragraphs in a large text corpus, such as the Gigaword corpus (Graff and Cieri, 2003), we identify candidate narrative paragraphs and only apply the statistical classifier to these candidate paragraphs. Specifically, we require a candidate paragraph to satisfy all the constraints used for identifying seed narrative paragraphs but contain only 30%[7] or more sentences with an actantial structure and have the longest character entity chain spanning over 20%[8] of or more sentences.

We choose Maximum Entropy (Berger et al., 1996) as the classifier. Specifically, we use the MaxEnt model implementation in the LIBLINEAR library[9] (Fan et al., 2008) with default parameter settings. Next, we describe the features used to capture the key elements of narratives.

Features for Identifying Plot Events: Realizing that grammar production rules are effective in identifying sentences that contain a plot event, we encode all the production rules as features in the statistical classifier. Specifically, for each narrative paragraph, we use the frequency of all syntactic production rules as features. Note that the bottom level syntactic production rules have the form of POS tag $\rightarrow$ WORD and contain a lexical word, which made these rules dependent on specific contexts of a paragraph. Therefore, we exclude these bottom level production rules from the feature set in order to model generalizable narrative elements rather than specific contents of a paragraph.

In addition, to capture potential event sequence overlaps between new narratives and the already learned narratives, we build a verb bigram language model using verb sequences extracted from the learned narrative paragraphs and calculate the perplexity score (as a feature) of the verb sequence in a candidate narrative paragraph. Specifically, we calculate the perplexity score of an event sequence that is normalized by the number of events, $PP(e_1, ..., e_N) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(e_i|e_{i-1})}}$, where N is the total number of events in a sequence and e_i is a event word. We approximate $P(e_i|e_{i-1}) = \frac{C(e_{i-1}, e_i)}{C(e_{i-1})}$, where $C(e_{i-1})$ is the number of occurrences of e_{i-1} and $C(e_{i-1}, e_i)$ is the number of co-occurrences of e_{i-1} and e_i. $C(e_{i-1}, e_i)$ and $C(e_{i-1})$ are calculated based on all event sequences from known narrative paragraphs.

Features for the Protagonist Characters: We consider the longest three coreferent entity chains in a paragraph that have at least one mention recognized as a *Person* or *Organization*, or a gendered pronoun. Similar to the seed narrative identification stage, we obtain the normalized length of each entity chain by dividing the number of entity mentions with the number of sentences in the paragraph. In addition, we also observe that a protagonist character appears frequently in the surrounding paragraphs as well, therefore, we calculate the normalized length of each entity chain based on its presences in the target paragraph as well as one preceding paragraph and one follow-

[5]40% was chosen to reflect that a narrative paragraph often contains a main character that is commonly mentioned across sentences (half or a bit less than half of all the sentences).

[6]We used the skewed pos:neg ratio of 1:5 in all bootstrapping iterations to reflect the observation that there are generally many more non-narrative paragraphs than narrative paragraphs in a document.

[7]This value is half of the corresponding thresshold used for identifying seed narrative paragraphs.

[8]This value is half of the corresponding thresshold used for identifying seed narrative paragraphs.

[9]https://www.csie.ntu.edu.tw/~cjlin/liblinear/

	0 (Seeds)	1	2	3	4	Total
News	20k	40k	12k	5k	1k	78k
Novels	75k	82k	24k	6k	2k	189k
Blogs	6k	10k	3k	1k	-	20k
Sum	101k	132k	39k	12k	3k	287k

Table 1: Number of new narratives generated after each bootstrapping iteration

ing paragraph. We use 6 normalized lengths (3 from the target paragraph [10] and 3 from surrounding paragraphs) as features.

Other Writing Style Features: We create a feature for each semantic category in the Linguistic Inquiry and Word Count (LIWC) dictionary (Pennebaker et al., 2015), and the feature value is the total number of occurrences of all words in that category. These LIWC features capture presences of certain types of words, such as words denoting relativity (e.g., motion, time, space) and words referring to psychological processes (e.g., emotion and cognitive). In addition, we encode Parts-of-Speech (POS) tag frequencies as features as well which have been shown effective in identifying text genres and writing styles.

4.4 Identifying Narrative Paragraphs from Three Text Corpora

Our weakly supervised system is based on the principles shared across all narratives, so it can be applied to different text sources for identifying narratives. We considered three types of texts: (1) **News Articles**. News articles contain narrative paragraphs to describe the background of an important figure or to provide details for a significant event. We use English Gigaword 5th edition (Graff and Cieri, 2003; Napoles et al., 2012), which contains 10 million news articles. (2) **Novel Books**. Novels contain rich narratives to describe actions by characters. BookCorpus (Zhu et al., 2015) is a large collection of free novel books written by unpublished authors, which contains 11,038 books of 16 different sub-genres (e.g., Romance, Historical, Adventure, etc.). (3) **Blogs**. Vast publicly accessible blogs also contain narratives because "personal life and experiences" is a primary topic of blog posts (Lenhart, 2006). We use the Blog Authorship Corpus (Schler et al., 2006) collected from the blogger.com website, which consists of 680k posts written by thousands of authors. We applied the Stanford CoreNLP tools (Manning et al., 2014) to the three text corpora to obtain POS tags, parse trees, named entities, coreference chains, etc.

In order to combat semantic drifts (McIntosh and Curran, 2009) in bootstrapping learning, we set the initial selection confidence score produced by the statistical classifier at 0.5 and increase it by 0.05 after each iteration. The bootstrapping system runs for four iterations and learns 287k narrative paragraphs in total. Table 1 shows the number of narratives that were obtained in the seeding stage and in each bootstrapping iteration from each text corpus.

5 Phase Two: Extract Event Temporal Knowledge from Narratives

Narratives we obtained from the first phase may describe specific stories and contain uncommon events or event transitions. Therefore, we apply Pointwise Mutual Information (PMI) based statistical metrics to measure strengths of event temporal relations in order to identify general knowledge that is not specific to any particular story. Our goal is to learn event pairs and longer event chains with events completely ordered in the temporal "before/after" relation.

First, by leveraging the double temporality characteristic of narratives, we only consider event pairs and longer event chains with 3-5 events that have occurred as a segment in at least one event sequence extracted from a narrative paragraph. Specifically, we extract the event sequence (the plot) from a narrative paragraph by finding the main event in each sentence and chaining the main events[11] according to their textual order.

Then we rank candidate event pairs based on two factors, how strongly associated two events are and how common they appear in a particular temporal order. We adopt the existing metric, Causal Potential (CP), which has been applied to acquire causally related events (Beamer and Girju, 2009) and exactly measures the two aspects. Specifically, the CP score of an event pair is calculated using the following equation:

$$cp(e_i, e_j) = pmi(e_i, e_j) + log\frac{P(e_i \to e_j)}{P(e_j \to e_i)} \quad (1)$$

where, the first part refers to the Pointwise Mutual Information (PMI) between two events and the

[10]Specifically, the lengths of the longest, second longest and third longest entity chains.

[11]We only consider main events that are in base verb forms or in the past tense, by requiring their POS tags to be VB, VBP, VBZ or VBD.

second part measures the relative ordering or two events. $P(e_i \to e_j)$ refers to the probability that e_i occurs before e_j in a text, which is proportional to the raw frequency of the pair. PMI measures the association strength of two events, formally, $pmi(e_i, e_j) = log\frac{P(e_i,e_j)}{P(e_i)P(e_j)}$, $P(e_i) = \frac{C(e_i)}{\sum_x C(e_x)}$ and $P(e_i, e_j) = \frac{C(e_i,e_j)}{\sum_x \sum_y C(e_x,e_y)}$, where, x and y refer to all the events in a corpus, $C(e_i)$ is the number of occurrences of e_i, $C(e_i, e_j)$ is the number of co-occurrences of e_i and e_j.

While each candidate pair of events should have appeared consecutively as a segment in at least one narrative paragraph, when calculating the CP score, we consider event co-occurrences even when two events are not consecutive in a narrative paragraph but have one or two other events in between. Specifically, the same as in (Hu and Walker, 2017), we calculate separate CP scores based on event co-occurrences with zero (consecutive), one or two events in between, and use the weighted average CP score for ranking an event pair, formally, $CP(e_i, e_j) = \sum_{d=1}^{3} \frac{cp_d(e_i,e_j)}{d}$.

Then we rank longer event sequences based on CP scores for individual event pairs that are included in an event sequence. However, an event sequence of length n is more than $n-1$ event pairs with any two consecutive events as a pair. We prefer event sequences that are coherent overall, where the events that are one or two events away are highly related as well. Therefore, we define the following metric to measure the quality of an event sequence:

$$CP(e_1, e_2, \cdots, e_n) = \frac{\sum_{d=1}^{3} \sum_{j=1}^{n-d} \frac{CP(e_j,e_{j+d})}{d}}{n-1}. \quad (2)$$

6 Evaluation

6.1 Precision of Narrative Paragraphs

From all the learned narrative paragraphs, we randomly selected 150 texts, with 25 texts selected from narratives learned in each of the two stages (i.e., seed narratives and bootstrapped narratives) using each of the three text corpora (i.e., news, novels, and blogs). Following the same definition "A story is a narrative of events arranged in their time sequence" (Forster, 1962; Gordon and Swanson, 2009), two human adjudicators were asked to judge whether each text is a narrative or a non-narrative. In order to obtain high inter-agreements, before the official annotations, we trained the two annotators for several iterations. Note that the texts we used in training annotators are different from the final texts we used for evaluation purposes. The overall kappa inter-agreement between the two annotators is 0.77.

Narratives	Seed	Bootstrapped
News	0.84	0.72
Novel	0.88	0.92
Blogs	0.92	0.88
AVG	0.88	0.84

Table 2: Precision of narratives based on human annotation

pairs	graduate $\to$ teach (5.7), meet $\to$ marry (5.3) pick up $\to$ carry (6.3), park $\to$ get out (7.3) turn around $\to$ face (6.5), dial $\to$ ring (6.3)
chains	drive $\to$ park $\to$ get out (7.8) toss $\to$ fly $\to$ land (5.9) grow up $\to$ attend $\to$ graduate $\to$ marry (6.9) contact $\to$ call $\to$ invite $\to$ accept (4.2) knock $\to$ open $\to$ reach $\to$ pull out $\to$ hold (6.0)

Table 3: Examples of event pairs and chains (with CP scores). $\to$ represents *before* relation.

Table 2 shows the precision of narratives learned in the two stages using the three corpora. We determined that a text is a correct narrative if both annotators labeled it as a narrative. We can see that on average, the rule-based classifier achieves the precision of 88% on initializing seed narratives and the statistical classifier achieves the precision of 84% on bootstrapping new ones. Using narratology based features enables the statistical classifier to extensively learn new narrative, and meanwhile maintain a high precision.

6.2 Precision of Event Pairs and Chains

To evaluate the quality of the extracted event pairs and chains, we randomly sampled 20 pairs (2%) from every 1,000 event pairs up to the top 18,929 pairs with CP score $\geq$ 2.0 (380 pairs selected in total), and 10 chains (1%) from every 1,000 up to the top 25,000 event chains[12] (250 chains selected in total). The average CP scores for all event pairs and all event chains we considered are 2.9 and 5.1 respectively. Two human adjudicators were asked to judge whether or not events are likely to occur in the temporal order shown. For event chains, we have one additional criterion requiring that events form a coherent sequence overall. An

[12]It turns out that many event chains have a high CP score close to 5.0, so we decided not to use a cut-off CP score of event chains but simply chose to evaluate the top 25,000 event chains.

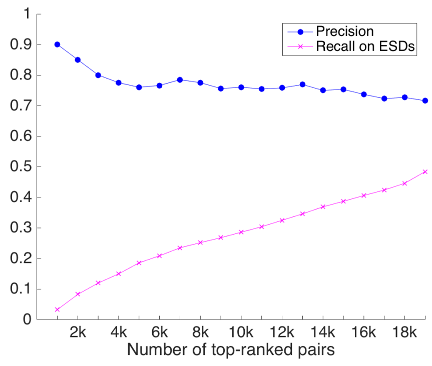


Figure 3: Top-ranked event pairs evaluation

# of top chains	5k	10k	15k	20k	25k
Precision	0.76	0.8	0.75	0.73	0.69

Table 4: Precision of top-ranked event chains

event pair/chain is deemed correct if both annotators labeled it as correct. The two annotators achieved kappa inter-agreement scores of 0.71 and 0.66, on annotating event pairs and event chains respectively.

As we know, coverage on acquired knowledge is often hard to evaluate because we do not have a complete knowledge base to compare to. Thus, we propose a pseudo recall metric to evaluate the coverage of event knowledge we acquired. Regneri et al. (2010) collected Event Sequence Descriptions (ESDs) of several types of human activities (e.g., baking a cake, going to the theater, etc.) using crowdsourcing. Our first pseudo recall score is calculated based on how many consecutive event pairs in human-written scripts can be found in our top-ranked event pairs. Figure 3 illustrates the precision of top-ranked pairs based on human annotation and the pseudo recall score based on ESDs. We can see that about 75% of the top 19k event pairs are correct, which captures 48% of human-written script knowledge in ESDs. In addition, table 4 shows the precision of top-ranked event chains with 3 to 5 events. Among the top 25k event chains, about 70% are correctly ordered with the temporal "after" relation. Table 3 shows several examples of event pairs and chains.

6.3 Improving Temporal Relation Classification by Incorporating Event Knowledge

To find out whether the learned temporal event knowledge can help with improving temporal re-

Models	Acc.(%)
Choubey and Huang (2017)	51.2
+ CP score	**52.3**

Table 5: Results on TimeBank corpus

Method	Acc.(%)
(Chambers and Jurafsky, 2008)	30.92
(Granroth-Wilding and Clark, 2016)	43.28
(Pichotta and Mooney, 2016)	43.17
(Wang et al., 2017)	46.67
Our Results	**48.83**

Table 6: Results on MCNC task

lation classification performance, we conducted experiments on a benchmark dataset - TimeBank corpus v1.2, which contains 2308 event pairs that are annotated with 14 temporal relations [13].

To facilitate direct comparisons, we used the same state-of-the-art temporal relation classification system as described in our previous work Choubey and Huang (2017) and considered all the 14 relations in classification. Choubey and Huang (2017) forms three sequences (i.e., word forms, POS tags, and dependency relations) of context words that align with the dependency path between two event mentions and uses three bi-directional LSTMs to get the embedding of each sequence. The final fully connected layer maps the concatenated embeddings of all sequences to 14 fine-grained temporal relations. We applied the same model here, but if an event pair appears in our learned list of event pairs, we concatenated the CP score of the event pair as additional evidence in the final layer. To be consistent with Choubey and Huang (2017), we used the same train/test splitting, the same parameters for the neural network and only considered intra-sentence event pairs. Table 5 shows that by incorporating our learned event knowledge, the overall prediction accuracy was improved by 1.1%. Not surprisingly, out of the 14 temporal relations, the performance on the relation *before* was improved the most by 4.9%.

6.4 Narrative Cloze

Multiple Choice version of the Narrative Cloze task (MCNC) proposed by Granroth-Wilding and Clark (2016); Wang et al. (2017), aims to eval-

[13]Specifically, the 14 relations are *simultaneous, before, after, ibefore, iafter, begins, begun by, ends, ended by, includes, is included, during, during inv, identity*

uate understanding of a script by predicting the next event given several context events. Presenting a chain of contextual events $e_1, e_2, ..., e_{n-1}$, the task is to select the next event from five event candidates, one of which is correct and the others are randomly sampled elsewhere in the corpus. Following the same settings of Wang et al. (2017) and Granroth-Wilding and Clark (2016), we adapted the dataset (test set) of Chambers and Jurafsky (2008) to the multiple choice setting. The dataset contains 69 documents and 349 multiple choice questions.

We calculated a PMI score between a candidate event and each context event $e_1, e_2, ..., e_{n-1}$ based on event sequences extracted from our learned 287k narratives and we chose the event that have the highest sum score of all individual PMI scores. Since the prediction accuracy on 349 multiple choice questions depends on the random initialization of four negative candidate events, we ran the experiment 10 times and took the average accuracy as the final performance.

Table 6 shows the comparisons of our results with the performance of several previous models, which were all trained with 1,500k event chains extracted from the NYT portion of the Gigaword corpus (Graff and Cieri, 2003). Each event chain consists of a sequence of verbs sharing an actor within a news article. Except Chambers and Jurafsky (2008), other recent models utilized more and more sophisticated neural language models. Granroth-Wilding and Clark (2016) proposed a two layer neural network model that learns embeddings of event predicates and their arguments for predicting the next event. Pichotta and Mooney (2016) introduced a LSTM-based language model for event prediction. Wang et al. (2017) used dynamic memory as attention in LSTM for prediction. It is encouraging that by using event knowledge extracted from automatically identified narratives, we achieved the best event prediction performance, which is 2.2% higher than the best neural network model.

7 Conclusions

This paper presents a novel approach for leveraging the double temporality characteristic of narrative texts and acquiring temporal event knowledge across sentences in narrative paragraphs. We developed a weakly supervised system that explores narratology principles and identifies narrative texts from three text corpora of distinct genres. The temporal event knowledge distilled from narrative texts were shown useful to improve temporal relation classification and outperform several neural language models on the narrative cloze task. For the future work, we plan to expand event temporal knowledge acquisition by dealing with event sense disambiguation and event synonym identification (e.g., drag, pull and haul).

8 Acknowledgments

We thank our anonymous reviewers for providing insightful review comments.

References

Mieke Bal. 2009. *Narratology: Introduction to the theory of narrative*. University of Toronto Press.

Brandon Beamer and Roxana Girju. 2009. Using a bigram event model to predict causal potential. In *CICLing*. Springer, pages 430–441.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22(1):39–71.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pages 1247–1250.

Kevin Burton, Akshay Java, and Ian Soboroff. 2009. The icwsm 2009 spinn3r dataset. In *Third Annual Conference on Weblogs and Social Media (ICWSM 2009)*. AAAI.

Betul Ceran, Ravi Karad, Steven Corman, and Hasan Davulcu. 2012. A hybrid model and memory based story classifier. In *Proceedings of the 3rd Workshop on Computational Models of Narrative*. pages 58–62.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, pages 602–610.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*. volume 94305, pages 789–797.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Prafulla Kumar Choubey and Ruihong Huang. 2017. A sequential model for classifying temporal relations between intra-sentence events. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1796–1802.

Oxford English Dictionary. 2007. Oxford english dictionary online.

Joshua Eisenberg and Mark Finlayson. 2017. A simpler and more generalizable story detector using verb and character features. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2698–2705.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research* 9(Aug):1871–1874.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 363–370.

Edward Morgan Forster. 1962. Aspects of the novel. 1927. *Ed. Oliver Stallybrass* .

Andrew Gordon and Reid Swanson. 2009. Identifying personal stories in millions of weblog entries. In *Third International Conference on Weblogs and Social Media, Data Challenge Workshop, San Jose, CA*. volume 46.

Hebert Grabes. 2013. Sequentiality. *Handbook of Narratology* 2:765–76.

David Graff and C Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium* .

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*. pages 2727–2733.

Algirdas Julien Greimas. 1971. Narrative grammar: Units and levels. *MLN* 86(6):793–806.

Zhichao Hu, Elahe Rahimtoroghi, Larissa Munishkina, Reid Swanson, and Marilyn A Walker. 2013. Unsupervised induction of contingent event pairs from film scenes. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 369–379.

Zhichao Hu and Marilyn Walker. 2017. Inferring narrative causality between event pairs in films. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. pages 342–351.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 336–344.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics* 39(4):885–916.

Amanda Lenhart. 2006. *Bloggers: A portrait of the internet's new storytellers*. Pew Internet & American Life Project.

Inderjeet Mani. 2012. Computational modeling of narrative. *Synthesis Lectures on Human Language Technologies* 5(3):1–142.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Tara McIntosh and James R Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, pages 396–404.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2016. Inscript: Narrative texts annotated with script information. In *LREC*. pages 3485–3493.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pages 95–100.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

Brian T Pentland. 1999. Building process theory with narrative: From description to explanation. *Academy of management Review* 24(4):711–724.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*. pages 2800–2806.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*. Lancaster, UK., volume 2003, page 40.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 979–988.

Mark O Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*. volume 6, pages 199–205.

Richard Walsh. 2001. Fabula and fictionality in narrative theory. *Style* 35(4):592–606.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 57–67.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pages 481–492.

Wenlin Yao, Saipravallika Nettyam, and Ruihong Huang. 2017. A weakly supervised approach to train temporal relation classifiers and acquire regular event pairs simultaneously. In *Proceedings of the 2017 Conference on Recent Advances in Natural Language Processing*. pages 803–812.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*. pages 19–27.

A Appendix

Here is the full list of grammar rules for identifying plot events in the seeding stage (Section 4.2).

Sentence rules (14):

S → S CC S
S → S PRN CC S
S → NP VP
S → NP ADVP VP
S → NP VP ADVP
S → CC NP VP
S → PP NP VP
S → NP PP VP
S → PP NP ADVP VP
S → ADVP S NP VP
S → ADVP NP VP
S → SBAR NP VP
S → SBAR ADVP NP VP
S → CC ADVP NP VP

Noun Phrase rules (12):

NP → PRP
NP → NNP
NP → NNS
NP → NNP NNP
NP → NNP CC NNP
NP → NP CC NP
NP → DT NN
NP → DT NNS
NP → DT NNP
NP → DT NNPS
NP → NP NNP
NP → NP NNP NNP

Text Deconvolution Saliency (TDS): a deep tool box for linguistic analysis

Laurent Vanni[1]* Melanie Ducoffe[2]* Damon Mayaffre[1] Frederic Precioso[2]

Dominique Longrée[3] Veeresh Elango[2] Nazly Santos[2] Juan Gonzalez[2] Luis Galdo[2]

Carlos Aguilar[1]

[1]Université Côte d'Azur, CNRS, BCL, France

[2]Université Côte d'Azur, CNRS, I3S, France

[3]Univ. Liège, L.A.S.L.A, Belgium

{laurent.vanni, melanie.ducoffe}@unice.fr

Abstract

In this paper, we propose a new strategy, called Text Deconvolution Saliency (TDS), to visualize linguistic information detected by a CNN for text classification. We extend Deconvolution Networks to text in order to present a new perspective on text analysis to the linguistic community. We empirically demonstrated the efficiency of our Text Deconvolution Saliency on corpora from three different languages: English, French, and Latin. For every tested dataset, our Text Deconvolution Saliency automatically encodes complex linguistic patterns based on co-occurrences and possibly on grammatical and syntax analysis.

1 Introduction

As in many other fields of data analysis, Natural Language Processing (NLP) has been strongly impacted by the recent advances in Machine Learning, more particularly with the emergence of Deep Learning techniques. These techniques outperform all other state-of-the-art approaches on a wide range of NLP tasks and so they have been quickly and intensively used in industrial systems. Such systems rely on end-to-end training on large amounts of data, making no prior assumptions about linguistic structure and focusing on stastically frequent patterns. Thus, they somehow step away from computational linguistics as they learn implicit linguistic information automatically without aiming at explaining or even exhibiting classic linguistic structures underlying the decision.

This is the question we raise in this article and that we intend to address by exhibiting classic linguistic patterns which are indeed exploited implictly in deep architectures to lead to higher performances. Do neural networks make use of co-occurrences and other standard features, considered in traditional Textual Data Analysis (TDA) (Textual Mining)? Do they also rely on complementary linguistic structure which is invisible to traditional techniques? If so, projecting neural networks features back onto the input space would highlight new linguistic structures and would lead to improving the analysis of a corpus and a better understanding on where the power of the Deep Learning techniques comes from.

Our hypothesis is that Deep Learning is sensitive to the linguistic units on which the computation of the key statistical sentences is based as well as to phenomena other than frequency and complex linguistic observables. The TDA has more difficulty taking such elements into account – such as linguistic linguistic patterns. Our contribution confronts Textual Data Analysis and Convolutional Neural Networks for text analysis. We take advantage of deconvolution networks for image analysis in order to present a new perspective on text analysis to the linguistic community that we call Text Deconvolution Saliency (TDS). Our deconvolution saliency corresponds to the sum over the word embedding of the deconvolution projection of a given feature map. Such a score provides a heat-map of words in a sentence that highlights the pattern relevant for the classification decision. We examine z-test (see section 4.2) and TDS for three languages: English, French and Latin. For all our datasets, TDS highlights new linguistic observables, invisible with z-test alone.

2 Related work

Convolutional Neural Networks (CNNs) are widely used in the computer vision community for

* L. Vanni and M. Ducoffe contributed equally to this work and should be considered as co-first authors.

a wide panel of tasks: ranging from image classification, object detection to semantic segmentation. It is a bottom-up approach where we applied an input image, stacked layers of convolutions, nonlinearities and sub-sampling.

Encouraged by the success for vision tasks, researchers applied CNNs to text-related problems Kalchbrenner et al. (2014); Kim (2014). The use of CNNs for sentence modeling traces back to Collobert and Weston (2008). Collobert adapted CNNs for various NLP problems including Part-of-Speech tagging, chunking, Named Entity Recognition and semantic labeling. CNNs for NLP work as an analogy between an image and a text representation. Indeed each word is embedded in a vector representation, then several words build a matrix (concatenation of the vectors).

We first discuss our choice of architectures. If Recurrent Neural Networks (*mostly GRU and LSTM*) are known to perform well on a broad range of tasks for text, recent comparisons have confirmed the advantage of CNNs over RNNs when the task at hand is essentially a keyphrase recognition task Yin et al. (2017).

In Textual Mining, we aim at highlighting linguistics patterns in order to analyze their constrast: specificities and similarities in a corpus Feldman, R., and J. Sanger (2007); L. Lebart, A. Salem and L. Berry (1998). It mostly relies on frequential based methods such as z-test. However, such existing methods have so far encountered difficulties in underlining more challenging linguistic knowledge, which up to now have not been empirically observed as for instance syntactical motifs Mellet and Longrée (2009).

In that context, supervised classification, especially CNNs, may be exploited for corpus analysis. Indeed, CNN learns automatically parameters to cluster similar instances and drive away instances from different categories. Eventually, their prediction relies on features which inferred specificities and similarities in a corpus. Projecting such features in the word embedding will reveal relevant spots and may automatize the discovery of new linguistic structures as in the previously cited syntactical motifs. Moreover, CNNs hold other advantages for linguistic analysis. They are static architectures that, according to specific settings are more robust to the vanishing gradient problem, and thus can also model long-term dependency in a sentence Dauphin et al. (2017); Wen et al. (2017); Adel and Schütze (2017). Such a property may help to detect structures relying on different parts of a sentence.

All previous works converged to a shared assessment: both CNNs and RNNs provide relevant, but different kinds of information for text classification. However, though several works have studied linguistic structures inherent in RNNs, to our knowledge, none of them have focused on CNNs. A first line of research has extensively studied the interpretability of word embeddings and their semantic representations Ji and Eisenstein (2014). When it comes to deep architectures, Karpathy et al. Karpathy et al. (2015) used LSTMs on character level language as a testbed. They demonstrate the existence of long-range dependencies on real word data. Their analysis is based on gate activation statistics and is thus global. On another side, Li et al. Li et al. (2015) provided new visualization tools for recurrent models. They use decoders, t-SNE and first derivative saliency, in order to shed light on how neural models work. Our perspective differs from their line of research, as we do not intend to explain how CNNs work on textual data, but rather use their features to provide complementary information for linguistic analysis.

Although the usage of RNNs is more common, there are various visualization tools for CNNs analysis, inspired by the computer vision field. Such works may help us to highlight the linguistic features learned by a CNN. Consequently, our method takes inspiration from those works. Visualization models in computer vision mainly consist in inverting hidden layers in order to spot active regions or features that are relevant to the classification decision. One can either train a decoder network or use backpropagation on the input instance to highlight its most relevant features. While those methods may hold accurate information in their input recovery, they have two main drawbacks: (i) they are computationally expensive: the first method requires training a model for each latent representation, and the second relies on backpropagation for each submitted sentence; (ii) they are highly hyperparameter dependent and may require some finetuning depending on the task at hand. On the other hand, Deconvolution Networks, proposed by Zeiler et al Zeiler and Fergus (2014), provide an off-the-shelf method to project a feature map in the input space. It consists in inverting each convolutional layer iteratively, back to the input space. The inverse of a discrete convolution is computationally challenging. In response, a coarse approximation may be employed which consists of inverting channels and filter weights in a convolutional layer and then transposing their kernel matrix. More details of the deconvolution heuristic are provided in section 3. Deconvolution has several advantages. First, it induces minimal computational requirements compared to previous visualization methods. Also, it has been used with success for semantic segmentation on images: in Noh et al. (2015); Noh et al demonstrate the efficiency of deconvolution networks to predict segmentation masks to identify pixel-wise class labels. Thus deconvolution is able to localize meaningful structure in the input space.

3 Model

3.1 CNN for Text Classification

We propose a deep neural model to capture linguistics patterns in text. This model is based on Convolutional Neural Networks with an embedding layer for word representations, one convolutional with pooling layer and non-linearities. Finally we have two fully-connected layers. The final output size corresponds to the number of classes. The model is trained by cross-entropy with an Adam optimizer. Figure 1 shows the global structure of our architecture. The input is a sequence of words $w_1, w_2...w_n$ and the output contains class probabilities (for text classification).

The embedding is built on top of a Word2Vec architecture, here we consider a Skip-gram model. This embedding is also finetuned by the model to to increase the accuracy. Notice that we do not use lemmatisation, as in Collobert and Weston (2008), thus the linguistic material which is automatically detected does not rely on any prior assumptions about the part of speech. In computer vision, we consider images as 2-dimensional isotropic signals. A text representation may also be considered as a matrix: each word is embedded in a feature vector and their concatenation builds a matrix. However, we cannot assume both dimensions the sequence of words and their embedding representation are isotropic. Thus the filters of CNNs for text typically differ from their counterparts designed for images. Consequently in text, the width of the filter is usually equal to the dimension of the embedding, as illustrated with the red, yellow, blue and green filters in figure 1

Using CNNs has another advantage in our context: due to the convolution operators involved, they can be easily parallelized and may also be easily used by the CPU, which is a practical solution for avoiding the use of GPUs at test time.

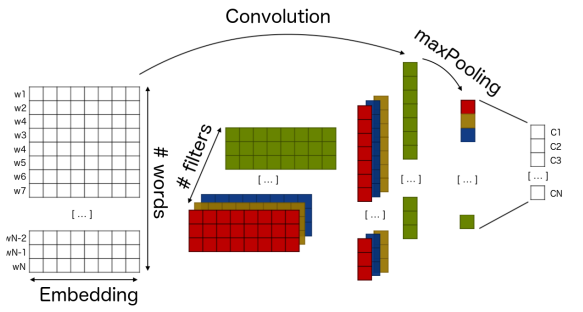


Figure 1: CNN for Text Classification

3.2 Deconvolution

Extending Deconvolution Networks for text is not straightforward. Usually, in computer vision, the deconvolution is represented by a convolution whose weights depends on the filters of the CNN: we invert the weights of the channels and the filters and then transpose each kernel matrix. When considering deconvolution for text, transposing the kernel matrices is not realistic since we are dealing with nonisotropic dimensions - the word sequences and the filter dimension. Eventually, the kernel matrix is not transposed.

Another drawback concerns the dimension of the feature map. Here feature map means the output of the convolution before applying max pooling. Its shape is actually the tuple *(# words, # filters)*. Because the filters' width (red, yellow, blue and green in fig 1) matches the embedding dimension, the feature maps cannot contain this information. To project the feature map in the embedding space, we need to convolve our feature map with the kernel matrices. To this aim, we upsample the feature map to obtain a 3-dimensional sample of size *(# words, embedding dimension, # filters)*.

To analyze the relevance of a word in a sentence, we only keep one value per word which corresponds to the sum along the embedding axis of the output of the deconvolution. We call this sum Text Deconvolution Saliency (TDS).

For the sake of consistency, we sum up our method in figure 2

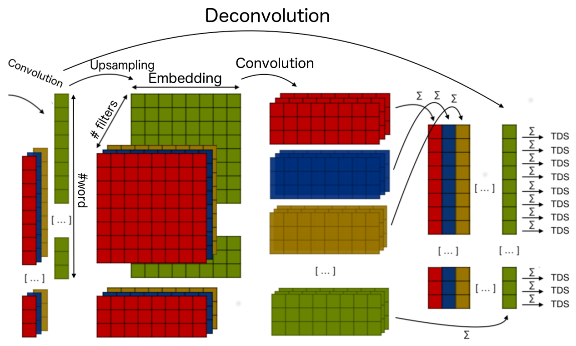


Figure 2: Textual Deconvolution Saliency (TDS)

Eventually, every word in a sentence has a unique TDS score whose value is related to the others. In the next section, we analyze the relevance of TDS. We thoroughly demonstrate empirically, that the TDS encodes complex linguistic patterns based on co-occurrences and possibly also on grammatical and syntaxic analysis.

4 Experiments

4.1 Datasets

In order to understand what the linguistic markers found by the convolutional neural network approach are, we conducted several tests on different languages and our model seems to get the same behavior in all of them. In order to perform all the linguistic statistical tests, we used our own simple linguistic toolbox Hyperbase, which allows the creation of databases from textual corpus, the analysis and the calculations such as z-test, co-occurrences, PCA, K-Means distance,... We use it to evaluate TDS against z-test scoring. We compel our analysis by only presenting cases on which z-test fail while TDS does not. Indeed TDS captures z-test, as we did not find any sentence on which z-test succeeds while TDS fails. Red words in the studied examples are the highest TDS.

The first dataset we used for our experiments is the well known IMDB movie review corpus for sentiment classification. It consists of 25,000 reviews labeled by positive or negative sentiment with around 230,000 words.

The second dataset targets French political discourses. It is a corpus of 2.5 millions of words of French Presidents from 1958 (with De Gaulle, the first President of the Fifth Republic) to 2018 with the first speeches by Macron. In this corpus we have removed Macron's speech from the 31st of December 2017, to use it as a test data set. The training task is to recognize each french president.

The last dataset we used is based on Latin. We assembled a contrastive corpus of 2 million words with 22 principle authors writting in classical Latin. As with the French dataset, the learning task here is to be able to predict each author according to new sequences of words. The next example is an excerpt of chapter 26 of the 23th book of Livy:

> *[...] tutus tenebat se quoad multum ac diu obtestanti quattuor milia peditum et quingenti equites in supplementum missi ex Africa sunt . tum refecta tandem spe* ***castra propius hostem*** *mouit classem que et ipse instrui parari que iubet ad insulas maritimam que oram tutandam . in* ***ipso impetu*** *mouendarum de [...]*

4.2 Z-test Versus Text Deconvolution Saliency

Z-test is one of the standard metrics used in linguistic statistics, in particular to measure the occurrences of word collocations Manning and Schütze (1999). Indeed, the z-test provides a statistical score of the co-occurrence of a sequence of words to appear more frequently than any other sequence of words of the same length. This score results from the comparison between the frequency of the observerd word sequence with the frequency expected in the case of a "Normal" distribution. In the context of constrative corpus analysis, this same calculation applied to single words can readily provide, for example, the most specific vocabulary of a given author. The highest z-test are the most specific words of this given author in this case. This is a simple but strong method for analyzing features of text. It can also be used to classify word sentences according to the global z-test (sum of the scores) of all the words in the given sentence. We can thus use this global z-test as a very simple metric for authorship classification. The resulting authorship of a given sentence is for instance given by the author corresponding to the highest global z-test on that sentence compared to all other global z-test obtained by summing up the z-test of each word of the same sentence but with the vocabulary specificity of another author. The mean accuracy of assigning the right author to the right sentence, in our data set, is around 87%, which confirms that z-test is indeed meaningful for

	z-test	Deep Learning
Latin	84%	93%
French	89%	91%
English	90%	97%

Table 1: Test accuray with z-test and Deep Learning

contrast pattern analysis. On the other hand, most of the time CNN reaches an accuracy greater than 90% for text classification (as shown in Table 1).

This means that the CNN approaches can learn also on their own some of the linguistic specificities useful in discriminating text categories. Previous works on image classification have highlighted the key role of convolutional layers which learn different level of abstractions of the data to make classification easier.

The question is: what is the nature of the abstraction on text?

We show in this article that CNN approach detects automatically words with high z-test but obviously this is not the only linguistic structure detected.

To make the two values comparable, we normalize them. The values can be either positive or negative. And we distinguish between two thresholds[1] for the z-test: over 2 a word is considered as specific and over 5 it is strongly specific (and the oposite with negative values). For the TDS it is just a matter of activation strength.

The Figure 3 shows us a comparison between z-test and TDS on a sentence extracted from our Latin corpora (Livy Book XXIII Chap. 26). This sentence is an example of specific words used by Livy[2]. As we can see, when the z-test is the highest, the TDS is also the highest and the TDS values are high also for the neighbor words (for example around the word *castra*). However, this is not always the case: for example small words as *que* or *et* are also high in z-test but they do not impact the network at the same level. We can see also on Figure 3 that words like *tenebat*, *multum* or *propius* are totally uncorrelated. The Pearson correlation coefficient[3] tells us that in this sentence there is no linear correlation between z-test and TDS (with a Pearson of 0.38). This example is one of the most correlated examples of our dataset, thus CNN seems to learn more than a simple z-test.

[1]The z-test can be approximated by a normal distribution. The score we obtain by the z-test is the standard deviation. A low standard deviation indicates that the data points tend to be close to the mean (the expected value). Over 2 this score means there is less than 2% of chance to have this distribution. Over 5 it's less than 0.1%.

[2]Titus Livius Patavinus – (64 or 59 BC - AD 12 or 17) – was a Roman historian.

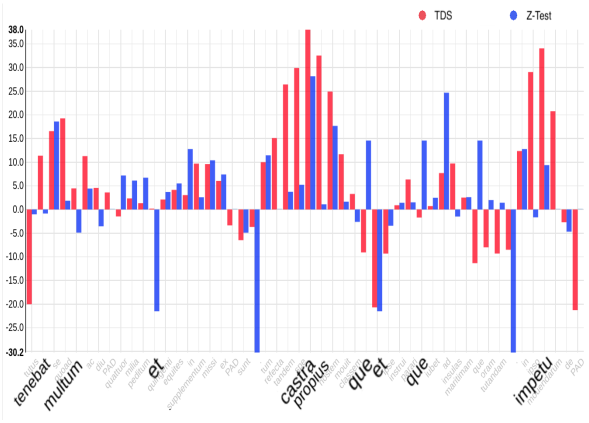


Figure 3: z-test versus Text Deconvolution Saliency (TDS) - Example on Livy Book XXIII Chap. 26

4.3 Dataset: English

For English, we used the IMDB movie review corpus for sentiment classification. With the default methods, we can easily show the specific vocabulary of each class (positive/negative), according to the z-test. There are for example the words *too*, *bad*, *no* or *boring* as most indicitive of negative sentiment, and the words *and*, *performance*, *powerful* or *best* for positive. Is it enough to detect automatically if a new review is positive or not? Let's see an example excerpted from a review from December 2017 (not in the training set) on the last American blockbuster:

> *[...]* ***i enjoyed three moments*** *in the film in total ,* ***and if i am being honest and*** *the person* ***next to me fell asleep*** *in the middle and started snoring during the slow space chasescenes .* ***the story failed to*** *draw me in and entertain* ***me the way*** *[...]*

In general the z-test is sufficient to predict the class of this kind of comment. But in this case, the CNN seems to do better, but why?

[3]Pearson correlation coefficient measures the linear relationship between two datasets. It has a value between $+1$ and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative

If we sum all the z-test (for negative and positive), the positive class obtains a greater score than the negative. The words *film*, *and*, *honest* and *entertain* – with scores 5.38, 12.23, 4 and 2.4 – make this example positive. CNN has activated different parts of this sentence (as we show in bold/red in the example). If we take the sub-sequence *and if i am being honest and*, there are two occurences of *and* but the first one is followed by *if* and our toolbox gives us 0.84 for *and if* as a negative class. This is far from the 12.23 in the positive. And if we go further, we can do a co-occurrence analysis on *and if* on the training set. As we see with our co-occurrence analysis[4] (Figure 4), *honest* is among the most specific adjectivals[5] associated with *and if*. Exactly what we found in our example.

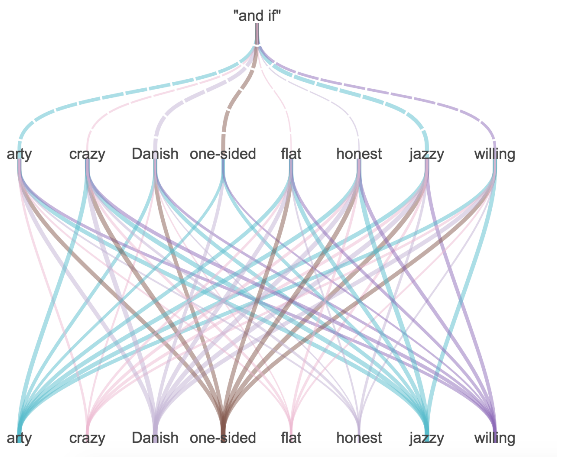


Figure 4: co-occurrences analysis of *and if* (Hyperbase)

In addition, we have the same behavior with the verb *fall*. There is the word *asleep* next to it. *Asleep* alone is not really specific of negative review (z-test of 1.13). But the association of both words become highly specific of negative sentences (see the co-occurrences analysis - Figure 5).

[4]Those figures shows the major co-occurrences for a given word (or lemma or PartOfSpeech). There two layers of co-occurrences, the first one (on top) show the direct co-occurrence and the second (on bottom) show a second level of co-occurrence. This level is given by the context of two words (taken together). The colors and the dotted lines are only used to make it more readable (dotted lines are used for the first level). The width of each line is related to the z-test score (more the z-test is big, more the line is wide).

[5]With our toolbox, we can focus on different part of speech.

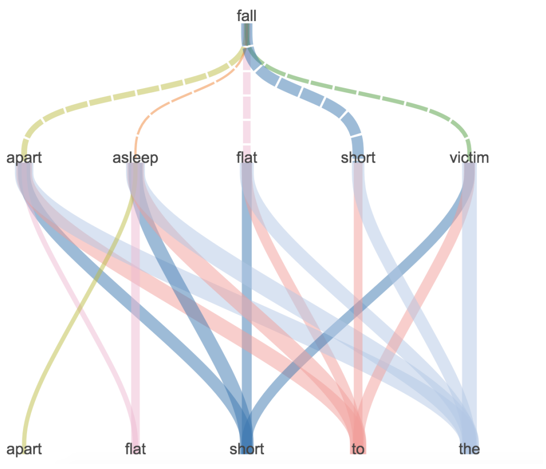


Figure 5: co-occurrences analysis of *fall* (Hyperbase)

The Text Deconvolution Saliency here confirms that the CNN seems to focus not only on high z-test but on more complex patterns and maybe detects the lemma or the part of speech linked to each word. We will see now that these observations are still valid for other languages and can even be generalized between different TDS.

4.4 Dataset: French

In this corpus we have removed Macron's speech from the 31st of December 2017, to use it as a test data set. In this speech, the CNN primarily recognizes Macron (the training task was to be able to predict the correct President). To achieve this task the CNN seems to succeed in finding really complex patterns specific to Macron. For example in this sequence:

> *[...] notre pays* ***advienne à*** *l'école pour nos enfants, au travail pour l' ensemble de* ***nos concitoyens*** *pour le climat pour le quotidien de chacune et chacun d' entre vous .* ***Ces transformations profondes*** *ont commencé et se* ***poursuivront*** *avec la même force le même rythme la même intensité [...]*

The z-test gives a result statistically closer to De Gaulle than to Macron. The error in the statistical attribution can be explained by a Gaullist phraseology and the multiplication of linguistic markers strongly indexed with De Gaulle: De Gaulle had the specificity of making long and literary sentences articulated around co-ordination conjunctions as in *et* (z-test = 28 for de Gaulle, two oc-

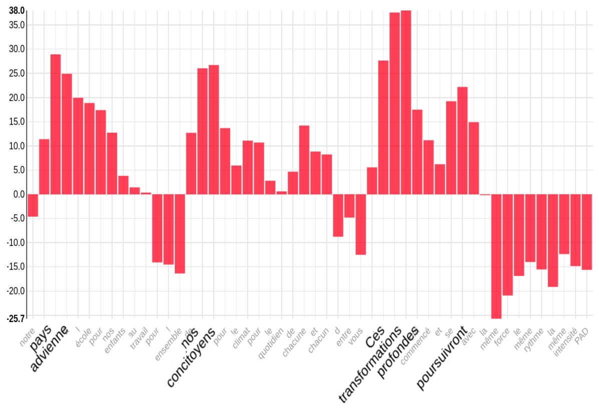


Figure 6: Deconvolution on Macron speech.

currences in the excerpt). His speech was also more conceptual than average, and this resulted in an over-use of the articles defined *le*, *la*, *l´*, *les*) very numerous in the excerpt (7 occurrences); especially in the feminine singular (*la république*, *la liberté*, *la nation*, *la guerre*, etc., here we have *la même force*, *la même intensité*.

The best results given by the CNN may be surprising for a linguist but match perfectly with what is known about the sociolinguistics of Macron's dynamic kind of speeches.

The part of the excerpt, which impacts most the CNN classification, is related to the nominal syntagm *transformations profondes*. Taken separately, neither of the phrase's two words are very Macronian from a statistical point of view (*transformations* = 1.9 *profondes* = 2.9). Better, the syntagm itself does not appear in the President's learning corpus (0 occurrence). However, it can be seen that the co-occurrence of *transformation* and *profondes* amounts to 4.81 at Macron: so it is not the occurrence of one word alone, or the other, which is Macronian but the simultaneous appearance of both in the same window. The second and complementary most impacting part of the excerpt thus is related to the two verbs *advienne* and *poursuivront*. From a semantic point of view, the two verbs perfectly contribute, after the phrase *transformations profondes*, to give the necessary dynamic to a discourse that advocates change. However it is the verb tenses (carried by the morphology of the verbs) that appear to be the determining factor in the analysis. The calculation of the grammatical codes co-occurring with the word *transformations* thus indicates that the verbs in the subjunctive and the verbs in the future (and also the nouns) are the privileged codes for Macron (Figure 7).

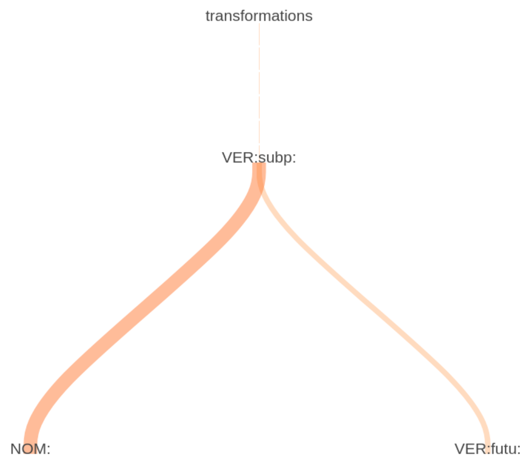


Figure 7: Main part-of-speech co-occurrences for *transformations* (Hyperbase)

More precisely the algorithm indicates that, for Macron, when *transformation* is associated with a verb in the subjunctive (here *advienne*), then there is usually a verb in the future co-present (here *poursuivront*). *transformations profondes*, *advienne* to the subjunctive, *poursuivront* to the future: all these elements together form a speech promising action, from the mouth of a young and dynamic President. Finally, the graph indicates that *transformations* is especially associated with nouns in the President's speeches: in an extraordinary concentration, the excerpt lists 11 (*pays*, *école*, *enfants*, *travail*, *concitoyens*, *climat*, *quotidien*, *transformations*, *force*, *rythme*, *intensité*).

4.5 Dataset: Latin

As with the French dataset, the learning task here is to be able to predict the identity of each author from a contrastive corpus of 2 million words with 22 principle authors writting in classical Latin.

The statistics here identify this sentence as Caesar[6] but Livy is not far off. As historians, Caesar and Livy share a number of specific words: for example tool words like *se* (reflexive pronoun) or *que* (a coordinator) and prepositions like *in*, *ad*, *ex*, *of*. There are also nouns like *equites* (cavalry) or *castra* (fortified camp).

The attribution of the sentence to Caesar cannot only rely only on z-test: *que* or *in* or *castra*, with

[6]Gaius Julius Caesar, 100 BC - 44 BC, usually called Julius Caesar, was a Roman politician and general and a notable author of Latin prose.

differences thereof equivalent or inferior to Livy. On the other hand, the differences of *se*, *ex*, are greater, as is that of *equites*. Two very Caesarian terms undoubtedly make the difference *iubet* (he orders) and *milia* (thousands).

The greater score of *quattuor* (four), *castra*, *hostem* (the enemy), *impetu* (the assault) in Livy are not enough to switch the attribution to this author.

On the other hand, CNN activates several zones appearing at the beginning of sentences and corresponding to coherent syntactic structures (for Livy) – *Tandem reflexes spe castra propius hostem mouit* (then, hope having finally returned, he moved the camp closer to the camp of the enemy) – despite the fact that *castra* in *hostem mouit* is attested only by Tacitus[7].

There are also *in ipso metu* (in fear itself), while *in* followed by *metu* is counted one time with Caesar and one time also with Quinte-Curce[8].

More complex structures are possibly also detected by the CNN: the structure *tum* + participates Ablative Absolute (*tum refecta*) is more characteristic of Livy (z-test 3.3 with 8 occurrences) than of Caesar (z-test 1.7 with 3 occurrences), even if it is even more specific of Tacitus (z-test 4.2 with 10 occurrences).

Finally and more likely, the co-occurrence between *castra*, *hostem* and *impetu* may have played a major role: Figure 8

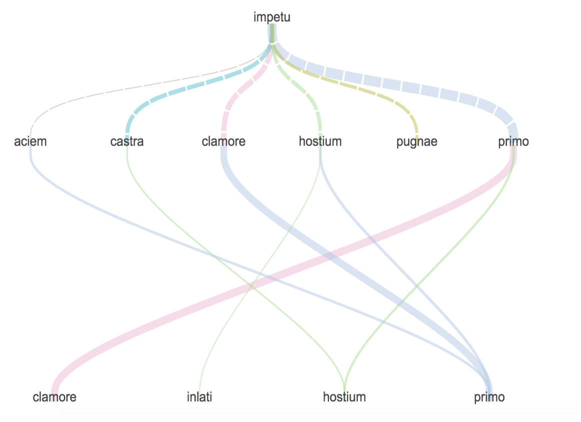


Figure 8: Specific co-occurrences between *impetu* and *castra* (Hyperbase)

With Livy, *impetu* appears as a co-occurrent with the lemmas *hostis* (z-test 9.42) and *castra* (z-test 6.75), while *hostis* only has a gap of 3.41 in Caesar and that *castra* does not appear in the list of co-occurrents.

For *castra*, the first co-occurent for Livy is *hostis* (z-test 22.72), before *castra* (z-test 10.18), *ad* (z-test 10.85), *in* (z-test 8.21), *impetus* (z-test 7.35), *que* (z-test 5.86) while in Caesar, *impetus* does not appear and the scores of all other lemmas are lower except *castra* (z-test 15.15), *hostis* (8), *ad* (10,35), *in* (5,17), *que* (4.79).

Thus, our results suggest that CNNs manage to account for specificity, phrase structure, and co-occurence networks...

4.6 Preprocessings and hyperparameters

In order to make our experiments reproductible, we detail here all the hyperparameters used in our architecture. The neural network is written in python with the library Keras (an tensorflow as backend).

The embedding uses a Word2Vec implementation given by the gensim Library. Here we use the SkipGram model with a window size of 10 words and output vectors of 128 values (embedding dimension).

The textual datas are tokenized by a home-made tokensizer (which work on English, Latin and French). The corpus is splited into 50 length sequence of words (punctuation is keeped) and each word is converted inta an unique vector of 128 value.

The first layer of our model takes the text sequence (as word vectors) and applies a weight corresponding to our WordToVec values. Those weights are still trainable during model training.

The second layer is the convolution, a Conv2D in Keras with 512 filters of size $3 * 128$ (filtering three words at time), with a Relu activation method. Then, there is the Maxpooling (MaxPooling2D)

(The deconvolution model is identical until here. We replace the rest of the classification model (Dense) by a transposed convolution (Conv2DTranspose).)

The last layers of the model are Dense layers. One hidden layer of 100 neurons with a Relu activation and one final layer of size equal to the number of classes with a softmax activation.

All experiments in this paper share the same architecture and the same hyperparameters, and

[7]Publius (or Gaius) Cornelius Tacitus, 56 BC - 120 BC, was a senator and a historian of the Roman Empire.

[8]Quintus Curtius Rufus was a Roman historian, probably of the 1st century, his only known and only surviving work being "Histories of Alexander the Great"

are trained with a cross-entropy method (with an Adam optimizer) with 90% of the dataset for the training data and 10% for the validation. All the tests in this paper are done with new data not included in the original dataset.

5 Conclusion

In a nutshell, Text Deconvolution Saliency is efficient on a wide range of corpora. By crossing statistical approaches with neural networks, we propose a new strategy for automatically detecting complex linguistic observables, which up to now hardly detectable by frequency-based methods. Recall that the linguistic matter and the topology recovered by our TDS cannot return to chance: the zones of activation make it possible to obtain recognition rates of more than 91% on the French political speech and 93% on the Latin corpus; both rates equivalent to or higher than the rates obtained by the statistical calculation of the key passages. Improving the model and understanding all the mathematical and linguistic outcomes remains an import goal. In future work, we intend to thoroughly study the impact of TDS given morphosyntactic information.

Acknowledgments

This work has been partly funded by the French Government (National Research Agency, ANR) through the grant ANR-16-CE23-0006 *Deep in France*, through the "Investments for the Future" Program ANR-11-LABX-0031-01, and through the UCAJEDI Investments in the Future project ANR-15-IDEX-01.

References

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1723–1729.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941.

Feldman, R., and J. Sanger. 2007. *The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data*. New York: Cambridge University Press.

Hyperbase. Web based toolbox for linguistics analysis. `http://hyperbase.unice.fr`.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 655–665.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

L. Lebart, A. Salem and L. Berry. 1998. *Exploring Textual Data*. Ed. Springer.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

S. Mellet and D. Longrée. 2009. Syntactical motifs and textual structures. In *Belgian Journal of Linguistics 23*, pages 161–173.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 438–449.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Coherence Modeling of Asynchronous Conversations: A Neural Entity Grid Approach

Tasnim Mohiuddin[*] and **Shafiq Joty**[*]
Nanyang Technological University
{mohi0004,srjoty}@ntu.edu.sg

Dat Tien Nguyen[*]
University of Amsterdam
t.d.nguyen@uva.nl

Abstract

We propose a novel coherence model for written asynchronous conversations (e.g., forums, emails), and show its applications in coherence assessment and thread reconstruction tasks. We conduct our research in two steps. First, we propose improvements to the recently proposed neural entity grid model by lexicalizing its entity transitions. Then, we extend the model to asynchronous conversations by incorporating the underlying conversational structure in the entity grid representation and feature computation. Our model achieves state of the art results on standard coherence assessment tasks in monologue and conversations outperforming existing models. We also demonstrate its effectiveness in reconstructing thread structures.

1 Introduction

Sentences in a text or a conversation do not occur independently, rather they are connected to form a coherent discourse that is easy to comprehend. **Coherence models** are computational models that can distinguish a coherent discourse from incoherent ones. It has ranges of applications in text generation, summarization, and coherence scoring.

Inspired by formal theories of discourse, a number of coherence models have been proposed (Barzilay and Lapata, 2008; Lin et al., 2011; Li and Jurafsky, 2017). The **entity grid** model (Barzilay and Lapata, 2008) is one of the most popular coherence models that has received much attention over the years. As exemplified in Table 1, the model represents a text by a grid that captures how grammatical roles of different discourse entities (*e.g.,* nouns) change from one sentence to another in the text. The grid is then converted into a feature vector containing probabilities of local entity transitions, enabling machine learning models to measure the degree of coherence. Earlier extensions of this basic model incorporate entity-specific features (Elsner and Charniak, 2011b), multiple ranks (Feng and Hirst, 2012), and coherence relations (Feng et al., 2014).

Recently, Nguyen and Joty (2017) proposed a neural version of the grid models. Their model first transforms the grammatical roles in a grid into their distributed representations, and employs a convolution operation over it to model entity transitions in the distributed space. The spatially max-pooled features from the convoluted features are used for coherence scoring. This model achieves state-of-the-art results in standard evaluation tasks on the Wall Street Journal (WSJ) corpus.

Although the neural grid model effectively captures long entity transitions, it is still limited in that it does not consider any lexical information regarding the entities, thereby, fails to distinguish

s_0: **LDI** Corp., Cleveland, said it will offer \$50 million in commercial **paper** backed by leaserental receivables.

s_1: The program matches funds raised from the sale of the commercial **paper** with small to medium-sized leases.

s_2: **LDI** termed the **paper** "non-recourse financing", meaning that investors would be repaid from the lease receivables, rather than directly by LDI Corp.

s_3: **LDI** leases and sells data-processing, telecommunications and other high-tech equipment.

	INVESTORS	MILLION	FUNDS	EQUIPMENT	CORP.	**PAPER**	SALE	TELECOMM.	LEASE	PROGRAM	CLEVELAND	RECEIVABLES	LEASES	DATA-PROCESS.	**LDI**	NON-RECOURSE
s_0	–	O	–	–	S	X	–	–	–	–	X	X	–	–	X	–
s_1	–	–	O	–	–	X	X	–	–	S	–	–	X	–	–	–
s_2	S	–	–	–	X	S	–	–	X	–	–	X	–	–	S	X
s_3	–	–	–	O	–	–	–	X	–	–	–	–	–	X	S	–

Table 1: Entity grid representation (bottom) for a document (top) from the WSJ corpus.

[*]All authors contributed equally.

between entity types. Although the extended neural grid considers entity features like named entity and proper mention, it requires an explicit feature extraction step, which can prevent us to transfer the model to a resource-poor language or domain.

Apart from these limitations, previous research on coherence models has mainly focused on monologic discourse (*e.g.,* news article). The only exception is the work of Elsner and Charniak (2011a), who applied coherence models to the task of conversation disentanglement in **synchronous** conversations like phone and chat conversations.

With the emergence of Internet technologies, **asynchronous** communication media like emails, blogs, and forums have become a commonplace for discussing events and issues, seeking answers, and sharing personal experiences. Participants in these media interact with each other asynchronously, by writing at different times. We believe coherence models for asynchronous conversations can help many downstream applications in these domains. For example, we will demonstrate later that coherence models can be used to predict the underlying thread structure of a conversation, which provides crucial information for building effective conversation summarization systems (Carenini et al., 2008) and community question answering systems (Barron-Cedeno et al., 2015).

To the best of our knowledge, none has studied the problem of coherence modeling in asynchronous conversation before. Because of its asynchronous nature, information flow in these conversations is often not sequential as in monologue or synchronous conversation. This poses a novel set of challenges for discourse analysis models (Joty et al., 2013; Louis and Cohen, 2015). For example, consider the forum conversation in Figure 2(a). It is not obvious how a coherence model like the entity grid can represent the conversation, and use it in downstream tasks effectively.

In this paper we aim to remedy the above limitations of existing models in two steps. First, we propose improvements to the existing neural grid model by *lexicalizing* its entity transitions. We propose methods based on word embeddings to achieve better generalization with the lexicalized model. Second, we adapt the model to asynchronous conversations by incorporating the underlying *conversational structure* in the grid representation and subsequently in feature computation. For this, we propose a novel grid representation for asynchronous conversations, and adapt the convolution layer of the neural model accordingly.

We evaluate our approach on two discrimination tasks. The first task is the standard one, where we assess the models based on their performance in discriminating an original document from its random permutation. In our second task, we ask the models to distinguish an original document from its inverse order of the sentences. For our adapted model to asynchronous conversation, we also evaluate it on *thread reconstruction*, a task specific to asynchronous conversation. We performed a series of experiments, and our main findings are:

(a) Our experiments on the WSJ corpus validate the utility of our proposed extension to the existing neural grid model, yielding absolute F_1 improvements of up to 4.2% in the standard task and up to 5.2% in the inverse-order discrimination task, setting a new state-of-the-art.

(b) Our experiments on a forum dataset show that our adapted model that considers the conversational structure outperforms the temporal baseline by more than 4% F_1 in the standard task and by about 10% F_1 in the inverse order discrimination task.

(c) When applied to the thread reconstruction task, our model achieves promising results outperforming several strong baselines.

We have released our source code and datasets at `https://ntunlpsg.github.io/project/coherence/n-coh-acl18/`

2 Background

In this section we give an overview of existing coherence models. In the interest of coherence, we defer description of the neural grid model (Nguyen and Joty, 2017) until next section, where we present our extension to this model.

2.1 Traditional Entity Grid Models

Introduced by Barzilay and Lapata (2008), the **entity grid** model represents a text by a two-dimensional matrix. As shown in Table 1, the rows correspond to sentences, and the columns correspond to entities (noun phrases). Each entry $E_{i,j}$ represents the syntactic role that entity e_j plays in sentence s_i, which can be one of: subject (S), object (O), other (X), or absent (–). In cases where an entity appears more than once with different grammatical roles in the same sentence, the role with the highest rank (S $\succ$ O $\succ$ X) is considered.

Motivated by the Centering Theory (Grosz et al., 1995), the model considers **local entity transitions** as the deciding patterns for assessing coherence. A local entity transition of length k is a sequence of $\{\text{S,O,X,–}\}^k$, representing grammatical roles played by an entity in k consecutive sentences. Each grid is represented by a vector of 4^k transition probabilities computed from the grid. To distinguish between transitions of important entities from unimportant ones, the model considers the *salience* of the entities, which is measured by their occurrence frequency in the document. With the feature vector representation, coherence assessment task is formulated as a ranking problem in a SVM preference ranking framework (Joachims, 2002). Barzilay and Lapata (2008) showed significant improvements in two out of three evaluation tasks when a coreference resolver is used to identify coreferent entities in a text.

Elsner and Charniak (2011b) show improvements to the grid model by including non-head nouns as entities. Instead of employing a coreference resolver, they match the nouns to detect coreferent entities. They demonstrate further improvements by extending the grid to distinguish between entities of different types. They do so by incorporating entity-specific features like named entity, noun class and modifiers. Lin et al. (2011) model transitions of discourse roles for entities as opposed to their grammatical roles. They instantiate discourse roles by discourse relations in Penn Discourse Treebank (Prasad et al., 2008). In a follow up work, Feng et al. (2014) trained the same model but using relations derived from deep discourse structures annotated with Rhetorical Structure Theory (Mann and Thompson, 1988).

2.2 Other Existing Models

Guinaudeau and Strube (2013) proposed a **graph-based** unsupervised method. They convert an entity grid into a bipartite graph consisting of two sets of nodes, representing sentences and entities, respectively. The edges are assigned weights based on the grammatical role of the entities in the respective sentences. They perform one-mode projections to transform the bipartite graph to a directed graph containing only sentence nodes. The coherence score of the document is then computed as the average *out-degree* of sentence nodes.

Louis and Nenkova (2012) introduced a coherence model based on **syntactic patterns** by assuming that sentences in a coherent text exhibit certain syntactic regularities. They propose a local coherence model that captures the co-occurrence of structural features in adjacent sentences, and a global model based on a hidden Markov model, which learns the global syntactic patterns from clusters of sentences with similar syntax.

Li and Hovy (2014) proposed a **neural** framework to compute the coherence score of a document by estimating coherence probability for every window of three sentences. They encode each sentence in the window using either a recurrent or a recursive neural network. To get a document-level coherence score, they sum up the window-level log probabilities. Li and Jurafsky (2017) proposed two encoder-decoder models augmented with latent variables for both coherence evaluation and discourse generation. Their first model incorporates global discourse information (topics) by feeding the output of a sentence-level HMM-LDA model (Gruber et al., 2007) into the encoder-decoder model. Their second model is trained end-to-end with variational inference.

In our work, we take an entity-based approach, and extend the neural grid model proposed recently by Nguyen and Joty (2017).

3 Extending Neural Entity Grid

In this section we first briefly describe the neural entity grid model proposed by Nguyen and Joty (2017). Then, we propose our extension to this model that leads to improved performance. We present our coherence model for asynchronous conversation in the next section.

3.1 Neural Entity Grid

Figure 1 depicts the neural grid model of Nguyen and Joty (2017). Given an entity grid E, they first transform each entry $E_{i,j}$ (a grammatical role) into a distributed representation of d dimensions by looking up a shared embedding matrix $M \in \mathbb{R}^{|G|\times d}$, where G is the vocabulary of possible grammatical roles, *i.e.,* $G = \{S, O, X, -\}$. Formally, the look-up operation can be expressed as:

$$L = \Big[M(E_{1,1}) \cdots M(E_{i,j}) \cdots M(E_{I,J})\Big] \quad (1)$$

where $M(E_{i,j})$ refers to the row in M that corresponds to grammatical role $E_{i,j}$, and I and J are

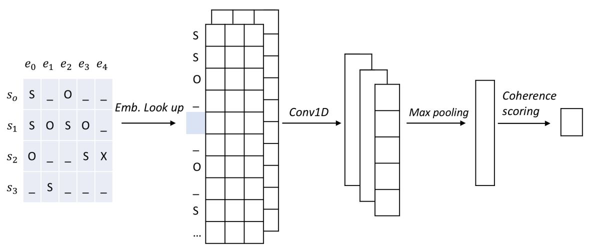


Figure 1: Neural entity grid model proposed by Nguyen and Joty (2017). The model is trained using a pairwise ranking approach with shared parameters for positive and negative documents.

the number of rows (sentences) and columns (entities) in the entity grid, respectively. The result of the look-up operation is a tensor $L \in \mathbb{R}^{I \times J \times d}$, which is fed to a convolution layer to model local entity transitions in the distributed space.

The convolution layer of the neural network composes patches of entity transitions into high-level abstract features by treating entities independently (*i.e.,* 1D convolution). Formally, it applies a *filter* $\mathbf{w} \in \mathbb{R}^{m.d}$ to each local entity transition of length m to generate a new abstract feature z_i:

$$z_i = h(\mathbf{w}^T L_{i:i+m,j} + b_i) \qquad (2)$$

where $L_{i:i+m,j}$ denotes concatenation of m vectors in L for entity e_j, b_i is a bias term, and h is a nonlinear activation function. Repeated application of this filter to each possible m-length transitions of different entities in the grid generates a *feature map*, $\mathbf{z}^i = [z_1, \cdots, z_{I.J+m-1}]$. This process is repeated N times with N different filters to get N different feature maps, $[\mathbf{z}^1, \cdots, \mathbf{z}^N]$. A *max-pooling* operation is then applied to extract the most salient features from each feature map:

$$\mathbf{p} = [\mu_l(\mathbf{z}^1), \cdots, \mu_l(\mathbf{z}^N)] \qquad (3)$$

where $\mu_l(\mathbf{z}^i)$ refers to the $\max$ operation applied to each non-overlapping window of l features in the feature map $\mathbf{z}^i$. Finally, the pooled features are used in a linear layer to produce a *coherence score*:

$$y = \mathbf{u}^T \mathbf{p} + b \qquad (4)$$

where $\mathbf{u}$ is the weight vector and b is a bias term. The model is trained with a *pairwise ranking* loss based on ordered training pairs (E_i, E_j):

$$\mathcal{L}(\theta) = \max\{0, 1 - \phi(E_i|\theta) + \phi(E_j|\theta)\} \qquad (5)$$

where entity grid E_i exhibits a higher degree of coherence than grid E_j, and $y = \phi(E_k|\theta)$ denotes the transformation of input grid E_k to a coherence score y done by the model with parameters θ. We will see later that such ordering of documents (grids) can be obtained automatically by permuting the original document. Notice that the network shares its parameters (θ) between the positive (E_i) and the negative (E_j) instances in a pair.

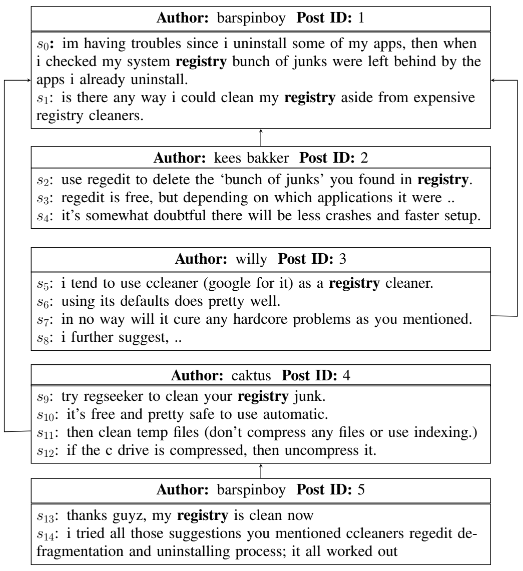
Since entity transitions in the convolution step are modeled in a continuous space, it can effectively capture longer transitions compared to traditional grid models. Unlike traditional grid models that compute transition probabilities from a *single* grid, convolution filters and role embeddings in the neural model are learned from all training instances, which helps the model to generalize well.

Since the abstract features in the feature maps are generated by convolving over role transitions of different entities in a document, the model implicitly considers relations between entities in a document, whereas transition probabilities in traditional entity grid models are computed without considering any such relation between entities. Convolution over the entire grid also incorporates *global* information (*e.g.,* topic) of a discourse.

3.2 Lexicalized Neural Entity Grid

Despite its effectiveness, the neural grid model presented above has a limitation. It does not consider any lexical information regarding the entities, thus, cannot distinguish between transitions of different entities. Although the extended neural grid model proposed in (Nguyen and Joty, 2017) does incorporate entity features like named entity type and proper mention, it requires an explicit feature extraction step using tools like named entity recognizer. This can prevent us in transferring the model to resource-poor languages or domains.

To address this limitation, we propose to lexicalize entity transitions. This can be achieved by attaching the entity with the grammatical roles. For example, if an entity e_j appears as a subject (S) in sentence s_i, the grid entry $E_{i,j}$ will be encoded as e_j-S. This way, an entity OBAMA as subject (OBAMA-S) and as object (OBAMA-O) will have separate entries in the embedding matrix M. We can initialize the word-role embeddings randomly, or with pre-trained embeddings for the word (OBAMA). In another variation, we kept word and role embeddings separate and con-



Author: barspinboy **Post ID:** 1

s_0**:** im having troubles since i uninstall some of my apps, then when i checked my system **registry** bunch of junks were left behind by the apps i already uninstall.
s_1: is there any way i could clean my **registry** aside from expensive registry cleaners.

Author: kees bakker **Post ID:** 2

s_2: use regedit to delete the ‘bunch of junks’ you found in **registry**.
s_3: regedit is free, but depending on which applications it were ..
s_4: it’s somewhat doubtful there will be less crashes and faster setup.

Author: willy **Post ID:** 3

s_5: i tend to use ccleaner (google for it) as a **registry** cleaner.
s_6: using its defaults does pretty well.
s_7: in no way will it cure any hardcore problems as you mentioned.
s_8: i further suggest, ..

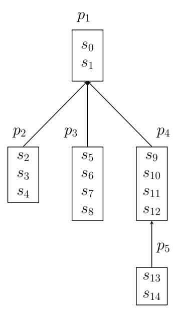
Author: caktus **Post ID:** 4

s_9: try regseeker to clean your **registry** junk.
s_{10}: it’s free and pretty safe to use automatic.
s_{11}: then clean temp files (don’t compress any files or use indexing.)
s_{12}: if the c drive is compressed, then uncompress it.

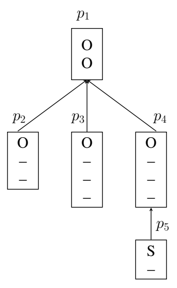
Author: barspinboy **Post ID:** 5

s_{13}: thanks guyz, my **registry** is clean now
s_{14}: i tried all those suggestions you mentioned ccleaners regedit de-fragmentation and uninstalling process; it all worked out

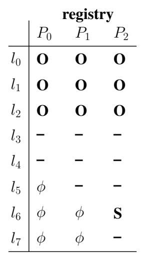
(a) A forum conversation



(b) Conversational tree



(c) Role transition for **‘registry’**



registry

	P_0	P_1	P_2
l_0	**O**	**O**	**O**
l_1	**O**	**O**	**O**
l_2	**O**	**O**	**O**
l_3	–	–	–
l_4	–	–	–
l_5	ϕ	–	–
l_6	ϕ	ϕ	**S**
l_7	ϕ	ϕ	–

(d) Grid representations

Figure 2: (a) A forum conversation, (b) Thread structure of the conversation, (c) Entity role transition over a conversation tree, and (d) 2D role transition matrix for an entity; ϕ denotes zero-padding.

catenated them after the look-up, thus enforcing OBAMA-S and OBAMA-O to share a part of their representations. However, in our experiments, we found the former approach to be more effective.

4 Coherence Models for Asynchronous Conversations

The main difference between monologue and asynchronous conversation is that information flow in asynchronous conversation is not sequential as in monologue, rather it is often interleaved. For example, consider the forum conversation in Figure 2(a). There are three possible subconversations, each corresponding to a path from the root node to a leaf node in the conversation graph in Figure 2(b). In response to seeking suggestions about how to clean *system registry*, the first path ($p_1 \leftarrow p_2$) suggests to use *regedit*, the second path ($p_1 \leftarrow p_3$) suggests *ccleaner*, and the third one ($p_1 \leftarrow p_4$) suggests using *regseeker*. These discussions are interleaved in the chronological order of the posts ($p_1 \leftarrow p_2 \leftarrow p_3 \leftarrow p_4 \leftarrow p_5$). Therefore, monologue-based coherence models may not be effective if applied directly to the conversation.

We hypothesize that coherence models for asynchronous conversation should incorporate the conversational structure like the tree structure in Figure 2(b), where the nodes represent posts and the edges represent ‘reply-to’ links between them. Since the grid models operate at the sentence level, we construct conversational structure at the sentence level. We do this by linking the boundary sentences across posts and by linking sentences in the same post chronologically. Specifically, we connect the first sentence of post p_j to the last sentence of post p_i if p_j replies to p_i, and sentence s_{t+1} is linked to s_t if both s_t and s_{t+1} are in the same post.[1] Now the question is, how can we represent a conversation tree with an entity grid, and then model entity transitions in the tree? In the following, we describe our approach to this problem.

4.1 Conversational Entity Grid

The conversation tree captures how topics flow in an asynchronous conversation. Our key hypothesis is that in a coherent conversation entities exhibit certain local patterns in the conversation tree in terms of their distribution and syntactic realization. Figure 2(c) shows how the grammatical roles of entity *‘registry’* in our example conversation change over the tree. For coherence assessment, we wish to model entity transitions along each of the conversation paths (top-to-bottom), and also their spatial relations across the paths (left-to-right). The existing grid representation is insufficient to model the *two-dimensional (2D) spatial* entity transitions in a conversation tree.

We propose a three-dimensional (3D) grid for representing entity transitions in an asynchronous conversation. The first dimension in our grid rep-

[1]The links between sentences are not explicitly shown in Figure 2(b) to avoid visual clutter.

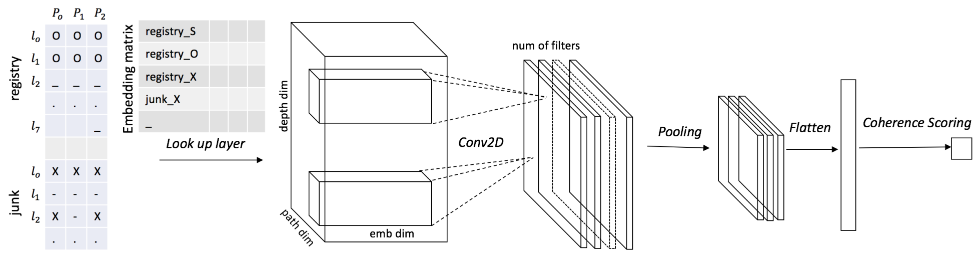


Figure 3: **Conversational Neural Grid** model for assessing coherence in asynchronous conversations.

resents *entities*, while the second and third dimensions represent *depth* and *path* of the tree, respectively. Figure 2(d) shows an example representation for an entity '*registry*'. Each column in the matrix represents transitions of the entity along a path, whereas each row represents transitions of the entity at a level of the conversation tree.

Although illustrated with a tree structure, our method is applicable to general graph-structured conversations, where a post can reply to multiple previous posts. Our model relies on paths from the root to the leaf nodes, which can be extracted for any graph as long as we avoid loops.

4.2 Modeling Entity Transitions

As shown in Figure 3, given a 3D entity grid as input, the look-up layer (Eq. 1) of our neural grid model produces a 4D tensor $L \in \mathbb{R}^{I \times J \times P \times d}$, where I is the total number of entities in the conversation, J is the depth of the tree, P is the number of paths in the tree, and d is the embedding dimension. The convolution layer then uses a 2D filter $\mathbf{w} \in \mathbb{R}^{m.n.d}$ to convolve local patches of entity transitions

$$z_i = h(\mathbf{w}^T L_{i,j:j+m,p:p+n} + b_i) \tag{6}$$

where m and n are the height and width of the filter, and $L_{i,j:j+m,p:p+n} \in \mathbb{R}^{m.n.d}$ denotes a concatenated vector containing $(m \times n)$ embeddings representing a 2D window of entity transitions. As we repeatedly apply the filter to each possible window with stride size 1, we get a 2D feature map Z^i of dimensions $(I.J+m-1) \times (I.P+n-1)$. Employing N different filters, we get N such 2D feature maps, $[Z^1, \cdots, Z^N]$, based on which the max pooling layer extracts the most salient features:

$$\mathbf{p} = [\mu_{l \times w}(Z^1), \cdots, \mu_{l \times w}(Z^N)] \tag{7}$$

where $\mu_{l \times w}$ refers to the $\max$ operation applied to each non-overlapping 2D window of $l \times w$ features in a feature map. The pooled features are then linearized and used for coherence scoring in the final layer of the network as described by Equation 4.

5 Experiments on Monologue

To validate our proposed extension to the neural grid model, we first evaluate our lexicalized neural grid model in the standard evaluation setting.

Evaluation Tasks and Dataset: We evaluate our models on the standard **discrimination** task (Barzilay and Lapata, 2008), where a coherence model is asked to distinguish an original document from its incoherent renderings generated by random permutations of its sentences. The model is considered correct if it ranks the original document higher than the permuted one.

We use the same train-test split of the WSJ dataset as used in (Nguyen and Joty, 2017) and other studies (Elsner and Charniak, 2011b; Feng et al., 2014). Following previous studies, we use 20 random permutations of each article for both training and testing, and exclude permutations that match the original article. Table 2 gives some statistics about the dataset along with the number of pairs used for training and testing. Nguyen and Joty (2017) randomly selected 10% of the training pairs for development purposes, which we also use for tuning hyperparameters in our models.

In addition to the standard setting, we also evaluate our models on an *inverse-order* setting, where we ask the models to distinguish an original document from the inverse order of its sentences (*i.e.,* from last to first). The transitions of roles in a negative grid are in the reverse order of the original grid. We do not train our models explicitly on this task, rather use the trained model from the standard setting. The number of test pairs in this setting is same as the number of test documents.

Model Settings and Training: We train the neural models with the pairwise ranking loss in Equation 5. For a fair comparison, we use

	Sections	# Doc.	Avg. # Sen.	# Pairs
Train	00-13	1,378	21.5	26,422
Test	14-24	1,053	22.3	20,411

Table 2: Statistics on the WSJ dataset.

similar model settings as in (Nguyen and Joty, 2017)[2] – ReLU as activation functions (h), RMSprop (Tieleman and Hinton, 2012) as the learning algorithm, Glorot-uniform (Glorot and Bengio, 2010) for initializing weight matrices, and uniform $\mathcal{U}(-0.01, 0.01)$ for initializing embeddings randomly. We applied batch normalization (Ioffe and Szegedy, 2015), which gave better results than using dropout. Minibatch size, embedding size and filter number were fixed to 32, 300 and 150, respectively. We tuned for optimal filter and pooling lengths in $\{2, \cdots, 12\}$. We train up to 25 epochs, and select the model that performs best on the development set; see **supplementary** documents for best hyperparameter settings for different models. We run each experiment five times, each time with a different random seed, and we report the average of the runs to avoid any randomness in results. Statistical significance tests are done using an *approximate randomization* test with SIGF V.2 (Padó, 2006).

Results and Discussions: We present our results on the standard discrimination task and the inverse-order task in Table 3; see Std (F_1) and Inv (F_1) columns, respectively. For space limitations, we only show F_1 scores here, and report both accuracy and F_1 in the supplementary document. We compare our lexicalized models (group III) with the unlexicalized models (group II) of Nguyen and Joty (2017).[3] We also report the results of non-neural entity grid models (Elsner and Charniak, 2011b) in group I. The extended versions use entity-specific features.

We experimented with both *random* and *pre-trained* initialization for word embeddings in our lexicalized models. As can be noticed in Table 3, both versions give significant improvements over the unlexicalized models on both the standard and the inverse-order discrimination tasks (2.7 - 4.3% absolute). Our best model with Google pre-trained embeddings (Mikolov et al., 2013) yields state-of-the-art results. We also experimented

[2]https://ntunlpsg.github.io/project/coherence/n-coh-acl17

[3]Our reproduced results for the neural grid model are slightly lower than their reported results ($\sim 1\%$). We suspect this is due to the randomness in the experimental setup.

	Model	Emb.	**Std** (F_1)	**Inv** (F_1)
I	Grid (E&C)	-	81.60	75.78
	Ext. Grid (E&C)	-	84.95	80.34
II	Neural Grid (N&J)	Random	84.36	83.94
	Ext. Neural Grid (N&J)	Random	85.93	83.00
III	Lex. Neural Grid	Random	$87.03^{\dagger}$	$86.88^{\dagger}$
	Lex. Neural Grid	Google	$\mathbf{88.56}^{\dagger}$	$\mathbf{88.23}^{\dagger}$

Table 3: Discrimination results on the **WSJ** dataset. Superscript † indicates a lexicalized model is significantly superior to the unlexicalized Neural Grid (N&J) model with p-value < 0.01.

with Glove (Pennington et al., 2014), which has more vocabulary coverage than word2vec – Glove covers 89.77% of our vocabulary items, whereas word2vec covers 85.66%. However, Glove did not perform well giving F_1 score of 86% in the standard discrimination task. Schnabel et al. (2015) also report similar results where word2vec was found to be superior to Glove in most evaluation tasks. Our model also outperforms the extended neural grid model that relies on an additional feature extraction step for entity features. These results demonstrate the efficacy of lexicalization in capturing fine-grained entity information without loosing generalizability, thanks to distributed representation and pre-trained embeddings.

6 Experiments on Conversation

We evaluate our coherence models for asynchronous conversations on two tasks: discrimination and thread reconstruction.

6.1 Evaluation on Discrimination

The discrimination tasks are applicable to conversations also. We first present the dataset we use, then we describe how we create coherent and incoherent examples to train and test our models.

Dataset: Our conversational corpus contains discussion threads regarding *computer troubleshooting* from the technology related news site CNET.[4] This corpus was originally collected by Louis and Cohen (2015), and it contains 13,352 threads. For our experiments, we selected 3,825 threads assuring that each contains at least 3 and at most 15 posts. We use 2,400 threads for training, 750 for testing and 675 for development purposes. Table 4 shows some basic statistics about the resulting dataset. The threads roughly contain 29 sentences and 6 comments on average.

[4]https://www.cnet.com/

	#Thread	Avg Com	Avg Sen	#Pairs (tree)	#Pairs (path)
Train	2,400	6.01	28.76	47,948	106,122
Test	750	5.75	27.79	14,986	33,852
Dev	675	6.27	30.70	13,485	28,897
Total	3,825	5.98	28.77	76,419	168,871

Table 4: Statistics on the **CNET** dataset.

Model Settings and Training: To validate the efficacy of our conversational grid model, we compare it with the following baseline settings:

• **Temporal:** In the temporal setting, we construct an entity grid from the chronological order of the sentences in a conversation, and use it with our monologue-based coherence models. Models in this setting thus disregard the structure of the conversation and treat it as a monologue.

• **Path-level:** This is a special case of our model, where we consider each path (a column in our conversational grid) in the conversation tree separately. We construct an entity grid for a path and provide as input to our monologue-based models.

To train the models with pairwise ranking, we create 20 incoherent conversations for each original conversation by shuffling the sentences in their temporal order. For models involving conversation trees (path-level and our model), the tree structure remains unchanged for original and permuted conversations, only the position of the sentences vary based on the permutation. Since the shuffling is done globally at the conversation level, this scheme allows us to compare the three representations (temporal, path-level and tree-level) fairly with the same set of permutations.

An incoherent conversation may have paths in the tree that match the original paths. We remove those matched paths when training the path-level model. See Table 4 for number of pairs used for training and testing our models. We evaluate path-level models by aggregating correct/wrong decisions for the paths – if the model makes more correct decisions for the original conversation than the incoherent one, it is counted as a correct decision overall. Aggregating path-level *coherence scores* (*e.g.,* by averaging or summing) would allow a coherence model to get awarded for assigning higher score to an original path (hence, correct) while making wrong decisions for the rest; see supplementary document for an example. Similar to the setting in Monologue, we did not train explicitly on the inverse-order task, rather use the trained model from the standard setting.

Conv. Rep	Model	Emb.	**Std** (F_1)	**Inv** (F_1)
Temporal	Neural Grid (N&J)	random	82.28	70.53
	Lex. Neural Grid	random	86.63	80.40
	Lex. Neural Grid	Google	87.17	80.76
Path-level	Neural Grid (N&J)	random	82.39	75.68†
	Lex. Neural Grid	random	88.13	88.38†
	Lex. Neural Grid	Google	88.44	89.31†
Tree-level	Neural Grid (N&J)	random	83.98†	77.33†
	Lex. Neural Grid	random	89.87†	89.23†
	Lex. Neural Grid	Google	**91.29**†	**90.40**†

Table 5: Discrimination results on **CNET**. Superscript † indicates a model is significantly superior to its temporal counterpart with p-value < 0.01.

Results and Discussions: Table 5 compares the results of our models on the two discrimination tasks. We observe more gains in conversation than in monologue for the lexicalized models – 4.9% to 7.3% on the standard task, and 10% to 13.6% on the inverse-order task. Notice especially the huge gains on the inverse-order task. This indicates lexicalization helps to better adapt to new domains.

A comparison of the results on the standard task across the representations shows that path-level models perform on par with the temporal models, whereas the tree-level models outperform others by a significant margin. The improvements are 2.7% for randomly initialized word vectors and 4% for Google embeddings. Although, the path-level model considers some conversational structures, it observes only a portion of the conversation in its input. The common topics (expressed by entities) of a conversation get distributed across multiple conversational paths. This limits the path-level model to learn complex relationships between entities in a conversation. By encoding an entire conversation into a single grid and by modeling the spatial relations between the entities, our conversational grid model captures both local and global information (topic) of a conversation.

Interestingly, the improvements are higher on the inverse-order task for both path- and tree-level models. The inverse order yields more dissimilarity at the paths with respect to the original order, thus making them easier to distinguish.

If we notice the hyperparameter settings for the best models on this task (see supplementary document), we see they use a filter width of 1. This indicates that to find the right order of the sentences in conversations, it is sufficient to consider entity transitions along the conversational paths in a tree.

6.2 Evaluation on Thread Reconstruction

One crucial advantage of our tree-level model over other models is that we can use it to build predictive models to uncover the thread structure of a conversation from its posts. Consider again the thread in Figure 2. Our goal is to train a coherence model that can recover the tree structure in Figure 2(b) from the sequence of posts $(p_1, p_2, \ldots, p_5)$.

This task has been addressed previously (Wang et al., 2008, 2011). Most methods learn an edge-level classifier to decide for a possible link between two posts using features like distance in position/time, cosine similarity, etc. To our knowledge, we are the first to use coherence models for this problem. However, our goal in this paper is not to build a state-of-the-art system for thread reconstruction, rather to evaluate coherence models by showing its effectiveness in scoring candidate tree hypotheses. In contrast to previous methods, our approach therefore considers the whole thread structure at once, and computes coherence scores for all possible candidate trees of a conversation. The tree that receives the highest score is predicted as the thread structure of the conversation.

Training: We train our coherence model for thread reconstruction using pairwise ranking loss as before. For a given sequence of comments in a thread, we construct a set of valid candidate trees; a valid tree is one that respects the chronological order of the comments, *i.e.,* a comment can only reply to a comment that precedes it. The training set contains ordered pairs (T_i, T_j), where T_i is a true (gold) tree and T_j is a valid but false tree.

Experiments: The number of valid trees grows exponentially with the number of posts in a thread, which makes the inference difficult. As a proof of concept that coherence models are useful for finding the right tree, we built a simpler dataset by selecting forum threads from the CNET corpus ensuring that a thread contains at most 5 posts. The final dataset contains 1200 threads with an average of 3.8 posts and 27.64 sentences per thread.

We assess the performance of the models at two levels: (*i*) **thread-level**, where we evaluate if the model could identify the entire conversation thread correctly, and (*ii*) **edge-level**, where we evaluate if the model could identify individual replies correctly. For comparison, we use a number of simple but well performing baselines:

- **All-previous** creates thread structure by linking a comment to its previous (in time) comment.
- **All-first** creates thread structure by linking all the comments to the initial comment.
- **COS-sim** creates thread structure by linking a comment to one of the previous comments with which it has the highest cosine similarity. We use TF.IDF representation for the comments.

	Thread-level	**Edge-level**	
	Acc	F_1	Acc
All-previous	27.00	52.00	61.83
All-first	25.67	48.23	58.19
COS-sim	27.66	50.56	60.30
Conv. Entity Grid	**30.33**†	**53.59**†	**62.81**†

Table 6: Thread reconstruction results; † indicates significant difference from COS-sim (p< .01).

Table 6 compares our best conversational grid model (tree-level with Google vectors) with the baselines. The low thread-level accuracy across all the systems prove that reconstructing an entire tree is a difficult task. Models are reasonably accurate at the edge level. Our coherence model shows promising results, yielding substantial improvements over the baselines. It delivers 2.7% improvements in thread-level and 2.5% in edge-level accuracy over the best baseline (COS-sim).

Interestingly, our best model for this task uses a filter width of 2 (maximum can be 4 for 5 posts). This indicates that spatial (left-to-right) relations between entity transitions are important to find the right thread structure of a conversation.

7 Conclusion

We presented a coherence model for asynchronous conversations. We first extended the existing neural grid model by lexicalizing its entity transitions. We then adapt the model to conversational discourse by incorporating the thread structure in its grid representation and feature computation. We designed a 3D grid representation for capturing spatio-temporal entity transitions in a conversation tree, and employed a 2D convolution to compose high-level features from this representation.

Our lexicalized grid model yields state of the art results on standard coherence assessment tasks in monologue and conversations. We also show a novel application of our model in forum thread reconstruction. Our future goal is to use the coherence model to generate new conversations.

References

Alberto Barron-Cedeno, Simone Filice, Giovanni Da San Martino, Shafiq Joty, Lluís Màrquez, Preslav Nakov, and Alessandro Moschitti. 2015. Thread-level information for comment classification in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL'15, pages 687–693, Beijing, China. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. 2008. Summarizing emails with conversational cohesion and subjectivity. In *Proceedings of the 46nd Annual Meeting on Association for Computational Linguistics, ACL'08*, pages 353–361, OH. ACL.

Micha Elsner and Eugene Charniak. 2011a. Disentangling chat with local coherence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1179–1189, Stroudsburg, PA, USA. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2011b. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 125–129, Portland, Oregon. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 315–324, Avignon, France. Association for Computational Linguistics.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *COLING*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 249–256, Sardinia, Italy.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Comput. Linguist.*, 21(2):203–225.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 163–170, San Juan, Puerto Rico. PMLR.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 93–103.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, Edmonton, Alberta, Canada. ACM.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2013. Topic segmentation and labeling in asynchronous conversations. *J. Artif. Int. Res.*, 47(1):521–573.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048, Doha, Qatar. Association for Computational Linguistics.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 198–209, Copenhagen, Denmark. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 997–1006, Portland, Oregon. Association for Computational Linguistics.

Annie Louis and Shay B. Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. In *EMNLP*.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1157–1168, Stroudsburg, PA, USA. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dat Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1320–1330. Association for Computational Linguistics.

Sebastian Padó. 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Tobias Schnabel, Igor Labutov, David M Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 298–307.

T. Tieleman and G Hinton. 2012. *RMSprop*. COURSERA: Neural Networks

Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre, and Timothy Baldwin. 2011. Predicting thread discourse structure over technical web forums. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 13–25, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yi-Chia Wang, Mahesh Joshi, William Cohen, and Carolyn Ros. 2008. Recovering implicit thread structure in newsgroup style conversations. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2008.*

Deep Reinforcement Learning for Chinese Zero pronoun Resolution

Qingyu Yin[♯], Yu Zhang[♯], Weinan Zhang[♯], Ting Liu[♯][*] William Yang Wang[♭]
[♯]Harbin Institute of Technology, China
[♭]University of California, Santa Barbara, USA
{qyyin, yzhang, wnzhang, tliu}@ir.hit.edu.cn
william@cs.ucsb.edu

Abstract

Deep neural network models for Chinese zero pronoun resolution learn semantic information for zero pronoun and candidate antecedents, but tend to be short-sighted—they often make local decisions. They typically predict coreference chains between the zero pronoun and one single candidate antecedent one link at a time, while overlooking their long-term influence on future decisions. Ideally, modeling useful information of preceding potential antecedents is critical when later predicting zero pronoun-candidate antecedent pairs. In this study, we show how to integrate local and global decision-making by exploiting deep reinforcement learning models. With the help of the reinforcement learning agent, our model learns the policy of selecting antecedents in a sequential manner, where useful information provided by earlier predicted antecedents could be utilized for making later coreference decisions. Experimental results on OntoNotes 5.0 dataset show that our technique surpasses the state-of-the-art models.

1 Introduction

Zero pronoun, as a special linguistic phenomenon in pro-dropped languages, is pervasive in Chinese documents (Zhao and Ng, 2007). A zero pronoun is a gap in the sentence, which refers to the component that is omitted because of the coherence of language. Following shows an example of zero pronoun in Chinese document, where zero pronouns are represented as "ϕ".

[当事人 李亚鼎] 除了 表示 ϕ_1 欣然 接受 但 ϕ_2 也 希望 国家 要 有 人 负责。

([Litigant Li Yading] not only shows ϕ_1 willing of acception, but also ϕ_2 hopes that there should be someone in charge of it.)

A zero pronoun can be an anaphoric zero pronoun if it coreferes to one or more mentions in the associated text, or unanaphoric, if there are no such mentions. In this example, the second zero pronoun "ϕ_2" is anaphoric and corefers to the mention "当事人李亚鼎/Litigant Li Yading" while the zero pronoun "ϕ_1" is unanaphoric. These mentions that contain the important information for interpreting the zero pronoun are called the antecedents.

In recent years, deep learning models for Chinese zero pronoun resolution have been widely investigated (Chen and Ng, 2016; Yin et al., 2017a,b). These solutions concentrate on anaphoric zero pronoun resolution, applying numerous neural network models to zero pronoun-candidate antecedent prediction. Neural network models have demonstrated their capabilities to learn vector-space semantics of zero pronouns and their antecedents (Yin et al., 2017a,b), and substantially surpass classic models (Zhao and Ng, 2007; Chen and Ng, 2013, 2015), obtaining state-of-the-art results on the benchmark dataset.

However, these models are heavily making local coreference decisions. They simply consider the coreference chain between the zero pronoun and one single candidate antecedent one link at a time while overlooking their impacts on future decisions. Intuitively, antecedents provide key linguistic cues for explaining the zero pronoun, it is therefore reasonable to leverage useful information provided by previously predicted antecedents as cues for predicting the later zero pronoun-candidate antecedent pairs. For instance, given a sentence "I have confidence that ϕ can do it." with its candidate mentions "he", "the boy" and "I", it is challenging to infer whether mention "I" is pos-

[*]Corresponding author.

sible to be the antecedent if it is considered separately. In that case, the resolver may incorrectly predict "I" to be the antecedent since "I" is the nearest mention. Nevertheless, if we know that "he" and "the boy" have already been predicted to be the antecedents, it is uncomplicated to infer the ϕ-"I" pair as "non-coreference" because "I" corefers to the disparate entity that is refered by "he". Hence, a desirable resolver should be able to 1) take advantage of cues of previously predicted antecedents, which could be incorporated to help classify later candidate antecedents and 2) model the long-term influence of the single coreference decision in a sequential manner.

To achieve these goals, we propose a deep reinforcement learning model for anaphoric zero pronoun resolution. On top of the neural network models (Yin et al., 2017a,b), two main innovations are introduced that are capable of efficaciously leveraging effective information provided by potential antecedents, and making long-term decisions from a global perspective. First, when dealing with a specific zero pronoun-candidate antecedent pair, our system encodes all its preceding candidate antecedents that are predicted to be the antecedents in the vector space. Consequently, this representative vector is regarded as the antecedent information, which can be utilized to measure the coreference probability of the zero pronoun-candidate antecedent pair. In addition, the policy-based deep reinforcement learning algorithm is applied to learn the policy of making coreference decisions for zero pronoun-candidate antecedent pairs. The innovative idea behind our reinforcement learning model is to model the antecedent determination as a sequential decision process, where our model learns to link the zero pronoun to its potential antecedents incrementally. By encoding the antecedents predicted in previous states, our model is capable of exploring the long-term influence of independent decisions, producing more accurate results than models that simply consider the limited information in one single state.

Our strategy is favorable in the following aspects. First, the proposed model learns to make decisions by linguistic cues of previously predicted antecedents. Instead of simply making local decisions, our technique allows the model to learn which action (predict to be an antecedent) available from the current state can eventually lead to a high-scoring overall performance. Second, instead of requiring supervised signals at each time step, deep reinforcement learning model optimizes its policy based on an overall reward signal. In other words, it learns to directly optimize the overall evaluation metrics, which is more effective than models that learn with loss functions that heuristically define the goodness of a particular single decision. Our experiments are conducted on the OntoNotes dataset. Comparing to baseline systems, our model obtains significant improvements, achieving the state-of-the-art performance for zero pronoun resolution. The major contributions of this paper are three-fold.

- We are the first to consider reinforcement learning models for zero pronoun resolution in Chinese documents;
- The proposed deep reinforcement learning model leverages linguistic cues provided by the antecedents predicted in earlier states when classifying later candidate antecedents;
- We evaluate our reinforcement learning model on a benchmark dataset, where a considerable improvement is gained over the state-of-the-art systems.

The rest of this paper is organized as follows. The next section describes our deep reinforcement learning model for anaphoric zero pronoun resolution. Section 3 presents our experiments, including the dataset description, evaluation metrics, experiment results, and analysis. We outline related work in Section 4. The Section 5 is about the conclusion and future work.

2 modelology

In this section, we introduce the technical details of the proposed reinforcement learning framework. The specific task of anaphoric zero pronoun resolution is to select antecedents from candidate antecedents for the zero pronoun. Here we formulate it as a sequential decision process in a reinforcement learning setting. We first describe the environment of the Markov decision making process and our reinforcement learning agent. Then, we introduce the modules. The last subsection is about the supervised pre-training technique of our model.

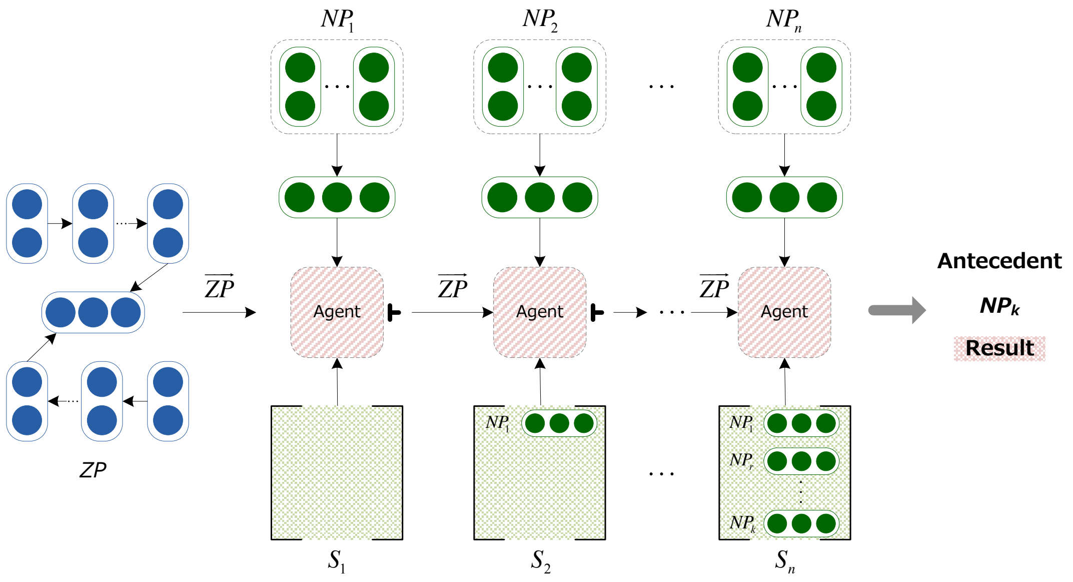


Figure 1: Illustration of our reinforcement learning framework. Given a zero pronoun with n candidate antecedents (presented as "NP"), for each time, the agent scores pairs of zero pronoun-candidate antecedent for their likelihood of coreference by 1) zero pronoun; 2) candidate antecedent and 3) antecedent information. Antecedent information at time t is generated by all the antecedents predicted in previous states.

2.1 Reinforcement Learning for Zero Pronoun Resolution

Given an anaphoric zero pronoun zp, a set of candidate antecedents are required to be selected from its associated text. In particular, we adopt the heuristic model utilized in recent Chinese anaphoric zero pronoun resolution work (Chen and Ng, 2016; Yin et al., 2017a,b) for this purpose. For those noun phrases that are two sentences away at most from the zero pronoun, we select those who are maximal noun phrases or modifier ones to compose the candidate set. These noun phrases ($\{np_1, np_2, ..., np_n\}$) and the zero pronoun ($zp$) are then encoded into representation vectors: $\{v_{np_1}, v_{np_2}, ..., v_{np_n}\}$ and v_{zp}.

Previous neural network models (Chen and Ng, 2016; Yin et al., 2017a,b) generally consider some pairwise models to select antecedents. In these work, candidate antecedents and the zero pronoun are first merged into pairs $\{(zp, np_1), (zp, np_2), ..., (zp, np_n)\}$, and then different neural networks are applied to deal with each pair independently. We argue that these models only make local decisions while overlooking their impacts on future decisions. In contrast, we formulate the antecedent determination process in as Markov decision process problem. An innovative reinforcement learning algorithm is designed that learns to classify candidate antecedents incrementally. When predicting one single zero pronoun-candidate antecedent pair, our model leverages antecedent information generated by previously predicted antecedents, making coreference decisions based on global signals.

The architecture of our reinforcement learning framework is shown in Figure 1. For each time step, our reinforcement learning agent predicts the zero pronoun-candidate antecedent pair by using 1) the zero pronoun; 2) information of current candidate antecedent and 3) antecedent information generated by antecedents predicted in previous states. In particular, our reinforcement learning agent is designed as a policy network $\pi_\theta(s, a) = p(a|s; \theta)$, where s represents the *state*; a indicates the *action* and θ represents the parameters of the model. The parameters θ are trained using stochastic gradient descent. Compared with Deep Q-Network (Mnih et al., 2013) that commonly learns a greedy policy, policy network is able to learn a stochastic policy that prevents the agent from getting stuck at an intermediate state (Xiong et al., 2017). Additionally, the learned policy is more explainable, comparing to learned value functions in Deep Q-Network. We here introduce the definitions of components of our reinforcement learning model, namely, *state*, *action* and *reward*.

2.1.1 State

Given a zero pronoun zp with its representation v_{zp} and all of its candidate antecedents representations $\{v_{np_1}, v_{np_2}, ..., v_{np_n}\}$, our model generate coreference decisions for zero pronoun-candidate antecedent pairs in sequence. More specifically, for each time, the *state* is generated by using both the vectors of the current zero pronoun-candidate antecedent pair and candidates that have been predicted to be the antecedents in the previous states. For time t, the state vector s_t is generated as follows:

$$s_t = (v_{zp}, v_{np_t}, v_{ante}(t), v_{feature_t}) \qquad (1)$$

where v_{zp} and v_{np_t} are the vectors of zp and np_t at time t. As shown in Chen and Ng (2016), human-designed handcrafted features are essential for the resolver since they reveal the syntactical, positional and other relations between a zero pronoun and its counterpart antecedents. Hence, to evaluate the coreference possibility of each candidate antecedent in a comprehensive manner, we integrate a group of features that are utilized in previous work (Zhao and Ng, 2007; Chen and Ng, 2013, 2016) into our model. For these multi-value features, we decompose them into a corresponding set of binary-value ones. $v_{feature_t}$ represents the feature vector. $v_{ante}(t)$ represents the antecedent information generated by candidates that have been predicted to be antecedents in previous states. After that, these vectors are concatenated to be the representation of *state* and fed into the deep reinforcement learning agent to generate the *action*.

2.1.2 Action

The action for each state is defined to be: *corefer* that indicates the zero pronoun and candidate antecedent are coreference; or otherwise, *non-corefer*. If an action *corefer* is made, we retain the vector of the counterpart antecedent together with those of the antecedents predicted in previous states to generate the vector v_{ante}, which is utilized to produce the antecedent information in the next state.

2.1.3 Reward

Normally, once the agent executes a series of actions, it observes a reward $R(a_{1:T})$ that could be any function. To encourage the agent to find accurate antecedents, we regard the F-score for the selected antecedents as the *reward* for each action in a path.

2.2 Reinforcement Learning Agent

Basically, our reinforcement learning agent is comprised of three parts, namely, the zero pronoun encoder that learns to encode a zero pronoun into vectors by using its context words; the candidate mention encoder that represents the candidate antecedents by content words; and the agent that maps the state vector s to a probability distribution over all possible actions.

In this work, the ZP-centered neural network model proposed by Yin et al. (2017a) is employed to be the zero pronoun encoder. The encoder learns to encode the zero pronoun by its associated text into its vector-space semantics. In particular, two standard recurrent neural networks are employed to encode the preceding text and the following text of a zero pronoun, separately. Such a model learns to encode the associated text around the zero pronoun, exploiting sentence-level information for the zero pronoun. For the candidate mentions encoder, we adopt the recurrent neural network-based model that encodes these phrases by using its content words. More specifically, we utilize a standard recurrent neural network to model the content of a phrase from left to right. This model learns to produce the vector of a phrase by considering its content, providing our model an ability to reveal its vector-space semantics. In this way, we generate the vector for zp, the v_{zp}, and representation vectors of all its candidate antecedents, which are denoted as $\{v_{np_1}, v_{np_2}, ..., v_{np_n}\}$.

Moreover, we employ pooling operations to encode antecedent information by using the antecedents that are predicted in previous states. In particular, we generate two vectors by applying the max-pooling and average-pooling, respectively. These two vectors are then concatenated together. Let the representative vector of the tth candidate antecedent to be $v_{np_t} \in \mathbb{R}^d$, and the predicted antecedents at time t be writen as $S(t) = [v_{np_i}, v_{np_j}, ..., v_{np_r}]$, the vector at time t, $v_{ante}(t)_k$ is generated by:

$$v_{ante}(t)_k = \begin{cases} max\{S(t)_{k,\cdot}\} & \text{for } 0 \leq \text{k} < \text{d} \\ ave\{S(t)_{k-d,\cdot}\} & \text{for d} \leq \text{k} < \text{2d} \end{cases}$$

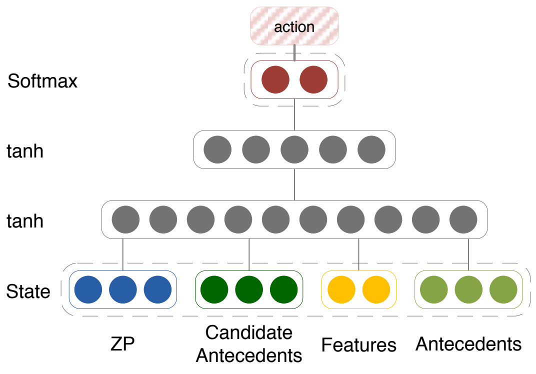


Figure 2: Illustration of the feedforward neural network model employed as the agent. Its input vector includes these parts: (1) Zero pronoun; (2) Candidate Antecedents; (3) Pair Features and (4) Antecedents. By going through all the full-connected hidden layers and one $softmax$ layer, the agent maps the state vector into the probability distribution over actions that indicates the coreference likelihood of the input zero pronoun-candidate antecedent pair.

The concatenation of these vectors is regarded as input and is fed into our reinforcement learning agent. More specifically, a feed-forward neural network is utilized to constitute the agent that maps the state vector to a probability distribution over all possible actions. Figure 2 shows the architecture of the agent. Two hidden layers are employed in our model, each of which utilizes the $tanh$ as the activation function. For each layer, we generate the output by:

$$h_i(s_t) = tanh(W_i h_{i-1}(s_t) + b_i) \tag{2}$$

where W_i and b_i are the parameters of the ith hidden layer; s_i represents the state vector. After going through all the layers, we can get the representative vector for the zero pronoun-candidate antecedent pair (zp, np_t). We then feed it into a scoring-layer to get their coreference score. The scoring-layer is a fully-connected layer of dimension 2:

$$score(zp, np_t) = W_s h_2(s_t) + b_s \tag{3}$$

where h_2 represents the output of the second hidden layer; $W_s \in \mathbb{R}^{2\times r}$ is the parameter of the layer and r is the dimension of h_2. Consequently, we generate the probability distribution over actions using the output generated by the scoring-layer of the neural network, where a $softmax$ layer is employed to gain the probability of each action:

$$p_\theta(a) \propto e^{score(zp,np_t)} \tag{4}$$

In this work, the policy-based reinforcement learning model is employed to train the parameter of the agent. More specifically, we explore using the REINFORCE policy gradient algorithm (Williams, 1992), which learns to maximize the expected reward:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{a_{1:T}\sim p(a|zp,np_t;\theta)} R(a_{1:T}) \\ &= \sum_t \sum_a p(a|zp, np_t; \theta) R(a_t) \end{aligned} \tag{5}$$

where $p(a|zp, np_t; \theta)$ indicates the probability of selecting action a.

Intuitively, the estimation of the gradient might have very high variance. One commonly used remedy to reduce the variance is to subtract a *baseline* value b from the reward. Hence, we utilize the gradient estimate as follows:

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_t \log p(a|zp, np_t; \theta)(R(a_t) - b_t) \tag{6}$$

Following Clark and Manning (2016), we interduce the baseline b and get the value of b_t at time t by $\mathbb{E}_{a_{t'}\sim p} R(a_1, ..., a_{t'}, ..., a_T)$.

2.3 Pretraining

Pretraining is crucial in reinforcement learning techniques (Clark and Manning, 2016; Xiong et al., 2017). In this work, we pretrain the model by using the loss function from Yin et al. (2017a):

$$loss = -\sum_{i=1}^{N} \sum_{np\in\mathcal{A}(zp_i)} \delta(zp_i, np) log(P(np|zp_i)) \tag{7}$$

where $P(np|zp_i)$ is the coreference score generated by the agent (the probability of choosing *corefer* action); $\mathcal{A}(zp_i)$ represents the candidate antecedents of zp_i; $\delta(zp, np)$ is 1 or 0, representing zp and np are coreference or not.

3 Experiments

3.1 Dataset and Settings

3.1.1 Dataset

Same to recent work on Chinese zero pronoun (Chen and Ng, 2016; Yin et al., 2017a,b), the proposed model is evaluated on the Chinese portion of the OntoNotes 5.0 dataset[1] that was used in the Conll-2012 Shared Task. Documents in this dataset are from six different sources, namely, Broadcast News (BN), Newswires (NW), Broadcast Conversations (BC), Telephone Conversations (TC), Web Blogs (WB) and Magazines (MZ). Since zero pronoun coreference annotations exist in only the training and development set (Chen and Ng, 2016), we utilize the training dataset for training purposes and test our model on the development set. The statistics of our dataset are reported in Table 1. To make equal comparison, we adopt the strategy as utilized in the existing work (Chen and Ng, 2016; Yin et al., 2017a), where 20% of the training dataset are randomly selected and reserved as a development dataset for tuning the model.

	#Documents	#Sentences	#AZPs
Training	1,391	36,487	12,111
Test	172	6,083	1,713

Table 1: Statistics on the training and test dataset.

3.1.2 Evaluation Measures

Following previous work on zero pronoun resolution (Zhao and Ng, 2007; Chen and Ng, 2016; Yin et al., 2017a,b), metrics employed to evaluate our model are: recall, precision, and F-score (F). We report the performance for each source in addition to the overall result.

3.1.3 Baselines and Experiment Settings

Five recent zero pronoun resolution systems are employed as our baselines, namely, Zhao and Ng (2007), Chen and Ng (2015), Chen and Ng (2016), Yin et al. (2017a) and Yin et al. (2017b). The first of them is machine learning-based, the second is the unsupervised and the other ones are all deep learning models. Since we concentrate on the anaphoric zero pronoun resolution process, we run experiments by employing the experiment setting with ground truth parse results and ground truth anaphoric zero pronoun, all of which are from the original dataset. Moreover, to illustrate the effectiveness of our reinforcement learning model, we run a set of ablation experiments by using different pretraining iterations and report the performance of our model with different iterations. Besides, to explore the randomness of the reinforcement learning technique, we report the performance variation of our model with different random seeds.

3.1.4 Implementation Details

We randomly initialize the parameters and minimize the objective function using Adagrad (Duchi et al., 2011). The embedding dimension is 100, and hidden layers are 256 and 512 dimensions, respectively. Moreover, the dropout (Hinton et al., 2012) regularization is added to the output of each layer. Table 2 shows the hyperparameters we utilized for both the pre-training and reinforcement learning process. Hyperparameters here are selected based on preliminary experiments and there remains considerable space for improvement, for instance, applying the annealing.

	Pre	RL
hidden dimentions	256 & 512	256 & 512
training epochs	70	50
batch	256	256
dropout rate	0.5	0.7
learning rate	0.003	0.00009

Table 2: Hyperparameters for the pre-training (Pre) and reinforcement learning (RL).

3.2 Experiment Results

In Table 3, we compare the results of our model with baselines in the test dataset. Our reinforcement learning model surpasses all previous baselines. More specifically, for the "Overall" results, our model obtains a considerable improvement by 2.3% in F-score over the best baseline (Yin et al., 2017a). Moreover, we run experiments in different sources of documents and report the results for each source. The number following a source's name indicates the amount of anaphoric zero pronoun in that source. Our model beats the best baseline in four of six sources, demonstrating the efficiency of our reinforcement learning model. The improvement gained over the best baseline in source "BC" is 4.3% in F-score, which is encouraging since it contains the most anaphoric zero pronoun. In all words, all these suggest that our model surpasses existed baselines, which demonstrates the efficiency of the proposed technique.

Ideally, our model learns useful information

[1]http://catalog.ldc.upenn.edu/LDC2013T19

	NW (84)	MZ (162)	WB (284)	BN (390)	BC (510)	TC (283)	**Overall**
Zhao and Ng (2007)	40.5	28.4	40.1	43.1	44.7	42.8	41.5
Chen and Ng (2015)	46.4	39.0	51.8	53.8	49.4	52.7	50.2
Chen and Ng (2016)	48.8	41.5	56.3	55.4	50.8	53.1	52.2
Yin et al. (2017b)	50.0	45.0	55.9	53.3	55.3	54.4	53.6
Yin et al. (2017a)	48.8	46.3	59.8	**58.4**	53.2	**54.8**	54.9
Our model	**63.1**	**50.2**	**63.1**	56.7	**57.5**	54.0	**57.2**

Table 3: Experiment results on the test data. The first six columns show the results on different source of documents and the last column is the overall results.

gathered from candidates that have been predicted to be the antecedents in previous states, which brings a global-view instead of simply making partial decisions. By applying the reinforcement learning, our model learns to directly optimize the overall performance in expectation, guiding benefit in making decisions in a sequential manner. Consequently, they bring benefit to predict accurate antecedents, leading to the better performance.

Moreover, on purpose of better illustrating the effectiveness of the proposed reinforcement learning model, we run a set of experiments with different settings. In particular, we compare the model with and without the proposed reinforcement learning process using different pre-training iterations. For each time, we report the performance of our model on both the test and development set. For all these experiments, we retain the rest of the model unchanged.

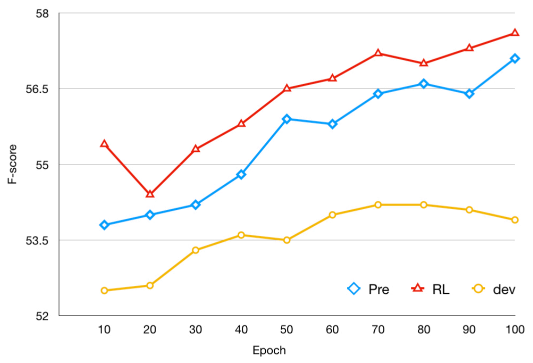


Figure 3: Experiment results of different models, where "RL" represents the reinforcement learning algorithm and "Pre" presents the model without reinforcement learning. "dev" shows the performance of our reinforcement learning model on the development dataset.

Figure 3 shows the performance of our model with and without reinforcement learning. We can see from the table that our model with reinforcement learning achieves better performance than the model without this all across the board. With the help of reinforcement learning, our model learns to choose effective actions in sequential decisions. It empowers the model to directly optimize the overall evaluation metrics, which brings a more effective and natural way of dealing with the task. Moreover, by seeing that the performance on development dataset stops increasing with iterations bigger than 70, we therefore set the pre-training iterations to 70.

Following Reimers and Gurevych (2017), to illustrate the impact of randomness in our reinforcement learning model, we run our model with different random seed values. Table 4 shows the performance of our model with different random seeds on the test dataset. We report the minimum, the maximum, the median F-scores results and the standard deviation σ of F-scores. We run

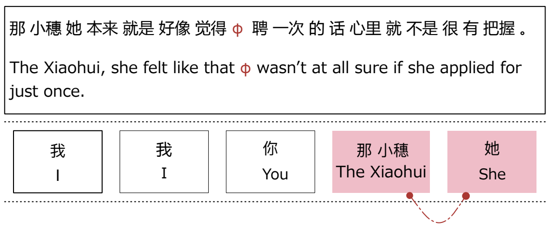
Min F	Median F	Max F	σ
56.5	57.1	57.5	0.00253

Table 4: Performance of our model with different random seeds.

the model with 38 different random seeds. The maximum F-score is 57.5% and the minimum one is 56.5%. Based on this observation, we can draw the conclusion that our proposed reinforcement learning model generally beats the baselines and achieves the state-of-the-art performance.

3.3 Case Study

Lastly, we show a case to illustrate the effectiveness of our proposed model, as is shown in Figure 4. In this case, we can see that our model correctly predict mentions "那小穗/The Xiaohui"



那 小穗 她 本来 就是 好像 觉得 ϕ 聘 一次 的 话 心里 就 不是 很 有 把握 。
The Xiaohui, she felt like that ϕ wasn't at all sure if she applied for just once.
我 I
我 I
你 You
那 小穗 The Xiaohui
她 She

Figure 4: Example of case study. Noun phrases with pink background color are the ones selected to be the antecedents by our model.

and "她/She" as the antecedents of the zero pronoun "ϕ". This case demonstrates the efficiency of our model. Instead of making only local decisions, our model learns to predict potential antecedents incrementally, selecting global-optimal antecedents in a sequential manner. In the end, our model successfully predicts "她/She" as the result.

4 Related Work

4.1 Zero Pronoun Resolution

A wide variety of techniques for machine learning models for Chinese zero pronoun resolution have been proposed. Zhao and Ng (2007) utilized the decision tree to learn the anaphoric zero pronoun resolver by using syntactical and positional features. It is the first time that machine learning techniques are applied for this task. To better explore syntactics, Kong and Zhou (2010) employed the tree kernel technique in their model. Chen and Ng (2013) extended Zhao and Ng (2007)'s model further by integrating innovative features and coreference chains between zero pronoun as bridges to find antecedents. In contrast, unsupervised techniques have been proposed and shown their efficiency. Chen and Ng (2014) proposed an unsupervised model, where a model trained on manually resolved pronoun was employed for the resolution of zero pronoun. Chen and Ng (2015) proposed an unsupervised anaphoric zero pronoun resolver, using the salience model to deal with the issue. Besides, there has been extensive work on zero anaphora for other languages. Efforts for zero pronoun resolution fall into two major categories, namely, (1) heuristic techniques (Han, 2006); and (2) learning-based models (Iida and Poesio, 2011; Isozaki and Hirao, 2003; Iida et al., 2006, 2007; Sasano and Kurohashi, 2011; Iida and Poesio, 2011; Iida et al., 2015, 2016).

In recent years, deep learning techniques have been extensively studied for zero pronoun resolution. Chen and Ng (2016) introduced a deep neural network resolver for this task. In their work, zero pronoun and candidates are encoded by a feed-forward neural network. Liu et al. (2017) explored to produce pseudo dataset for anaphoric zero pronoun resolution. They trained their deep learning model by adopting a two-step learning method that overcomes the discrepancy between the generated pseudo dataset and the real one. To better utilize vector-space semantics, Yin et al. (2017b) employed recurrent neural network to encode zero pronoun and antecedents. In particular, a two-layer antecedent encoder was employed to generate the hierarchical representation of antecedents. Yin et al. (2017a) developed an innovative deep memory network resolver, where zero pronouns are encoded by its potential antecedent mentions and associated text.

The major difference between our model and existed techniques lies in the applying of deep reinforcement learning. In this work, we formulate the anaphoric zero pronoun resolution as a sequential decision process in a reinforcement learning setting. With the help of reinforcement learning, our resolver learns to classify mentions in a sequential manner, making global-optimal decisions. Consequently, our model learns to take advantage of earlier predicted antecedents when making later coreference decisions.

4.2 Deep Reinforcement Learning

Recent advances in deep reinforcement learning have shown promise results in a variety of natural language processing tasks (Branavan et al., 2012; Narasimhan et al., 2015; Li et al., 2016). In recent time, Clark and Manning (2016) proposed a deep reinforcement learning model for coreference resolution, where an agent is utilized for linking mentions to their potential antecedents. They utilized the policy gradient algorithm to train the model and achieves better results compared with the counterpart neural network model. Narasimhan et al. (2016) introduced a deep Q-learning based slot-filling technique, where the agent's action is to retrieve or reconcile content from a new document. Xiong et al. (2017) proposed an innovative reinforcement learning framework for learning multi-hop relational paths. Deep reinforcement learning is a natural choice for tasks that require making incremental decisions. By combining non-linear function approximations with reinforcement learning, the deep reinforcement learning paradigm can integrate vector-space semantic into a robust joint learning and reasoning process. Moreover, by optimizing the policy-based on the reward signal, deep reinforcement learning model relies less on heuristic loss functions that require careful tuning.

5 Conclusion

We introduce a deep reinforcement learning framework for Chinese zero pronoun resolution. Our model learns the policy on selecting antecedents in a sequential manner, leveraging effective information provided by the earlier predicted antecedents. This strategy contributes to the predicting for later antecedents, bringing a natural view for the task. Experiments on the benchmark dataset show that our reinforcement learning model achieves an F-score of 67.2% on the test dataset, surpassing all the existed models by a considerable margin.

In the future, we plan to explore neural network models for efficaciously resolving anaphoric zero pronoun documents and research on some specific components which might influence the performance of the model, such as the embedding. Meanwhile, we plan to research on the possibility of applying adversarial learning (Goodfellow et al., 2014) to generate better rewards than the human-defined reward functions. Besides, to deal with the problematic scenario when ground truth parse tree and anaphoric zero pronoun are unavailable, we are interested in exploring the neural network model that integrates the anaphoric zero pronoun determination and anaphoric zero pronoun resolution jointly in a hierarchical architecture without using parser or anaphoric zero pronoun detector.

Our code is available at `https://github.com/qyyin/Reinforce4ZP.git`.

Acknowledgments

Thank the anonymous reviewers for their valuable comments. This work was supported by the Major State Basic Research Development 973 Program of China (No.2014CB340503), National Natural Science Foundation of China (No.61472105 and No.61502120). According to the meaning by Harbin Institute of Technology, the contact author of this paper is Ting Liu.

References

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *EMNLP*, pages 1360–1365.

Chen Chen and Vincent Ng. 2014. Chinese zero pronoun resolution: An unsupervised approach combining ranking and integer linear programming. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, page 320.

Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural networks. In *Proceedings of the 54rd Annual Meeting of the ACL.*

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *Proceedings of EMNLP'16.*

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Na-Rae Han. 2006. *Korean zero pronouns: analysis and resolution*. Ph.D. thesis, Citeseer.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 625–632. Association for Computational Linguistics.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(4):1.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ilp solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 804–813. Association for Computational Linguistics.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. *Proceedings of EMNLP'15*, pages 2179–2189.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of EMNLP*.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 184–191. Association for Computational Linguistics.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882–891. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Ting Liu, Yiming Cui, Qingyu Yin, Shijin Wang, Weinan Zhang, and Guoping Hu. 2017. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. In *ACL*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *NIPS*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *EMNLP'15*.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2355–2365.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *IJCNLP*, pages 758–766.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017a. Chinese zero pronoun resolution with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1318.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017b. A deep neural network for chinese zero pronoun resolution. In *IJCAI*.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of chinese zero pronouns: A machine learning approach. In *EMNLP-CoNLL*, volume 2007, pages 541–550.

Entity-Centric Joint Modeling of Japanese Coreference Resolution and Predicate Argument Structure Analysis

Tomohide Shibata[†‡] and **Sadao Kurohashi**[†‡]
[†]Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
[‡]CREST, JST
4-1-8, Honcho, Kawaguchi-shi, Saitama, 332-0012, Japan
{shibata, kuro}@i.kyoto-u.ac.jp

Abstract

Predicate argument structure analysis is a task of identifying structured events. To improve this field, we need to identify a salient entity, which cannot be identified without performing coreference resolution and predicate argument structure analysis simultaneously. This paper presents an entity-centric joint model for Japanese coreference resolution and predicate argument structure analysis. Each entity is assigned an embedding, and when the result of both analyses refers to an entity, the entity embedding is updated. The analyses take the entity embedding into consideration to access the global information of entities. Our experimental results demonstrate the proposed method can improve the performance of the inter-sentential zero anaphora resolution drastically, which is a notoriously difficult task in predicate argument structure analysis.

1 Introduction

Natural language often conveys a sequence of events like "who did what to whom", and extracting structured events from the raw text is a kind of touchstone for machine reading. This is realized by a combination of coreference resolution (called *CR*, hereafter) and predicate argument structure analysis (called *PA*, hereafter).

The characteristics and difficulties in the analyses vary among languages. In English, there are few omissions of arguments, and thus PA is relatively easy, around 83% accuracy (He et al., 2017), while CR is relatively difficult, around 70% accuracy (Lee et al., 2017).

On the other hand, in Japanese and Chinese, where arguments are often omitted, PA is a difficult task, and even state-of-the-art systems only achieve around 50% accuracy. Zero anaphora resolution (ZAR) is a difficult subtask of PA, detecting a zero pronoun and identifying a referent of the zero pronoun. As the following example shows, CR in English (identifying the antecedent of *it*) and ZAR in Japanese (identifying the omitted nominative argument) are similar problems.

(1) a. John bought a car last month.
It was made by Toyota.

b. ジョンは 先月 車を 買った。
John-TOP last month a car-ACC bought.
(ϕ が) トヨタ製だった。
(ϕ-NOM) Toyota made-COPULA.

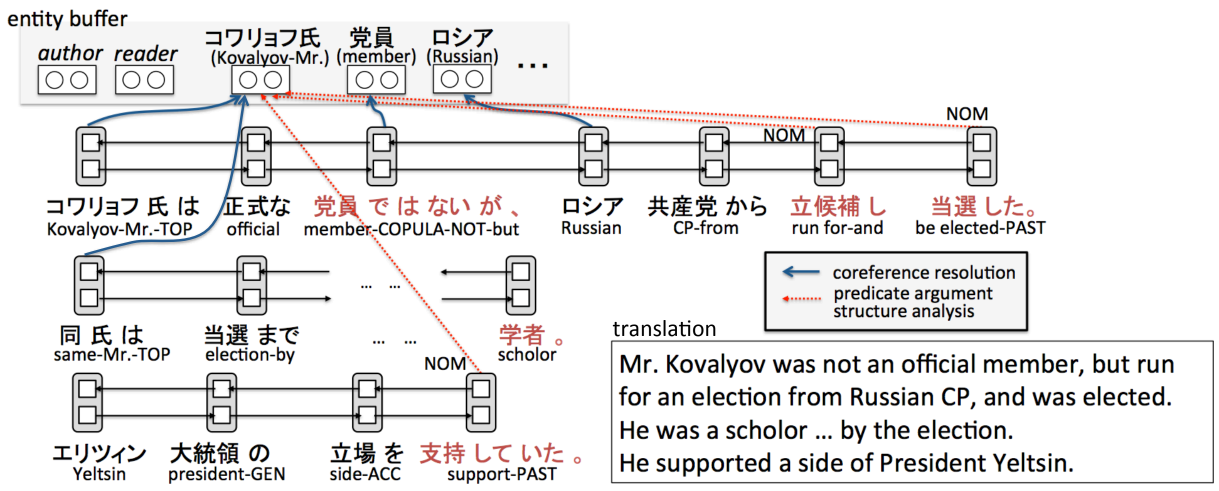
Note that CR such as the relation between "the company" and "Toyota" is also difficult in Japanese.

According to the argument position relative to the predicate, ZAR is classified into the following three types:

- *intra-sentential* (*intra* in short): an argument is located in the same sentence with the predicate
- *inter-sentential* (*inter* in short): an argument is located in the preceding sentences, such as "車" for "トヨタ製だった" (Toyota made-COPULA) in sentence (1b)
- *exophora*: an argument does not appear in a document, such as *author* and *reader*

Among these three types, the analysis of *inter* is extremely difficult because there are many candidates in preceding sentences, and clues such as a dependency path between a predicate and an argument cannot be used.

This paper presents a joint model of CR and PA in Japanese. It is necessary to perform them together because PA (especially inter-sentential



Mr. Kovalyov was not an official member, but run for an election from Russian CP, and was elected. He was a scholor ... by the election. He supported a side of President Yeltsin.

Figure 1: An overview of our proposed method. The phrases with red represent a predicate.

ZAR) needs to identify salient entities, which cannot be identified without performing CR and PA simultaneously. Our results support this claim, and suggest that the status quo of PA-exclusive research in Japanese is an insufficient approach.

Our work is inspired by (Wiseman et al., 2016), which described an English CR system, where entities are represented by embeddings, and they are updated by CR results dynamically. We perform Japanese CR and PA by extending this idea. Our experimental results demonstrate the proposed method can improve the performance of the inter-sentential zero anaphora resolution drastically.

2 Related Work

Predicate Argument Structure Analysis. Early studies have handled both *intra-* and *inter-*sentential anaphora (Taira et al., 2008; Sasano and Kurohashi, 2011), and Hangyo et al. (2013) present a method for handling *exophora*. Recent studies, however, focus on only *intra*-sentential anaphora (Ouchi et al., 2015; Shibata et al., 2016; Iida et al., 2016; Ouchi et al., 2017; Matsubayashi and Inui, 2017), because the analysis of *inter*-sentential anaphora is extremely difficult. Neural network-based approaches (Shibata et al., 2016; Iida et al., 2016; Ouchi et al., 2017; Matsubayashi and Inui, 2017) have improved its performance.

Although most of studies did not consider the notion *entity*, Sasano and Kurohashi (2011) consider an entity, and its salience score is calculated based on simple rules. However, they used gold coreference links to form the entities, and reported the salience score did not improve the performance. In contrast, we perform CR automatically, and capture the entity salience by using RNNs.

For Chinese, where zero anaphors are often used, neural network-based approaches (Chen and Ng, 2016; Yin et al., 2017) outperformed conventional machine learning approaches (Zhao and Ng, 2007).

Coreference Resolution. CR has been actively studied in English and Chinese. Neural network-based approaches (Wiseman et al., 2016; Clark and Manning, 2016b,a; Lee et al., 2017) outperformed conventional machine learning approaches (Clark and Manning, 2015). Wiseman et al. (2016) and Clark and Manning (2016b) learn an entity representation and integrate this into a mention-based model. Our work is inspired by Wiseman et al. (2016), which learn the entity representation by using Recurrent Neural Networks (RNNs). Clark and Manning (2016b) adopt a clustering approach for the entity representation. The reason why we do not use this is that if we take a clustering approach in our setting, zero pronouns need to be first identified before clustering, and thus, it is hard to perform CR and PA jointly. Lee et al. (2017) take an end-to-end approach, aiming at not relying on hand-engineering mention detector (consider all spans as potential mentions). In used Japanese evaluation corpora, since the basic unit for the annotations and our analyses (CR and PA) is fixed, we do not need consider all spans.

In Japanese, CR has not been actively studied other than Iida et al. (2003); Sasano et al. (2007)

since the use of zero pronouns is more common and problematic.

Semantic Role Labeling. Japanese PA is similar to Semantic Role Labeling (SRL) in English. Neural network-based approaches have improved the performance (Zhou and Xu, 2015; He et al., 2017). In these approaches, an appropriate argument for a predicate is searched among mentions in a text. The notion *entity* is not considered.

Other Entity-Centric Study. There are several studies that consider the notion *entity* in other areas: text comprehension (Kobayashi et al., 2016; Henaff et al., 2016) and language modeling (Ji et al., 2017).

3 Japanese Preliminaries

Before presenting our proposed method, we describe the basics of Japanese predicate argument structure and its analysis.

Since the word order is relatively free among arguments in Japanese, an argument is followed by a case marking postposition. The postpositions が (*ga*), を (*wo*), and に (*ni*) indicate nominative (NOM), accusative (ACC) and dative (DAT), respectively. In the double nominative construction such as "私が英語が上手だ" (My English is good), "英語" (English) is regarded as NOM, and "私" (I), the outer nominative is regarded as NOM2. This paper targets these four cases.

PA is tightly related to a dependency structure of a sentence. Considering the relation between a predicate and its argument, and a necessary analysis can be classified into the following three categories (see example sentence (2) below).

(2) ジョン は 買った パン を 食べた
John-TOP bought bread–ACC ate.

Overt case: When an argument with a case marking postposition has a dependency relation with a predicate, PA is not necessary. In example (2), since "パン を" (bread-ACC) has a dependency relation with "食べた" (ate), it is obvious that "食べた" takes "パン" as its ACC argument.

Case analysis: When a topic marker は (*wa*) is attached to an argument, the case marking postposition disappears, and the analysis of identifying the case role becomes necessary. The analysis is called *case analysis*. In the example, although "ジョン は" (John-TOP) has a dependency relation with "食べた" (ate), the analysis of identifying NOM is necessary. The same phenomenon happens when a relative clause is used. When an argument is modified by a relative clause, we do not know its case role to the predicate in the relative clause. In the example, although "パン" has a dependency relation with "買った" (bought), the analysis of identifying ACC is necessary.

Zero anaphora resolution (ZAR): Some arguments are not included in the phrases with which a predicate has a dependency relation. While pronouns are mostly used in English, they are rarely used in Japanese. This phenomenon is called *zero anaphora*, and the analysis of identifying an argument (referent of the zero pronoun) is called *zero anaphora resolution* (ZAR). In the example, although "買った" takes "ジョン" as its NOM argument, they do not have a dependency relation, and thus zero anaphora resolution is necessary.

When dependency relations are identified by parsing, what Japanese PA has to do is case analysis and zero anaphora resolution.

Each predicate has a set of required cases, but not all the four cases. For example, "買う" (buy) takes NOM and ACC, but neither DAT nor NOM2. PA for "買う" in sentence (2) has to find John as NOM, but also has to judge that it does not take DAT and NOM2 arguments.

Another difficulty lies in that a predicate takes a case, but in a sentence it does not take a specific argument. For example, in the sentence "it is difficult to bake a bread", NOM of "bake" is not a specific person, but means "anyone" or "in general". In such cases, PA has to regard arguments as *unspecified*.

4 Overview of Our Proposed Method

An overview of our proposed model is described with a motivated example (Figure 1). Our model equips an *entity buffer* for entity management. At first, it contains only special entities, *author* and *reader*.

In Japanese CR and PA, a basic phrase, which consists of one content word and zero or more function words, is adopted as a basic unit. When an input text is given, the contextual representations of basic phrases are obtained by using Convolutional Neural Network (CNN) and Bidirectional LSTM. Then, from the beginning of the text, CR is performed if a target phrase is a noun phrase, and PA is performed if a target phrase is a predicate phrase. Both of these analyses take into consideration not only the mentions in the text but also the entities in the entity buffer.

In CR, when a mention refers to an existing entity, the entity embedding in the entity buffer is updated. In Figure 1, "同氏" (said person) is analyzed to refer to "コワリョフ氏" (Mr.Kovalyov), and the entity embedding of "コワリョフ氏" is updated. When a mention is analyzed to have no antecedent, it is registered to the entity buffer as a new entity.

In PA, when a predicate has no argument for any case, its argument is searched among any mentions in the text, *author* and *reader*. In the same way as CR, PA takes into consideration not only the mentions but also entities in the entity buffer, and updates the entity embedding.

In Figure 1, the predicate "立候補し" (run for) has no NOM argument. Our method finds "コワリョフ氏" as its NOM argument, and then updates its entity embedding. As mentioned before, the entity embedding of "コワリョフ氏" is updated by the coreference relation with "同氏" in the second sentence. In the third sentence, the predicate "支持していた" (support) has also no NOM argument, and "コワリョフ氏" is identified as its NOM argument, because the frequent reference implies its salience.

5 Base Model

5.1 Input Encoding

Conventional machine learning techniques have extracted features from a basic phrase, which require much effort on feature engineering. Our method obtains an embedding of each basic phrase using CNN and bi-LSTM as shown in Figure 2.

Suppose the i-th basic phrase bp_i consists of $|bp_i|$ words. First, the embedding of each word is represented as a concatenation of word (lemma), part of speech (POS), sub-POS and conjugation embeddings. We append start-of-phrase and end-of-phrase special words to each phrase in order to better represent prefixes and suffixes. Let $W^i \in \mathbb{R}^{d\times(|bp_i|+2)}$ be an embedding matrix for bp_i where d denotes the dimension of word representation.

The embedding of the basic phrase is obtained by applying CNN to the sequence of words. A feature map $\boldsymbol{f}^i$ is obtained by applying a convolution between W^i and a filter H of width n. The m-th element of $\boldsymbol{f}^i$ is obtained as follows:

$$\boldsymbol{f}^i[m] = \tanh(\langle W^i[*, m : m+n-1], H\rangle), \quad (1)$$

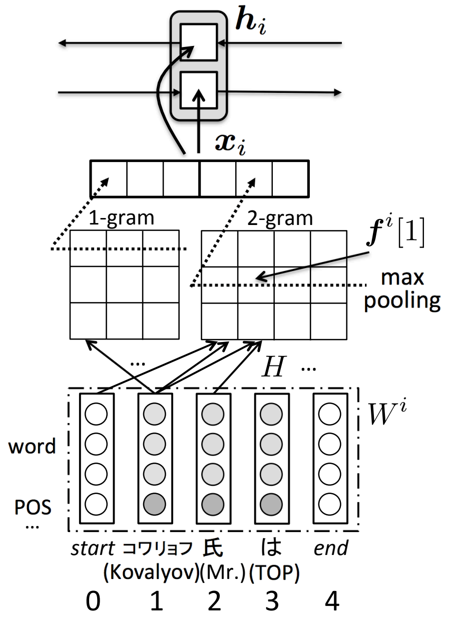


Figure 2: Basic phrase embedding obtained with CNN and Bi-LSTM.

where $W^i[*, m : m+n-1]$ denotes the m-to-($m+n-1$)-th column of W^i, and $\langle A, B\rangle = \mathrm{Tr}(AB^{\mathrm{T}})$ is the Frobenius inner product. Then, to capture the most important feature for a given filter in bp_i, the max pooling is applied as follows:

$$x^i = \max_m \boldsymbol{f}^i[m]. \quad (2)$$

The process described so far is for one filter. The multiple filters of varying widths are applied to obtain the representation of bp_i. When we set h filters, $\boldsymbol{x}_i$, the embedding of the i-th basic phrase, is represented as $[x^i_1, \cdots, x^i_h]$.

The embeddings of basic phrases are read by bi-LSTM to capture their context as follows:

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_i &= \overrightarrow{LSTM}(\boldsymbol{x}_i, \overrightarrow{\boldsymbol{h}}_{i-1}), \\ \overleftarrow{\boldsymbol{h}}_i &= \overleftarrow{LSTM}(\boldsymbol{x}_i, \overleftarrow{\boldsymbol{h}}_{i+1}), \end{aligned} \quad (3)$$

and the contextualized embedding of the i-th basic phrase is represented as a concatenation of the hidden layers of forward and backward LSTM.

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i] \quad (4)$$

This process is performed for each sentence. Since CR and PA are performed for a whole document D, the indices of basic phrases are reassigned from the beginning to the end of D in a consecutive order: $D = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_i, \cdots\}$.

To handle exophora, *author* and *reader* are assigned a unique trainable embedding, respectively.

5.2 Coreference Resolution

We adopt a mention-ranking model that assigns each mention its highest scoring candidate antecedent. This model assigns a score $s^m_{CR}(ant, m_i)$ to a target mention m_i and its candidate antecedent ant[1]. The candidate antecedents include i) mentions preceding m_i, ii) *author* and *reader*, and iii) $\texttt{NA}_{\texttt{CR}}$ (no antecedent). $s^m_{CR}(ant, m_i)$ is calculated as follows:

$$s^m_{CR}(ant, m_i) = W_2^{CR} ReLU(W_1^{CR} \boldsymbol{v}^{CR}_{input}), \quad (5)$$

where W_1^{CR} and W_2^{CR} are weight matrices, and $\boldsymbol{v}^{CR}_{input}$ is an input vector, a concatenation of the following vectors:

- embeddings of m_i and ant
- exact match or partial match between strings of m_i and ant
- sentence distance between m_i and ant. The distance is binned into one of the buckets [0, 1, 2, 3+].
- whether a pair of m_i and ant has an entry in a synonym dictionary.

When a candidate antecedent is $\texttt{NA}_{\texttt{CR}}$, the input vector is just the embedding of a target mention m_i, and the same neural network with different weight matrices calculates a score.

The following margin objective is trained:

$$\mathcal{L}_{CR} = \sum_i^{N_m} \max_{ant \in \mathcal{ANT}(m_i)} (1 + s^m_{CR}(ant, m_i) - s^m_{CR}(\hat{t}_i, m_i)), \quad (6)$$

where N_m denotes the number of mentions in a document, $\mathcal{ANT}(m_i)$ denotes the set of candidate antecedents of m_i, and $\hat{t}_i$ denotes the highest scoring true antecedent of m_i defined as follows:

$$\hat{t}_i = \operatorname*{argmax}_{ant \in \mathcal{T}(m_i)} s^m_{CR}(ant, m_i), \quad (7)$$

where $\mathcal{T}(m_i)$ denotes the set of true antecedents of m_i.

5.3 Predicate Argument Structure Analysis

When a target phrase is a predicate phrase, PA is performed. For each case of a predicate, PA searches an appropriate argument among candidate arguments: i) basic phrases located in the sentence including the predicate and preceding sentences, ii) *author* and *reader*, iii) unspecified, and

[1]The superscript m of $s^m_{CR}(ant, m_i)$ represents a *mention*-based score, which contrasts with an *entity*-based score introduced in Section 6.

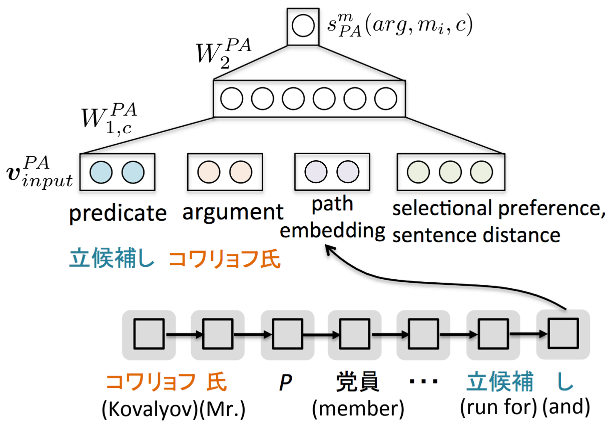


Figure 3: A neural network for PA.

iv) $\texttt{NA}_{\texttt{PA}}$ which means the predicate takes no argument of for the case.

The probability that the predicate m_i takes an argument arg for case c is defined as follows:

$$P(c = arg|m_i) = \frac{\exp(s^m_{PA}(arg, m_i, c))}{\sum_{\substack{carg \in \\ \mathcal{ARG}(m_i)}} \exp(s^m_{PA}(carg, m_i, c))}, \quad (8)$$

where $\mathcal{ARG}(m_i)$ denotes the set of candidate arguments of m_i, and a score $s^m_{PA}(arg, m_i, c)$ is calculated by a neural network as follows (Figure 3):

$$s^m_{PA}(arg, m_i, c) = W_2^{PA} \tanh(W_{1,c}^{PA} \boldsymbol{v}^{PA}_{input}), \quad (9)$$

where $W_{1,c}^{PA}$, W_2^{PA} are weight matrices, and $\boldsymbol{v}^{PA}_{input}$ is an input vector, a concatenation of the following vectors:

- embeddings of m_i and arg[2]
- path embedding: the dependency path between a predicate and an argument is an important clue. Roth and Lapata (2016) learn a representation of a lexicalized dependency path for SRL. An LSTM reads words[3] from an argument to a predicate along with a dependency path, and the final hidden state is adopted as the embedding of the dependency path.[4] For case analysis, the direct dependency relation between a predicate and its argument can be represented as the path embedding.

[2]An embedding for $\texttt{NA}_{\texttt{PA}}$ is assigned a trainable one.

[3]We add special words {Parent, Child}, which indicate a dependency direction between basic phrases.

[4]When an argument is an *inter* or *exophora*, the path embedding is set to be a zero vector.

- selectional preference: selectional preference is another important clue for PA. A selectional preference score is learned in an unsupervised manner from automatic parses of a raw corpus (Shibata et al., 2016).
- sentence distance between m_i and arg. The distance is binned in the same way as CR.

The objective is to minimize the cross entropy between predicted and true distributions:

$$\mathcal{L}_{PA} = -\sum_{i}^{N_p}\sum_{c} \log P(c = \widehat{arg}|p_i), \tag{10}$$

where N_p denotes the number of predicates in a document, and $\widehat{arg}$ denotes a true argument.

6 Entity-Centric Model

While the base model performs *mention*-based CR and PA, our proposed model performs *entity*-based analyses as shown in Figure 1.

6.1 Entity Embedding Update

The entity embeddings are managed in an entity buffer. First, let us introduce time stamp i for the entity embedding update. Time i corresponds to the analysis for the i-th basic phrase in a document. If an entity is referred to by the analysis, its embedding is updated. Let $\boldsymbol{e}_i^{(k)}$ be the embedding of an entity k at time i (after the entity embedding is updated).

In CR, following Wiseman et al. (2016), when a target phrase m_i refers to the entity k, $\boldsymbol{e}_i^{(k)}$ is updated as follows:

$$\boldsymbol{e}_i^{(k)} \leftarrow LSTM_e(\boldsymbol{h}_i, \boldsymbol{e}_{i-1}^{(k)}) \tag{11}$$

where $LSTM_e$ denotes an LSTM for the entity embedding update. When an antecedent is $\mathtt{NA}_{\mathrm{CR}}$, a new entity embedding is set up, initialized by a zero vector. The entity buffer maintains K LSTMs (K is the number of entities in a document), and their parameters are shared.

The proposed method updates the entity embedding not only in CR but also in PA. When the referent of a zero pronoun of case c of predicate p_i is entity k, the entity embedding is updated by using the predicate embedding $\boldsymbol{h}_i$ multiplied by a weight matrix W_c for case c as follows:

$$\boldsymbol{e}_i^{(k)} \leftarrow LSTM_e(W_c\boldsymbol{h}_i, \boldsymbol{e}_{i-1}^{(k)}). \tag{12}$$

In both CR and PA, the embeddings of entities other than the referred entity k are not updated ($\boldsymbol{e}_i^{(l)} \leftarrow \boldsymbol{e}_{i-1}^{(l)} (l \neq k)$).

6.2 Use of Entity Embedding in CR and PA

Both CR and PA are allowed to take the entity embeddings into consideration. In CR, let z_{ant} denote the id of an entity to which the candidate antecedent ant belongs. The *entity*-based score s_{CR}^{e} is calculated as follows:

$$s_{CR}^{e}(ant, m_i) = \begin{cases} \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1}^{(z_{ant})} & (ant \neq \mathtt{NA}_{\mathrm{CR}}) \\ g_{NA}(m_i) & (ant = \mathtt{NA}_{\mathrm{CR}}). \end{cases} \tag{13}$$

The intuition behind the first case is that the dot-product of $\boldsymbol{h}_i$, the embedding of the target mention, and $\boldsymbol{e}_{i-1}^{(z_{ant})}$, the embedding of the entity that ant belongs to indicates the plausibility of their coreference. $g_{NA}(m_i)$ is defined as follows:

$$g_{NA}(m_i) = \boldsymbol{q}^{\mathrm{T}} \tanh(W_{NA}\left[\begin{smallmatrix} \boldsymbol{h}_i \\ \sum_k \boldsymbol{e}_{i-1}^{(k)} \end{smallmatrix}\right]), \tag{14}$$

where $\boldsymbol{q}$ is a weight vector, and W_{NA} is a weight matrix. The intuition is that whether a target phrase is $\mathtt{NA}_{\mathrm{CR}}$ can be judged from $\boldsymbol{h}_i$, the embedding of the target mention itself, and the sum of all the current entity embeddings. s_{CR}^{e} is added to s_{CR}^{m}, and the training objective is the same as the one described in Section 5.2.

In PA, the entity embedding corresponding to a candidate argument arg[5] is just added to the input vector $\boldsymbol{v}_{input}^{PA}$ described in Section 5.3, and *mention*- and *entity*-based score $s_{PA}^{m+e}(arg, m_i, c)$ is calculated in the same way as $s_{PA}^{m}(arg, m_i, c)$. The training objective is again the same as the one in Section 5.3.

In Wiseman et al. (2016), the oracle entity assignment is used for the entity embedding update in training, and the system output is used in a greedy manner in testing. Since the performance of PA is lower than that of English CR, there might be a more significant gap between training and testing. Therefore, scheduled sampling (Bengio et al., 2015) is adopted to bridge the gap: in training, the oracle entity assignment is used with probability ϵ_t (at the t-th iteration) and the system output otherwise. Exponential decay is used: $\epsilon_t = k^t$ (we set $k = 0.75$ for our experiments).

7 Experiments

7.1 Experimental Setting

The two kinds of evaluation sets were used for our experiments. One is the KWDLC (Kyoto Uni-

[5]When arg is $\mathtt{NA}_{\mathrm{PA}}$, the entity embedding is set to a zero vector.

versity Web Document Leads Corpus) evaluation set (Hangyo et al., 2012), and the other is Kyoto Corpus. KWDLC consists of the first three sentences of 5,000 Web documents (15,000 sentences) and Kyoto Corpus consists of 550 News documents (5,000 sentences). Word segmentations, POSs, dependencies, PASs, and coreferences were manually annotated (the closest referents and antecedents were annotated for zero anaphora and coreferences, respectively). Since we want to focus on the accuracy of CR and PA, gold segmentations, POSs, and dependencies were used. KWDLC (Web) was divided into 3,694 documents (11,558 sents.) for training, 512 documents (1,585 sents.) for development, and 700 documents (2,195 sents.) for testing; Kyoto Corpus (News) was divided into 360 documents (3,210 sents.) for training, 98 documents (971 sents.) for development, and 100 documents (967 sents.) for testing.

The evaluation measure is an F-measure, and the evaluation of both CR and PA was relaxed using a gold coreference chain, which leads to an entity-based evaluation. We did not use the conventional CR evaluation measures (MUC, B^3, CEAF and CoNLL) because our F-measure is almost the same as MUC, which is a link-based measure, and the other measures considering singletons get excessively high values[6], and thus they do not accord with the actual performance in our setting.[7]

7.2 Implementation Detail

The dimension of word embeddings was set to 100, and the word embeddings were initialized with pre-trained embeddings by Skip-gram with a negative sampling (Mikolov et al., 2013) on a Japanese Web corpus consisting of 100M sentences. The dimension of POS, sub-POS and conjugation were set to 10, respectively, and these embeddings were initialized randomly. The dimensions of the hidden layer in all the neural networks were set to 100. We used filter windows of 1,2,3 with 33 feature maps each for basic phrase CNN. Adam (Kingma and Ba, 2014) was adopted as the optimizer. F measures were averaged over four runs.

Checkpoint ensemble (Chen et al., 2017) was adopted, where the k best models were taken in terms of validation score, and then the parameters from the k models were averaged for testing. This method requires only one training process. In our experiments, k was set to 5, and the maximum number of epochs was set to 10.

We used a single-layer bi-LSTM for the input encoding (Section 5.1); preliminary experiments with stacked stacked bi-directional LSTM with residual connections were not favorable. Although we tried to use the character-level embedding of each word obtained with CNN, as the same way in the basic phrase embedding from the word sequences, the performance was not improved. The synonym dictionary used for CR (Section 5.2) was constructed from an ordinary dictionary and Web corpus, and has about 7,300 entries (Sasano et al., 2007).

7.3 Experimental Result

The following three methods were compared:

- Baseline: the method described in Section 5.
- "+entity (CR)": this method corresponds to (Wiseman et al., 2016). Entity embedding is updated based on the CR result, and CR takes the entity embedding into consideration.
- "+entity (CR,PA)" (**proposed method**): entity embedding is updated based on PA as well as CR result, and the CR and PA take the entity embedding into consideration.

The performance of CR and PA (case analysis and zero anaphora resolution (ZAR)) is shown in Table 1. The performance of CR and case analysis was almost the same for all the methods. For ZAR, "+entity (CR,PA)" improved the performance drastically.

CR surely benefits from the entity salience. Since entity embeddings are updated based on system outputs, its performance matters. The performance of Japanese CR is lower than that of English CR. Therefore, we assume there are improved/worsen examples, and our CR performance did not improve significantly. The performance of ZAR also matters. However, the performance of ZAR in our baseline model is extremely low, and thus there are few worsen examples and

[6]In Japanese, since zero pronouns are often used, there are many singletons. In example sentences (1) of the Introduction section, while "a car" and "It" form one cluster in English sentences (1-a), "a car" is a singleton in Japanese sentences (1-b) because a zero pronoun is used in the second sentence.

[7]For the Web evaluation set, the F-measure of our proposed method is 0.685, and the conventional evaluation measures are as follows; MUC: 69.1, B^3: 97.2, CEAF: 95.7, and CoNLL: 87.3.

	Web			News		
method	coreference resolution	case analysis	zero anaphora resolution (ZAR)	coreference resolution	case analysis	zero anaphora resolution (ZAR)
Baseline	0.661	0.887	0.516	**0.543**	**0.896**	0.278
+entity (CR)	0.666	0.890	0.518	0.539	0.894	0.275
+entity (CR,PA)	**0.685**	**0.892**	**0.581**	0.541	0.895	**0.356**

Table 1: Performance (F-measure) of coreference resolution, case analysis and zero anaphora resolution.

		Web					News				
case	method	case analysis	zero anaphora resolution (ZAR)				case analysis	zero anaphora resolution (ZAR)			
			all	*intra*	*inter*	*exophora*		all	*intra*	*inter*	*exophora*
NOM	Baseline	0.942	0.575	0.466	0.083	0.695	0.939	0.316	0.455	0.042	0.261
	+entity (CR)	**0.945**	0.579	0.475	0.117	0.693	**0.940**	0.315	0.452	0.037	0.239
	+entity (CR,PA)	**0.945**	**0.646**	**0.508**	**0.502**	**0.721**	**0.940**	**0.390**	**0.486**	**0.256**	**0.357**
	# of arguments	(1,461)	(2,009)	(338)	(393)	(1,278)	(905)	(1,016)	(451)	(388)	(177)
ACC	Baseline	0.853	0.268	0.368	0.119	0.000	**0.679**	**0.053**	**0.093**	0.000	0.000
	+entity (CR)	0.855	0.254	0.357	0.108	0.000	0.631	0.025	0.048	0.000	0.000
	+entity (CR,PA)	**0.857**	**0.343**	**0.413**	**0.282**	0.000	0.651	0.016	0.028	0.000	0.000
	# of arguments	(299)	(224)	(106)	(105)	(13)	(105)	(97)	(41)	(56)	(0)
DAT	Baseline	**0.498**	0.432	0.115	0.016	0.581	**0.308**	0.183	**0.039**	0.000	0.367
	+entity (CR)	0.445	0.422	0.119	0.016	0.574	0.223	0.162	0.005	0.000	0.334
	+entity (CR,PA)	0.411	**0.465**	**0.133**	**0.126**	**0.600**	0.292	**0.328**	0.030	**0.005**	**0.566**
	# of arguments	(101)	(576)	(86)	(149)	(341)	(26)	(286)	(82)	(89)	(115)
NOM2	Baseline	0.478	0.216	**0.259**	0.000	0.245	**0.098**	0.000	0.000	0.000	0.000
	+entity (CR)	0.501	0.212	0.226	0.000	0.257	0.069	0.000	0.000	0.000	0.000
	+entity (CR,PA)	**0.526**	**0.283**	0.240	**0.112**	**0.341**	0.092	0.000	0.000	0.000	0.000
	# of arguments	(110)	(140)	(29)	(28)	(83)	(13)	(37)	(17)	(13)	(7)
all	Baseline	0.887	0.516	0.400	0.074	0.654	**0.896**	0.278	0.394	0.032	0.291
	+entity (CR)	0.890	0.518	0.405	0.093	0.654	0.894	0.275	0.396	0.027	0.265
	+entity (CR,PA)	**0.892**	**0.581**	**0.439**	**0.399**	**0.681**	0.895	**0.356**	**0.417**	**0.204**	**0.432**
	# of arguments	(1,971)	(2,949)	(559)	(675)	(1,715)	(1,049)	(1,436)	(591)	(546)	(299)

Table 2: Performance of case analysis and zero anaphora resolution for each case, and each argument position for zero anaphora resolution. The underlined values indicate the proposed method outperforms the baseline by a large margin.

a number of improved examples. Therefore, ZAR can benefit from the entity representation obtained by both CR and PA.

Table 2 shows performance of case analysis and zero anaphora resolution for each case, and each argument position. *Unspecified* was counted for *exophora*. Both for the News and Web evaluation sets, the performance for *inter* arguments of zero anaphora resolution, which was extremely difficult in the baseline method, was improved by a large margin by our proposed method.

7.4 Ablation Study

To reveal the importance of each clue for CR and PA, each clue was ablated. Table 3 shows the result on the development set. We found that, the path embedding was effective for PA, and the string match was effective for CR. The sentence distance for both CR and PA was effective for News, but not for Web since the Web evaluation corpus consists of three-sentence documents.

7.5 Comparison with Other Work

It is difficult to compare the performance of our method with other studies directly because there are no studies handling both CR and PA. The comparisons with other studies are summarized as follows:

- Shibata et al. (2016) proposed a neural-network based PA. Their target was *intra* and *exophora* for three major cases (NOM, ACC and DAT), and the performance was 0.534 on the same Web corpus as ours. The performance of our proposed method for the same three cases was 0.626. Furthermore, since their model assumes a static PA graph, their model is difficult to be extended to handle CR.
- Ouchi et al. (2017) proposed a grid-type RNN model for capturing the multi-predicate interaction. Their target was only *intra* on the NAIST text corpus (News), and the performance was 47.1. Since the NAIST text

	coreference resolution				zero anaphora resolution (ZAR)			
	Web		News		Web		News	
	F1	Δ	F1	Δ	F1	Δ	F1	Δ
Our proposed model	0.633		0.613		0.512		0.361	
CR								
- string match	0.212	-0.420	0.184	-0.429	0.474	-0.038	0.348	-0.013
- sentence distance	0.643	+0.011	0.588	-0.025	0.505	-0.007	0.343	-0.018
- synonym dictionary	0.643	+0.010	0.613	0.000	0.510	-0.002	0.348	-0.013
PA								
- path embedding	0.643	+0.010	0.625	+0.012	0.459	-0.054	0.268	-0.093
- selectional preference	0.638	+0.005	0.316	-0.297	0.507	-0.005	0.173	-0.188
- sentence distance	0.647	+0.014	0.606	-0.007	0.516	+0.004	0.327	-0.034

Table 3: Ablation study on the development set. The cells shaded gray represent they are not directly affected from the ablation, but from the counterpart analysis result.

corpus contains a lot of annotation errors as pointed out in Iida et al. (2016), we did not conduct our experiments on the NAIST text corpus.

- Iida et al. (2003) reported an F-measure of about 0.7 on News domain. The possible reason why our performance on News (0.541) is lower than theirs is that their basic unit is a compound noun while our basic unit is a noun, and thus our setting is difficult in comparison with theirs.

Since we handle *inter* as well as *intra* and *exophora* arguments in PA, together with CR, we can say that our experimental setting is more practical in comparison with other studies.

7.6 Error Analysis

In example (3), although the NOM argument of the predicate "通院ですよ!" (go to hospital) is *author*, our method wrongly classified it as *unspecified*.

(3) 毎日のように 通院ですよ！ 私自身は
every day go to hospital! I myself-TOP
とても 健康なんですけど。
very healthy.
((I) go to hospital every day!
(I am) very healthy, though.)

In the second sentence, our method correctly identified the antecedent of "私" (I) as *author*, and the NOM of "健康なんですけど" (healthy) as "私" (I). Our method adopts the greedy search so that it cannot exploit this handy information in the analysis of the first sentence. The global modeling using reinforcement learning (Clark and Manning, 2016a) for a whole document is our future work.

In example (4), although the NOM argument of "装飾されています" (be decorated) in the second sentence is "ドレス" (dress) in the first sentence, our method wrongly classified it as $\mathtt{NA}_{\mathtt{PA}}$.

(4) 大変 印象的な ドレスです。
very impressive dress-COPULA.
オーガンジーの 上に ラインを 描くように
organdie-GEN top-DAT line-ACC draw-as
小さな ビーズで 装飾されています。
small bead-INS decorated
((This is) a very impressive dress.
(The dress) is decorated by small beads as they draw a line on its organdy.)

"オーガンジー" (organdie) has a bridging relation to "ドレス", which might help capture the salience of "ドレス". The bridging reference resolution is our next target and must be easily incorporated into our model.

8 Conclusion

This paper has presented an entity-centric neural network-based joint model of coreference resolution and predicate argument structure analysis. Each entity has its embedding, and the embeddings are updated according to the result of both of these analyses dynamically. Both of these analyses took the entity embedding into consideration to access the global information of entities. The experimental results demonstrated that the proposed method could improve the performance of the inter-sentential zero anaphora resolution drastically, which has been regarded as a notoriously difficult task. We believe that our proposed method is also effective for other pro-drop languages such as Chinese and Korean.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR1301, Japan.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR* abs/1506.03099. http://arxiv.org/abs/1506.03099.

Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 778–788. http://www.aclweb.org/anthology/P16-1074.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *CoRR* abs/1710.03282. http://arxiv.org/abs/1710.03282.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Association for Computational Linguistics (ACL)*.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 643–653. https://doi.org/10.18653/v1/P16-1061.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*. Faculty of Computer Science, Universitas Indonesia, Bali,Indonesia, pages 535–544. http://www.aclweb.org/anthology/Y12-1058.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 924–934. http://www.aclweb.org/anthology/D13-1095.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *CoRR* abs/1612.03969. http://arxiv.org/abs/1612.03969.

Ryu Iida, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *In Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*. pages 23–30.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1244–1254. https://aclweb.org/anthology/D16-1132.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1831–1840. http://www.aclweb.org/anthology/D17-1195.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the NAACL HLT 2016*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 188–197. http://www.aclweb.org/anthology/D17-1018.

Yuichiroh Matsubayashi and Kentaro Inui. 2017. Revisiting the design issues of local models for japanese predicate-argument structure analysis. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 128–133. http://www.aclweb.org/anthology/I17-2022.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 961–970. `http://www.aclweb.org/anthology/P15-1093`.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Neural modeling of multi-predicate interactions for Japanese predicate argument structure analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1591–1600. `http://aclweb.org/anthology/P17-1146`.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1192–1202. `http://www.aclweb.org/anthology/P16-1113`.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2007. Improving coreference resolution using bridging reference resolution and automatically acquired synonyms. In *DAARC*.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 758–766. `http://www.aclweb.org/anthology/I11-1085`.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural network-based model for Japanese predicate argument structure analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1235–1244. `http://www.aclweb.org/anthology/P16-1117`.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, pages 523–532. `http://www.aclweb.org/anthology/D08-1055`.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 994–1004. `http://www.aclweb.org/anthology/N16-1114`.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017. Chinese zero pronoun resolution with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1320–1329. `https://www.aclweb.org/anthology/D17-1136`.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, pages 541–550. `http://www.aclweb.org/anthology/D/D07/D07-1057`.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1127–1137. `http://www.aclweb.org/anthology/P15-1109`.

Constraining MGbank: Agreement, L-Selection and Supertagging in Minimalist Grammars

John Torr
School of Informatics
University of Edinburgh
11 Crichton Street, Edinburgh, UK
john.torr@cantab.net

Abstract

This paper reports on two strategies that have been implemented for improving the efficiency and precision of wide-coverage Minimalist Grammar (MG) parsing. The first extends the formalism presented in Torr and Stabler (2016) with a mechanism for enforcing fine-grained selectional restrictions and agreements. The second is a method for factoring computationally costly null heads out from bottom-up MG parsing; this has the additional benefit of rendering the formalism fully compatible for the first time with highly efficient Markovian supertaggers. These techniques aided in the task of generating MGbank, the first wide-coverage corpus of Minimalist Grammar derivation trees.

1 Introduction

Parsers based on deep grammatical formalisms, such as CCG (Steedman and Baldridge, 2011) and HPSG (Pollard and Sag, 1994), exhibit superior performance on certain semantically crucial (unbounded) dependency types when compared to those with relatively shallow context free grammars (in the spirit of Collins (1997) and Charniak (2000)) or, in the case of modern dependency parsers (McDonald and Pereira (2006), Nivre et al. (2006)), no explicit formal grammar at all (Rimell et al. (2009), Nivre et al. (2010)). As parsing technology advances, the importance of correctly analysing these more complex construction types will also inevitably increase, making research into deep parsing technology an important goal within NLP.

One deep grammatical framework that has not so far been applied to NLP tasks is the Minimalist Grammar (MG) formalism (Stabler, 1997). Linguistically, MG is a computationally-oriented formalization of many aspects of Chomsky's (1995) Minimalist Program, arguably still the dominant framework in theoretical syntax, but so far conspicuously absent from NLP conferences. Part of the reason for this has been that until now no Minimalist treebank existed on which to train efficient statistical Minimalist parsers.

The Autobank (Torr, 2017) system was designed to address this issue. It provides a GUI for creating a wide-coverage MG together with a module for automatically generating MG trees for the sentences of the Wall Street Journal section of the Penn Treebank (PTB) (Marcus et al., 1993), which it does using an exhaustive bottom-up MG chart parser[1]. This system has been used to create MGbank, the first wide coverage (precision-oriented) Minimalist Grammar and MG treebank of English, which consists of 1078 hand-crafted MG lexical categories (355 of which are phonetically null) and currently covers approximately half of the WSJ PTB sentences. A problem which arose during its construction was that without any statistical model to constrain the derivation, MG parsing had to be exhaustive, and this presented some significant efficiency challenges once the grammar grew beyond a certain size[2], mainly because of the problem of identifying the location and category of phonetically silent heads (equivalent to type-changing unary rules) allowed by the theory. This problem was particularly acute for the MGbank grammar, which makes extensive use of such heads to multiply out the lexicon during pars-

[1]The parser is based on Harkema's (2001) CKY variant.

[2]As Cramer and Zhang (2010) (who pursue a similar treebanking strategy for HPSG) observe, there is very often considerable tension between the competing goals of efficiency and coverage for deep, hand-written and precision-oriented parsers, which aim not only to provide detailed linguistic analyses for grammatical sentences, but also to reject ungrammatical ones wherever possible.

ing. This approach reduces the amount of time needed for manual annotation, and also enables the parser to better generalise to unseen constructions, but it can quickly lead to an explosion in the search space if left unconstrained.

This paper provides details on two strategies that were developed for constraining the hypothesis space for wide-coverage MG parsing. The first of these is an implementation of the sorts of selectional restrictions[3] standardly used by other formalisms, which allow a head to specify certain fine-grained properties about its arguments. Pesetsky (1991) refers to this type of fine-grained selection as *l(exical)-selection*, in contrast to coarser-grained *c(ategory)-selection* and semantic *s-selection*. The same system is also used here to enforce morphosyntactic agreements, such as subject-verb agreement[4] and case 'assignment'. It is simpler and flatter than the structured feature value matrices one finds in formalisms such as HPSG and LFG, which arguably makes it less linguistically plausible. However, it is also considerably easier to read and to annotate, which greatly facilitated the manual treebanking task.

The second technique to be presented is a method for extracting a set of complex overt categories from a corpus of MG derivation trees which has the dual effect of factoring computationally costly null heads out from parsing (but not from the resulting parse trees) and rendering MGs fully compatible for the first time with existing supertagging techniques. Supertagging was originally introduced in Bangalore and Joshi (1999) for the Lexicalised Tree Adjoining Grammar (LTAG) formalism (Schabes et al., 1988), and involves applying Markovian part-of-speech tagging techniques to strongly lexicalised tag sets that are much larger and richer than the 45 tags used by the PTB. Because each supertag contains a great deal of information about the syntactic environment of the word it labels, such as its subcategorization frame, supertagging is sometimes referred to as 'almost parsing'. It has proven highly effective at making CCG (Clark and Curran, 2007; Lewis et al., 2016; Xu, 2016; Wu et al., 2017) parsing in particular efficient enough to support large-scale NLP tasks, making it desirable to apply this technique to MGs. However, existing supertaggers can only tag what they can see, presenting a problem for MGs, which include phonetically unpronounced heads. Our extraction algorithm addresses this by anchoring null heads to overt ones within complex LTAG-like supertag categories.

The paper is arranged as follows: section 2 gives an informal overview of MGs; section 3 introduces the selectional mechanisms and shows how these are used in MGbank to enforce case 'assignment' (3.1), l-selection (3.2) and subject-verb agreement (3.3); section 4 presents the algorithm for extracting supertags from a corpus of MG derivation trees (4.1), gives details of how a standard CKY MG parser can straightforwardly be adapted to make use of these complex tags (4.2), and presents some preliminary supertagging results (4.3) and a discussion of these (4.4); section 5 concludes the paper.

2 Minimalist Grammars

For a more detailed and formal account of the MG formalism assumed in this paper, see Torr and Stabler (2016) (henceforth T&S); here we give only an informal overview. MG is introduced in Stabler (1997); it is a strongly lexicalised formalism in which categories are comprised of lists of structure building features ordered from left to right. These features must be checked against each other and deleted during the derivation, except for a single c feature on the complementizer (C) heading the sentence, which survives intact (equivalent to reaching the S root in classical CFG parsing). Features are checked and deleted via the application of a small set of abstract Merge and Move rules. Two simple MG lexical entries are given below (The :: is a type identifier[5]):

him :: d
helps :: d= v

The structure building features themselves can be categorized into four classes: selector =x/x= features, selectee x features, licensor +y features, and licensee -y features. In a directional MG, such as that presented in T&S, the = symbol on the selector can appear on either side of the x category symbol, and this indicates whether selection is to the left or to the right. For instance, in our toy lexicon *helps*'s first feature is a d= selector, indicating that it is looking for a DP on its right. Since the

[3]These were briefly introduced in Torr (2017), but are expounded here in much greater depth.

[4]The approach to agreement adopted here differs in various respects from the operation Agree (Chomsky (2000) (2001)) assumed in current mainstream Minimalism.

[5]:: indicates a non-derived item and : a derived one.

first feature of *him* is a d selectee, we can merge these two words to obtain the following VP category, where ϵ is the empty string (The reason for the commas separating the left and right dependent string components from the head string component is to allow for subsequent head movement of the latter (see Stabler (2001)):

ϵ, *helps, him* : v

The strings of the two merged elements have been here been concatenated, but this will not always be the case. In particular, if the selected item has additional features behind its selectee, then it will need to check these in subsequent derivational steps via applications of Move. In that case the two constituents must be kept separate within a single expression following Merge. To illustrate this, we will update the lexicon as follows:

him :: d -case
helps :: d= +CASE v

Merging these two items results in the following expression:

ϵ, *helps*, ϵ : +CASE v, *him* : -case

The two subconstituents, separated above by the rightmost comma, are referred to as *chains*; the leftmost chain in any expression is the head of the expression; all other chains are movers. The +CASE licensor on the head chain must now attract a chain within the expression with a matching -case licensee as its first feature to move overtly to its left dependent (specifier) position[6]. Exactly one moving chain *must* satisfy this condition, or this expression will be unable to enter into any further operations (if more than one chain has the same licensee feature, it will violate a constraint on MG derivations known as the *Shortest Move Constraint* (SMC) and automatically be discarded). As this condition is satisfied by just *him*'s -case feature, we can perform the unary operation Move on this expression, resulting in the following new, single-chained expression:

him, helps, ϵ : v

We can represent these binary Merge and unary Move operations using the MG derivation tree in fig 1a. Derivation trees such as this are used frequently in work on Stablerian Minimalist Grammars, but they can be deterministically mapped into phrase structure trees like fig 1b[7].

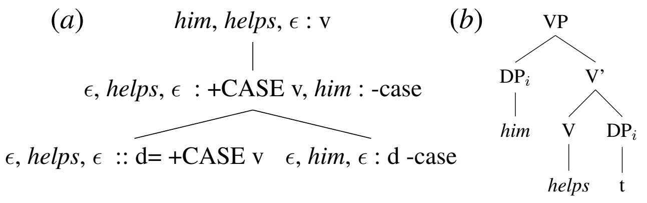


Figure 1: An MG Derivation tree for the VP *him, helps* (a); and its corresponding Xbar phrase structure tree (b). At this stage in the derivation the verb and its object are incorrectly ordered. This will be rectified by subsequent V-to-v head movement placing the verb to the left of its object.

To continue this derivation and derive the transitive sentence *he helps him*, we will expand our lexicon with the following categories, where square brackets indicate a null head and a $>$ diacritic on a selector feature indicates that a variant of Merge is triggered in which the head string of the selected constituent undergoes head movement to the left of the selecting constituent's head string:

he :: d -case
[trans] :: $>$v= =d lv[8]
[pres] :: lv= +CASE t
[decl] :: t= c

The full derivation tree and corresponding Xbar phrase structure tree for the sentence are given in fig 2 and fig 3 respectively.

3 Case, L-selection and Agreement

3.1 Case 'Assignment'

Notice that at present both the nominative and accusative forms of the masculine personal pronoun

[6]Uppercase licensors specify overt movement; lowercase licensors, by contrast, trigger covert movement, where only the features move, not the string (see T&S). Note that the MGbank grammar follows Chomsky's (2008) suggestion that it is the lexical verb V, rather than the null 'little v' head governing it, which checks the object's features, having inherited the relevant licensors (offline we assume) from v. This unifies the analysis of standard transitives with ECM constructions (*Jack expected Mary to help*), which in MGbank involve overt raising of the subject of the embedded infinitival clause to spec-VP to check accusative case (object control *Jack persuaded Mary to help* involves two such movements, the first for theta and the second for case).

[7]MGbank includes MG derivation tree, MG derived (bare phrase structure) tree, and Xbar tree formats.

[8]Note that little v is written as lv in MGbank derivation trees because upper vs lowercase letters are used to trigger different rules. In the corresponding MGbank Xbar trees, however, v has been converted to V and lv to v, to make these trees more familiar.

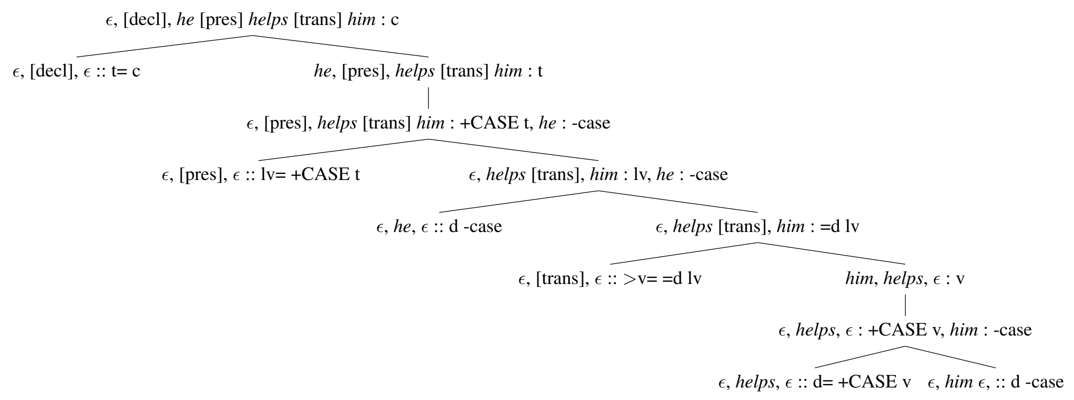

ϵ, [decl], he [pres] helps [trans] him : c
ϵ, [decl], ϵ :: t= c
he, [pres], helps [trans] him : t
ϵ, [pres], helps [trans] him : +CASE t, he : -case
ϵ, [pres], ϵ :: lv= +CASE t
ϵ, helps [trans], him : lv, he : -case
ϵ, he, ϵ :: d -case
ϵ, helps [trans], him : =d lv
ϵ, [trans], ϵ :: >v= =d lv
him, helps, ϵ : v
ϵ, helps, ϵ : +CASE v, him : -case
ϵ, helps, ϵ :: d= +CASE v
ϵ, him ϵ, :: d -case

Figure 2: MG derivation tree for the sentence *he helps him*.

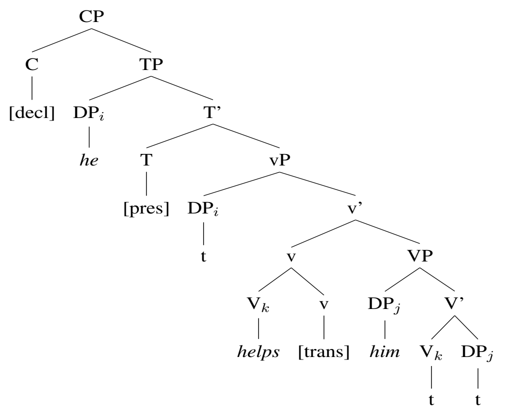


Figure 3: Xbar phrase structure tree for the sentence *he helps him*.

in our lexicon have the same feature sequence. This means that as well as correctly generating *he helps him*, our grammar also overgenerates *him helps he*. One way to solve this would be to split +/-case features into +/-nom and +/-acc. However, many items of category d in English (e.g. *the*, *a*, *you*, *there*, *it*) are syncretised (i.e. have the same phonetic form) for nominative vs. accusative case. This solution therefore lacks elegance as it expands the lexicon with duplicate homophonic entries differing in just a single (semantically meaningless) feature. Furthermore, increasing the size of the set k of licensees could adversely impact parsing efficiency, given that the worst case theoretical time complexity of MG chart parsing is known to be n^{2k+3} (Fowlie and Koller, 2017), where k is the number of moving chains allowed in any single expression by the grammar.

Instead, we will retain the single -case licensee feature and introduce NOM and ACC as subcategories, or *selectional properties*, of this feature. We will also subcategorize licensor features using *selectional requirements* of the form +X and -X, where X is some selectional property. Positive +X features require the presence of the specified property on the licensee feature being checked, while -X features require its absence. For example, consider the following updated lexical entries, where individual selectional features are separated by the . symbol:

him :: d -case{ACC}

he :: d -case{NOM}

helps :: d= +CASE{+ACC} v{PRES.TRANS}

[pres] :: lv{+PRES}= +CASE{+NOM} t{FIN.PRES}

[trans] :: >v{+TRANS}= =d lv

The +ACC selectional requirement on the V head's +CASE licensor specifies that the object's licensee feature must bear an ACC selectional property, while +NOM on the T(ense) head indicates that the subject's licensee must have a NOM property. For SMC purposes, however, these two different subcategories of -case will still block one another, meaning that k remains unaffected. The reader should satisfy themselves that our grammar now correctly blocks the ungrammatical *him helps he*.

We can now also address the aforementioned syncretism issue without increasing the size of the grammar. To do this, we simply allow features to bear multiple selectional properties from the same paradigm. For example, representing the pronoun *it* as follows will allow it to appear in either a nominative or an accusative case licensing position:

it :: d -case{ACC.NOM}

3.2 L-selection

As well as constraining Move, selectional restrictions can also constrain Merge. For instance, we can ensure that a subject control verb like *want* subcategorizes for a *to*-infinitival CP complement, and thereby avoid overgenerating *Jack wants that she help(s)*, simply by using the following categories for *want* and *that*:

want :: c{+INF}= v{TRANS}
that :: t{+FIN}= c{DECL.FIN}

Because *that* lacks the INF feature required by *want*, the ungrammatical derivation is blocked. We also need to block **Jack wants she help(s)*, where the overt C head is omitted. Minimalists assume that finite embedded declaratives lacking an overt C are nevertheless headed by a null C - a silent counterpart of *that*. A complicating factor is that a null complementizer is also assumed to head certain types of embedded infinitivals, including the embedded *help* clause in *Jack wants* [$_{CP}$ *to help*]. Given that these null C heads are (trivially) homophones and that they arguably exist to encode the same illocutionary force[9], an elegant approach would be to minimize the size of the lexicon - and hence the grammar - by treating them as one and the same item. On the other hand, using a single null C head syncretised with both FIN and INF will fail to block **Jack wants she help(s)*.

At present both C and T are specified as FIN, suggesting a redundancy. Instead, therefore, we will assume that T, being the locus of tense, is also the sole locus of inherent finiteness, but that C's selectee may inherit FIN or INF from its TP complement as the derivation proceeds[10]. Only a null C which inherits INF from a *to*-TP complement will be selectable by a verb like *want*, blocking the ungrammatical **Jack wants she help(s)*. However, although lacking inherent tense *properties*, certain C heads continue to bear inherent tense *requirements*[11]; for instance, *that*'s selector will retain its inherent +FIN, identifying it as a *finite* complementizer.

To implement this percolation[12] mechanism, we now introduce *selectional variables*, which we write as x, y, z etc. A variable on a selector or licensor feature will cause all the selectional properties and requirements (but not other variables) contained on the selectee or licensee feature that it checks to be copied onto all other instances of that variable on the selecting or licensing category's remaining unchecked feature sequence. Consider the following:

[trans] :: >v{+TRANS.x}= =d lv{x}
[pres] :: lv{+PRES.x}= +CASE{+NOM.x} t{FIN.x}
to :: lv{+BARE.x}= t{INF.x}
[decl] :: t{x}= c{DECL.x}
that :: t{+FIN.x}= c{DECL.x}

The [pres] T head has an x variable on its lv= selector feature and this same variable also appears to the right on its +CASE licensor and t selectee; any selectional properties or requirements contained on the lv selectee of its vP complement will thus percolate onto these two features (see fig 4). The x's on the two C heads will percolate the FIN property from the t selectee of [pres] to the c selectee of [decl], where it can be selected for by a verb like *say*, but not *want*, which requires INF (contained on the *to* T head); this will correctly block **Jack wants (that) she help(s)*.

Although we will not discuss the details here, it is worth noting that the MGbank grammar also uses this same percolation mechanism to capture long distance subcategorization in English subjunctives, thereby allowing *Jack demanded that she be there on time* while also blocking **Jack demanded that she is there on time*.

[9]Infinitival complementizers are sometimes assumed to encode *irrealis* force (see e.g. Radford (2004)) in contrast to *that* and its null counterpart which encode declarative force. However, the fact that *Jack expects her to help* is (on one reading) virtually synonymous with *Jack expects that she will help* suggests that in both cases the C head is encoding the same semantic property, with any subtle difference in meaning attributable to the contents of the Tense (T) head (i.e. *to* vs. *will*). Consider also *Mary wondered whether to help* vs. *Mary wondered whether she should help*, where the embedded infinitival and finite clauses are both clearly interrogative.

[10]If Grimshaw (1991) is correct that functional projections like DP, TP and CP are part of *extended projections* of the N and V heads they most closely c-command, then we should not be surprised to find instances where fine-grained syntactic properties are projected up through these functional layers.

[11]The property vs. requirement distinction mirrors Chomsky's (1995) interpretable vs. uninterpretable one.

[12]Note that because we are only allowing selectional properties and requirements to percolate, rather than the structure building feature themselves, this system is fundamentally different from that described in Kobele (2005), where it was shown that allowing licensee features to be percolated leads to type 0 MGs. Furthermore, by unifying any multiple instances of the same selectional property or requirement that arise on a structure building feature owing to percolation, we can ensure that the set of MG terminals and non-terminals remains finite and thus that the weak equivalence to MCFG (Michaelis, 1998; Harkema, 2001) is maintained.

ϵ, [pres], *helps* [trans] *him* : +CASE{+NOM.PRES.TRANS.+3SG} t{FIN.PRES.TRANS.+3SG}, *he* : -case{NOM.3SG}

ϵ, [pres], ϵ :: lv{+PRES.x}= +CASE{+NOM.x} t{FIN.x} ϵ, *helps* [trans], *him* : lv{PRES.TRANS.+3SG}, *he* : -case{NOM.3SG}

Figure 4: Merge of T with vP with percolation of selectional properties and requirements.

3.3 Subject-Verb Agreement

The percolation mechanism introduced above can also be used to capture agreement between the subject and the inflected verb. In Minimalist theory, this agreement is only indirect: the subject actually agrees directly with T when it moves to become the latter's specifier, having been initially selected for either by V (in the case of non-agent arguments) or by v (in the case of agent subjects - see fig 3)[13]. There is also assumed to be some sort of syntactic agreement (Roberts (2010)) and/or phonetic (Chomsky (2001)) process operating between T and the inflected verb, resulting in any tense/agreement inflectional material generated in T(ense) being suffixed onto the finite verb.

In MGbank, tense agreement is enforced between T and the finite verb by percolating a PRES or PAST selectional property from the selectee of the latter up through the tree so that it can be selected for by the [pres] or [past] T head. Subject-verb agreement, meanwhile, is enforced by also placing an agreement selectional restriction (+3SG, +1PL, -3SG etc) on the finite verb's selectee, and then percolating this up to the +CASE licensor of the T head. We thus have the following updated entries:

him :: d -case{ACC.3SG}
he :: d -case{NOM.3SG}
helps :: d= +CASE{+ACC} v{+3SG.PRES}

The percolation step from little v (lv) to T is shown in fig 4; lv has already inherited PRES and +3SG from V (*helps*) at this point, and these features now percolate to T's licensor and selectee[14] owing to the x variables; the PRES feature inherited from V by v is selected for by T, enforcing non-local tense agreement between T and V, while the +3SG enforces subject verb agreement[15].

4 MG Supertagging

The above selectional system restricts the parser's search space sufficiently well that it is feasible to generate an initial MG treebank for many of the sentences in the PTB, particularly the shorter ones and those longer ones which do not require the full range of null heads to be allowed into the chart[16]. However, for longer sentences requiring null heads such as extraposers, topicalizers or focalizers, parsing remains impractically slow. In this section we show how computationally costly null heads can be factored out from MG parsing al-

[13]A reviewer asks why all subjects are not directly selected for by V, suggesting that this appears to be a deviation from semantics, and more generally calls for some explanation of the underlying modelling decisions adopted here (e.g. head movements, case movements, null heads etc) which clearly deviate from the more surface oriented analyses of other formalisms used in NLP. In many cases these decisions rest on decades of research which we cannot hope to summarise here; for good introductions to Minimalism, see Radford (2004) and Hornstein et al. (2005). It is worth noting, however, that the null v head in fig 3 is essentially a valency increasing causative morpheme which ends up suffixed to the main verb (via head movement of the latter), effectively enabling it to take an additional 'external' argument. We can therefore view the V-v complex as a single synthetic verbal head, so that just as in a language like Turkish the verb *öl* meaning 'to die' can be transformed from an intransitive to a transitive (meaning 'to kill') by appending to it the causative suffix *dür*, in English a verb like *break* can be transformed from an intransitive (*the window broke*) to a transitive (*he broke the window*) by applying a null version of this morpheme. This cross-linguistic perspective (which makes this formalism potentially very relevant for machine translation) reflects a central goal of Minimalism, which is to show that at a relevant level of abstract representation, all languages share a common syntax (making them easier for children to learn). Most of the analyses adopted here are standard ones from the literature (see e.g. Larson's (1988) VP Shell Hypothesis, Baker's (1988) Uniform Theta Assignment Hypothesis, Koopman and Sportiche's (1991) Verb Phrase Internal Subject Hypothesis, and Chomsky (1995; 2008) on little v).

[14]Note that selectional requirements are entirely inert on selectee and licensee features while, conversely, selectional properties are inert on selectors and licensors.

[15]For non-3SG present tense verbs, MGbank uses a -3SG negative selectional requirement; for verbs with more complex paradigms, however, the grammar allows for inclusive disjunctive selectional requirements. For example, the selectee feature of the *was* form of the verb *be* bears the feature [+1SG|+3SG], allowing it to take either a first or third singular subject.

[16]The Autobank parser holds certain costly null heads back from the chart and only introduces these incrementally if it fails to parse the sentence without them. The advantage of this strategy is that it improves efficiency for many sentences, but the disadvantage is that it can also result in correct analyses being bled by incorrect ones. The supertagging approach introduced in this section eliminates this problem, since null heads are now anchored to overt ones as part of complex categories, any of which may freely be assigned by the supertagger.

together by anchoring them to overt heads within complex overt categories extracted from this initial treebank. This allows much more of the disambiguation work to be undertaken by a statistical Markovian supertagger[17], a strategy which has proven highly effective at rendering CCG parsing in particular efficient enough for large-scale NLP tasks. We also show how a standard CKY MG parser can be adapted to make use of these complex categories, and present some preliminary supertagging results.

4.1 Factoring null heads out from MG parsing

Consider again the lexical items which appear along the spine of the clause in fig 2.

[decl] :: t= c
[pres] :: lv= +CASE t
[trans] :: >v= =d lv
helps :: d= +CASE v

Recall that the null [trans] little v merges with the VP headed by overt *helps*, while the null [pres] T head merges with the vP, and the null [decl] C with TP. If we view each of these head-complement merge operations as a link in a chain, then all of these null heads are either directly (in the case of v) or indirectly (in the case of T and C) linked to the overt verb. All of the information represented on V, v, T and C heads in Minimalism is in LTAG represented on a single overt lexical category (known as an initial tree). We can adopt this perspective for Minimalist parsing if we view chains of merges that start with some null head and end with some overt head as constituting complex overt categories. Given a corpus of derivation trees, it is possible to extract all such chains appearing in the corpus, essentially precompiling all of the attested combinations of null heads with their overt anchors into the lexicon. A very simple algorithm for doing this is given below.

```
for each derivation tree $\tau$:
  for each null head $\eta$ in $\tau$:
     if $\eta$ is a proform:
        linkWithGovernor($\eta$);
     else:
        linkWithHeadOfComplement($\eta$);
  groupLinksIntoSupertags()
```

[17]During treebank generation we used the C&C (Clark and Curran, 2007) supertagger retrained to take gold CCGbank categories and words as input and output MGbank supertags.

For each derivation tree, we first anchor all null heads either directly or indirectly to some overt head; this is achieved by extracting a set of links, each of which represents one merge operation in the tree. Each link is comprised of the two atomic MG lexical categories that are the arguments to the merge operation along with matching indices indicating which features are checked by the operation. Applying the algorithm to our example sentence would result in the following 3 links:

link1: [decl] :: t=^{1} c, [pres] :: lv= +CASE t^1
link2: [pres] :: lv=^{2} +CASE t, [trans] :: v= =d lv^2
link3: [trans] :: v=^{3} =d lv, *helps* :: d= +CASE v^3

The majority of null heads are simply linked with the head of their complement, the only exception being that null proforms, such as PRO in arbitrary control constructions[18] (named [pro-d] in MGbank) and the null verbal heads used for VP ellipsis ([pro-v] in MGbank), are linked to whichever head selects for them (i.e. their governor). Assuming that null proforms are the only null heads appearing at the bottom of any extended projection (ep)[19] in the corpus, this ensures that all of the lexical items inside a given supertag are part of the same ep, except for PRO, which is trivially an ep in its own right and must therefore be anchored to the verb that selects it. Note that some atomic overt heads (such as *he* and *him* in our example sentence) will not be involved in any links and will therefore form simplex supertags.

Once the merge links and unattached overt heads are extracted, the algorithm then groups them together in such a way that any lexical items which are chained together either directly or indirectly by merge links are contained in the same group. Because links are only formed between null heads and their complements (except in the case of the null proform heads), and not between heads and specifiers or adjuncts, each chain ends with the first overt head encountered, so that every (null or overt) head is guaranteed to appear in just one group and each group is guaranteed to contain at most one overt lexical item.

The above merge links would form one group, or supertag, represented compactly as follows:

[18]Other instances of control are treated as cases of A-movement following Boeckx et al. (2010).

[19]Here, we define the clausal extended projection as running from V up to the closest CP (or TP if CP is absent, as in ECM constructions), and for nominals from N up to the closest PP (or DP if PP is absent).

[decl] :: t=^{1} c
[pres] :: lv=^{2} +CASE t^1
[trans] :: v=^{3} =d lv^2
helps :: d= +CASE v^3

All of the subcategorization information of the main verb is contained within this supertag, but unlike in the case of LTAG categories, this is not always the case: if an auxiliary verb were present between little vP and TP, for instance, then only little v would be anchored to the main verb, while T and C would be anchored to the structurally higher auxiliary. C is the head triggering A'-movements, such as wh-movement and topicalization. A consequence of this is that, although like LTAG (but unlike CCG) A'-movement is lexicalised onto an overt category here, that overt category is often structurally and linearly much closer to the A'-moved element than in LTAG. For instance, in the sentence *what did she say that Pete eats for breakfast?*, an LTAG would precompile the wh-movement onto the supertag for *eats*, whereas here the [int] C head licensing this movement would be precompiled onto *did*.

As noted in Kasai et al. (2017), LTAG's lexicalisation of unbounded A'-movement is one reason why supertagging has proven more difficult to apply successfully to TAG than to CCG, Markovian supertaggers being inherently better at identifying local dependencies. We hope that lexicalising A'-movement into a supertag that is linearly closer to the moved item will therefore ultimately prove advantageous.

4.2 Adapting an existing CKY MG parser to use MG supertags

The MG supertags can be integrated into an existing CKY MG parser quite straight forwardly as follows: first, for each supertag token assigned to each word in the sentence, we map the indices that indicate which features check each other into globally unique identifiers. This is necessary to ensure that different supertags and different instances of the same supertag assigned to different words are differentiated by the system. Then, whenever one of the constrained features is encountered, the parser ensures that it is only checked against the feature with the matching identifier. The parser otherwise operates as usual except that thousands of potential merge operations are now disallowed, with the result that the search space is drastically reduced (though this of course depends on the number of supertags assigned to each word).

One complication concerns the dynamic programming of the chart. In standard CKY MG parsing, as with classical CFG CKY, items with the same category spanning the same substring are combined into a single chart entry during parsing. This prevents the system having to create identical tree fragments multiple times. But the current approach complicates this because many items now have different predetermined futures (i.e. their unchecked features are differentially constrained), and when the system later attempts to reconstruct the trees by following the backpointers, things can become very complicated. We can avoid this issue, however, simply by treating the unique identifiers that were assigned to certain selector features as part of the category. This has the effect of splitting the categories and will, for instance, prevent two single chain categories =d^1 d= v and =d^2 d= v from being treated as a single chart entry until their =d features have been checked.

4.3 Preliminary Results

An LSTM supertagger similar to that in (Lewis et al., 2016) was trained on 13,000 sentences randomly chosen from MGbank, extracting various types of (super)tag from the derivation trees. A further 742 sentences were used for development, and 753 for testing, again randomly chosen. We tried training on just the automatically generated corpus and testing on the hand-crafted trees, but this hurt 1-best performances by 2-4%, no doubt owing to the fact that this hand-crafted set deliberately contains many of the rarer constructions in the Zipfian tail which didn't make it into the automatically generated corpus[20]. With more data this effect should lessen. The results for n-best supertagging accuracies are given in table 1.

4.4 Discussion

Unsurprisingly, the accuracies improve as the number of tags decreases. The CCGbank data contains by far the least tag types and has the highest performance. However, it is worth noting that the MG supertags contain a lot more information than their CCGbank counterparts, even once A'-movement and selectional restrictions are removed. For example, MGbank encodes all predicate-argument relations directly in the syntax, distinguishing for instance between subject

[20]There are 831 category types in the automatically generated corpus from a total of 1078 for the entire treebank.

	rei	ab	rei-A’	ab-A’	ov	ccg
|tags|	3087	2087	1883	1181	717	342
1-best	79.1	81.1	83.0	84.2	88.0	92.4
2-best	88.4	90.2	91.1	91.9	95.3	97.1
3-best	91.6	93.5	94.1	94.8	97.1	98.3
10-best	96.4	97.4	97.9	98.2	99.2	99.5
25-best	97.6	98.5	98.9	99.1	99.7	99.7
40-best	98.0	98.7	99.0	99.4	99.8	99.8

Table 1: Accuracies on different MG (super)tag types showing the % of cases where the correct tag appears in the n-best list. The first row gives the number of different (super)tag types in the data; rei(fied) is supertags with all selectional properties and requirements; ab(stract) is supertags with all but 5 of these features removed[22]; -A’ indicates that null C heads, and [focalizer], [topicalizer], [wh] and [relativizer] heads were not included in the supertags, thereby delexicalising A’-movement and moving the formalism towards CCG; ov(ert) is the (reified) atomic overt tags; ccg is the ccgbank supertags.

raising and subject control verbs, and between object raising (ECM) and object control verbs, whereas CCGbank itself does not. For a fairer comparison, therefore, we would need to combine CCGbank syntactic types with the semantic types of Bos (Bos et al., 2004). There are also many types of dependencies, such as those for rightward movement and correlative focus (*either..or, neither..nor, both..and*), which could be delixicalised to reduce the size of the supertag sets further. Of course, the more null heads that are allowed freely into the chart, the stronger the statistical model of the derivation itself must be. Finally, the MGbank grammar (particularly in its reified versions) is precision-oriented, in the sense that it blocks many ungrammatical sentence types (agreement/l-selection violations, binding theory violations, (anti)*that*-trace violations, wh-island violations etc). The extra information needed to attain this precision expands the tag set but should also ultimately help in pruning the search space, enabling the parser to try more tags. The CCGbank grammar, meanwhile, is much more flexible (making it very robust), and therefore leaves a much greater proportion of the task of constraining the search space to the probability model.

The 1-best accuracies are clearly not high enough to be practical for wide-coverage MG parsing at present. By the time the 3-best supertags per word are considered, however, the accuracies are in all cases quite high, and by the 25-best they are very high, although it is difficult to say at this point what level will be sufficient for wide-coverage parsing. The overt atomic tagging is much better, achieving high accuracy by the 3-best, but these tags contain the least information and therefore leave much more disambiguation to the parsing model. Clearly, using MG supertags will require an algorithm that navigates the search space as efficiently as possible and allows the supertagger to try as many tags for each word as possible. We are in the process of re-implementing the A* search algorithm of (Lewis and Steedman, 2014), which allows their CCG parser to consider the complete distribution of 425 supertags for each word.

The potential efficiency advantages of parsing with MG supertags are considerable: reparsing the seed set of 960 trees (which includes 207 sentences which were added to cover some constructions not found in the Penn Treebank) takes over 8 hours on a 1.4GHz Intel Core i5 Macbook Air with a perfect oracle providing the 1-best overt atomic tag, but just over 6 minutes using reified supertags.

5 Conclusion

We presented two methods for constraining the parser’s search space and improving efficiency during wide-coverage MG parsing. The first extends the formalism with mechanisms for enforcing morphosyntactic agreements and selectional restrictions. The second anchors computationally costly null heads to overt heads inside complex overt categories, rendering the formalism fully compatible with Markovian supertagging techniques. Both techniques have proven useful for the generation of MGbank. We are now working on an A* MG parser which can consider the full distribution of supertags for each word and exploit the potential of these rich lexical categories.

Acknowledgments

A big thank you is due to Mark Steedman, as well as to the four anonymous reviewers for all their helpful comments and suggestions. Most especially, I would like to thank Miloš Stanojević, who coded up the LSTM supertagger, ran the experiments reported in this paper and made some very helpful suggestions regarding the supertagging method described above. This project was supported by the Engineering and Physical Sciences Research Council (EPSRC) and an ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a Google Faculty Award.

References

Mark C Baker. 1988. *Incorporation: A theory of grammatical function changing*. University of Chicago Press.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265.

Cedric Boeckx, Norbert Hornstein, and Jairo Nunes. 2010. *Control as Movement*. Cambridge University Press, Cambridge, UK.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1240. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

Noam Chomsky. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels, and Juan Uriagereka, editors, *Step by Step: Essays in Minimalist Syntax in Honor of Howard Lasnik*, pages 89–155. MIT Press, Cambridge, MA.

Noam Chomsky. 2001. Derivation by phase. *Ken Hale: A life in language*, pages 1–52.

Noam Chomsky. 2008. On phases. In Robert Freidin, Carlos Peregrin Otero, and Maria Zubizarreta, editors, *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, pages 133–166. MIT Press.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid. ACL.

B. Cramer and Y. Zhang. 2010. Constraining robust constructions for broad-coverage parsing with precision grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 223–231, Beijing.

Meaghan Fowlie and Alexander Koller. 2017. Parsing minimalist languages with interpreted regular tree grammars. In *Proceedings of the Thirteenth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+13)*, pages 11–20.

Jane Grimshaw. 1991. Extended projection. Unpublished manuscript, Brandeis University, Waltham, Mass. (Also appeared in J. Grimshaw (2005), Words and Structure, Stanford: CSLI).

Hendrik Harkema. 2001. *Parsing Minimalist Languages*. Ph.D. thesis, UCLA, Los Angeles, California.

Norbert Hornstein, Jairo Nunes, and Kleanthes Grohmann. 2005. *Understanding Minimalism*. Cambridge University Press.

Jungo Kasai, Bob Frank, Tom McCoy, Owen Rambow, and Alexis Nasr. 2017. Tag parsing with neural networks and vector representations of supertags. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Gregory M. Kobele. 2005. Features moving madly: A formal perspective on feature percolation in the minimalist program. *Research on Language and Computation*, 3(4):391–410.

Hilda Koopman and Dominique Sportiche. 1991. The position of subjects. *Lingua*, 85(2-3):211–258.

Richard Larson. 1988. On the double object construction. *Linguistic Inquiry*, 19:335–392.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Mike Lewis and Mark Steedman. 2014. A* ccg parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Ryan McDonald and Fernando Pereira. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. University of Pennsylvania.

Jens Michaelis. 1998. Derivational minimalism is mildly context–sensitive. In *International Conference on Logical Aspects of Computational Linguistics*, pages 179–198. Springer.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gomez-Rodriguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 833–841. Association for Computational Linguistics.

David Pesetsky. 1991. Zero syntax: Vol. 2: Infinitives. Unpublished MS., MIT.

Carl Pollard and Ivan Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA.

Andrew Radford. 2004. *Minimalist Syntax: Exploring the Structure of English*. Cambridge University Press.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore. ACL.

Ian G Roberts. 2010. *Agreement and head movement: Clitics, incorporation, and defective goals*, volume 59 of *Linguistic Inquiry Monograph*. MIT Press.

Yves Schabes, Anne Abeille, and Aravind K Joshi. 1988. Parsing strategies with 'lexicalized' grammars: application to tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 578–583. Association for Computational Linguistics.

Edward Stabler. 1997. Derivational minimalism. In *Logical Aspects of Computational Linguistics (LACL'96)*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95, New York. Springer.

Edward P. Stabler. 2001. Recognizing head movement. In *Logical Aspects of Computational Linguistics: 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings.*, volume 4, pages 245–260.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In Robert Borsley and Kirsti Börjars, editors, *Non-Transformational Syntax: A Guide to Current Models*, pages 181–224. Blackwell, Oxford.

John Torr. 2017. Autobank: a semi-automatic annotation tool for developing deep minimalist grammar treebanks. In *Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017*, pages 81–86.

John Torr and Edward P. Stabler. 2016. Coordination in minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the Twelfth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+12)*, pages 1–17.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2017. A dynamic window neural network for ccg supertagging. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3337–3343.

Wenduan Xu. 2016. Lstm shift-reduce ccg parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1764.

Not that much power: Linguistic alignment is influenced more by low-level linguistic features rather than social power

Yang Xu, and **Jeremy Cole** and **David Reitter**
College of Information Sciences and Technology
The Pennsylvania State University
yang.xu@psu.edu and jrcole@psu.edu and reitter@psu.edu

Abstract

Linguistic alignment between dialogue partners has been claimed to be affected by their relative social power. A common finding has been that interlocutors of higher power tend to receive more alignment than those of lower power. However, these studies overlook some low-level linguistic features that can also affect alignment, which casts doubts on these findings. This work characterizes the effect of power on alignment with logistic regression models in two datasets, finding that the effect vanishes or is reversed after controlling for low-level features such as utterance length. Thus, linguistic alignment is explained better by low-level features than by social power. We argue that a wider range of factors, especially cognitive factors, need to be taken into account for future studies on observational data when social factors of language use are in question.

1 Introduction

The effect of social power on language use in conversations has been widely studied. The Communication Accommodation Theory (Giles, 2008) states that the *social power* of speakers influence the extent to which conversation partners accommodate (or align, coordinate) their communicating styles towards them. This theory is supported by findings from qualitative studies on employment interviews (Willemyns et al., 1997), classroom talks (Jones et al., 1999), and the more recent data-driven studies on large online communities and court conversations (Danescu-Niculescu-Mizil et al., 2012; Jones et al., 2014; Noble and Fernández, 2015). In particular, Danescu-Niculescu-Mizil et al. (2012) uses a probability-based measure of linguistic alignment to demonstrate that people align more towards conversation partners of higher power, i.e., the admin users in Wikipedia talk-page, and the justices in U.S. supreme court conversations, than those of lower power, i.e., the non-admin users and the lawyers.

However, while these results find sound explanations from socio-linguistic theories, they are still somewhat surprising from the perspective of cognitive mechanisms of language production, because the mutual alignment between interlocutors of in natural dialogue can be explained by an automatic and low-level priming process (Pickering and Garrod, 2004). It is known that the strength of alignment is sensitive to low-level linguistic features (e.g., words, syntactic structures etc.), such as temporal clustering properties (Myslín and Levy, 2016), syntactic surprisal measured by prediction error (Jaeger and Snider, 2013), and lexical information density (Xu and Reitter, 2018).

Then why, or under what mechanisms, can alignment be affected by the relatively high-level social perceptions of power as reported? Could it be the case that the effect of power on alignment is actually due to the other low level features in language, such as the ones mentioned above? Is the effect of power still observable, if we control for other factors? How large is the effect? Is it significant enough to be captured by computational measures of alignment? Answering these questions will help clarify the role of social factors in linguistic alignment, and improve our understanding of language production in general.

In this study, we conduct a two-step model analysis. First, we use a basic model that has two predictors, *count* (number of a certain linguistic marker in the preceding utterance) and *power* (power status of the preceding speaker), to predict the occurrence of the same marker in the following utterance. Here, the linguistic markers are derived from 11 Linguistic Inquiry and Word Count (LIWC; Pennebaker et al., 2001) categories (e.g., *article*, *adverb*, etc.). With the basic model, the main effect of *count* characterizes the strength of alignment, and the interaction between *count* and *power* characterizes the effect of power on alignment (Section 3). Second, we use an extended model that includes a third predictor, *utterance length* (It is chosen as a typical low-level linguistic feature, discussed in Section 2.3), on top of the basic model. With the extended model, we aim to examine whether the inclusion of *utterance length* will influence the interaction between *count* and *power* (Section 4). Therefore, we can examine the extent to which the effect of power on alignment is confounded by low-level linguistic features.

To clarify, our goal is not to disprove the existence of social accommodation in dialogue. Nonetheless, it is important to distinguish between what is caused by automatic priming-based alignment and what is caused by high-level, intentional accommodation. As we will discuss, these are different processes with different predictions. Throughout this paper we use the term *alignment* to refer to the priming-based process, and accommodation to refer to the intentional process.

2 Related Work

2.1 Social power and linguistic alignment

The social factors of language use have been widely studied. Communication Accommodation Theory (Giles, 2008) posits that individuals adapt their communication styles to increase or decrease the social distance from their interlocutors. One factor that affects the adaptation of linguistic styles is *social power*. Typically, people of lower power converge their linguistic styles to those of higher power; for example, interviewees towards interviewers (Willemyns et al., 1997), or students towards teachers (Jones et al., 1999).

More recently, sensitive quantitative methods have been applied to this line of inquiry. Danescu-Niculescu-Mizil et al. (2012) computed the probability-based linguistic coordination measure among Wikipedia editors and participants of the US supreme court, and they showed that people with low power (e.g., lawyers, non-admins) exhibit greater coordination than people with high power (Justices, admins). Using the same data, Noble and Fernández (2015) found that linguistic coordination is positively correlated with social network centrality, and this effect is even greater than the effect of power status distinction.

The aforementioned studies do not include low-level language features in their analysis and thus overlook the possibility that cognitive mechanisms may be able to more readily explain the data. Importantly, as we will later discuss, these studies use a measurement of alignment that we believe is more appropriately measuring the automatic process, rather than the intentional one.

2.2 Quantifying linguistic alignment

A variety of computational measures of linguistic alignment have been developed. Some quantify the increase in conditional probability of certain elements (words or word types) given that they have appeared earlier (Church, 2000; Danescu-Niculescu-Mizil et al., 2012). Some compute the proportion of repeated lexical entries or syntactic rules between two pieces of text (Fusaroli et al., 2012; Wang et al., 2014; Xu and Reitter, 2015). Some use the coefficients returned by generalized linear models (McCullagh, 1984; Breslow and Clayton, 1993; Lindstrom and Bates, 1990) as an index of alignment (Reitter and Moore, 2014). A large body of the existing computational measures intensively use LIWC (Pennebaker et al., 2001) to construct representations of language users' styles, which can be used to measure alignment with distance-like metrics (Niederhoffer and Pennebaker, 2002; Jones et al., 2014). Many of these approaches do not distinguish between different levels of linguistic analysis and different psycholinguistic processes (phonological, lexical, syntactic, etc.), and neither do we. Alignment is consistently present across these levels and processes, although it is not as clear in naturalistic language as it is in the constrained utterances of experiments, particularly at the syntactic level (Healey et al., 2014). We are concerned with the question of whether alignment is a socially linked, intentional adaptation process, as opposed to addressing any particular cognitive model.

More recently, Doyle et al. (2016) pointed out that most existing measures are difficult to compare, and emphasized the need for a universal measure. The Hierarchical Alignment Model (HAM; Doyle et al., 2016) and Word-Based HAM (WHAM; Doyle and Frank, 2016) use statistical inference techniques, which out-perform other measures in terms of robustness of capturing linguistic alignment in social media conversations.

In this study, we choose to use generalized linear models to quantify linguistic alignment, avoiding issues with more complex, and less inspectable models. For instance, the commonly used probability based methods and their more advanced variants (HAM and WHAM) lack the flexibility to jointly examine multiple factors (e.g., speaker groups, utterance length etc.) that influence alignment. Another issue is that they do not take into account the number of occurrences of linguistic markers, which is known to affect alignment (see Section 2.3). Conversely, though linear models do not give an accurate per-speaker estimate of alignment (which we do not need for the purpose of this study), they do provide the ability to examine multiple factors that influence alignment by simply including multiple predictors in the model. As should be clear, a generalized linear model also already takes into account baseline usage with a fitted intercept. Given these considerations, we use generalized linear models for quantitative analysis. The formulation of our models is described in Sections 3.2 and 4.1.

2.3 Cognitive constraints on linguistic alignment: why utterance length matters

There are many, at times competing, cognitive explanations of linguistic alignment in both comprehension and production. Jaeger and Snider (2013) explained alignment as a consequence of expectation adaptation, and they found that stronger alignment is associated with syntactic structures that have higher surprisal (roughly speaking, less common). Alignment in language production can also be modeled as a general memory phenomenon (Reitter et al., 2011), which explains a number of known interaction effects. Myslín and Levy (2016) found that sentence comprehension is faster when the same syntactic structure clusters in time in prior experience than when it is evenly spaced in time. Myslín and Levy (2016) cast comprehension priming as the rational expectation for repetition of stimuli. Though this result is not directly related to comprehension-to-production priming, it makes sense to anticipate that production could also be sensitive to the clustering patterns of linguistic elements, because comprehension and production are closely coupled processes (Pickering and Garrod, 2007).

Utterance length, i.e., the number of words in utterance, is a feature that closely relates to both surprisal and clustering properties. Longer utterances tend to have higher syntactic surprisal (Xu and Reitter, 2016a), and it is reasonable to assume they tend to contain more evenly distributed stimuli. Thus, utterance length is a low-level linguistic feature that correlates with many of the causes of alignment. In this way, we use utterance length as a stand-in for low-level linguistic features as a whole when comparing it with social power, a much higher-level feature. Examining alignment (in social science research and elsewhere) therefore calls for controlling sentence length.

3 Experiment 1: Basic model

In Experiment 1, we justify the practice of using generalized linear models to quantify linguistic alignment. We compare two ways of characterizing the occurrence of LIWC-derived markers in a preceding utterance, binary presence and numeric count, to determine which results in better model. We use an interaction term in the model to quantify the effect of the power status of speakers on linguistic alignment, which serves as the basis for the following sections.

3.1 Corpus data

We use two datasets compiled by Danescu-Niculescu-Mizil et al. (2012): Wikipedia talk-page corpus (*Wiki*) and a corpus of United States supreme court conversations (*SC*). Wiki is a collection of conversations from Wikipedia editor's talk Pages[1], which contains 125,292 conversations contributed by 30,732 editors. SC is a collection of conversations from the U.S. Supreme Court Oral Arguments[2], with 51,498 utterances making up 50,389 conversational exchanges, from 204 cases involving 11 Justices and 311 other participants (lawyers or amici curiae).

A conversation consists of a sequence of utterances, $\{u_i\}(i = 1, 2, \ldots, N)$, where N is the total number of utterances in the conversation. Because people take turns to talk in conversation, u_i and u_{i+1} are always from different speakers. Since our interest here is the alignment between different speakers (as opposed to within the same speaker), we use a sliding window of size 2 to go through

[1]http://en.wikipedia.org/wiki/Wikipedia:Talk_page_guidelines

[2]http://www.supremecourt.gov/oral_arguments/

the whole conversation, generating a sequence of adjacent utterance pairs, $\{\langle prime_i, target_i\rangle\}(i = 1, 2 \ldots N - 1)$.

Next, we process each utterance u_i by counting the number of occurrences of 14 linguistic markers that are derived from LIWC categories, resulting in 14 counts for each utterance. These 14 linguistic markers are: high frequency adverbs (*adv*), articles (*art*), auxiliary verbs (*auxv*), certainty (*certain*), conjunctions (*conj*), discrepancy (*discrep*), exclusion (*excl*), inclusion (*incl*), impersonal pronouns (*ipron*), negations (*negate*), personal pronouns (*ppron*), prepositions (*prep*), quantifiers (*quant*), and tentativeness (*tentat*). These fourteen markers come from taking the union of the 8 markers used by Danescu-Niculescu-Mizil et al. (2012) and the 11 markers used by Doyle and Frank (2016), which are the main studies we wanted to compare with.

3.2 Statistical models

We formulate alignment as the impact of using certain linguistic elements in the preceding utterance on their chance to appear again in the following utterance. In the language of generalized linear models, we use the occurrence of linguistic markers in *target* as the response variable and the predictor is their occurrence in *prime*. These occurrences can be represented as either a boolean or a count. Thus alignment is characterized by the β coefficient of the predictor, which allows the model to distinguish the prevalence of *Occurrence* or another feature in primed situations as compared to its prior in the corpus. Factors that may influence alignment (e.g., social power) can then be examined by adding a corresponding interaction term to the model.

Our first step, then, is to replicate the previous studies' findings of the effect of social power on alignment. Two models were fitted, predicting the *presence* of the linguistic marker *m* in *target* utterance over its absence. We fit models both corresponding to a binary predictor (C_{presence}) and a count-based one (C_{count}). Both models include a second binary predictor, C_{power}, indicating the power status of the *prime* speaker (*high* vs. *low*), and its interaction with C_{presence} and C_{count}, respectively. Additionally, random intercepts on linguistic marker and *target* speaker are fitted, based on the consideration that individuals might have different levels of alignment towards different markers. C_{count} is log-transformed to maximize model fit according to Bayesian Information Criterion; this is commensurate with standard psycholinguistic practice and known cumulative priming and memory effects. Equation (1) shows the count-based model. To reiterate, the interaction term $C_{\text{count}} * C_{\text{power}}$ characterizes the effect of power on alignment.

$$\begin{aligned}\text{logit}(m) &= \ln \frac{p(m \text{ in } \textit{target})}{p(m \text{ not in } \textit{target})} \\ &= \beta_0 + \beta_1 C_{\text{count}} + \beta_2 C_{\text{power}} \\ &+ \beta_3 \mathbf{C_{count}} * \mathbf{C_{power}}\end{aligned} \tag{1}$$

3.3 Model coefficients

The main effects of C_{presence} and C_{count} are significant ($p < 0.001$) and positive in both corpora (SC: $\beta_{\text{presence}} = 0.439$, $\beta_{\text{count}} = 0.291$; Wiki: $\beta_{\text{presence}} = 0.440$, $\beta_{\text{count}} = 0.395$), which captures the linguistic alignment from *prime* to *target*. However, there is difference in how alignment is influenced by power between the two corpora: In SC, $C_{\text{count}} * C_{\text{power}}$ is significant ($\beta = 0.078$, $p < 0.001$), but $C_{\text{presence}} * C_{\text{power}}$ is non-significant; In Wiki, on the contrary, $C_{\text{presence}} * C_{\text{power}}$ is marginally significant ($\beta = 0.014$, $p = .055$), but $C_{\text{count}} * C_{\text{power}}$ in not significant. No collinearity is found between C_{count} (or C_{presence}) and C_{power} (Pearson correlation $r < 0.2$).

To explore why using C_{presence} vs. C_{count} results in different significance levels for SC and Wiki, we fit a individual linear model for each linguistic marker, using 14 *disjoint* subsets of each corpus. We present the z scores and significance levels of the two interaction terms are reported in Table 1. First, in SC the interaction term $C_{\text{presence}} * C_{\text{power}}$ is significant for 9 out of 14 markers. In Wiki, $C_{\text{count}} * C_{\text{power}}$ is significant for 5 out of 14 markers. This suggests that the interaction between the occurrence of linguistic markers and the power status of speakers exists within a subset of the linguistic categories, but not across all of them. Thus, we consider this first experiment a replication of past findings of the effect of social power on alignment: social power has a significant effect across certain markers, but its overall effect is neutralized in the full model since some markers at not significant. This analysis also revealed that $C_{\text{count}} * C_{\text{power}}$ is more reliable in capturing this effect, which is what we will use in the following experiment.

Table 1: Summary of the 14 models that fit individual markers on disjoint data subsets. Wald's z-score and significance level (*** for $p < 0.001$, ** for $p < 0.01$, * for $p < 0.05$, and † for $0.05 < p < 0.1$) of the interaction terms ($C_{\text{presence}} * C_{\text{power}}$ or $C_{\text{count}} * C_{\text{power}}$) are reported.

Marker	z score			
	$C_{\text{presence}} * C_{\text{power}}$		$C_{\text{count}} * C_{\text{power}}$	
	SC	Wiki	SC	Wiki
adv	1.19	-0.33	**6.16*****	-0.40
art	**1.99***	0.36	**4.60*****	1.27
auxv	**3.72*****	-0.62	**5.81*****	-0.83
certain	-0.02	**3.19****	**1.94**†	**2.84****
conj	0.54	-0.20	**6.79*****	0.39
discrep	**5.44*****	-0.05	**8.03*****	0.25
excl	-0.53	**1.96***	**2.94****	**2.16***
incl	**2.86****	0.80	**5.24*****	**2.15***
ipron	**6.84*****	**1.70**†	**10.22*****	**1.90**†
negate	**2.83****	**3.14****	**5.49*****	**3.11****
ppron	**2.74****	**-1.86**†	1.29	-1.13
prep	**4.76*****	**2.37***	**6.87*****	-0.19
quant	0.89	1.01	**4.14*****	-0.04
tentat	**3.69*****	0.17	**4.52*****	-0.78

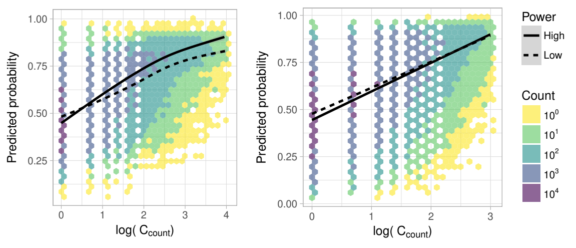
3.4 Visualizing the effect of power

To better understand the interaction term $C_{\text{count}} * C_{\text{power}}$, we divide the data into two groups by whether C_{power} is *high* or *low*, and fit a model on each of the groups. In the models we include only one predictor C_{count} (see Equation (2)). Then we compare the main effects (β_1 coefficients) from the two groups.

$$\text{logit}(m) = \beta_0 + \beta_1 C_{\text{count}} \tag{2}$$

Unsurprisingly, the main effects of C_{count} are significant for both groups ($p < 0.001$). But more importantly, the β_1 coefficients of the high power group are larger than those of the low power group. For SC, the difference is very salient: $\beta_1^{\text{high}} = 0.416$ ($SE = 0.006$), $\beta_1^{\text{low}} = 0.272$ ($SE = 0.005$). For Wiki, the difference is smaller: $\beta_1^{\text{high}} = 0.424$ ($SE = 0.007$), $\beta_1^{\text{low}} = 0.386$ ($SE = 0.005$). This is in line with the non-significant coefficient of $C_{\text{count}} * C_{\text{power}}$ in Wiki. In fact, the models of Wiki are fitted on a subset of data that contain the 5 (out of 14) markers that have significant coefficients of $C_{\text{count}} * C_{\text{power}}$ in the individual models shown in Table 1 (*certain*, *excl*, *incl*, *ipron*, *negate*), so that the difference in slopes is presented at maximal degree.

In Figure 1 we illustrate the β_{high} and β_{low} coefficients of C_{count} by plotting the predicted probability (the reversed logit transformation of the left-hand side term of Equation (2)) against C_{count} (log-transformed). It is obvious that the slope of β_{high} is larger than that of β_{low} (more salient in SC), indicating the significant interaction between C_{count} and C_{power}.



(a) Supreme Court (b) Wikipedia

Figure 1: The predicted probability of marker appearing in *target* (the reverse logit transform of the left hand side of Equation (2)) against the number of markers in *prime*, i.e., C_{count} (log-transformed), grouped by the power of *prime* speaker, i.e., *high* vs. *low*. Divergent slopes indicate significant interactions. Colored hexagons indicate the number of data points within that region.

3.5 Discussion

The occurrence of linguistic markers in *prime* is a strong predictor of whether the same marker will appear again in *target*. The coefficients of C_{count} can be viewed as indicators of the linguistic alignment between interlocutors: larger positive βs indicate stronger alignment, while smaller or even negative βs indicate weaker and reverse alignment, respectively (not found in our data).

Our results confirm the previously reported effect of power on linguistic alignment. The significant β' coefficient of $C_{\text{count}} * C_{\text{power}}$ means that the β of C_{count} is dependent on C_{power}. In other words, the strength of alignment varies significantly depending on different power levels (i.e., *high* vs. *low*) of the *prime* speaker (reflected by the different slopes in Figure 1). However, we need to keep in mind that this affirmative finding is *not* safe, because it based on a simple model that has only one key predictor, C_{power}. According to our hypothesis, the strength of alignment can be influenced by a lot of low-level linguistic features, and we are not sure yet if the effect of power will still be visible after we includes more predictors representing those features. This will be the next step experiment.

Additionally, the results also suggest that the influence of power on linguistic alignment is better characterized by the more fine-grained cumulative effect of linguistic markers than when it is simply explained by the mere difference between their *absence* or *presence*. Thus, we will discard C_{presence} and proceed with C_{count}.

4 Experiment 2: Extended model

In our first experiment, we replicated the effect of *prime* speakers' power status on the linguistic alignment from *target* speakers, from the significant interaction term $C_{\text{count}} * C_{\text{power}}$. Now, we want to determine if the effect of power remains significant after taking into account utterance length. As discussed, our hypothesis is that alignment (as measured by changes in probability of using LIWC categories) is best explained by low-level linguistic features that would be taken into account by an automatic priming process.

4.1 Statistical models

We add a new predictor to Equation (1), C_{pLen}, which is the number of words in *prime*, resulting in an extended model shown in Equation (3). We are interested to see if β_4 remains significant when the other interaction terms (with corresponding coefficients β_5, β_6 and β_7) are added.

$$\begin{aligned}\text{logit}(m) &= \ln \frac{p(m \text{ in } \textit{target})}{p(m \text{ not in } \textit{target})} \\ &= \beta_0 + \beta_1 C_{\text{count}} + \beta_2 C_{\text{power}} + \beta_3 C_{\text{pLen}} \\ &+ \boldsymbol{\beta_4 C_{\textbf{count}} * C_{\textbf{power}}} \\ &+ \boldsymbol{\beta_5 C_{\textbf{count}} * C_{\textbf{pLen}}} \\ &+ \boldsymbol{\beta_6 C_{\textbf{power}} * C_{\textbf{pLen}}} \\ &+ \boldsymbol{\beta_7 C_{\textbf{count}} * C_{\textbf{power}} * C_{\textbf{pLen}}}\end{aligned} \tag{3}$$

Note that we used the same subset of Wiki as used in Section 3.4 (using the five most significant LIWC categories), so that the strongest effect of $C_{\text{count}} * C_{\text{power}}$ is considered.

4.2 Model coefficients

The coefficients of the full model are in Table 2. Surprisingly, the coefficient of $C_{\text{count}} * C_{\text{power}}$ is significantly *negative* in SC, and non-significant in Wiki (see highlighted rows), which are in contrast to the *positive* coefficients of the same term in Table 1. It indicates that the observed effect of power on alignment depends on the presence of C_{pLen} in the model. No collinearity is found between C_{power} and other predictors: Pearson correlation $r < 0.2$; Variance inflation factor (VIF) is low (< 2.0) (O'brien, 2007).

Table 2: Summary of the model described in Equation (3): β coefficients, Wald's z-score and significance level (*** for $p < 0.001$, ** for $p < 0.01$, * for $p < 0.05$) for all predictors and interactions.

Corpus	Predictor	β	z
SC	Intercept	0.360	2.40*
	C_{count}	0.213	26.92***
	C_{power}	-0.060	-3.39***
	C_{pLen}	0.080	13.03***
	$\boldsymbol{C_{\textbf{count}} * C_{\textbf{power}}}$	**-0.103**	**-9.95*** **
	$C_{\text{count}} * C_{\text{pLen}}$	-0.066	-15.35***
	$C_{\text{power}} * C_{\text{pLen}}$	0.231	25.25***
	$C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$	0.036	4.79***
Wiki	Intercept	0.330	1.40
	C_{count}	0.149	31.11***
	C_{power}	-0.074	-10.52***
	C_{pLen}	0.179	40.80***
	$\boldsymbol{C_{\textbf{count}} * C_{\textbf{power}}}$	**0.001**	**0.14**
	$C_{\text{count}} * C_{\text{pLen}}$	0.022	6.13***
	$C_{\text{power}} * C_{\text{pLen}}$	0.042	5.52***
	$C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$	-0.010	-1.61

To further demonstrate how the coefficient of $C_{\text{power}} * C_{\text{count}}$ is dependent on C_{pLen}, we remove $C_{\text{count}} * C_{\text{pLen}}$, $C_{\text{power}} * C_{\text{pLen}}$ and $C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$ from Equation (3) stepwisely, and examine $C_{\text{count}} * C_{\text{power}}$ in the corresponding remaining models. z-scores, significance levels, and the Akaike information criterion (AIC) score (Akaike, 1998) of the remainder models are reported in Table 3. In the full model, and when $C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$ or $C_{\text{count}} * C_{\text{pLen}}$ is removed from the model, the coefficients of $C_{\text{power}} * C_{\text{count}}$ are significantly negative in SC and non-significant in Wiki. Only when $C_{\text{power}} * C_{\text{pLen}}$ is removed, the coefficients of $C_{\text{count}} * C_{\text{power}}$ become significantly positive (the last two rows in Table 3). However, the models that have negative or non-significant coefficient for $C_{\text{power}} * C_{\text{count}}$ have lower AIC scores than those that have positive coefficient (The full model has the lowest AIC score), which indicates that the former ones have higher quality. Altogether, the stepwise analysis not only indicates that the positive interaction between C_{power} and C_{count} shown in our basic model (Section 3) is unreliable, but also suggests that a negative interaction (SC) or non-significant interaction is more preferable.

4.3 Visualizing interaction effect

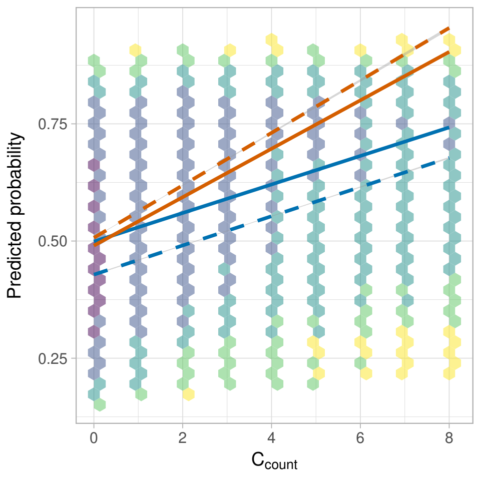
To illustrate how the interaction $C_{\text{power}} * C_{\text{count}}$ diminishes after adding C_{pLen} into the extended model, we cluster different ranges of C_{pLen} and determine how the amount of priming changes with C_{count} w.r.t. different combinations of C_{power} and C_{pLen}. This is a common practice to interpret linear models with three-way interactions (Houslay, 2014).

To cluster, we first compute the mean of C_{pLen} (i.e., the average utterance length), M_{pLen}. Then we divide the data by whether C_{pLen} is above or below M_{pLen}. Then we compute the mean of C_{pLen} for the upper and lower parts of data, resulting in M^L_{pLen} and M^S_{pLen} respectively (L for long and S for short). Now, we can replace the continuous variable C_{pLen} to a categorical and ordinal one that has two values, $\{M^S_{\text{pLen}}, M^L_{\text{pLen}}\}$, which represent the length of relatively short and long utterances respectively. Together with the other categorical variable, C_{power}, which has two values, *high* and *low*, we have four combinations: $C_{\text{pLen}} = M^S_{\text{pLen}}$ and $C_{\text{power}} = high$ (SH), $C_{\text{pLen}} = M^L_{\text{pLen}}$ and $C_{\text{power}} = high$ (LH), $C_{\text{pLen}} = M^S_{\text{pLen}}$ and $C_{\text{power}} = low$ (SL), $C_{\text{pLen}} = M^L_{\text{pLen}}$ and $C_{\text{power}} = low$ (LL). In Figure 2 we plot the smoothed regression lines of predicted probability against C_{count}, w.r.t. the above four groups of C_{pLen} and C_{power} combinations. Here C_{count} is not log-transformed, because it better demonstrates the trend of the fitted regression lines.

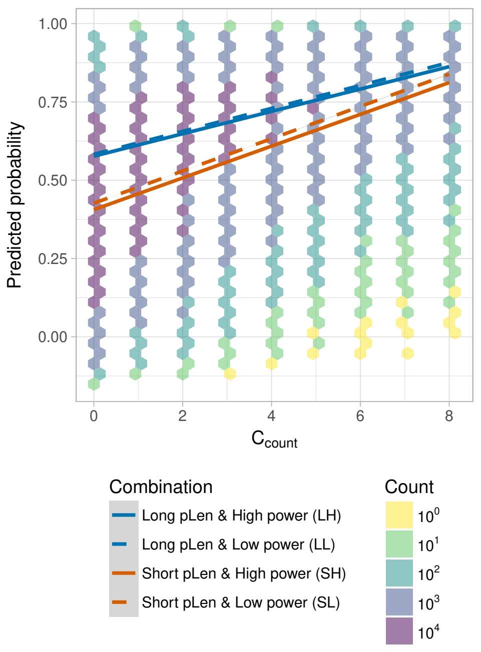
Figure 2 intuitively shows that C_{pLen} is a more determinant predictor than C_{power}. Division by power, i.e., *high* (SH and LH groups) vs. *low* (SL and LL groups), does not result in a salient difference in slopes, as it can be seen that the slopes of *high* (solid) and *low* (dashed) power lines do not differ much from each other within the same *prime* utterance length group (indicated by color). However, division by *prime* utterance length, i.e. *short* (SH and SL) vs. *long* (LH and LL), results in very significant differences in slopes: in Figure 2a, *short* C_{pLen} group (orange) has larger slopes than *long* C_{pLen} group (blue), while in Figure 2b, *short* group has smaller slopes than *long* group.

4.4 Discussion

Adding C_{pLen} to the model has strong impact on the previous conclusion about the effect of power



(a) Supreme Court



(b) Wikipedia

Figure 2: The predicted probability of marker appearing in *target* against C_{count}, grouped by the four combinations of C_{pLen} (*long* vs. *short*, indicated by color) and C_{power} (*high* vs. *low*, indicated by line type): LH, LL, SH, and SL. Colored hexagon indicates the number of data points.

on alignment. First of all, we find a negative interaction between C_{count} and C_{power} in SC and a non-significant effect in Wiki, which is contrary to the previous findings reported by Danescu-Niculescu-Mizil et al. (2012). Moreover, we doubt the reliability of a positive interaction because the valence of its β varies when other interaction terms (associated with C_{pLen}) are removed or added, and a negative or non-significant interaction is preferred

Table 3: Wald's z-score and significance level (*** for $p < 0.001$) of the $C_{\text{count}} * C_{\text{power}}$ term, and the AIC scores of the remainder models after removing other interaction terms from the full model stepwisely. The full model is described in Equation (3).

Remainder model	SC		Wiki	
	z-score	AIC	z-score	AIC
Full	-9.95***	697588	0.14	890685
Full $- C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$	-8.75***	697609	-0.62	890686
Full $- C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$ $- C_{\text{count}} * C_{\text{pLen}}$	-5.61***	697838.9	-0.74	890723.5
Full $- C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$ $- C_{\text{power}} * C_{\text{pLen}}$	10.90***	698254.7	3.85***	890726.7
Full $- C_{\text{count}} * C_{\text{power}} * C_{\text{pLen}}$ $- C_{\text{count}} * C_{\text{pLen}}$ $- C_{\text{power}} * C_{\text{pLen}}$	15.02***	698461.8	3.72***	890763.8

by a simple model selection criterion.

Second, there is a significant interaction between C_{count} and C_{pLen}, though it is in different directions for the two corpora: negative β in SC and positive β in Wiki. Both observations have some theoretical justification from previous studies. Myslín and Levy (2016)'s work is in favor of the negative β: language comprehension is facilitated by the clustering of linguistic stimuli in time. In our case, the linguistic markers in the utterance of speaker A function as stimuli to speaker B. A longer utterance means that all the stimuli span wider in time, and thus demonstrate less clustering, which make the stimuli less salient features for speaker B to adapt to. This in turn causes speaker B to be less likely reuse those stimuli in the near future. Meanwhile, evidence from the line of works on surprisal and syntactic priming supports the positive β. In syntactic alignment, structures with higher surprisal (less common) are associated with stronger alignment (Jaeger and Snider, 2013; Reitter and Moore, 2014). Since surprisal has been found to be closely related with utterance length in dialogue (Genzel and Charniak, 2003; Xu and Reitter, 2016b,a), it is reasonable to expect that longer utterances receive stronger alignment because they contain content of higher surprisal.

The discrepancy between Wiki and SC in terms of the direction of $C_{\text{count}} * C_{\text{pLen}}$ is an interesting phenomenon to explore, because it can tell us something about how the form of dialogue (Wiki consists of online conversations and SC consists of face-to-face ones) affects the underlying cognitive mechanism of language production.

Regardless, our main finding is that low-level linguistic features, such as utterance length, have a strong effect on linguistic alignment. These effects are an important confound to take into account when examining higher-level features, such as social power. In particular, the effect of social power cannot be reliably detected by linear models once introducing utterance length.

Another interesting piece of result is the significant interaction term $C_{\text{power}} * C_{\text{pLen}}$, which implies that the power status of speaker and how long he/she tends to speak are not totally unrelated. Significant but weak correlation are found between C_{power} and C_{pLen} (using Pearson's correlation score): $r = -0.059$ in SC; $r = -0.018$ in Wiki. This correlation may show some kind of a linguistic manifestation of social power, but since it is not directly related to the alignment process, we do not further discuss it in this paper.

In summary of the results, we conjecture that the previously reported effect of power (Danescu-Niculescu-Mizil et al., 2012) is likely to be caused by the correlation between power status and utterance length, though further investigation is needed to confirm this. Moreover, utterance length is just one simple factor, and there are many more other linguistic features that can correlate with social power: e.g., the surprisal based measure of lexical information etc.

5 Conclusion

To sum up, our findings suggest that the previously reported effect of power on linguistic alignment is not reliable. Instead, we consistently align towards language that shares certain low-level features. We call for the inclusion of a wider range of factors in future studies of social influences on language use, especially low-level but interpretable cognitive factors. Perhaps in most scenarios, alignment is primarily influenced by linguistic features themselves, rather than social power.

We are not denying the existence of accommodation caused by the social distance between interlocutors. However, we want to stress the difference between the priming-induced alignment at lower linguistic levels and the intentional accommodation that is caused by higher-level perception of social power. The latter should be a relatively stable effect that is independent on the low-level linguistic features. In particular, our findings suggest that the probability change of LIWC categories is more likely to be a case of automatic alignment, rather than an intentional accommodation, because it is better explained by lower-level linguistic features (utterance length). Therefore, we suggest that future work on social power and language use should consider other (maybe higher-level) linguistic elements.

Acknowledgement

We sincerely thank Lizhao Ge for her useful advice on the statistical methods used. The work leading to this paper was funded by the National Science Foundation (IIS-1459300 and BCS-1457992).

References

Hirotogu Akaike. 1998. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer.

Norman E Breslow and David G Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88(421):9–25.

Kenneth W Church. 2000. Empirical estimates of adaptation: the chance of two Noriega's is closer to $p/2$ than p^2. In *Proceedings of the 18th Conference on Computational Linguistics*, volume 1, pages 180–186, Saarbrücken, Germany.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st International Conference on World Wide Web*, pages 699–708, Lyon, France.

Gabriel Doyle and Michael C Frank. 2016. Investigating the sources of linguistic alignment in conversation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 526–536, Berlin, Germany.

Gabriel Doyle, Dan Yurovsky, and Michael C Frank. 2016. A robust framework for estimating linguistic alignment in Twitter conversations. In *Proceedings of the 25th International Conference on World Wide Web*, pages 637–648, Montreal, Canada.

Riccardo Fusaroli, Bahador Bahrami, Karsten Olsen, Andreas Roepstorff, Geraint Rees, Chris Frith, and Kristian Tylén. 2012. Coming to terms quantifying the benefits of linguistic coordination. *Psychological Science*, 23(8):931–939.

Dmitriy Genzel and Eugene Charniak. 2003. Variation of entropy and parse trees of sentences as a function of the sentence number. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 65–72. Association for Computational Linguistics.

Howard Giles. 2008. Communication accommodation theory. In L. A. Baxter and D. O. Braithewaite, editors, *Engaging theories in interpersonal communication: Multiple perspectives*, pages 161–173. Sage, Thousand Oaks, CA.

Patrick G. T. Healey, Matthew Purver, and Christine Howes. 2014. Divergence in dialogue. *PLOS ONE*, 9(6):1–6.

Thomas M. Houslay. 2014. Understanding 3-way interactions between continuous variables. `https://tomhouslay.com/2014/03/21/understanding-3-way-interactions-between-continuous-variables/`.

T Florian Jaeger and Neal E Snider. 2013. Alignment as a consequence of expectation adaptation: Syntactic priming is affected by the prime's prediction error given both prior and recent experience. *Cognition*, 127(1):57–83.

Elizabeth Jones, Cynthia Gallois, Victor Callan, and Michelle Barker. 1999. Strategies of accommodation: Development of a coding system for conversational interaction. *Journal of Language and Social Psychology*, 18(2):123–151.

Simon Jones, Rachel Cotterill, Nigel Dewdney, Kate Muir, and Adam Joinson. 2014. Finding zelig in text: A measure for normalizing linguistic accommodation. In *25th International Conference on Computational Linguistics*, Bath, UK.

Mary J Lindstrom and Douglas M Bates. 1990. Nonlinear mixed effects models for repeated measures data. *Biometrics*, pages 673–687.

Peter McCullagh. 1984. Generalized linear models. *European Journal of Operational Research*, 16(3):285–292.

Mark Myslín and Roger Levy. 2016. Comprehension priming as rational expectation for repetition: Evidence from syntactic processing. *Cognition*, 147:29–56.

Kate G Niederhoffer and James W Pennebaker. 2002. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21(4):337–360.

Bill Noble and Raquel Fernández. 2015. Centre stage: How social network position shapes linguistic coordination. In *Proceedings of CMCL 2015*, pages 29–38, Denver, CO.

Robert M O'brien. 2007. A caution regarding rules of thumb for variance inflation factors. *Quality & Quantity*, 41(5):673–690.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

Martin J Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(02):169–190.

Martin J Pickering and Simon Garrod. 2007. Do people use language production to make predictions during comprehension? *Trends in cognitive sciences*, 11(3):105–110.

David Reitter, Frank Keller, and Johanna D Moore. 2011. A computational cognitive model of syntactic priming. *Cognitive Science*, 35(4):587–637.

David Reitter and Johanna D Moore. 2014. Alignment and task success in spoken dialogue. *Journal of Memory and Language*, 76:29–46.

Yafei Wang, David Reitter, and John Yen. 2014. Linguistic adaptation in conversation threads: Analyzing alignment in online health communities. In *Proceedings of Cognitive Modeling and Computational Linguistics. Workshop at the Annual Meeting of the Association for Computational Linguistics*.

Michael Willemyns, Cynthia Gallois, Victor Callan, and J Pittam. 1997. Accent accommodation in the employment interview. *Journal of Language and Social Psychology*, 15(1):3–22.

Yang Xu and David Reitter. 2015. An evaluation and comparison of linguistic alignment measures. In *Proceedings of Cognitive Modeling and Computational Linguistics (CMCL)*, pages 58–67, Denver, DO.

Yang Xu and David Reitter. 2016a. Convergence of syntactic complexity in conversation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 443–448, Berlin, Germany.

Yang Xu and David Reitter. 2016b. Entropy converges between dialogue participants: Explanations from an information-theoretic perspective. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 537–546, Berlin, Germany.

Yang Xu and David Reitter. 2018. Information density converges in dialogue: Towards an information-theoretic model. *Cognition*, 170:147–163.

TutorialBank: A Manually-Collected Corpus for Prerequisite Chains, Survey Extraction and Resource Recommendation

Alexander R. Fabbri **Irene Li** **Prawat Trairatvorakul** **Yijiao He**
Wei Tai Ting **Robert Tung** **Caitlin Westerfield** **Dragomir R. Radev**

Department of Computer Science, Yale University
{alexander.fabbri,irene.li,prawat.trairatvorakul,yijiao.he,
robert.tung,weitai.ting,caitlin.westerfield,dragomir.radev}@yale.edu

Abstract

The field of Natural Language Processing (NLP) is growing rapidly, with new research published daily along with an abundance of tutorials, codebases and other online resources. In order to learn this dynamic field or stay up-to-date on the latest research, students as well as educators and researchers must constantly sift through multiple sources to find valuable, relevant information. To address this situation, we introduce TutorialBank, a new, publicly available dataset which aims to facilitate NLP education and research. We have manually collected and categorized over 6,300 resources on NLP as well as the related fields of Artificial Intelligence (AI), Machine Learning (ML) and Information Retrieval (IR). Our dataset is notably the largest manually-picked corpus of resources intended for NLP education which does not include only academic papers. Additionally, we have created both a search engine [1] and a command-line tool for the resources and have annotated the corpus to include lists of research topics, relevant resources for each topic, prerequisite relations among topics, relevant subparts of individual resources, among other annotations. We are releasing the dataset and present several avenues for further research.

1 Introduction

NLP has seen rapid growth over recent years. A Google search of "Natural Language Processing" returns over 100 million hits with papers, tutorials, blog posts, codebases and other related online resources. Additionally, advances in related fields such as Artificial Intelligence and Deep Learning are strongly influencing current NLP research. With these developments, an increasing number of tutorials and online references are being published daily. As a result, the task of students, educators and researchers of tracking the changing landscape in this field has become increasingly difficult.

Recent work has studied the educational aspect of mining text for presenting scientific topics. One goal has been to develop concept maps of topics, graphs showing which topics are prerequisites for learning a given topic (Gordon et al., 2016; Liu et al., 2016; Pan et al., 2017a,b; Liang et al., 2017). Another goal has been to automatically create reading lists for a subject either by building upon concept graphs (Gordon et al., 2017) or through an unstructured approach (Jardine, 2014).

Additionally, other work has aimed to automatically summarize scientific topics, either by extractively summarizing academic papers (Jha et al., 2013, 2015; Jaidka et al., 2016) or by producing Wikipedia articles on these topics from multiple sources (Sauper and Barzilay, 2009; Liu et al., 2018). Scientific articles constitute primary texts which describe an author's work on a particular subject, while Wikipedia articles can be viewed as tertiary sources which summarize both results from primary works as well as explanations from secondary sources. Tang and McCalla (2004, 2009) and Sheng et al. (2017) explore the pedagogical function among the types of sources.

To address the problem of the scientific education of NLP more directly, we focus on the annotation and utilization of secondary sources presented in a manner immediately useful to the NLP community. We introduce the TutorialBank corpus, a manually-collected dataset of links to over

[1] `http://aan.how`

6,300 high-quality resources on NLP and related fields. The corpus's magnitude, manual collection and focus on annotation for education in addition to research differentiates it from other corpora. Throughout this paper we use the general term "resource" to describe any tutorial, research survey, blog post, codebase or other online source with a focus on educating on a particular subject. We have created a search engine for these resources and have annotated them according to a taxonomy to facilitate their sharing. Additionally, we have annotated for pedagogical role, prerequisite relations and relevance of resources to hand-selected topics and provide a command-line interface for our annotations.

Our main contribution is the manual collection of good quality resources related to NLP and the annotation and presentation of these resources in a manner conducive to NLP education. Additionally, we show initial work on topic modeling and resource recommendation. We present a variant of standard reading-list generation which recommends resources based on a title and abstract pair and demonstrate additional uses and research directions for the corpus.

2 Related Work

2.1 Pedagogical Value of Resources

Online resources are found in formats which vary in their roles in education. Sheng et al. (2017) identify seven types of pedagogical roles found in technical works: Tutorial, Survey, Software Manual, Resource, Reference Work, Empirical Results, and Other. They annotate a dataset of over 1,000 resources according to these types. Beyond these types, resources differ in their pedagogical value, which they define as "the estimate of how useful a document is to an individual who seeks to learn about specific concepts described in the document". Tang and McCalla (2004, 2009) discuss the pedagogical value of a single type, academic papers, in relation to a larger recommendation system.

2.2 Prerequisite Chains

Prerequisite chains refer to edges in a graph describing which topics are dependent on the knowledge of another topic. Prerequisite chains play an important role in curriculum planning and reading list generation. Liu et al. (2016) propose "Concept Graph Learning" in order to induce a graph from which they can predict prerequisite relations among university courses. Their framework consists of two graphs: (1) a higher-level graph which consists of university courses and (2) a lower-level graph which consists of induced concepts and pair-wise sequential preferences in learning or teaching the concept.

Liang et al. (2017) experiment with prerequisite chains on education data but focus on the recovery of a concept graph rather than on predicting unseen course relations as in Liu et al. (2016). They introduce both a synthetic dataset as well as one scraped from 11 universities which includes course prerequisites as well as concept-prerequisite labels. Concept graphs are also used in (Gordon et al., 2016) to address the problem of developing reading lists for students. The concept graph in this case is a labeled graph where nodes represent both documents and concepts (determined using Latent Dirichlet Allocation (LDA) (Blei et al., 2003)), and edges represent dependencies. They propose methods based on cross entropy and information flow for determining edges in the graph. Finally, finding prerequisite relationships has also been used in other contexts such as Massive Open Online Courses (MOOCs) (Pan et al., 2017a,b).

2.3 Reading List Generation

Jardine (2014) generates recommended reading lists from a corpus of technical papers in an unstructured manner in which a topic model weighs the relevant topics and relevant papers are chosen through his ThemedPageRank approach. He also provides a set of expert-generated reading lists. Conversely, Gordon et al. (2017) approach reading list generation from a structured perspective, first generating a concept graph from the corpus and then traversing the graph to select the most relevant document.

2.4 Survey Extraction

Recent work on survey generation for scientific topics has focused on creating summaries from academic papers (Jha et al., 2013, 2015; Jaidka et al., 2016). Jha et al. (2013) present a system that generates summaries given a topic keyword. From a base corpus of papers found by query matching, they expand the corpus via a citation network using a heuristic called Restricted Expansion. This process is repeated for seven standard NLP topics. In a similar manner, Jha et al. (2015) experiment with fifteen topics in computational linguistics and collect at least surveys written by experts on each topic, also making use of citation networks to expand their corpus. They introduce a content model as well as a discourse model and perform a qualitative comparisons of coherence with a standard summarization model.

The task of creating surveys for specified topics has also been viewed in the multi-document summarization setting of generating Wikipedia articles (Sauper and Barzilay, 2009; Liu et al., 2018). Sauper and Barzilay (2009) induce domain-specific templates from Wikipedia and fill these templates with content from the Internet. More recently Liu et al. (2018) explore a diverse set of domains for summarization and are the first to attempt abstractive summarization of the first section of Wikipedia articles, by combining extractive and abstractive summarization methods.

3 Dataset Collection

3.1 An Overview of TutorialBank

As opposed to other collections like the ACL Anthology (Bird et al., 2008; Radev et al., 2009, 2013, 2016), which contain solely academic papers, our corpus focuses mainly on resources other than academic papers. The main goal in our decision process of what to include in our corpus has been the quality-control of resources which can be used for an educational purpose. Initially, the resources collected were conference tutorials as well as surveys, books and longer papers on broader topics, as these genres contain an inherent amount of quality-control. Later on, other online resources were added to the corpus, as explained below. Student annotators, described later on, as well as the professor examined resources which they encountered in their studies. The resources were added to the corpus if deemed of good quality. Important to note is that not all resources which were found on the Internet were added to TutorialBank; one could scrape the web according to search terms, but quality control of the results would be largely missing. The quality of a resource is a somewhat subjective measure, but we aimed to find resources which would serve a pedagogical function to either students or researchers, with a professor of NLP making the final decision. This collection of resources and meta-data annotation has been done over multiple years, while this year we created the search engine and added additional annotations mentioned below.

1 - Introduction and Linguistics
2 - Language Modeling, Syntax and Parsing
3 - Semantics and Logic
4 - Pragmatics, Discourse, Dialogue and Applications
5 - Classification and Clustering
6 - Information Retrieval and Topic Modeling
7 - Neural Networks and Deep Learning
8 - Artificial Intelligence
9 - Other Topics

Table 1: Top-level Taxonomy Topics

Topic Category	Count
Introduction to Neural Networks and Deep Learning	635
Tools for Deep Learning	475
Miscellaneous Deep Learning	287
Machine Learning	225
Word Embeddings	139
Recurrent Neural Networks	134
Python Basics	133
Reinforcement learning	132
Convolutional Neural Networks	129
Introduction to AI	89

Table 2: Corpus count by taxonomy topic for the most frequent topics (excluding topic "Other").

3.1.1 TutorialBank Taxonomy

In order to facilitate the sharing of resources about NLP, we developed a taxonomy of 305 topics of varying granularity. The top levels of our taxonomy tree are shown in Table 1. The backbone of our Taxonomy corresponds to the syllabus of a university-level NLP course and was expanded to include related topics from other courses in ML, IR and AI. As a result, there is a bias in the corpus towards NLP resources and resources from other fields in so far as they are relevant to NLP. However, this bias is planned, as our focus remains teaching NLP. The resource count for the most frequent taxonomy topics is shown in Table 2.

3.2 Data Preprocessing

For each resource in the corpus, we downloaded the corresponding PDF, PowerPoint presentations and other source formats and used PDFBox to perform OCR in translating the files to textual format. For HTML pages we downloaded both the raw HTML with all images as well as a formatted text version of the pages. For copyright purposes we release only the meta data such as urls and annotations and provide scripts for reproducing the dataset.

Resource Category	Count
corpus	131
lecture	126
library	1014
link set	1186
naclo	154
paper	1176
survey	390
tutorial	2079

Table 3: Corpus count by pedagogical feature.

4 Dataset Annotation

Annotations were performed by a group of 3 PhD students in NLP, and 6 undergraduate Computer Science students who have taken at least one course in AI or NLP.

4.1 Pedagogical Function

When collecting resources from the Internet, each item was labeled according to the medium in which it was found, analogous to the pedagogical function of (Sheng et al., 2017). We will use this term throughout the paper to describe this categorization. The categories along with their counts are shown in Table 3:

- **Corpus:** A corpus provides access to and a description of a scientific dataset.
- **Lecture:** A lecture consists of slides/notes from a university lecture.
- **Library:** A library consists of github pages and other codebases which aid in the implementation of algorithms.
- **NACLO:** NACLO problems refer to linguistics puzzles from the North American Computational Linguistics Olympiad.
- **Paper:** A paper is a short/long conference paper taken from sites such as https://arxiv.org/ and which is not included in the ACL Anthology.
- **Link set:** A link set provides a collection of helpful links in one location.
- **Survey:** A survey is a long paper or book which describes a broader subject.
- **Tutorial:** A tutorial is a slide deck from a conference tutorial or an HTML page that describes a contained topic.

4.2 Topic to Resource Collection

We first identified by hand 200 potential topics for survey generation in the fields of NLP, ML, AI and IR. Topics were added according to the following criteria:

1. It is conceivable that someone would write a Wikipedia page on this topic (an actual page may or may not exist).
2. The topic is not overly general (e.g., "Natural Language Processing") or too obscure or narrow.
3. In order to write a survey on the topic, one would need to include information from a number of sources.

Capsule Networks
Domain Adaptation
Document Representation
Matrix factorization
Natural language generation
Q Learning
Recursive Neural Networks
Shift-Reduce Parsing
Speech Recognition
Word2Vec

Table 4: Random sample of the list of 200 topics used for prerequisite chains, readling lists and survey extraction.

While some of the topics come from our taxonomy, many of the taxonomy topics have a different granularity than we desired, which motivated our topic collection. Topics were added to the list along with their corresponding Wikipedia pages, if they exist. A sample of the topics selected is shown in 4. Once the list of topics was compiled, annotators were assigned topics and asked to search that topic in the TutorialBank search engine and find relevant resources. In order to impose some uniformity on the dataset, we chose to only include resources which consisted of PowerPoint slides as well as HTML pages labeled as tutorials. We divided the topics among the annotators and asked them to choose five resources per topic using our search engine. The resource need not solely focus on the given topic; the resource may be on a more general topic and include a section on the given topic. As in general searching for resources, often resources include related information, so we believe this setting is fitting. For some topics the annotators chose fewer than five resources (partially due to the constraint we impose on the form of the resources). We noted topics for which no resources were found, and rather than replace the topics to reflect TutorialBank coverage, we leave these topics in and plan to add additional resources in a future release.

4.3 Prerequisite Chains

Even with a collection of resources and a list of topics, a student may not know where to begin studying a topic of interest. For example, in order to understand sentiment analysis the student should be familiar with Bayes' Theorem, the basics of ML as well as other topics. For this purpose, the annotators annotated which topics are prerequisites of others for the given topics from their reading lists. We expanded our list of potential prerequisites to include eight additional topics which were too broad for survey generation (e.g., Linear Algebra) but which are important prerequisites to capture. Following the method of (Gordon et al., 2016), we define labeling a topic Y as a prerequisite of X according to the following question:

- Would understanding Topic Y help you to understand Topic X?

As in (Gordon et al., 2016), the annotators can answer this question as "no", "somewhat" or "yes."

4.4 Reading Lists

When annotators were collecting relevant resources for a particular topic, we asked them to order the resources they found in terms of the usefulness of the resource for learning that particular topic. We also include the Wikipedia pages corresponding to the topics, when available, as an additional source of information. We do not perform additional annotation of the order of the resources or experiment in automatically reproducing these ordered lists but rather offer this annotation as a pedagogical tool for students and educators. We plan the expansion of these lists and analysis in future experiments.

4.5 Survey Extraction

We frame the task of creating surveys of scientific topics as a document retrieval task. A student searching for resources in order to learn about a topic such as Recurrent Neural Networks (RNN's) may encounter resources 1) which solely cover RNN's as well as 2) resources which cover RNN's within the context of a larger topic (e.g., Deep Learning). Within the first type, not every piece of content (a single PowerPoint slide or section in a blog post) contributes equally well to an understanding of RNN's; the content may focus on background information or may not clearly explain the topic. Within the second type, larger tutorials may contain valuable information on the topic, but may also contain much information not immediately relevant to the query. Given a query topic and a set of parsed documents we want to retrieve the parts most relevant to the topic.

In order to prepare the dataset for extracting surveys of topics, we first divide resources into units of content which we call "cards". PowerPoint slides inherently contain a division in the form of each individual slide, so we divide PowerPoint presentations into individual slides/cards. For HTML pages, the division is less clear. However, we convert the HTML pages to a markdown file and then automatically split the markdown file using header markers. We believe this is a reasonable heuristic as tutorials and similar content tend to be broken up into sections signalled by headers.

For each of the resources which the annotators gathered for the reading lists on a given topic, that same annotator was presented with each card from that resource and asked to rate the usefulness of the card. The annotator could rate the card from 0-2, with 0 meaning the card is not useful for learning the specified topic, 1 meaning the card is somewhat useful and 2 meaning the card is useful. We chose a 3-point scale as initial trials showed a 5-point scale to be too subjective. The annotators also had the option in our annotation interface to drop cards which were parsed incorrectly or were repeated one after the other as well as skip cards and return to score a card.

4.6 Illustrations

Whether needed for understanding a subject more deeply or for preparing a blog post on a subject, images play an important role in presenting concepts more concretely. Simply extracting the text from HTML pages leaves behind this valuable information, and OCR software often fails to parse complex graphs and images in a non-destructive fashion. To alleviate this problem and promote the sharing of images, we extracted all images from our collected HTML pages. Since many images were simply HTML icons and other extraneous images, we manually checked the images and selected those which are of value to the NLP student. We collected a total of 2,000 images and matched them with the taxonomy topic name of the resource it came from as well as the url of the resource. While we cannot outdo the countless images from Google search, we believe illustrations can be an additional feature of our search engine, and we describe an interface for this collection below.

5 Additional Features and Analysis

5.1 Search Engine

In order to present our corpus in a user-friendly manner, we created a search engine using Apache Lucene[2]. We allow the user to query key words to search our resource corpus, and the results can then be sorted based on relevance, year, topic, medium, and other meta data. In addition to searching by term, users can browse the resources by topic according to our taxonomy. For each child topic from the top-level taxonomy downward, we display resources according to their pedagogical functions. In addition to searching for general resources, we also provide search functionality for a corpus of papers, where the user can search by keyword as well as by author and venue.

While the search engine described above provides access to our base corpus and meta data, we also provide a command-line interface tool with our release so that students and researchers can easily use our annotations for prerequisite topics, illustrations and survey generation for educational purposes. The tool allows the user to input a topic from the taxonomy and retrieve all images related to that topic according to our meta data. Additionally, the user can input a topic from our list of 200 topics, and our tool outputs the prerequisites of that topic according to our annotation as well as the cards labelled as relevant for that topic.

5.2 Resource Recommendation from Title and Abstract Pairs

In addition to needing to search for a general term, often a researcher begins with an idea for a project which is already focused on a nuanced sub-task. An employee at an engineering company may be starting a project on image captioning. Ideas about the potential direction of this project may be clear, but what resources may be helpful or what papers have already been published on the subject may not be immediately obvious. To this end we propose the task of recommending resources from title and abstract pairs. The employee will input the title and abstract of the project and obtain a list of resources which can help complete the project. This task is analogous to reproducing the reference section of a paper, however, with a focus on tutorials and other resources rather than solely on papers. As an addition to our search engine, we allow a user to input a title and an abstract of variable length. We then propose taxonomy topics based on string matches with the query as well as a list of resources and papers and their scores as determined by the search engine. We later explore two baseline models for recommending resources based on document and topic modeling.

[2]http://lucene.apache.org/

5.3 Dataset and Annotation Statistics

We created reading lists for 182 of the 200 topics we identify in Section 4.2. Resources were not found for 18 topics due to the granularity of the topic (e.g., Radial Basis Function Networks) as well as our intended restriction of the chosen resources to PowerPoint presentations and HTML pages. The average number of resources per reading list for the 182 topics is 3.94. As an extension to the reading lists we collected Wikipedia pages for 184 of the topics and present these urls as part of the dataset.

We annotated prerequisite relations for the 200 topics described above. We present a subset of our annotations in Figure 1, which shows the network of topic relations (nodes without incoming edges were not annotated for their prerequisites as part of this shown inter-annotation round). Our network consists of 794 unidirectional edges and 33 bidirectional edges. The presence of bidirectional edges stems from our definition of a prerequisite, which does not preclude bidirectionality (one topic can help explain another and vice-versa) as well as the similarity of the topics. The set of bidirectional edges consists of topic pairs (BLEU - ROUGE; Word Embedding - Distributional Semantics; Backpropagation - Gradient descent) which could be collapsed into one topic to create a directed acyclic graph in the future.

For survey extraction, we automatically split 313 resources into content cards which we annotated for usefulness in survey extraction. These resources are a subset of the reading lists limited in number due to constraints in downloading urls and parsing to our annotation interface. The total number of cards which were not marked as repeats/mis-parsed totals 17,088, with 54.59 per resource. 6,099 cards were labeled as somewhat relevant or relevant for the target topic. The resources marked as non-relevant may be poorly

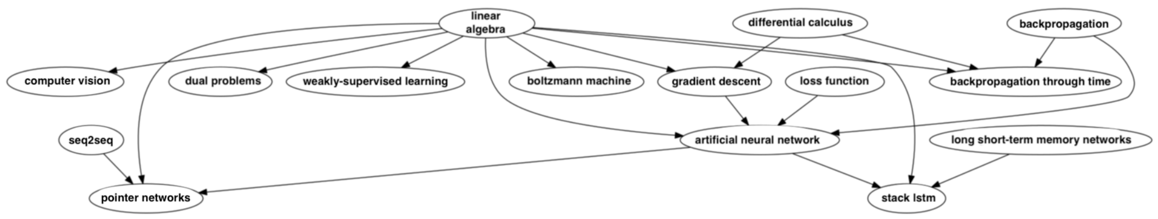


Figure 1: Subset of prerequisite annotations taken from inter-annotator agreement round.

Annotation	Kappa
Pedagogical Function	0.69
Prerequisites	0.30
Survey Extraction	0.33

Table 5: Inter-annotator agreement.

presented or may not pertain fully to the topic of that survey. These numbers confirm the appropriateness of this survey corpus as a non-trivial information retrieval task.

To better understand the difficulty of our annotation tasks, we performed inter-annotator agreement experiments for each of our annotations. We randomly sampled twenty-five resources and had annotators label for pedagogical function. Additionally, we sampled twenty-five topics for prerequisite annotations and five topics with reading list lengths of five for survey annotation. We used Fleiss's Kappa (Fleiss et al., 2004), a variant of Cohen's Kappa (Cohen, 1960) designed to measure annotator agreement for more than two annotators. The results are shown in Table 5. Using the scale as defined in Landis and Koch (1977), pedagogical function annotation exhibits *substantial agreement* while prerequisite annotation and survey extraction annotation show *fair agreement*. The Kappa score for pedagogical function is comparable to that of Sheng et al. (2017) (0.68) while the prerequisite annotation is slightly lower than the agreement metric used in Gordon et al. (2016) (0.36) although they measure agreement through Pearson correlation. We believe that the sparsity of the labels plays a role in these scores.

5.4 Comparison to Similar Datasets

Our corpus distinguishes itself in its magnitude, manual collection and focus on annotation for educational purposes in addition to research tasks. We use similar categories for classifying pedagogical function as Sheng et al. (2017), but our corpus is hand-picked and over four-times larger, while exhibiting similar annotation agreement.

Gordon et al. (2016) present a corpus for prerequisite relations among topics, but this corpus differs in coverage. They used LDA topic modeling to generate a list of 300 topics, while we manually create a list of 200 topics based on criteria described above. Although their topics are generated from the ACL Anthology and related to NLP, we find less than a 40% overlap in topics. Additionally, they only annotate a subset of the topics for prerequisite annotations while we focus on broad coverage, annotating two orders of magnitude larger in terms of prerequisite edges while exhibiting fair inter-annotator agreement.

Previous work and datasets on generating surveys for scientific topics have focused on scientific articles (Jha et al., 2013, 2015; Jaidka et al., 2016) and Wikipedia pages (Sauper and Barzilay, 2009; Liu et al., 2018) as a summarization task. We, on the other hand, view this problem as an information retrieval task and focus on extracting content from manually-collected PowerPoint slides and online tutorials. Sauper and Barzilay (2009) differ in their domain coverage, and while the surveys of Jha et al. (2013, 2015) focus on NLP, we collect resources for an order of magnitude larger set of topics. Finally, our focus here in creating surveys, as well as the other annotations, is first and foremost to create a useful tool for students and researchers. Websites such as the ACL Anthology[3] and arXiv[4] provide an abundance of resources, but do not focus on the pedagogical aspect of their content. Meanwhile, websites such as Wikipedia which aim to create a survey of a topic may not reflect the latest trends in rapidly changing fields.

6 Topic Modeling and Resource Recommendation

As an example usage of our corpus, we experimented with topic modeling and its extension to

[3]http://aclweb.org/anthology/
[4]https://arxiv.org/

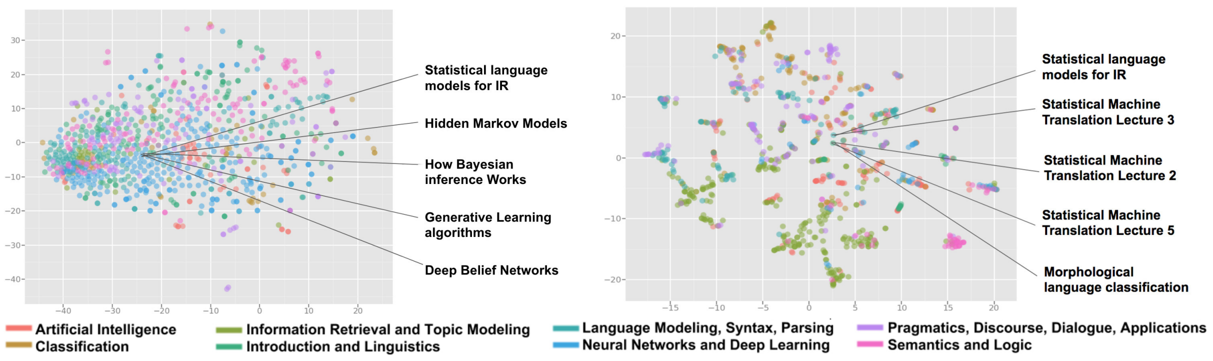


Figure 2: Plot showing a query document with title "Statistical language models for IR" and its neighbour document clusters as obtained through tSNE dimension reduction for Doc2Vec (left) and LDA topic modeling (right). Nearest neighbor documents titles are shown to the right of each plot.

resource recommendation. We restricted our corpus for this study to non-HTML files to examine the single domain of PDF's and PowerPoint presentations. This set consists of about 1,480 files with a vocabulary size 191,446 and a token count of 9,134,452. For each file, the tokens were processed, stop tokens were stripped, and then each token was stemmed. Words with counts less than five across the entire corpus were dropped. We experimented with two models: LDA, a generative probabilistic model mentioned earlier, and Doc2Vec (Le and Mikolov, 2014), an extension of Word2Vec (Mikolov et al., 2013) which creates representations of arbitrarily-sized documents. Figure 2 shows the document representations obtained with Doc2Vec as well as the topic clusters created with LDA. The grouping of related resources around a point demonstrates the clustering abilities of these models. We applied LDA in an unsupervised way, using 60 topics over 300 iterations as obtained through experimentation, and then colored each document dot with its category to observe the distribution. Our Doc2Vec model used hidden dimension 300, a window size of 10 and a constant learning rate of 0.025. Then, the model was trained for 10 epochs.

We tested these models for the task of resource recommendation from title+abstract pairs. We collected 10 random papers from ACL 2017. For LDA, the document was classified to a topic, and then the top resources from that topic were chosen, while Doc2Vec computed the similarity between the query document and the training set and chose the most similar documents. We concatenated the title and abstract as input and had our models predict the top 20 documents. We then had five annotators rate the recommendations for

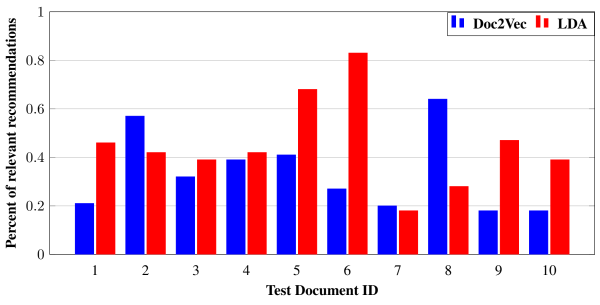


Figure 3: Relevance accuracies of the Doc2Vec and LDA resource recommendation models.

helpfulness as 0 (not helpful) or 1 (helpful). Recommended resources were rated according to the criterion of whether reading this resource would be useful in doing a project as described in the title and abstract. The results are found in Figure 3. Averaging the performance over each test case, the LDA model performed better than Doc2Vec (0.45 to 0.34), although both leave large room for improvements. LDA recommended resources notably better for cases 5 and 6, which correspond to papers with very well defined topics areas (Question Answering and Machine Translation) while Doc2Vec was able to find similar documents for cases 2 and 8 which are a mixture of topics, yet are well-represented in our corpus (Reinforcement Learning with dialog agents and emotion (sentiment) detection with classification). The low performance for both models also corresponds to differences in corpus coverage, and we plan to explore this bias in the future. We believe that this variant of reading list generation as well as the relationship between titles and abstracts is an unexplored and exciting area for future research.

7 Conclusion and Future Work

In this paper we introduce the TutorialBank Corpus, a collection of over 6,300 hand-collected resources on NLP and related fields. Our corpus is notably larger than similar datasets which deal with pedagogical resources and topic dependencies and unique in use as an educational tool. To this point, we believe that this dataset, with its multiple layers of annotation and usable interface, will be an invaluable tool to the students, educators and researchers of NLP. Additionally, the corpus promotes research on tasks not limited to pedagogical function classification, topic modeling and prerequisite relation labelling. Finally, we formulate the problem of recommending resources for a given title and abstract pair as a new way to approach reading list generation and propose two baseline models. For future work we plan to continue the collection and annotation of resources and to separately explore each of the above research tasks.

Acknowledgments

We would like to thank all those who worked on the development of the search engine and website as well as those whose discussion and annotated greatly helped this work, especially Jungo Kasai, Alexander Strzalkowski, Michihiro Yasunaga and Rui Zhang.

References

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan R. Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics. In *LREC*. European Language Resources Association.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and psychological measurement*, 20(1):37–46.

Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. 2004. *The Measurement of Interrater Agreement*. John Wiley & Sons, Inc.

Jonathan Gordon, Stephen Aguilar, Emily Sheng, and Gully Burns. 2017. Structured Generation of Technical Reading Lists. In *BEA@EMNLP*, pages 261–270. Association for Computational Linguistics.

Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. 2016. Modeling Concept Dependencies in a Scientific Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. 2016. Overview of the Cl-SciSumm 2016 Shared Task. In *BIRNDL@JCDL*, volume 1610 of *CEUR Workshop Proceedings*, pages 93–102. CEUR-WS.org.

James Gregory Jardine. 2014. *Automatically Generating Reading Lists*. Ph.D. thesis, University of Cambridge, UK.

Rahul Jha, Amjad Abu-Jbara, and Dragomir R. Radev. 2013. A System for Summarizing Scientific Topics Starting from Keywords. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 572–577.

Rahul Jha, Reed Coke, and Dragomir R. Radev. 2015. Surveyor: A System for Generating Coherent Survey Articles for Scientific Topics. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2167–2173.

J Richard Landis and Gary G Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, pages 159–174.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *CoRR*, abs/1405.4053.

Chen Liang, Jianbo Ye, Zhaohui Wu, Bart Pursel, and C. Lee Giles. 2017. Recovering Concept Prerequisite Relations from University Course Dependencies. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4786–4791.

Hanxiao Liu, Wanli Ma, Yiming Yang, and Jaime G. Carbonell. 2016. Learning Concept Graphs from Online Educational Data. *J. Artif. Intell. Res.*, 55:1059–1090.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by Summarizing Long Sequences. *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Liangming Pan, Chengjiang Li, Juanzi Li, and Jie Tang. 2017a. Prerequisite Relation Learning for Concepts in MOOCs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1447–1456.

Liangming Pan, Xiaochen Wang, Chengjiang Li, Juanzi Li, and Jie Tang. 2017b. Course Concept Extraction in MOOCs via Embedding-Based Graph Propagation. In *IJCNLP(1)*, pages 875–884. Asian Federation of Natural Language Processing.

Dragomir R. Radev, Mark Thomas Joseph, Bryan R. Gibson, and Pradeep Muthukrishnan. 2016. A Bibliometric and Network Analysis of the Field of Computational Linguistics. *JASIST*, 67(3):683–706.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL Anthology Network Corpus. In *Proceedings, ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, Singapore.

Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL Anthology Network Corpus. *Language Resources and Evaluation*, 47(4):919–944.

Christina Sauper and Regina Barzilay. 2009. Automatically Generating Wikipedia Articles: A Structure-Aware Approach. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 208–216.

Emily Sheng, Prem Natarajan, Jonathan Gordon, and Gully Burns. 2017. An Investigation into the Pedagogical Features of Documents. In *BEA@EMNLP*, pages 109–120. Association for Computational Linguistics.

Tiffany Ya Tang and Gordon I. McCalla. 2004. On the Pedagogically Guided Paper Recommendation for an Evolving Web-Based Learning System. In *FLAIRS Conference*, pages 86–92. AAAI Press.

Tiffany Ya Tang and Gordon I. McCalla. 2009. The Pedagogical Value of Papers: a Collaborative-Filtering based Paper recommender. *J. Digit. Inf.*, 10(2).

Give Me More Feedback: Annotating Argument Persuasiveness and Related Attributes in Student Essays

Winston Carlile Nishant Gurrapadi Zixuan Ke Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{winston,zixuan,vince}@hlt.utdallas.edu,Nishant.Gurrapadi@utdallas.edu

Abstract

While argument persuasiveness is one of the most important dimensions of argumentative essay quality, it is relatively little studied in automated essay scoring research. Progress on scoring argument persuasiveness is hindered in part by the scarcity of annotated corpora. We present the first corpus of essays that are simultaneously annotated with argument components, argument persuasiveness scores, and attributes of argument components that impact an argument's persuasiveness. This corpus could trigger the development of novel computational models concerning argument persuasiveness that provide useful feedback to students on *why* their arguments are (un)persuasive in addition to *how* persuasive they are.

1 Introduction

The vast majority of existing work on automated essay scoring has focused on *holistic* scoring, which summarizes the quality of an essay with a single score and thus provides very limited feedback to the writer (see Shermis and Burstein (2013) for the state of the art). While recent attempts address this problem by scoring a particular dimension of essay quality such as coherence (Miltsakaki and Kukich, 2004), technical errors, relevance to prompt (Higgins et al., 2004; Persing and Ng, 2014), organization (Persing et al., 2010), and thesis clarity (Persing and Ng, 2013), argument persuasiveness is largely ignored in existing automated essay scoring research despite being one of the most important dimensions of essay quality.

Nevertheless, scoring the persuasiveness of arguments in student essays is by no means easy. The difficulty stems in part from the scarcity of persuasiveness-annotated corpora of student essays. While persuasiveness-annotated corpora exist for other domains such as online debates (e.g., Habernal and Gurevych (2016a; 2016b)), to our knowledge only one corpus of persuasiveness-annotated student essays has been made publicly available so far (Persing and Ng, 2015).

Though a valuable resource, Persing and Ng's (2015) (P&N) corpus has several weaknesses that limit its impact on automated essay scoring research. First, P&N assign only *one* persuasiveness score to each essay that indicates the persuasiveness of the argument an essay makes for its *thesis*. However, multiple arguments are typically made in a persuasive essay. Specifically, the arguments of an essay are typically structured as an argument tree, where the major claim, which is situated at the root of the tree, is supported by one or more claims (the children of the root node), each of which is in turn supported by one or more premises. Hence, each node and its children constitute an argument. In P&N's dataset, only the persuasiveness of the overall argument (i.e., the argument represented at the root and its children) of each essay is scored. Hence, any system trained on their dataset cannot provide any feedback to students on the persuasiveness of any arguments other than the overall argument. Second, P&N's corpus does not contain annotations that explain *why* the overall argument is not persuasive if its score is low. This is undesirable from a feedback perspective, as a student will not understand why her argument is not persuasive if its score is low.

Our goal in this paper is to annotate and make publicly available a corpus of persuasive student essays that addresses the aforementioned weaknesses via designing appropriate annotation schemes and scoring rubrics. Specifically, not only do we score the persuasiveness of *each* argument in each essay (rather than simply the persuasiveness of the overall argument), but we also identify a set of attributes that can explain an argument's persuasiveness and annotate each argument with the values of these attributes. These annotations enable the development of systems that can provide useful feedback to students, as the attribute values predicted by these systems can help a student understand why her essay receives a particular persuasiveness score. To our knowledge, this is the first corpus of essays that are simultaneously annotated with argument components, persuasiveness scores, and related attributes.[1]

2 Related Work

While argument mining research has traditionally focused on determining the argumentative structure of a text document (i.e., identifying its major claim, claims, and premises, as well as the relationships between these argument components) (Stab and Gurevych, 2014b, 2017a; Eger et al., 2017), researchers have recently begun to study new argument mining tasks, as described below.

Persuasiveness-related tasks. Most related to our study is work involving argument persuasiveness. For instance, Habernal and Gurevych (2016b) and Wei et al. (2016) study the persuasiveness *ranking* task, where the goal is to rank two internet debate arguments written for the same topic w.r.t. their persuasiveness. As noted by Habernal and Gurevych, ranking arguments is a relatively easier task than scoring an argument's persuasiveness: in ranking, a system simply determines whether one argument is more persuasive than the other, but not *how much more* persuasive one argument is than the other; in scoring, however, a system has to determine how persuasive an argument is on an absolute scale. Note that ranking is not an acceptable evaluation setting for studying argument persuasiveness in the essay domain, as feedback for an essay has to be provided *independently* of other essays.

In contrast, there are studies that focus on factors affecting argument persuasiveness in internet debates. For instance, Lukin et al. (2017) examine how audience variables (e.g., personalities) interact with argument style (e.g., factual vs. emotional arguments) to affect argument persuasiveness. Persing and Ng (2017) identify factors that *negatively* impact persuasiveness, so their factors, unlike ours, cannot explain what makes an argument persuasive.

Other argument mining tasks. Some of the attributes that we annotate our corpus with have been studied. For instance, Hidey et al. (2017) examine the different semantic types of claims and premises, whereas Higgins and Walker (2012) investigate persuasion strategies (i.e., ethos, pathos, logos). Unlike ours, these studies use data from online debate forums and social/environment reports. Perhaps more importantly, they study these attributes independently of persuasiveness.

Several argument mining tasks have recently been proposed. For instance, Stab and Gurevych (2017b) examine the task of whether an argument is sufficiently supported. Al Khatib et al. (2016) identify and annotate a news editorial corpus with fine-grained argumentative discourse units for the purpose of analyzing the argumentation strategies used to persuade readers. Wachsmuth et al. (2017) focus on identifying and annotating 15 logical, rhetorical, and dialectical dimensions that would be useful for automatically accessing the quality of an argument. Most recently, the Argument Reasoning Comprehension task organized as part of SemEval 2018 has focused on selecting the correct warrant that explains reasoning of an argument that consists of a claim and a reason.[2]

3 Corpus

The corpus we chose to annotate is composed of 102 essays randomly chosen from the Argument Annotated Essays corpus (Stab and Gurevych, 2014a). This collection of essays was taken from *essayforum*[3], a site offering feedback to students wishing to improve their ability to write persuasive essays for tests. Each essay is written in response to a topic such as "should high school make music lessons compulsory?" and has already been annotated by Stab and Gurevych with an argument tree. Hence, rather than annotate everything from scratch, we annotate the persuasiveness score of each argument in the already-annotated argument trees in this essay collection as well as the attributes that potentially impact persuasiveness.

Each argument tree is composed of three types of tree nodes that correspond to argument compo-

[1]Our annotated corpus and annotation manual are publicly available at the website http://www.hlt.utdallas.edu/∼zixuan/EssayScoring.

[2]https://competitions.codalab.org/competitions/17327

[3]www.essayforum.com

Essays: 102	Sentences: 1462	Tokens: 24518
Major Claims: 185	Claims: 567	Premises: 707
Support Relations: 3615	Attack Relations: 219	

Table 1: Corpus statistics.

nents. The three annotated argument component types include: **MajorClaim**, which expresses the author's stance with respect to the essay's topic; **Claims**, which are controversial statements that should not be accepted by readers without additional support; and **Premises**, which are reasons authors give to persuade readers about the truth of another argument component statement. The two relation types include: **Support**, which indicates that one argument component supports another, and **Attack**, which indicates that one argument component attacks another.

Each argument tree has three to four levels. The root is a major claim. Each node in the second level is a claim that supports or attacks its parent (i.e., the major claim). Each node is the third level is a premise that supports or attacks its parent (i.e., a claim). There is an optional fourth level consisting of nodes that correspond to premises. Each of these premises either supports or attacks its (premise) parent. Stab and Gurevych (2014a) report high inter-annotator agreement on these annotations: for the annotations of major claims, claims, and premises, the Krippendorff's α values (Krippendorff, 1980) are 0.77, 0.70, and 0.76 respectively, and for the annotations of support and attack relations, the α values are both 0.81.

Note that Stab and Gurevych (2014a) determine premises and claims by their position in the argument tree and not by their semantic meaning. Due to the difficulty of treating an opinion as a non-negotiable unit of evidence, we convert all subjective premises into claims to demonstrate that they are subjective and require backing. At the end of this process, several essays contain argument trees that violate the scheme used by Stab and Gurevych, due to some premises supported by opinion premises, now converted to claims. Although the ideal argument should not violate the canonical structure, students attempting to improve their persuasive writing skills may not understand this, and mistakenly support evidence with their own opinions.

Statistics of this corpus are shown in Table 1. Its extensive use in argument mining research in recent years together with its reliably annotated argument trees makes it an ideal corpus to use for our annotation task.

4 Annotation

4.1 Definition

Since persuasiveness is defined on an argument, in order to annotate persuasiveness we need to define precisely what an argument is. Following van Eemeren et al. (2014), we define an argument as consisting of a conclusion that may or may not be supported/attacked by a set of evidences. Given an argument tree, a non-leaf node can be interpreted as a "conclusion" that is supported or attacked by its children, which can therefore be interpreted as "evidences" for the conclusion. In contrast, a leaf node can be interpreted as an unsupported conclusion. Hence, for the purposes of our work, an argument is composed of a node in an argument tree and all of its children, if any.

4.2 Annotation Scheme

Recall that the goal of our annotation is to score each argument w.r.t. its persuasiveness (see Table 2 for the rubric for scoring persuasiveness) and annotate each of its components with a set of predefined attributes that could impact the argument's persuasiveness. Table 3 presents a summary of the attributes we annotate. The rest of this subsection describes these attributes.

Each component type (MajorClaim, Claim, Premise) has a distinct set of attributes. All component types have three attributes in common: Eloquence, Specificity, and Evidence. *Eloquence* is how well the author uses language to convey ideas, similar to clarity and fluency. *Specificity* refers to the narrowness of a statement's scope. Statements that are specific are more believable because they indicate an author's confidence and depth of knowledge about a subject matter. Argument assertions (major claims and claims) need not be believable on their own since that is the job of the supporting evidence. The *Evidence* score describes how well the supporting components support the parent component. The rubrics for scoring Eloquence, Evidence, Claim/MajorClaim Specificity, and Premise Specificity are shown in Tables 4, 5, 6, and 7 respectively.

MajorClaim Since the major claim represents the overall argument of the essay, it is in this component that we annotate the persuasive strategies employed (i.e., *Ethos, Pathos* and *Logos*). These

Score	Description
6	A very strong, clear argument. It would persuade most readers and is devoid of errors that might detract from its strength or make it difficult to understand.
5	A strong, pretty clear argument. It would persuade most readers, but may contain some minor errors that detract from its strength or understandability.
4	A decent, fairly clear argument. It could persuade some readers, but contains errors that detract from its strength or understandability.
3	A poor, understandable argument. It might persuade readers who are already inclined to agree with it, but contains severe errors that detract from its strength or understandability.
2	It is unclear what the author is trying to argue or the argument is poor and just so riddled with errors as to be completely unpersuasive.
1	The author does not appear to make any argument (e.g. he may just describe some incident without explaining why it is important). It could not persuade any readers because there is nothing to be persuaded of. It may or may not contain detectable errors, but errors are moot since there is not an argument for them to interfere with.

Table 2: Description of the Persuasiveness scores.

Attribute	Possible Values	Applicability	Description
Specificity	1–5	MC,C,P	How detailed and specific the statement is
Eloquence	1–5	MC,C,P	How well the idea is presented
Evidence	1–6	MC,C,P	How well the supporting statements support their parent
Logos/Pathos/Ethos	yes,no	MC,C	Whether the argument uses the respective persuasive strategy
Relevance	1–6	C,P	The relevance of the statement to the parent statement
ClaimType	value,fact,policy	C	The category of what is being claimed
PremiseType	see Section 4.2	P	The type of Premise, e.g. statistics, definition, real example, etc.
Strength	1–6	P	How well a single statement contributes to persuasiveness

Table 3: Summary of the attributes together with their possible values, the argument component type(s) each attribute is applicable to (**MC**: MajorClaim, **C**: Claim, **P**: Premise), and a brief description.

Score	Description
5	Demonstrates mastery of English. There are no grammatical errors that distract from the meaning of the sentence. Exhibits a well thought out, flowing sentence structure that is easy to read and conveys the idea exceptionally well.
4	Demonstrates fluency in English. If there are any grammatical or syntactical errors, their affect on the meaning is negligible. Word choice suggests a broad vocabulary.
3	Demonstrates competence in English. There might be a few errors that are noticeable but forgivable, such as an incorrect verb tense or unnecessary pluralization. Demonstrates a typical vocabulary and a simple sentence structure.
2	Demonstrates poor understanding of sentence composition and/or poor vocabulary. The choice of words or grammatical errors force the reader to reread the sentence before moving on.
1	Demonstrates minimal eloquence. The sentence contains errors so severe that the sentence must be carefully analyzed to deduce its meaning.

Table 4: Description of the Eloquence scores.

Score	Description
6	A very strong, very persuasive argument body. There are many supporting components that have high Relevance scores. There may be a few attacking child components, but these components must be used for either concession or refuting counterarguments as opposed to making the argument indecisive or contradictory.
5	A strong, persuasive argument body. There are sufficient supporting components with respectable scores.
4	A decent, fairly persuasive argument body.
3	A poor, possibly persuasive argument body.
2	A totally unpersuasive argument body.
1	There is no argument body for the given component.

Table 5: Description of the Evidence scores.

three attributes are not inherent to the text identifying the major claim but instead summarize the child components in the argument tree.

Claim The claim argument component possesses all of the attributes of a major claim in addition to a *Relevance* score and a *ClaimType*. In order for an argument to be persuasive, all supporting components must be relevant to the component that they support/attack. The scoring rubric for Relevance is shown in Table 8. The ClaimType can be *value* (e.g., something is good or bad, important or not important, etc.), *fact* (e.g. something

Score	Description
5	The claim summarizes the argument well and has a qualifier that indicates the extent to which the claim holds true. Claims that summarize the argument well must reference most or all of the supporting components.
4	The claim summarizes the argument very well by mentioning most or all of the supporting components, but does not have a qualifier indicating the conditions under which the claim holds true. Alternatively, the claim may moderately summarize the argument by referencing a minority of supporting components and contain qualifier.
3	The claim has a qualifier clause or references a minority of the supporting components, but not both.
2	The claim does not make an attempt to summarize the argument nor does it contain a qualifier clause.
1	Simply rephrases the major claim or is outside scope of the major claim (argument components were annotated incorrectly: major claim could be used to support claim).

Table 6: Description of the Claim and MajorClaim Specificity scores.

Score	Description
5	An elaborate, very specific statement. The statement contains numerical data, or a historical example from the real world. There is (1) both a sufficient qualifier indicating the extent to which the statement holds true and an explanation of why the statement is true, or (2) at least one real world example, or (3) a sufficient description of a hypothetical situation that would evoke a mental image of the situation in the minds of most readers.
4	A more specific statement. It is characterized by either an explanation of why the statement is true, or a qualifier indicating when/to what extent the statement is true. Alternatively, it may list examples of items that do not qualify as historical events.
3	A sufficiently specific statement. It simply states a relationship or a fact with little ambiguity.
2	A broad statement. A statement with hedge words and without other redeeming factors such as explicit examples, or elaborate reasoning. Additionally, there are few adjectives or adverbs.
1	An extremely broad statement. There is no underlying explanation, qualifiers, or real-world examples.

Table 7: Description of the Premise Specificity scores.

Score	Description
6	Anyone can see how the support relates to the parent claim. The relationship between the two components is either explicit or extremely easy to infer. The relationship is thoroughly explained in the text because the two components contain the same words or exhibit coreference.
5	There is an implied relationship that is obvious, but it could be improved upon to remove all doubt. If the relationship is obvious, both relating components must have high Eloquence and Specificity scores.
4	The relationship is fairly clear. The relationship can be inferred from the context of the two statements. One component must have a high Eloquence and Specificity scores and the other must have lower but sufficient Eloquence and Specificity scores for the relationship to be fairly clear.
3	Somewhat related. It takes some thinking to imagine how the components relate. The parent component or the child component have low clarity scores. The two statements are about the same topic but unrelated ideas within the domain of said topic.
2	Mostly unrelated. It takes some major assumptions to relate the two components. A component may also receive this score if both components have low clarity scores.
1	Totally unrelated. Very few people could see how the two components relate to each other. The statement was annotated to show that it relates to the claim, but this was clearly in error.

Table 8: Description of the Relevance scores.

is true or false), or *policy* (claiming that some action should or should not be taken).

Premise The attributes exclusive to premises are *PremiseType* and *Strength*. To understand Strength, recall that only premises can persuade readers, but also that an argument can be composed of a premise and a set of supporting/attacking premises. In an argument of this kind, Strength refers to how well the parent premise contributes to the persuasiveness independently of the contributions from its children. The scoring rubric for Strength is shown in Table 9. PremiseType takes on a discrete value from one of the following: real_example, invented_instance, analogy, testimony, statistics, definition, common_knowledge, and warrant. Analogy, testimony, statistics, and definition are self-explanatory. A premise is labeled *invented_instance* when it describes a hypothetical situation, and *definition* when it provides a definition to be used elsewhere in the argument. A premise has type *warrant* when it does not fit any other type, but serves a functional purpose to explain the relationship between two entities or clarify/quantify another statement. The *real_example* premise type indicates that the statement is a historical event that actually occurred, or something that is verfiably true about the real world.

Score	Description
6	A very strong premise. Not much can be improved in order to contribute better to the argument.
5	A strong premise. It contributes to the persuasiveness of the argument very well on its own.
4	A decent premise. It is a fairly strong point but lacking in one or more areas possibly affecting its perception by the audience.
3	A fairly weak premise. It is not a strong point and might only resonate with a minority of readers.
2	A totally weak statement. May only help to persuade a small number of readers.
1	The statement does not contribute at all.

Table 9: Description of the Strength scores.

Attribute	Value	MC	C	P
Specificity	1	0	80	64
	2	73	259	134
	3	72	155	238
	4	32	59	173
	5	8	14	98
Logos	Yes	181	304	
	No	4	263	
Pathos	Yes	67	59	
	No	118	508	
Ethos	Yes	16	9	
	No	169	558	
Relevance	1		1	5
	2		33	45
	3		58	59
	4		132	145
	5		97	147
	6		246	306
Evidence	1	3	246	614
	2	62	115	28
	3	57	85	12
	4	33	80	26
	5	16	35	15
	6	14	6	12
Eloquence	1	3	23	24
	2	19	106	97
	3	116	320	383
	4	42	102	154
	5	5	16	49
ClaimType	fact		368	
	value		145	
	policy		54	
PremiseType	real_example			93
	invented_instance			53
	analogy			2
	testimony			4
	statistics			15
	definition			3
	common_know.			493
	warrant			44
Persuasiveness	1	3	82	8
	2	62	278	112
	3	60	84	145
	4	28	74	249
	5	17	39	123
	6	15	10	70

Table 10: Class/Score distributions by component type.

Attribute	MC	C	P
Persuasiveness	.739	.701	.552
Eloquence	.590	.580	.557
Specificity	.560	.530	.690
Evidence	.755	.878	.928
Relevance		.678	.555
Strength			.549
Logos	1	.842	
Pathos	.654	.637	
Ethos	1	1	
ClaimType		.589	
PremiseType			.553

Table 11: Krippendorff's α agreement on each attribute by component type.

4.3 Annotation Procedure

Our 102 essays were annotated by two native speakers of English. We first familiarized them with the rubrics and definitions and then trained them on five essays (not included in our corpus). After that, they were both asked to annotate a randomly selected set of 30 essays and discuss the resulting annotations to resolve any discrepancies. Finally, the remaining essays were partitioned into two sets, and each annotator received one set to annotate. The resulting distributions of scores/classes for persuasiveness and the attributes are shown in Table 10.

4.4 Inter-Annotator Agreement

We use Krippendorff's α to measure inter-annotator agreement. Results are shown in Table 11. As we can see, all attributes exhibit an agreement above 0.5, showing a correlation much more significant than random chance. Persuasiveness has an agreement of 0.688, which suggests that it can be agreed upon in a reasonably general sense. The MajorClaim components have the highest Persuasiveness agreement, and it declines as the type changes to Claim and then to Premise. This would indicate that persuasiveness is easier to articulate in a wholistic sense, but difficult to explain as the number of details involved in the explanation increases.

The agreement scores that immediately stand out are the perfect 1.0's for Logos and Ethos. The perfect Logos score is explained by the fact that every major claim was marked to use logos. Although ethos is far less common, both annotators easily recognized it. This is largely due to the indisputability of recognizing a reference to an accepted authority on a given subject. Very few authors utilize this approach, so when they do it is extremely apparent. Contrary to Persuasiveness, Evidence agreement exhibits an upward trend as the component scope narrows. Even with this pattern, the Evidence agreement is always higher than Persuasiveness agreement, which suggests that it is not the only determiner of persuasiveness.

In spite of a rubric defining how to score Eloquence, it remains one of the attributes with the lowest agreement. This indicates that it is difficult to agree on exact eloquence levels beyond basic English fluency. Additionally, Specificity produced unexpectedly low agreement in claims and major claims. Precisely quantifying how well a claim summarizes its argument turned out to be a complicated and subjective task. Relevance agreement for premises is one of the lowest, partly because there are multiple scores for high relevance, and no examples were given in the rubric.

All attributes but those with the highest agreement are plagued by inherent subjectivity, regardless of how specific the rubric is written. There are often multiple interpretations of a given sentence, sometimes due to the complexity of natural language, and sometimes due to the poor writing of the author. Naturally, this makes it difficult to identify certain attributes such as Pathos, ClaimType, and PremiseType.

Although great care was taken to make each attribute as independent of the others as possible, they are all related to each other to a minuscule degree (e.g., Eloquence and Specificity). While annotators generally agree on what makes a persuasive argument, the act of assigning blame to the persuasiveness (or lack thereof) is tainted by this overlapping of attributes.

4.5 Analysis of Annotations

To understand whether the attributes we annotated are indeed useful for predicting persuasiveness, we compute the Pearson's Correlation Coefficient (PC) between persuasiveness and each of the attributes along with the corresponding p-values. Results are shown in Table 12. Among the correlations that are statistically significant at the $p < .05$ level, we see, as expected, that Persuasiveness is positively correlated with Specificity, Evidence, Eloquence, and Strength. Neither is it surprising that support provided by a premise in the form of statistics and examples is positively correlated with Persuasiveness. While Logos and invented_instance also have significant correlations with Persuasiveness, the correlation is very weak.

Attribute	PC	p**-value**
Specificity	.5680	0
Relevance	−.0435	.163
Eloquence	.4723	0
Evidence	.2658	0
Strength	.9456	0
Logos	−.1618	0
Ethos	−.0616	.1666
Pathos	−.0835	.0605
ClaimType:fact	.0901	.1072
ClaimType:value	−.0858	.1251
ClaimType:policy	−.0212	.7046
PremiseType:real_example	.2414	0
PremiseType:invented_instance	.0829	.0276
PremiseType:analogy	.0300	.4261
PremiseType:testimony	.0269	.4746
PremiseType:statistics	.1515	0
PremiseType:definition	.0278	.4608
PremiseType:common_knowledge	−.2948	1.228
PremiseType:warrant	.0198	.6009

Table 12: Correlation of each attribute with Persuasiveness and the corresponding p-value.

	MC	**C**	**P**	**Avg**
PC	.9688	.9400	.9494	.9495
ME	.0710	.1486	.0954	.1061

Table 13: Persuasiveness scoring using gold attributes.

Next, we conduct an oracle experiment in an attempt to understand how well these attributes, when used together, can explain the persuasiveness of an argument. Specifically, we train three linear SVM regressors (using the SVM^{light} software (Joachims, 1999) with default learning parameters except for C (the regularization parameter), which is tuned on development data using grid search) to score an argument's persuasiveness using the *gold* attributes as features. The three regressors are trained on arguments having MajorClaims, Claims, and Premises as parents. For instance, to train the regressor involving MajorClaims, each instance corresponds to an argument represented by all and only those attributes involved in the major claim and all of its children.[4]

Five-fold cross-validation results, which are

[4]There is a caveat. If we define features for each of the children, the number of features will be proportional to the number of children. However, SVMs cannot handle a variable number of features. Hence, all of the children will be represented by one set of features. For instance, the Specificity feature value of the children will be the Specificity values averaged over all of the children.

Prompt: Government budget focus, young children or university?

Education plays a significant role in a country's long-lasting prosperity. It is no wonder that governments throughout the world lay special emphasis on education development. As for the two integral components within the system, elementary and advanced education, there's no doubt that a government is supposed to offer sufficient financial support for both.

Concerning that elementary education is the fundamental requirement to be a qualified citizen in today's society, government should guarantee that all people have equal and convenient access to it. So a lack of well-established primary education goes hand in hand with a high rate of illiteracy, and this interplay compromises a country's future development. In other words, if countries, especially developing ones, are determined to take off, one of the key points governments should set on agenda is to educate more qualified future citizens through elementary education.
...

Table 14: An example essay. Owing to space limitations, only its first two paragraphs are shown.

		P	E	S	Ev	R	St	Lo	Pa	Et	cType	pType
M1	government is supposed to offer sufficient financial support for both	3	4	2	3			T	F	F		
C1	if countries, especially developing ones, are determined to take off, one of the key points governments should set on agenda is to educate more qualified future citizens through elementary education	4	5	4	4	6		T	F	F	policy	
P1	elementary education is the fundamental requirement to be a qualified citizen in today's society	4	5	3	1	6	4					A
C2	government should guarantee that all people have equal and convenient access to it	2	3	1	1	6		F	F	F	policy	
P2	a lack of well-established primary education goes hand in hand with a high rate of illiteracy, and this interplay compromises a country's future development	4	5	3	1	6	4					C

Table 15: The argument components in the example in Table 14 and the scores of their associated attributes: **P**ersuasiveness, **E**loquence, **S**pecificity, **Ev**idence, **R**elevance, **St**rength, **Lo**gos, **Pa**thos, **Et**hos, **c**laim**Type**, and **p**remise**Type**.

shown in Table 13, are expressed in terms of two evaluation metrics, PC and ME (the mean absolute distance between a system's prediction and the gold score). Since PC is a *correlation* metric, higher correlation implies better performance. In contrast, ME is an *error* metric, so lower scores imply better performance. As we can see, the large PC values and the relatively low ME values provide suggestive evidence that these attributes, when used in combination, can largely explain the persuasiveness of an argument.

What these results imply in practice is that models that are trained on these attributes for persuasiveness scoring could provide useful feedback to students on *why* their arguments are (un)persuasive. For instance, one can build a pipeline system for persuasiveness scoring as follows. Given an argument, this system first predicts its attributes and then scores its persuasiveness using the predicted attribute values computed in the first step. Since the persuasiveness score of an argument is computed using its predicted attributes, these attributes can explain the persuasiveness score. Hence, a student can figure out which aspect of persuasiveness needs improvements by examining the values of the predicted attributes.

4.6 Example

To better understand our annotation scheme, we use the essay in Table 14 to illustrate how we obtain the attribute values in Table 15. In this essay, Claim **C1**, which supports MajorClaim **M1**, is supported by three children, Premises **P1** and **P2** as well as Claim **C2**.

After reading the essay in its entirety and acquiring a holistic impression of the argument's strengths and weaknesses, we begin annotating the atomic argument components bottom up, starting with the leaf nodes of the argument tree. First, we consider **P2**. Its Evidence score is 1 because it is a leaf node with no supporting evidence. Its Eloquence score is 5 because the sentence has no serious grammatical or syntactic errors, has a flowing, well thought out sentence structure, and uses articulate vocabulary. Its Specificity score is 3 because it is essentially saying that poor primary education causes illiteracy and consequently inhibits a country's development. It does not state why or to what extent, so we cannot assign a score of 4. However, it does explain a simple relationship with little ambiguity due to the lack of hedge words, so we can assign a score of 3. Its PremiseType is *common_knowledge* because it is reasonable to assume most people would agree that poor primary education causes illiteracy, and also that illiteracy inhibits a country's development. Its Relevance score is 6: its relationship with its parent is clear because the two components exhibit coreference. Specifically, **P2** contains a reference to primary/elementary education and shows how this affects a country's inability to transition from developing to developed. Its Strength is 4: though eloquent and relevant, **P2** is lacking substance in order to be considered for a score of 5 or 6. The PremiseType is *common_knowledge*, which is mediocre compared to statistics and real_example. In order for a premise that is not grounded in the real world to be strong, it must be very specific. **P2** only scored a 3 in Specificity, so we assign a Strength score of 4. Finally, the argument headed by **P2**, which does not have any children, has a Persuasiveness score of 4, which is obtained by summarizing the inherent strength of the premise and the supporting evidence. Although there is no supporting evidence for this premise, this does not adversely affect persuasiveness due to the standalone nature of premises. In this case the persuasiveness is derived totally from the strength.

Next, the annotator would score **C2** and **P1**, but for demonstration purposes we will examine the scoring of **C1**. **C1**'s Eloquence score is 5 because it shows fluency, broad vocabulary, and attention to how well the sentence structure reads. Its ClaimType is *policy* because it specifically says that the government should put something on their agenda. Its Specificity score is 4: while it contains information relevant to all the child premises (i.e., creating qualified citizens, whose role it is to provide the education, and the effect of education on a country's development), it does not contain a qualifier stating the extent to which the assertion holds true. Its Evidence score is 4: **C1** has two premises with decent persuasiveness scores and one claim with a poor persuasiveness score, and there are no attacking premises, so intuitively, we may say that this is a midpoint between many low quality premises and few high quality premises. We mark Logos as true, Pathos as false, and Ethos as false: rather than use an emotional appeal or an appeal to authority of any sort, the author attempts to use logical reasoning in order to prove their point. Its Persuasiveness score is 4: this score is mainly determined by the strength of the supporting evidence, given that the assertion is precise and clear as determined by the specificity and eloquence. Its Relevance score is 6, as anyone can see how endorsement of elementary education in **C1** relates to the endorsement of elementary and university education in its parent (i.e., **M1**).

After all of the claims have been annotated in the bottom-up method, the annotator moves on to the major claim, **M1**. **M1**'s Eloquence score is 4: while it shows fluency and a large vocabulary, it is terse and does not convey the idea exceptionally well. Its persuasion strategies are obtained by simply taking the logical disjunction of those used in its child claims. Since every claim in this essay relied on logos and did not employ pathos nor ethos, **M1** is marked with Logos as true, Pathos as false, and Ethos as false. Its Evidence score is 3: in this essay there are two other supporting claims not in the excerpt, with persuasiveness scores of only 3 and 2, so **M1**'s evidence has one decently persuasive claim, one claim that is poor but understandable, and one claim that is so poor as to be completely unpersuasive (in this case it has no supporting premises). Its Specificity score is 2 because it does not have a quantifier nor does it attempt to summarize the main points of the evidence. Finally, its Persuasiveness score is 3: all supporting claims rely on logos, so there is no added persuasiveness from a variety of persuasion strategies, and since the eloquence and specificity are adequate, they do not detract from the Evidence score.

5 Conclusion

We presented the first corpus of 102 persuasive student essays that are simultaneously annotated with argument trees, persuasiveness scores, and attributes of argument components that impact these scores. We believe that this corpus will push the frontiers of research in content-based essay grading by triggering the development of novel computational models concerning argument persuasiveness that could provide useful feedback to students on why their arguments are (un)persuasive in addition to how persuasive they are.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of the paper. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037.

References

Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, Francisca A. Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. In *Handbook of Argumentation Theory*. Springer, Dordrecht.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in Web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21.

Colin Higgins and Robyn Walker. 2012. Ethos, logos, pathos: Strategies of persuasion in social/environmental reports. *Accounting Forum*, 36:194-208.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *Human Language Technologies: The 2004 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 185–192.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology*. Sage commtext series. Sage, Thousand Oaks, CA.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder: Audience effects in persuasion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 742–753.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(1):25–55.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239.

Isaac Persing and Vincent Ng. 2013. Modeling thesis clarity in student essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 260–269.

Isaac Persing and Vincent Ng. 2014. Modeling prompt adherence in student essays. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1534–1543.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552.

Isaac Persing and Vincent Ng. 2017. Why can't you convince me? Modeling weaknesses in unpersuasive arguments. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4082–4088.

Mark D. Shermis and Jill Burstein. 2013. *Handbook of Automated Essay Evaluation: Current Applications and New Directions*. Routledge Chapman & Hall.

Christian Stab and Iryna Gurevych. 2014a. Annotating argument components and relations in persuasive essays. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510.

Christian Stab and Iryna Gurevych. 2014b. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Christian Stab and Iryna Gurevych. 2017a. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Christian Stab and Iryna Gurevych. 2017b. Recognizing insufficiently supported arguments in argumentative essays. In *Proceedings of the 15th Conference of the European Chapter of the Association for*

Computational Linguistics: Volume 1, Long Papers, pages 980–990.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? Ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200.

Inherent Biases in Reference-based Evaluation for Grammatical Error Correction and Text Simplification

Leshem Choshen[1] and Omri Abend[2]
[1]School of Computer Science and Engineering, [2] Department of Cognitive Sciences
The Hebrew University of Jerusalem
leshem.choshen@mail.huji.ac.il, oabend@cs.huji.ac.il

Abstract

The prevalent use of too few references for evaluating text-to-text generation is known to bias estimates of their quality (henceforth, *low coverage bias* or LCB). This paper shows that overcoming LCB in Grammatical Error Correction (GEC) evaluation cannot be attained by re-scaling or by increasing the number of references in any feasible range, contrary to previous suggestions. This is due to the long-tailed distribution of valid corrections for a sentence. Concretely, we show that LCB incentivizes GEC systems to avoid correcting even when they can generate a valid correction. Consequently, existing systems obtain comparable or superior performance compared to humans, by making few but targeted changes to the input. Similar effects on Text Simplification further support our claims.

1 Introduction

Evaluation in monolingual translation (Xu et al., 2015; Mani, 2009) and in particular in GEC (Tetreault and Chodorow, 2008; Madnani et al., 2011; Felice and Briscoe, 2015; Bryant and Ng, 2015; Napoles et al., 2015) has gained notoriety for its difficulty, due in part to the heterogeneity and size of the space of valid corrections (Chodorow et al., 2012; Dreyer and Marcu, 2012). Reference-based evaluation measures (RBM) are the common practice in GEC, including the standard M^2 (Dahlmeier and Ng, 2012), GLEU (Napoles et al., 2015) and I-measure (Felice and Briscoe, 2015).

The Low Coverage Bias (LCB) was previously discussed by Bryant and Ng (2015), who showed that inter-annotator agreement in producing references is low, and concluded that RBMs underestimate the performance of GEC systems. To address this, they proposed a new measure, Ratio Scoring, which re-scales M^2 by the inter-annotator agreement (i.e., the score of a human corrector), interpreted as an upper bound.

We claim that the LCB has more far-reaching implications than previously discussed. First, while we agree with Bryant and Ng (2015) that a human correction should receive a perfect score, we show that LCB does not merely scale system performance by a constant factor, but rather that some correction policies are less prone to be biased against. Concretely, we show that by only correcting closed class errors, where few possible corrections are valid, systems can outperform humans. Indeed, in Section 2.3 we show that some existing systems outperform humans on M^2 and GLEU, while only applying few changes to the source.

We thus argue that the development of GEC systems against low coverage RBMs disincentivizes systems from making changes to the source in cases where there are plentiful valid corrections (open class errors), as necessarily only some of them are covered by the reference set. To support our claim we show that (1) existing GEC systems under-correct, often performing an order of magnitude less corrections than a human does (§3.2); (2) increasing the number of references alleviates under-correction (§3.3); and (3) under-correction is more pronounced in error types that are more varied in their valid corrections (§3.4).

A different approach for addressing LCB was taken by (Bryant and Ng, 2015; Sakaguchi et al., 2016), who propose to increase the number of references (henceforth, M). In Section 2 we estimate the distribution of corrections per sentence, and find that increasing M is unlikely to overcome LCB, due to the vast number of valid corrections for a sentence and their long-tailed distribution. Indeed, even short sentences have over 1000 valid corrections on average. Empirically assessing the effect of increasing M on the bias, we find diminishing returns using three standard GEC measures (M^2, accuracy and GLEU), underscoring the difficulty in this approach.

Similar trends are found when conducting such experiments to Text Simplification (TS) (§4). Specifically we show that (1) the distribution of valid simplifications for a given sentence is long-tailed; (2) common measures for TS dramatically under-estimate performance; (3) additional references alleviate this under-prediction.

To recap, we find that the LCB hinders the reliability of RBMs for GEC, and incentivizes systems developed to optimize these measures not to correct. LCB cannot be overcome by re-scaling or increasing M in any feasible range.

2 Coverage in RBMs

We begin by formulating a methodology for studying the distribution of valid corrections for a sentence (§2.1), and then turn to assessing the effect inadequate coverage has on common RBMs (§2.2). Finally, we compare human and system scores by common RBMs (§2.3).

Notation. We assume each ungrammatical sentence x has a set of valid corrections $Correct_x$, and a discrete distribution $\mathcal{D}_x$ over them, where $P_{\mathcal{D}_x}(y)$ for $y \in Correct_x$ is the probability a human annotator would correct x as y.

Let $X = x_1 \ldots x_N$ be the evaluated set of source sentences and denote $\mathcal{D}_i := \mathcal{D}_{x_i}$. Each x_i is independently sampled from some distribution $\mathcal{L}$ over input sentences, and is paired with M corrections $Y_i = \left\{y_i^1, \ldots, y_i^M\right\}$, which are independently sampled from $\mathcal{D}_i$. Our analysis assumes a fixed number of references across sentences, but generalizing to sentence-dependent M is straightforward. The *coverage* of a reference set Y_i of size M for a sentence x_i is defined as $P_{y \sim \mathcal{D}_i}(y \in Y_i)$.

A system C is a function from input sentences to proposed corrections (strings). An evaluation measure is a function $f\colon X \times Y \times C \to \mathbb{R}$. We use the term "true measure" to refer to a measure's output where the reference set includes all valid corrections, i.e., $\forall i\colon Y_i = Correct_i$.

Experimental Setup. We conduct all experiments on the NUCLE test dataset (Dahlmeier et al., 2013). NUCLE is a parallel corpus of essays written by language learners and their corrected versions, containing 1414 essays and 50 test essays, each of about 500 words.

We evaluate all participating systems in the CoNLL 2014 shared task, in addition to three of the best performing systems on this dataset, a hybrid system (Rozovskaya and Roth, 2016), a phrase-based MT system (Junczys-Dowmunt and Grundkiewicz, 2016) and a neural network system (Xie et al., 2016). Appendix A lists system names and abbreviations.

2.1 Estimating the Corrections Distribution

Data. We turn to estimating the number of corrections per sentence, and their histogram. The experiments in the following section are run on a random sample of 52 short sentences from the NUCLE test data, i.e. with 15 words or less. Through the length restriction, we avoid introducing too many independent errors that may drastically increase the number of annotation variants (as every combination of corrections for these errors is possible), thus resulting in unreliable estimation for $\mathcal{D}_x$.

Proven effective in GEC and related tasks such as MT (Zaidan and Callison-Burch, 2011; Madnani et al., 2011; Post et al., 2012), we use crowdsourcing to sample from $\mathcal{D}_x$ (see Appendix B). Aiming to judge grammaticality rather than fluency, we instructed the workers to correct only when necessary, not for styling. We begin by estimating the histogram of $\mathcal{D}_x$ for each sentence, using the crowdsourced corrections. We use UNSEENEST (Zou et al., 2016), a non-parametric algorithm to estimate a discrete distribution in which the individual values do not matter, only their probability. UNSEENEST aims to minimize the "earthmover distance", between the estimated histogram and the histogram of the distribution. Intuitively, if histograms are piles of dirt, UNSEENEST minimizes the amount of dirt moved times the distance it moved. UNSEENEST was originally developed and tested for estimating the histogram of variants a gene may have, including undiscovered ones, a setting similar to ours. Our manual tests of UNSEENEST with small artificially created datasets showed satisfactory results.[1]

[1]An implementation of UNSEENEST, the data we collected, the estimated distributions and efficient implementations of computations with Poisson binomial distributions can be found in `https://github.com/borgr/IBGEC`.

Our estimates show that most input sentences have a large number of infrequent corrections that account for much of the probability mass and a rather small number of frequent corrections. Table 1 presents the mean number of different corrections with frequency at least γ (for different γs), and their total probability mass. For instance, 74.34 corrections account for 75% of the probability mass, each occurring with frequency $\geq 0.1\%$.

	Frequency Threshold (γ)			
	0	0.001	0.01	0.1
Variants	1351.24	74.34	8.72	1.35
Mass	1	0.75	0.58	0.37

Table 1: Estimating the distribution of corrections $\mathcal{D}_x$. The table presents the mean number of corrections per sentence with probability more than γ (top row), as well as their total probability mass (bottom row).

The high number of rare corrections raises the question of whether these can be regarded as noise. To test this we conducted another crowd-sourcing experiment, where 3 annotators were asked to judge whether a correction produced in the first experiment, is indeed valid. We plot the validity of corrections against their frequencies, finding that frequency has little effect, where even the rarest corrections are judged valid 78% of the time. Details in Appendix C.

2.2 Under-estimation as a Function of M

After estimating the histogram of valid corrections for a sentence, we turn to estimating the resulting bias (LCB), for different M values. We study sentence-level accuracy, F-Score and GLEU.

Sentence-level Accuracy. Sentence-level accuracy is the percentage of corrections that exactly match one of the references. Accuracy is a basic, interpretable measure, used in GEC by, e.g., Rozovskaya and Roth (2010). It is also closely related to the 0-1 loss function commonly used for training in GEC (Chodorow et al., 2012; Rozovskaya and Roth, 2013).

Formally, given test sentences $X = \{x_1, \ldots, x_N\}$, their references $Y_1, \ldots, Y_N$ and a system C, we define C's accuracy to be

$$Acc\,(C; X, Y) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{C(x_i)\in Y_i}. \tag{1}$$

Note that C's accuracy is, in fact, an estimate of C's *true accuracy*, the probability to produce a valid correction for a sentence. Formally:

$$TrueAcc\,(C) = P_{x\sim L}\left(C\,(x) \in Correct_x\right). \tag{2}$$

The bias of $Acc\,(C; X, Y)$ for a sample of N sentences, each paired with M references is then

$$\begin{aligned} &TrueAcc\,(C) - \mathbb{E}_{X,Y}\left[Acc\,(C; X, Y)\right] = && (3)\\ &TrueAcc\,(C) - P\left(C\,(x) \in Y\right) = && (4)\\ &P\left(C\,(x) \in Correct_x\right) \cdot && (5)\\ &\left(1 - P\left(C\,(x) \in Y | C\,(x) \in Correct_x\right)\right) && (6) \end{aligned}$$

We observe that the bias, denoted b_M, is not affected by N, only by M. As M grows, Y better approximates $Correct_x$, and b_M tends to 0.

In order to abstract away from the idiosyncrasies of specific systems, we consider an idealized learner, which, when correct, produces a valid correction with the same distribution as a human annotator (i.e., according to $\mathcal{D}_x$). Formally, we assume that, if $C(x) \in Correct_x$ then $C(x) \sim \mathcal{D}_x$. Hence the bias b_M (Eq. 6) can be re-written as

$$P(C(x) \in Correct_x) \cdot (1 - P_{Y\sim\mathcal{D}_x^M, y\sim\mathcal{D}_x}(y \in Y)).$$

We will henceforth assume that C is perfect (i.e., its *true accuracy* is 1). Note that assuming any other value for C's *true accuracy* would simply scale b_M by that accuracy. Similarly, assuming only a fraction p of the sentences require correction scales b_M by p.

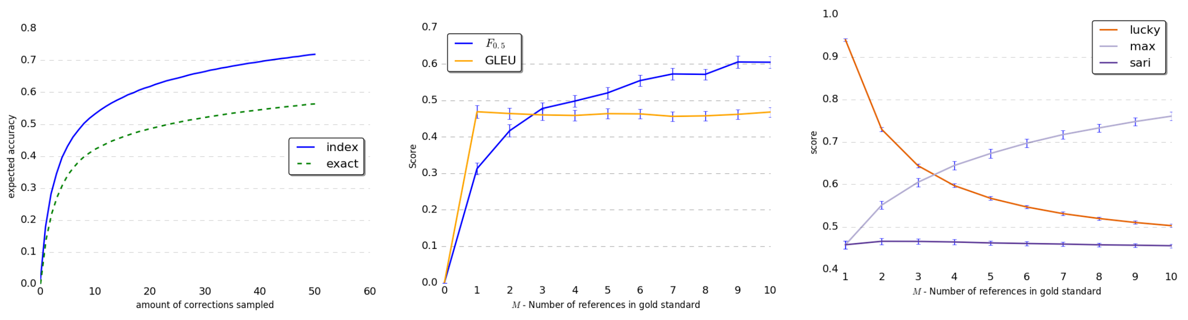
We estimate b_M empirically using its empirical mean on our experimental corpus:

$$\hat{b}_M = 1 - \frac{1}{N}\sum_{i=1}^{N} P_{Y\sim\mathcal{D}_i^M, y\sim\mathcal{D}_i}\left(y \in Y\right).$$

Using the UNSEENEST estimations of $\mathcal{D}_i$, we can compute $\hat{b}_M$ for any size of Y_i (M). However, as this is highly computationally demanding, we estimate it using sampling. Specifically, for every $M = 1, ..., 20$ and x_i, we sample Y_i 1000 times (with replacement), and estimate $P\left(y \in Y_i\right)$ as the covered probability mass $P_{\mathcal{D}_i}\{y : y \in Y_i\}$. Based on that we compute the accuracy distribution and expectation (see Appendix D).

We repeated all our experiments where Y_i is sampled without replacement, and find similar trends with a faster increase in accuracy reaching over 0.47 with $M = 10$.

Figure 1a presents the expected accuracy of a perfect system (i.e., 1-$\hat{b}_M$) for different Ms. Results show that even for M values which are much larger than the standard (e.g., $M = 20$), expected



(a) Accuracy and Exact Index Match.

(b) $F_{0.5}$ and GLEU

(c) (lucky) perfect SARI and MAX-SARI

Figure 1: The score obtained by perfect systems according to GEC accuracy (1a), GEC F-score and GLEU (1b). Figure 1c reports TS experimental results, namely the score of a perfect and lucky perfect system using SARI, and a perfect system using MAX-SARI. The y-axis corresponds to the measure values, and the x-axis to the number of references M. For bootstrapping experiments points are paired with a confidence interval ($p = .95$).

accuracy is only around 0.5. As M increases, the contribution of each additional correction diminishes sharply (the slope is 0.004 for $M = 20$).

We also experiment with a more relaxed measure, *Exact Index Match*, which is only sensitive to the identity of the changed words and not to what they were changed to. Formally, two corrections c and c' over a source sentence x match if for their word alignments with the source (computed as above) $a : \{1, ..., |x|\} \rightarrow \{1, ..., |c|, Null\}$ and $a' : \{1, ..., |x|\} \rightarrow \{1, ..., |c'|, Null\}$, it holds that $c_{a(i)} \neq x_i \Leftrightarrow c'_{a'(i)} \neq x_i$, where $c_{Null} = c'_{Null}$. Results, while somewhat higher, are still only 0.54 with $M = 10$. (Figure 1a)

F-Score. While accuracy is commonly used as a loss function for training GEC systems, F_α-score is standard for evaluating system performance. The score is computed in terms of *edit* overlap between edits that constitute a correction and ones that constitute a reference, where edits are substring replacements to the source. We use the standard M^2 scorer (Dahlmeier and Ng, 2012), which defines edits optimistically, maximizing over all possible annotations that generate the correction from the source. Since our crowdsourced corrections are not annotated for edits, we produce edits to the reference heuristically.

The complexity of the measure prohibits an analytic approach (Yeh, 2000). We instead use bootstrapping to estimate the bias incurred by not being able to exhaustively enumerate the set of valid corrections. As with accuracy, in order to avoid confounding our results with system-specific biases, we assume the evaluated system is perfect and sample its corrections from the human distribution of corrections $\mathcal{D}_x$.

Concretely, given a value for M and for N, we uniformly sample from our experimental corpus source sentences $x_1, ..., x_N$, and M corrections for each $Y_1, ..., Y_N$ (with replacement). Setting a realistic value for N in our experiments is important for obtaining comparable results to those obtained on the NUCLE corpus (see §2.3), as the expected value of F-score depends on N and the number of sentences that do not need correction (N_{cor}). Following the statistics of NUCLE's test set, we set $N = 1312$ and $N_{cor} = 136$.

Bootstrapping is carried out by the accelerated bootstrap procedure (Efron, 1987), with 1000 iterations. We also report confidence intervals ($p = .95$), computed using the same procedure.

Results (Figure 1b) again show the insufficiency of commonly-used M values for reliably estimating system performance. For instance, the $F_{0.5}$-score for our perfect system is only 0.42 with $M = 2$. The saturation effect, observed for accuracy, is even more pronounced in this setting.

GLEU. We repeat the procedure using the mean GLEU sentence score (Figure 1b), which was shown to better correlate with human judgments than M^2 (Napoles et al., 2016). Results are about 2% higher than M^2's with a similar saturation effect. Sakaguchi et al. (2016) observed a similar effect when evaluating against fluency-oriented references; this has led them to assume that saturation is due to covering most of the probability mass, which we now show is not the case.[2]

[2] We do not experiment with I-measure (Felice and Briscoe, 2015), as its run-time is prohibitively high for experimenting with bootstrapping that requires many applications of the measure (Choshen and Abend, 2018a), and as empirical validation studies showed that it has a low correlation with human judgments (Sakaguchi et al., 2016).

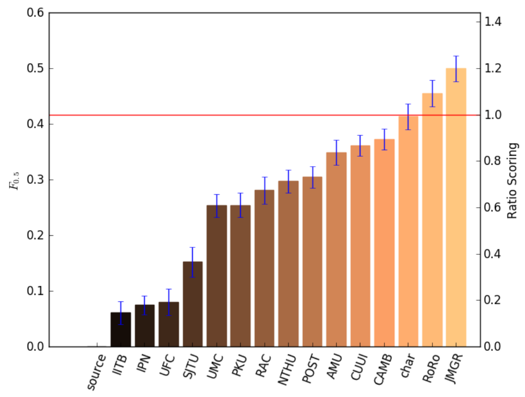


Figure 2: $F_{0.5}$ values with $M = 2$ for different systems, including confidence interval ($p = .95$). The left-most column ("source") presents the F-score of a system that doesn't make any changes to the source sentences. In red is human performance. See §2 for a legend of the systems.

2.3 Human and System Performance

The bootstrapping method for computing the significance of the F-score (§2.2) can also be used for assessing the significance of the differences in system performance reported in the literature. We compute confidence intervals of different systems on the NUCLE test data ($M = 2$).

Results (Figure 2) present mixed trends: some differences between previously reported F-scores are indeed significant and some are not. For example, the best performing system is significantly better than all but the second one.

Considering the F-score of the best-performing systems, and comparing them to the F-score of a perfect system with $M = 2$ (in accordance with systems' reported results), we find that their scores are comparable, where the systems RoRo and JMGR surpass a perfect system's F-score. Similar experiments with GLEU show that the two systems obtain comparable or superior performance to humans on this measure as well.

2.4 Discussion

In this section we have established that (1) as systems can surpass human performance on RBMs, re-scaling cannot be used to overcome the LCB, and that (2) as the distribution of valid corrections is long-tailed, the number of references needed for reliable RBMs is exceedingly high. Indeed, an average sentence has hundreds or more valid low-probability corrections, whose total probability mass is substantial. Our analysis with Exact Index Match suggests that similar effects are applicable to Grammatical Error Detection as well. The proposal of Sakaguchi et al. (2016), to emphasize fluency over grammaticality in reference corrections, only compounds this problem, as it results in a larger number of valid corrections.

3 Implications of the LCB

We discuss the adverse effects of LCB not only on the reliability of RBMs, but on the development of GEC systems. We argue that evaluation with inadequate reference coverage incentivizes systems to under-correct, and to mostly target errors that have few valid corrections (closed-class). We first show that low coverage can lead to under-correction (§3.1), then show that modern systems make far fewer corrections to the source, compared to humans (§3.2). §3.3 shows that increasing the number of references can alleviate this effect. §3.4 shows that open-class errors are more likely to be under-corrected than closed-class ones.

3.1 Motivating Analysis

For simplicity, we abstract away from the details of the learning model and assume that systems attempt to maximize an objective function, over some training or development data. We assume maximization is achieved by iterating over the samples, as with the Perceptron or SGD.

Assume the system is faced with a phrase it predicts to be ungrammatical. Assume p_{detect} is the probability this prediction is correct, and $p_{correct}$ is the probability it is able to predict a valid correction for this phrase (including correctly identifying it as erroneous). Finally, assume evaluation is against M references with coverage $p_{coverage}$ (the probability that a valid correction will be found among M randomly sampled references).

We will now assume that the system may either choose to correct with the correction it finds the most likely or not at all. If it chooses not to correct, its probability of being rewarded (i.e., its output is in the reference set) is $(1 - p_{detect})$. Otherwise, its probability of being rewarded is $p_{correct} \cdot p_{coverage}$. A system is disincentivized from altering the phrase in cases where:

$$p_{correct} \cdot p_{coverage} < 1 - p_{detect} \quad (7)$$

We expect Condition (7) to frequently hold in cases that require non-trivial changes, which are characterized both by low $p_{coverage}$ (as non-trivial changes are often open-class), and by lower system performance.

Corrector	Sentence
Source	This is especially to people who are overseas.
CHAR, UMC, JMGR	This is especially **for** people who are overseas.
IPN	This is especially to **peoples** who are overseas.
CUUI	This is especially to **the** people who are overseas.
NUCLE$_A$	This is especially **true for** people who are overseas.
NUCLE$_B$	This is especially **relevant** to people who are overseas.

Table 2: Example for a sentence and proposed corrections by different systems (top part) and by the two NUCLE annotators (bottom part). Systems not mentioned in the table retain the source. No system produces a new word as needed. The two references differ in their corrections.

Precision-oriented measures (e.g., $F_{0.5}$) penalize invalidly correcting more harshly than not correcting an ungrammatical sentence. In these cases, Condition (7) should be written as

$$p_{correct} \cdot p_{coverage} - (1 - p_{correct} \cdot p_{coverage})\,\alpha < 1 - p_{detect}$$

where α is the ratio between the penalty for introducing a wrong correction and the reward for a valid correction. The condition is even more likely to hold with such measures.

3.2 Under-correction in GEC Systems

In this section we compare the prevalence of changes made to the source by the systems, to their prevalence in the NUCLE references. To strengthen our claim, we exclude all non-alphanumeric characters, both within tokens or as separate tokens. See Table 2 for an example.

We consider three types of divergences between the source and the reference. First, we measure the extent to which *words* were changed: altered, deleted or added. To do so, we compute word alignment between the source and the reference, casting it as a weighted bipartite matching problem. Edge weights are assigned to be the token edit distances.[3] Following word alignment, we define WORDCHANGE as the number of aligned words and unaligned words changed. Second, we quantify word *order* differences using Spearman's ρ between the order of the words in the source sentence and the order of their corresponding-aligned words in the correction. $\rho = 0$ where the word order is uncorrelated, and $\rho = 1$ where the orders exactly match. We report the average ρ over all source sentence pairs. Third, we report how many source sentences were split and how many concatenated by the reference and by the systems. One annotator was arbitrarily selected for the figures.

Results. Results (Figure 3) show that humans make considerably more changes than systems according to all measures of under-correction, both in terms of the number of sentences modified and the number of modifications within them. Differences are often an order of magnitude large. For example, 36 reference sentences include 6 word changes, where the maximal number of sentences with 6 word changes by any system is 5. We find similar trends on the references of the TreeBank of Learner English (Yannakoudakis et al., 2011).

3.3 Higher M Alleviates Under-correction

This section reports an experiment for determining whether increasing the number of references in training indeed reduces under-correction. There is no corpus available with multiple references which is large enough for re-training a system. Instead, we simulate such a setting with an oracle reranking approach, and test whether the availability of increasingly more training references reduces a system's under-correction.

Concretely, given a set of sentences, each paired with M references, a measure and a system's k-best list, we define an oracle re-ranker that selects for each sentence the highest scoring correction. As a test case, we use the RoRo system with $k = 100$, and apply it to the largest available language learner corpus which is paired with a substantial amount of GEC references, namely the NUCLE test corpus. We use the standard F-score as the evaluation measure, examining the under-correction of the oracle re-ranker for different M values, averaging over the 1312 samples of M references from the available set of ten references provided by Bryant and Ng (2015).

As the argument is not trivial, we turn to explaining why decreased under-correction with an increase in M indicates that tuning against a small set of references (low coverage) yields under-correction. Assume an input sentence with some sub-string e. There are three cases: (1) e is an error, (2) e is valid but there are valid references that alter it, (3) e is uniquely valid. In case (3) or-

[3]Aligning words in GEC is much simpler than in MT, as most of the words are unchanged, deleted fully, added, or changed slightly.

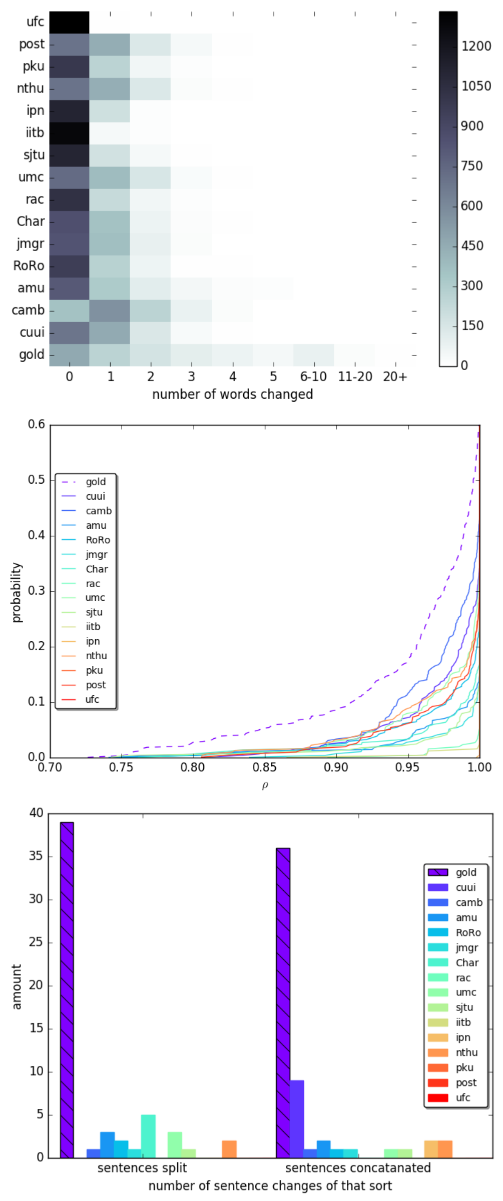


Figure 3: The prevalence of changes in system outputs and in the NUCLE reference. The top figure presents the number of sentences (heat) for each amount of word changes (x-axis; measured by WORDCHANGE) done by the outputs and the reference (y-axis). The middle figure presents the percentage of sentence pairs (y-axis) where the Spearman ρ values do not exceed a certain threshold (x-axis). The bottom figure presents the counts of source sentences (y-axis) concatenated (right bars) or split (left bars) by the references (striped column) and the outputs (coloured columns). See Appendix A for a legend of the systems. Under all measures, the gold standard references make substantially more changes to the source sentences than any of the systems, in some cases an order of magnitude more.

	L_{val} empty	L_{val} not empty	
		e valid	e error
Small M	0	$P_Y(\overline{e}, L_{val})$	$P_Y(L_{val})$
Large M	0	0	1
Correction Rate	=	$\downarrow$	$\uparrow$

Table 3: The expected effect of oracle re-ranking on under-correction. Values represent the probability of altering a substring of the input e, which is a proxy to the expected correction rate. L_{val} is the valid alterations in the k-best list. $P_Y(L_{val})$ is the probability that a valid correction from the list is also in the reference set Y, $P_Y(\overline{e}, L_{val})$ is the probability that, in addition, the reference that keeps e is not in Y. When M increases, the expected correction rate is expected to increase only if e is an error and a valid correction of it is found in the k-best list.

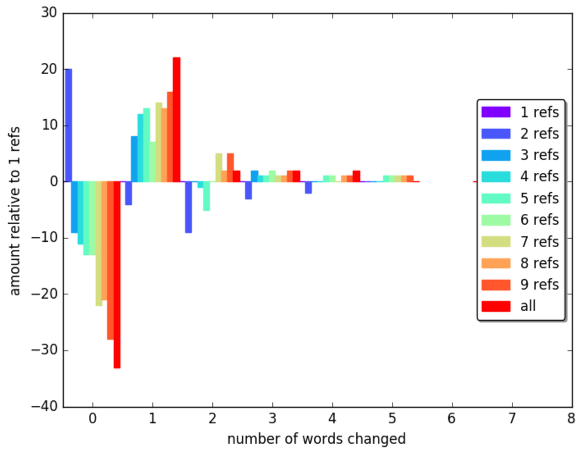


Figure 4: The amount of sentences (y-axis) with a given number of words changed (x-axis) following oracle reranking with different M values (column colors), where the amount for $M = 1$ is subtracted from them. All references are randomly sampled except the "all" column that contains all ten references. In conclusion, tuning against additional references indeed reduces under-correction.

acle re-ranking has no effect and can be ignored. The corrections in the k-best list can then be partitioned to those that keep e as it is; those that invalidly alter e; and those that validly alter e.

Table 3 presents the probability that e will be altered in the different cases. Analysis shows that under-correction is likely to decrease with M only in the case where e is an error and the k-best list contains a valid correction of it. Whenever the reference allows both keeping e and altering e, the re-ranker selects keeping e.

Indeed, our experimental results show that word changes increase with M (Figure 4), indicating that low coverage may play a role in the observed tendency of GEC systems to under-correct. No significant difference is found for word order.

3.4 Under-correction by Error Types

In this section we study the prevalence of under-correction according to edit types, finding that open-class types of errors (such as replacing a word with another word) are more starkly under-corrected, than closed-class errors. Evaluating with low coverage RBMs does not incentivize systems to address open-class errors (in fact, it disincentivizes them to). Therefore, even if LCB is not the cause for this trend, current evaluation procedures may perpetuate it.

We use the data of Bryant et al. (2017), which automatically assigned types to each edit in the output of all CoNLL 2014 systems on the NUCLE test set. As a measure of under-correction tendency, we take the ratio between the mean number of corrections produced by the systems and by the references. We note that this analysis does not consider whether the predicted correction is valid or not, but only how many of the errors of each type the systems attempted to correct.

We find that all edit types are under-predicted on average, but that the least under-predicted ones are mostly closed-class types. Concretely, the top quarter of error types consists of orthographical errors, plurality inflection of nouns, adjective inflections to superlative or comparative forms and determiner selection. The bottom quarter includes the categories verb selection, noun selection, particle/preposition selection, pronoun selection, and the type OTHER, which is a residual category. The only exception to this regularity is the closed-class punctuation selection type, which is found in the lower quarter. See Appendix E.

This trend cannot be explained by assuming that common error types are targeted more. Indeed, error type frequency is slightly negatively correlated with the under-correction ratio (ρ=-0.29 p-value=0.16). A more probable account of this effect is the disincentive of GEC systems to correct open-class error types, for which even valid corrections are unlikely to be rewarded.

4 Similar Effects on Simplification

We now turn to replicating our experiments on Text Simplification (TS). From a formal point of view, evaluation of the tasks is similar: the output is obtained by making zero or more edits to the source. RBMs are the standard for TS evaluation, much like they are in GEC.

Our experiments on TS demonstrate that similar trends recur in this setting as well. The tendency of TS systems to under-predict changes to the source has already been observed by previous work (Alva-Manchego et al., 2017), showing that TS systems under-predict word additions, deletions, substitutions, and sequence shifts (Zhang and Lapata, 2017), and have low edit distance from the source (Narayan and Gardent, 2016). Our experiments show that LCB may account for this under-prediction. Concretely, we show that (1) the distribution of valid references for a given sentence is long-tailed; (2) common evaluation measures suffer from LCB, taking SARI (Xu et al., 2016) as an example RBM (similar trends are obtained with Accuracy); (3) under-prediction is alleviated with M in oracle re-ranking experiments.

We crowd-sourced 2500 reference simplifications for 47 sentences, using the corpus and the annotation protocol of Xu et al. (2016), and applying UNSEENEST to estimate $\mathcal{D}_x$ (Appendix B). Table 4 shows that the expected number of references is even greater in this setting.

Assessing the effect of M on SARI, we find that SARI diverges from Accuracy and F-score in that its multi-reference version is not a maximum over the single-reference scores, but some combination of them. This can potentially increase coverage, but it also leads to an unintuitive situation: an output identical to a reference does not receive a perfect score, but rather the score depends on how similar the output is to the other references. A more in-depth analysis of SARI's handling of multiple references is found in Appendix F. In order to neutralize this effect of SARI, we also report results with MAX-SARI, which coincides with SARI on $M = 1$, and is defined as the maximum single-reference SARI score for $M > 1$.

Figure 1c presents the coverage of SARI and MAX-SARI of a perfect TS system that selects a random correction from the estimated distribution of corrections using the same bootstrapping protocol as in §2.1. We also include the SARI score of a "lucky perfect" system, that randomly selects one of the given references (the MAX-SARI score for such a system is 1). Results show that SARI has a coverage of about 0.45, and that this score is largely independent of M. The score of predicting one of the available references drops with the number of references, indicating that SARI scores may not be comparable across different M values.

We therefore restrict oracle re-ranking experi-

	Frequency Threshold (γ)			
	0	0.001	0.01	0.1
Variants	2636.29	111.19	4.68	0.13
Mass	1	0.42	0.14	0.02

Table 4: Estimating the distribution of simplifications $\mathcal{D}_x$. The table presents the mean number of simplifications per sentence with probability more than γ (top row), as well as their total probability mass (bottom row).

ments to MAX-SARI, conducting re-ranking experiments on k-best lists in two settings: Moses (Koehn et al., 2007) with $k = 100$, and a neural model (Nisioi et al., 2017) with $k = 12$. Our results indeed show that under-prediction is alleviated with M in both settings. For example, the least under-predicting model (the neural one) did not change 50 sentences with $M = 1$, but only 29 weren't changed with $M = 8$. See Appendix G.

5 Conclusion

We argue that using low-coverage reference sets has adverse effects on the reliability of reference-based evaluation, with GEC and TS as a test case, and consequently on the incentives offered to systems. We further argue that these effects cannot be overcome by re-scaling or increasing the number of references in a feasible way. The paper makes two methodological contributions to the monolingual translation evaluation literature: (1) a methodology for evaluating evaluation measures by the scores they assign a perfect system, using a bootstrapping procedure; (2) a methodology for assessing the distribution of valid monolingual translations. Our findings demonstrate how these tools can help characterize the biases of existing systems and evaluation measures. We believe our findings and methodologies can be useful for similar tasks such as style conversion and automatic post-editing of raw MT outputs.

We note that the LCB further jeopardizes the reliability of common validation experiments for RBMs, that assess the correlation between human and measure rankings of system outputs (Grundkiewicz et al., 2015). Indeed, if outputs all similarly under-correct, correlation studies will not be affected by whether an RBM is sensitive to under-correction. Therefore, the tendency of RBMs to reward under-correction cannot be detected by such correlation experiments (cf. Choshen and Abend, 2018a).

Our results underscore the importance of developing alternative evaluation measures that transcend n-gram overlap, and use deeper analysis tools, e.g., by comparing the semantics of the reference and the source to the output (cf. Lo and Wu, 2011). Napoles et al. (2016) have made progress towards this goal in proposing a reference-less grammaticality measure, using Grammatical Error Detection tools, as did Asano et al. (2017), who added a fluency measure to the grammaticality. In a recent project (Choshen and Abend, 2018b), we proposed a complementary measure that measures the semantic faithfulness of the output to the source, in order to form a combined semantic measure that bypasses the pitfalls of low coverage.

Acknowledgments

This work was supported by the Israel Science Foundation (grant No. 929/17), and by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. We thank Nathan Schneider, Courtney Napoles and Joel Tetreault for helpful feedback.

References

Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hiroki Asano, Tomoya Mizumoto, and Kentaro Inui. 2017. Reference-based metrics can be replaced with reference-less metrics in evaluating grammatical error correction systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 343–348.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *ACL (1)*, pages 697–707.

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel R Tetreault. 2012. Problems in evaluating grammatical error detection systems. In *COLING*, pages 611–628. Citeseer.

Leshem Choshen and Omri Abend. 2018a. Automatic metric validation for grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Leshem Choshen and Omri Abend. 2018b. Reference-less measure of faithfulness for grammatical error correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171. Association for Computational Linguistics.

Bradley Efron. 1987. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 82(397):171–185.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *HLT-NAACL*, pages 578–587.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, Edward Gillian, et al. 2015. Human evaluation of grammatical error correction systems. In *EMNLP*, pages 461–470.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Chi-kiu Lo and Dekai Wu. 2011. Meant: an inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility via semantic frames. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 220–229. Association for Computational Linguistics.

Nitin Madnani, Joel Tetreault, Martin Chodorow, and Alla Rozovskaya. 2011. They can help: Using crowdsourcing to improve the evaluation of grammatical error detection systems. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 508–513. Association for Computational Linguistics.

Inderjeet Mani. 2009. Summarization evaluation: an overview. In *Proceedings of the NTCIR Workshop*, volume 2.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2109–2115. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proceedings of the 9th International Natural Language Generation conference*, pages 111–120, Edinburgh, UK. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 85–91.

Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2010. Annotating esl errors: Challenges and rewards. In *Proceedings of the NAACL HLT 2010 fifth workshop on innovative use of NLP for building educational applications*, pages 28–36. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2013. Joint learning and inference for grammatical error correction. *Urbana*, 51:61801.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proc. of ACL*, pages 2205–2215.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Joel R Tetreault and Martin Chodorow. 2008. Native judgments of non-native usage: Experiments in preposition error detection. In *Proceedings of the Workshop on Human Judgements in Computational Linguistics*, pages 24–32. Association for Computational Linguistics.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 947–953. Association for Computational Linguistics.

Omar F Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1220–1229. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

James Zou, Gregory Valiant, Paul Valiant, Konrad Karczewski, Siu On Chan, Kaitlin Samocha, Monkol Lek, Shamil Sunyaev, Mark Daly, and Daniel G MacArthur. 2016. Quantifying unobserved protein-coding variants in human populations provides a roadmap for large-scale sequencing projects. *Nature Communications*, 7.

The price of debiasing automatic metrics in natural language evaluation

Arun Tejasvi Chaganty[*] and **Stephen Mussmann**[*] and **Percy Liang**
Computer Science Department, Stanford University
{chaganty,mussmann,pliang}@cs.stanford.edu

Abstract

For evaluating generation systems, automatic metrics such as BLEU cost nothing to run but have been shown to correlate poorly with human judgment, leading to systematic bias against certain model improvements. On the other hand, averaging human judgments, the unbiased gold standard, is often too expensive. In this paper, we use control variates to combine automatic metrics with human evaluation to obtain an unbiased estimator with lower cost than human evaluation alone. In practice, however, we obtain only a 7–13% cost reduction on evaluating summarization and open-response question answering systems. We then prove that our estimator is optimal: there is no unbiased estimator with lower cost. Our theory further highlights the two fundamental bottlenecks—the automatic metric and the prompt shown to human evaluators—both of which need to be improved to obtain greater cost savings.

1 Introduction

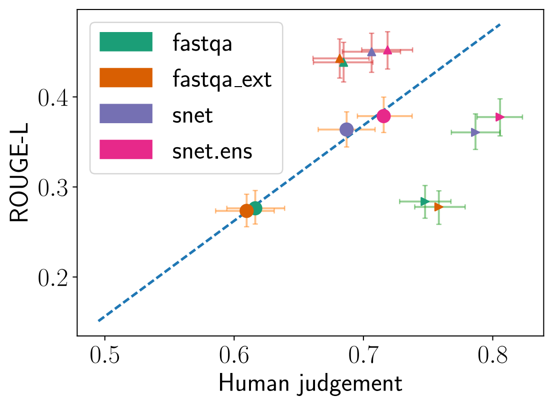
In recent years, there has been an increasing interest in tasks that require generating natural language, including abstractive summarization (Nallapati et al., 2016), open-response question answering (Nguyen et al., 2016; Kočisky et al., 2017), image captioning (Lin et al., 2014), and open-domain dialogue (Lowe et al., 2017b). Unfortunately, the evaluation of these systems remains a thorny issue because of the diversity of possible correct responses. As the gold standard of performing human evaluation is often too expensive, there has been a large effort developing automatic metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin and Rey, 2004), METEOR (Lavie and Denkowski, 2009; Denkowski and Lavie, 2014) and CiDER (Vedantam et al., 2015). However, these have shown to be biased, correlating poorly with human metrics across different datasets and systems (Liu et al., 2016b; Novikova et al., 2017).

Can we combine automatic metrics and human evaluation to obtain an *unbiased* estimate at *lower cost* than human evaluation alone? In this paper, we propose a simple estimator based on control variates (Ripley, 2009), where we average differences between human judgments and automatic metrics rather than averaging the human judgments alone. Provided the two are correlated, our estimator will have lower variance and thus reduce cost.

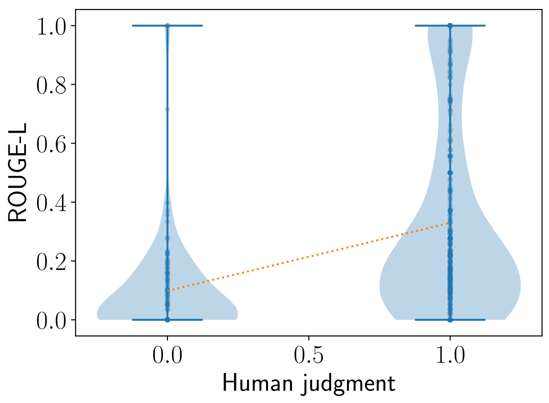
We prove that our estimator is *optimal* in the sense that no unbiased estimator using the same automatic metric can have lower variance. We also analyze its data efficiency (equivalently, cost savings)—the factor reduction in number of human judgments needed to obtain the same accuracy versus naive human evaluation—and show that it depends solely on two factors: (a) the annotator variance (which is a function of the human evaluation prompt) and (b) the correlation between human judgments and the automatic metric. This factorization allows us to calculate typical and best-case data efficiencies and accordingly refine the evaluation prompt or automatic metric.

Finally, we evaluate our estimator on state-of-the-art systems from two tasks, summarization on the CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016) and open-response question answering on the MS MARCOv1.0 dataset (Nguyen et al., 2016). To study our estimators offline, we preemptively collected 10,000 human judgments which cover several

[*]Authors contributed equally.



(a) System-level correlation on the MS MARCO task



(b) Instance-level correlation for the `fastqa` system

Figure 1: (a) At a system-level, automatic metrics (ROUGE-L) and human judgment correlate well, but (b) the instance-level correlation plot (where each point is a system prediction) shows that the instance-level correlation is quite low ($\rho = 0.31$). As a consequence, if we try to locally improve systems to produce better answers ($\triangleright$ in (a)), they do not significantly improve ROUGE scores and vice versa ($\triangle$).

tasks and systems.[1] As predicted by the theory, we find that the data efficiency depends not only on the correlation between the human and automatic metrics, but also on the evaluation prompt. If the automatic metric had perfect correlation, our data efficiency would be around 3, while if we had noiseless human judgments, our data efficiency would be about 1.5. In reality, the reduction in cost we obtained was only about 10%, suggesting that improvements in both automatic metric and evaluation prompt are needed. As one case study in improving the latter, we show that, when compared to a Likert survey, measuring the amount of post-editing needed to fix a generated sentence reduced the annotator variance by three-fold.

2 Bias in automatic evaluation

It is well understood that current automatic metrics tend to correlate poorly with human judgment at the instance-level. For example, Novikova et al. (2017) report correlations less than 0.3 for a large suite of word-based and grammar-based evaluation methods on a generation task. Similarly, Liu et al. (2016b) find correlations less than 0.35 for automatic metrics on a dialog generation task in one domain, but find correlations with the same metric dropped significantly to less than 0.16 when used in another domain. Still, somewhat surprisingly, several automatic metrics have been found to have high *system-level* correlations (Novikova et al., 2017). What, then, are the implications of having a low instance-level correlation?

As a case study, consider the task of open-response question answering: here, a system receives a human-generated question and must *generate* an answer from some given context, e.g. a document or several webpages. We collected the responses of several systems on the MS MARCOv1 dataset (Nguyen et al., 2016) and crowd-sourced human evaluations of the system output (see Section 4 for details).

The instance-level correlation (Figure 1b) is only $\rho = 0.31$. A closer look at the instance-level correlation reveals that while ROUGE is able to correctly assign low scores to bad examples (lower left), it is bad at judging good examples and often assigns them low ROUGE scores (lower right)—see Table 1 for examples. This observation agrees with a finding reported in Novikova et al. (2017) that automatic metrics correlate better with human judgments on bad examples than average or good examples.

Thus, as Figure 1(a) shows, we can improve low-scoring ROUGE examples without improving their human judgment ($\triangle$) and vice versa ($\triangleright$). Indeed, Conroy and Dang (2008) report that summarization systems were optimized for ROUGE during the DUC challenge (Dang, 2006) until they were indistinguishable from the ROUGE scores of human-generated summaries, but the systems

[1]An anonymized version of this data and the annotation interfaces used can be found at `https://bit.ly/price-of-debiasing`.

Question and reference answer	System answer (System; `Corr` / ROUGE-L)
Examples where system is correct and ROUGE-L > 0.5 *(*19.6% *or 285 of 1455 unique responses)*	
Q. what is anti-mullerian hormone **A.** Anti-Mullerian Hormone (AMH) is a protein hormone produced by granulosa cells (cells lining the egg sacs or follicles) within the ovary.	it is a protein hormone produced by granulosa cells (cells lining the egg sacs or follicles) within the ovary. (`snet.ens`; ✓ / 0.86)
Examples where system is incorrect and ROUGE-L > 0.5 *(*1.3% *or 19 of 1455 unique responses)*	
Q. at what gestational age can you feel a fetus move **A.** 37 to 41 weeks *(incorrect reference answer)*	37 to 41 weeks (`fastqa, fastqa.ext`; × / 1.0)
Examples where system is correct and ROUGE-L < 0.5 *(*56.0% *or 815 of 1455 unique responses)*	
Q. what is the definition of onomatopoeia **A.** It is defined as a word, which imitates the natural sounds of a thing.	the naming of a thing or action by a vocal imitation of the sound associated with it (as buzz, hiss). (`fastqa`; ✓ / 0.23)
Examples where system is incorrect and ROUGE-L < 0.5 *(*23.1% *or 336 of 1455 unique responses)*	
Q. what kind root stem does a dandelion have **A.** Fibrous roots and hollow stem.	vitamin a, vitamin c, vitamin d and vitamin b complex, as well as zinc, iron and potassium. (`snet`, `snet.ens`; × / 0.09)

(a) **MS MARCO.** Human annotators rated answer correctness (`AnyCorrect`) and the automatic metric used is ROUGE-L (higher is better).

Reference summary	System summary (System; `Edit` / VecSim)
Examples where system `Edit` < 0.3 *and VecSim* > 0.5 *(*53.9% *or 1078 of 2000 responses)*	
Bhullar is set to sign a ■-day contract with the Kings. The ■-year-old will become the NBA's first player of Indian descent. Bhullar will be on the roster when the Kings host New Orleans Pelicans.	~~Bhullar and~~**The** Kings are signing Bhullar to a ■-day contract. The ■-year-old will be on the roster on friday when David Wear's ■-season contract expires thursday. Bhullar is set to become the NBA's first player of Indian descent. (`ml`; 0.13 / 0.82)
Examples where system `Edit` > 0.3 *and VecSim* > 0.5 *(*18.0% *or 360 of 2000 responses)*	
The Direct Marketing Commission probing B2C Data and Data Bubble. Investigating whether they breached rules on the sale of private data. Chief commissioner described allegations made about firms as 'serious'.	~~■ Data obtained by the Mail's marketing commission said it would probe both companies over claims that they had breached the rules on the sale of private data.~~ The FSA said it would probe both companies over claims they had breached the rules on the sale of private data. (`se2seq`; 1.00 / 0.72)
Examples where system `Edit` < 0.3 *and VecSim* < 0.5 *(*14.5% *or 290 of 2000 responses)*	
Death toll rises to more than ■. Pemba Tamang, ■, shows no apparent signs of serious injury after rescue. Americans special forces helicopter ■, including ■ Americans, to safety.	~~Six of~~ **Despite** Nepal's tragedy, life triumphed in Kathmandu's hard-hit neighborhoods. Rescuers pulled an 15-year-old from the rubble of a multistory residential building. He was wearing a New York shirt and a blue neck brace. (`pointer`; 0.04 / 0.27)
Examples where system `Edit` > 0.3 *and VecSim* < 0.5 *(*13.6% *or 272 of 2000 responses)*	
"Mad Men's" final seven episodes begin airing April ■. The show has never had high ratings but is considered one of the great TV series. It's unknown what will happen to characters, but we can always guess.	'~~This's~~ "Mad Men" is the end of a series of an era', ~~This~~ **he** says. Stores have created fashion lines inspired by the show.~~"The Sopranos". The in ■ the Kent State shootings in may ■ or Richard Nixonś ■ re-election..~~ (`ml+rl`; 0.95 / 0.24)

(b) **CNN/Daily Mail.** Human judgment scores used are post-edit distance (`Edit`) (lower is better) and the automatic metric used is sentence vector similarity with the reference (higher is better).

Table 1: Examples highlighting the different modes in which the automatic metric and human judgments may agree or disagree. On the MS MARCO task, a majority of responses from systems were actually correct but poorly scored according to ROUGE-L. On the CNN/Daily Mail task, a significant number of examples which are scored highly by VecSim are poorly rated by humans, and likewise many examples scored poorly by VecSim are highly rated by humans.

had hardly improved on human evaluation. Hill-climbing on ROUGE can also lead to a system that does worse on human scores, e.g. in machine translation (Wu et al., 2016). Conversely, genuine quality improvements might not be reflected in improvements in ROUGE. This bias also appears in pool-based evaluation for knowledge base population (Chaganty et al., 2017). Thus the problems with automatic metrics clearly motivate the need for human evaluation, but can we still use the automatic metrics somehow to save costs?

3 Statistical estimation for unbiased evaluation

We will now formalize the problem of combining human evaluation with an automatic metric. Let $\mathcal{X}$ be a set of inputs (e.g., articles), and let S be the *system* (e.g. for summarization), which takes $x \in \mathcal{X}$ and returns output $S(x)$ (e.g. a summary). Let $\mathcal{Z} = \{(x, S(x)) : x \in \mathcal{X}\}$ be the set of system predictions. Let $Y(z)$ be the random variable representing the human judgment according to some evaluation prompt (e.g. grammaticality or correctness), and define $f(z) = \mathbb{E}[Y(z)]$ to be the (unknown) *human metric* corresponding to averaging over an infinite number of human judgments. Our goal is to estimate the average across all examples:

$$\mu \stackrel{\text{def}}{=} \mathbb{E}_z[f(z)] = \frac{1}{|\mathcal{Z}|}\sum_{z\in\mathcal{Z}} f(z) \qquad (1)$$

with as few queries to Y as possible.

Let g be an automatic metric (e.g. ROUGE), which maps z to a real number. We assume evaluating $g(z)$ is free. The central question is how to use g in conjunction with calls to Y to produce an unbiased estimate $\hat{\mu}$ (that is, $\mathbb{E}[\hat{\mu}] = \mu$). In this section, we will construct a simple estimator based on control variates (Ripley, 2009), and prove that it is minimax optimal.

3.1 Sample mean

We warm up with the most basic unbiased estimate, the sample mean. We sample $z^{(1)}, \dots, z^{(n)}$ independently with replacement from $\mathcal{Z}$. Then, we sample each human judgment $y^{(i)} = Y(z^{(i)})$ independently.[2] Define the estimator to be $\hat{\mu}_{\text{mean}} = \frac{1}{n}\sum_{i=1}^n y^{(i)}$. Note that $\hat{\mu}_{\text{mean}}$ is unbiased ($\mathbb{E}[\hat{\mu}_{\text{mean}}] = \mu$).

[2]Note that this independence assumption isn't quite true in practice since we do not control who annotates our data.

We can define $\sigma_f^2 \stackrel{\text{def}}{=} \text{Var}(f(z))$ as the variance of the human metric and $\sigma_a^2 \stackrel{\text{def}}{=} \mathbb{E}_z[\text{Var}(Y(z))]$ as the variance of human judgment averaged over $\mathcal{Z}$. By the law of total variance, the variance of our estimator is

$$\text{Var}(\hat{\mu}_{\text{mean}}) = \frac{1}{n}(\sigma_f^2 + \sigma_a^2). \qquad (2)$$

3.2 Control variates estimator

Now let us see how an automatic metric g can reduce variance. If there is no annotator variance ($\sigma_a^2 = 0$) so that $Y(z) = f(z)$, we should expect the variance of $f(z) - g(z)$ to be lower than the variance of $f(z)$, assuming g is correlated with f—see Figure 2 for an illustration.

The actual control variates estimator needs to handle noisy $Y(z)$ (i.e. $\sigma_a^2 > 0$) and guard against a $g(z)$ with low correlation. Let us standardize g to have zero mean and unit variance, because we have assumed it is free to evaluate. As before, let $z^{(1)}, \dots, z^{(n)}$ be independent samples from $\mathcal{Z}$ and draw $y^{(i)} = Y(z^{(i)})$ independently as well. We define the *control variates estimator* as

$$\hat{\mu}_{\text{cv}} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} - \alpha g(z^{(i)}), \qquad (3)$$

where

$$\alpha \stackrel{\text{def}}{=} \text{Cov}(f(z), g(z)). \qquad (4)$$

Intuitively, we have averaged over $y^{(i)}$ to handle the noise introduced by $Y(z)$, and scaled $g(z)$ to prevent an uncorrelated automatic metric from introducing too much noise.

An important quantity governing the quality of an automatic metric g is the correlation between $f(z)$ and $g(z)$ (recall that g has unit variance):

$$\rho \stackrel{\text{def}}{=} \frac{\alpha}{\sigma_f}. \qquad (5)$$

We can show that among all distributions with fixed σ_f^2, σ_a^2, and α (equivalently ρ), this estimator is minimax optimal, i.e. it has the least variance among all unbiased estimators:

Theorem 3.1. *Among all unbiased estimators that are functions of $y^{(i)}$ and $g(z^{(i)})$, and for all distributions with a given σ_f^2, σ_a^2, and α,*

$$\text{Var}(\hat{\mu}_{cv}) = \frac{1}{n}(\sigma_f^2(1-\rho^2) + \sigma_a^2), \qquad (6)$$

and no other estimator has a lower worst-case variance.

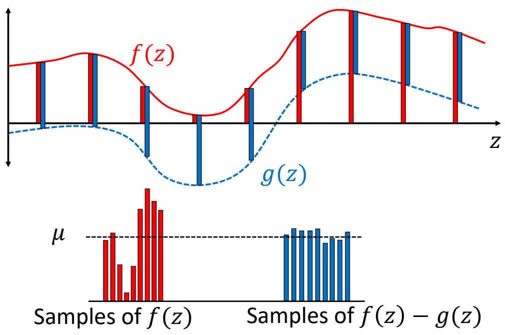


Figure 2: The samples from $f(z)$ have a higher variance than the samples from $f(z) - g(z)$ but the same mean. This is the key idea behind using control variates to reduce variance.

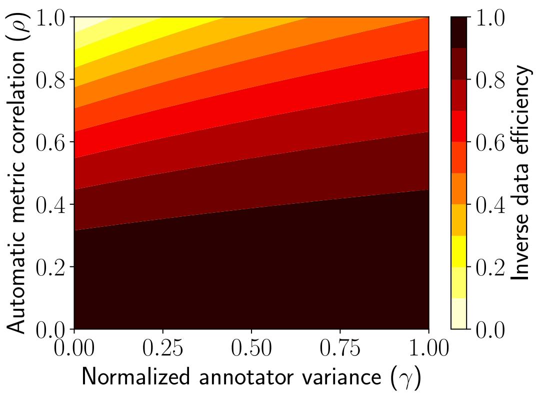


Figure 3: Inverse data efficiency for various values of γ and ρ. We need both low γ and high ρ to obtain significant gains.

Comparing the variances of the two estimators ((2) and (6)), we define the *data efficiency* as the ratio of the variances:

$$\mathrm{DE} \overset{\text{def}}{=} \frac{\mathrm{Var}(\hat{\mu}_{\text{mean}})}{\mathrm{Var}(\hat{\mu}_{\text{cv}})} = \frac{1+\gamma}{1-\rho^2+\gamma}, \qquad (7)$$

where $\gamma \overset{\text{def}}{=} \sigma_a^2/\sigma_f^2$ is the normalized annotator variance. Data efficiency is the key quantity in this paper: it is the multiplicative reduction in the number of samples required when using the control variates estimator $\hat{\mu}_{\text{cv}}$ versus the sample mean $\hat{\mu}_{\text{mean}}$. Figure 3 shows the inverse data efficiency contours as a function of the correlation ρ and γ.

When there is no correlation between human and automatic metrics ($\rho = 0$), the data efficiency is naturally 1 (no gain). In order to achieve a data efficiency of 2 (half the labeling cost), we need $|\rho| \geq \sqrt{2}/2 \approx 0.707$. Interestingly, even for an automatic metric with perfect correlation ($\rho = 1$), the data efficiency is still capped by $\frac{1+\gamma}{\gamma}$: unless $\gamma \to 0$ the data efficiency cannot increase unboundedly. Intuitively, even if we knew that $\rho = 1$, $f(z)$ would be undetermined up to a constant additive shift and just estimating the shift would incur a variance of $\frac{1}{n}\sigma_a^2$.

3.3 Using the control variates estimator

The control variates estimator can be easily integrated into an existing evaluation: we run human evaluation on a random sample of system outputs, automatic evaluation on all the system outputs, and plug in these results into Algorithm 1.

It is vital that we are able to evaluate the automatic metric on a significantly larger set of examples than those with human evaluations to reliably normalize $g(z)$: without these additional examples, it be can shown that the optimal minimax estimator for μ is simply the naive estimate $\hat{\mu}_{\text{mean}}$. Intuitively, this is because estimating the mean of $g(z)$ incurs an equally large variance as estimating μ. In other words, $g(z)$ is only useful if we have additional information about g beyond the samples $\{z^{(i)}\}$.

Algorithm 1 shows the estimator. In practice, we do not know $\alpha = \mathrm{Cov}(f(z), g(z))$, so we use a plug-in estimate $\hat{\alpha}$ in line 3 to compute the estimate $\widetilde{\mu}$ in line 4. We note that estimating α from data does introduce a $O(1/n)$ bias, but when compared to the standard deviation which decays as $\Theta(1/\sqrt{n})$, this bias quickly goes to 0.

Proposition 3.1. *The estimator $\widetilde{\mu}$ in Algorithm 1 has $O(1/n)$ bias.*

Algorithm 1 Control variates estimator

```
1: Input: $n$ human evaluations $y^{(i)}$ on system
   outputs $z^{(i)}$, normalized automatic metric $g$
2: $\bar{y} = \frac{1}{n}\sum_i y^{(i)}$
3: $\hat{\alpha} = \frac{1}{n}\sum_i (y^{(i)} - \bar{y}) g(z^{(i)})$
4: $\widetilde{\mu} = \frac{1}{n}\sum_i y^{(i)} - \hat{\alpha} g(z^{(i)})$
5: return $\widetilde{\mu}$
```

An additional question that arises when applying Algorithm 1 is figuring out how many samples n to use. Given a target variance, the number of samples can be estimated using (6) with conservative estimates of σ_f^2, σ_a^2 and ρ. Alternatively, our estimator can be combined with a dynamic stopping rule (Mnih et al., 2008) to stop data collection once we reach a target confidence interval.

Task	Eval.	σ_a^2	σ_f^2	$\gamma = \frac{\sigma_a^2}{\sigma_f^2}$
CDM	Fluency	0.32	0.26	1.23
CDM	Redund.	0.26	0.43	0.61
CDM	Overall	0.28	0.28	1.00
CDM	**Edit**	**0.07**	**0.18**	**0.36**
MS MARCO	AnyCorr.	0.14	0.15	0.95
MS MARCO	AvgCorr.	0.12	0.13	0.91

Table 2: A summary of the key statistics, human metric variance (σ_f^2) and annotator variance (σ_a^2) for different datasets, CNN/Daily Mail (CDM) and MS MARCO in our evaluation benchmark. We observe that the relative variance (γ) is fairly high for most evaluation prompts, upper bounding the data efficiency on these tasks. A notable exception is the `Edit` prompt wherein systems are compared on the number of post-edits required to improve their quality.

3.4 Discussion of assumptions

We will soon see that empirical instantiations of γ and ρ lead to rather underwhelming data efficiencies in practice. In light of our optimality result, does this mean there is no hope for gains? Let us probe our assumptions. We assumed that the human judgments are uncorrelated across different system outputs; it is possible that a more accurate model of human annotators (e.g. Passonneau and Carpenter (2014)) could offer improvements. Perhaps with additional information about $g(z)$ such as calibrated confidence estimates, we would be able to sample more adaptively. Of course the most direct routes to improvement involve increasing the correlation of g with human judgments and reducing annotator variance, which we will discuss more later.

4 Tasks and datasets

In order to compare different approaches to evaluating systems, we first collected human judgments for the output of several automatic summarization and open-response question answering systems using Amazon Mechanical Turk. Details of instructions provided and quality assurance steps taken are provided in Appendix A of the supplementary material. In this section, we'll briefly describe how we collected this data.

Evaluating language quality in automatic summarization. In automatic summarization, systems must generate a short (on average two or three sentence) summary of an article: for our study, we chose articles from the CNN/Daily Mail (CDM) dataset (Hermann et al., 2015; Nallapati et al., 2016) which come paired with reference summaries in the form of story highlights. We focus on the *language quality* of summaries and leave evaluating content selection to future work.

For each summary, we collected human judgments on a scale from 1–3 (Figure 4a) for fluency, (lack of) redundancy, and overall quality of the summary using guidelines from the DUC summarization challenge (Dang, 2006). As an alternate human metric, we also asked workers to post-edit the system's summary to improve its quality, similar to the post-editing step in MT evaluations (Snover et al., 2006). Obtaining judgments costs about \$0.15 per summary and this cost rises to about \$0.40 per summary for post-editing.

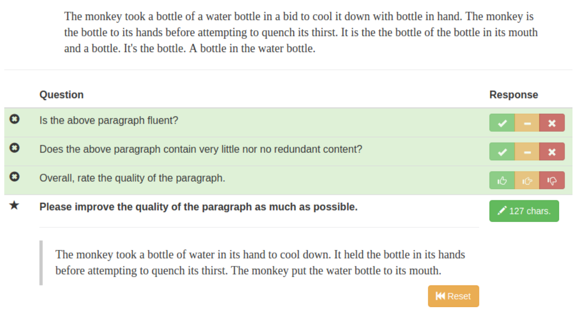
We collected judgments on the summaries generated by the `seq2seq` and `pointer` models of See et al. (2017), the `ml` and `ml+rl` models of Paulus et al. (2018), and the reference summaries.[3] Before presenting the summaries to human annotators, we performed some minimal post-processing: we true-cased and de-tokenized the output of `seq2seq` and `pointer` using Stanford CoreNLP (Manning et al., 2014) and replaced "unknown" tokens in each system with a special symbol (■).

Evaluating answer correctness. Next, we look at evaluating the correctness of system outputs in question answering using the MS MARCO question answering dataset (Nguyen et al., 2016). Here, each system is provided with a question and up to 10 paragraphs of context. The system generates open-response answers that do not need to be tied to a span in any paragraph.

We first ask annotators to judge if the output is even plausible for the question, and if yes, ask them identify if it is correct according to each context paragraph. We found that requiring annotators to highlight regions in the text that support their decision substantially improved the quality of the output without increasing costs. Annotations cost \$0.40 per system response.[4]

[3]All system output was obtained from the original authors through private communication.

[4]This cost could be significantly reduced if systems also

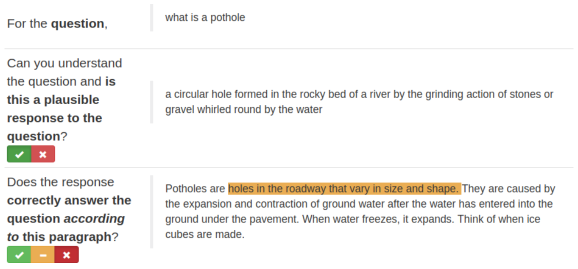


The monkey took a bottle of a water bottle in a bid to cool it down with bottle in hand. The monkey is the bottle to its hands before attempting to quench its thirst. It is the the bottle of the bottle in its mouth and a bottle. It's the bottle. A bottle in the water bottle.

	Question	Response
	Is the above paragraph fluent?	✓ – ✗
	Does the above paragraph contain very little nor no redundant content?	✓ – ✗
	Overall, rate the quality of the paragraph.	
★	**Please improve the quality of the paragraph as much as possible.**	127 chars.

The monkey took a bottle of water in its hand to cool down. It held the bottle in its hands before attempting to quench its thirst. The monkey put the water bottle to its mouth.

(a) Interface to evaluate language quality on CNN/Daily Mail



For the **question**,	what is a pothole
Can you understand the question and **is this a plausible response to the question**? ✓ ✗	a circular hole formed in the rocky bed of a river by the grinding action of stones or gravel whirled round by the water
Does the response **correctly answer the question *according to* this paragraph**? ✓ – ✗	Potholes are holes in the roadway that vary in size and shape. They are caused by the expansion and contraction of ground water after the water has entered into the ground under the pavement. When water freezes, it expands. Think of when ice cubes are made.

(b) Interface to judge answer correctness on MS MARCO

Figure 4: Screenshots of the annotation interfaces we used to measure (a) summary language quality on CNN/Daily Mail and (b) answer correctness on MS MARCO tasks.

While our goal is to evaluate the correctness of the provided answer, we found that there are often answers which may be correct or incorrect depending on the context. For example, the question "what is a pothole" is typically understood to refer to a hole in a roadway, but also refers to a geological feature (Figure 4b). This is reflected when annotators mark one context paragraph to support the given answer but mark another to contradict it. We evaluated systems based on both the average correctness (AvgCorrect) of their answers across all paragraphs as well as whether their answer is correct according to any paragraph (AnyCorrect).

We collected annotations on the systems generated by the `fastqa` and `fastqa_ext` from Weissenborn et al. (2017) and the `snet` and `snet.ens`(emble) models from Tan et al. (2018), along with reference answers. The answers generated by the systems were used without any post-processing. Surprisingly, we found that the correctness of the reference answers (according to the AnyCorrect metric) was only 73.5%, only 2% above that of the leading system (`snet.ens`). We manually inspected 30 reference answers which were annotated incorrectly and found that of those, about 95% were indeed incorrect. However, 62% are actually answerable from some paragraph, indicating that the real ceiling performance on this dataset is around 90% and that there is still room for improvement on this task.

5 Experimental results

We are now ready to evaluate the performance of our control variates estimator proposed in Section 3 using the datasets presented in Section 4.

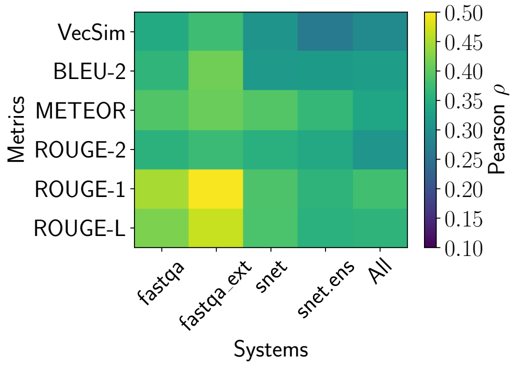
specify which passage they used to generate the answer.

Recall that our primary quantity of interest is *data efficiency*, the ratio of the number of human judgments required to estimate the overall human evaluation score for the control variates estimator versus the sample mean. We'll briefly review the automatic metrics used in our evaluation before analyzing the results.

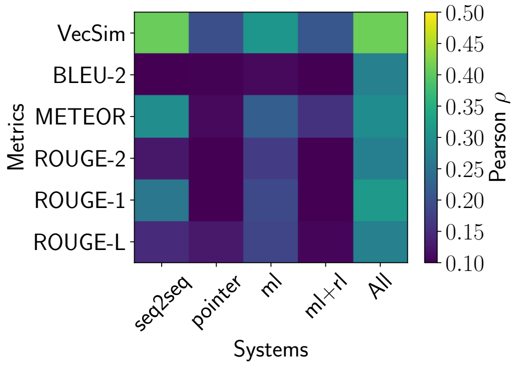
Automatic metrics. We consider the following frequently used automatic word-overlap based metrics in our work: **BLEU** (Papineni et al., 2002), **ROUGE** (Lin and Rey, 2004) and **METEOR** (Lavie and Denkowski, 2009). Following Novikova et al. (2017) and Liu et al. (2016b), we also compared a vector-based sentence-similarity using `sent2vec` (Pagliardini et al., 2017) to compare sentences (**VecSim**). Figure 5 shows how each of these metrics is correlated with human judgment for the systems being evaluated. Unsurprisingly, the correlation varies considerably across systems, with token-based metrics correlating more strongly for systems that are more extractive in nature (`fastqa` and `fastqa_ext`).

Results.[5] In Section 3 we proved that the control variates estimator is not only unbiased but also has the least variance among other unbiased estimators. Figure 6 plots the width of the 80% confidence interval, estimated using bootstrap, measured as a function of the number of samples collected for different tasks and prompts. As expected, the control variates estimator reduces the width of the confidence interval. We measure data efficiency by the averaging of the ratio of squared confidence intervals between the human baseline

[5]Extended results for other systems, metrics and prompts can be found at `https://bit.ly/price-of-debiasing/`.



(a) MS MARCO with the `AnyCorrect` prompt



(b) CNN/Daily Mail with the `Edit` prompt

Figure 5: Correlations of different automatic metrics on the MS MARCO and CNN/Daily Mail tasks. Certain systems are more correlated with certain automatic metrics than others, but overall the correlation is low to moderate for most systems and metrics.

and control variates estimates. We observe that the data efficiency depends on the task, prompt and system, ranging from about 1.08 (a 7% cost reduction) to 1.15 (a 13% cost reduction) using current automatic metrics.

As we showed in Section 3, further gains are fundamentally limited by the quality of the evaluation prompts and automatic metrics. Figures 6a and 6b show how improving the quality of the evaluation prompt from a Likert-scale prompt for quality (`Overall`) to using post-editing (`Edit`) noticeably decreases variance and hence allows better automatic metrics to increase data efficiency. Likewise, Figure 6c shows how using a better automatic metric (ROUGE-L instead of VecSim) also reduces variance.

Figure 6 also shows the conjectured confidence intervals if we were able to eliminate noise in human judgments (noiseless humans) or have a automatic metric that correlated perfectly with average human judgment (perfect metric). In particular, we use the mean of all (2–3) humans on each z for the perfect $g(z)$ and use the mean of all humans on each z for the "noiseless" $Y(z)$.

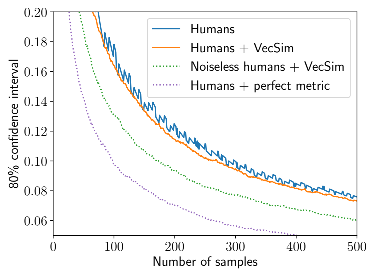
In both cases, we are able to significantly increase data efficiency (i.e. decrease estimator variance). With zero annotator variance and using existing automatic metrics, the data efficiency ranges from 1.42 to 1.69. With automatic metrics with perfect correlation and current variance of human judgments, it ranges from 2.38 to 7.25. Thus, we conclude that it is important not only to improve our automatic metrics but also the evaluation prompts we use during human evaluation.

6 Related work

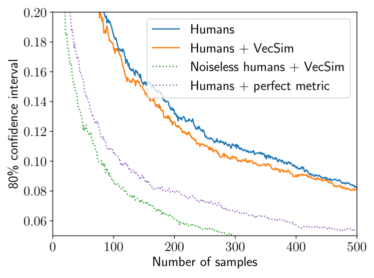
In this work, we focus on using existing automatic metrics to decrease the cost of human evaluations. There has been much work on improving the quality of automatic metrics. In particular, there is interest in learning models (Lowe et al., 2017a; Dusek et al., 2017) that are able to optimize for improved correlations with human judgment. However, in our experience, we have found that these learned automatic metrics have trouble generalizing to different systems. The framework we provide allows us to safely incorporate such models into evaluation, exploiting them when their correlation is high but also not introducing bias when it is low.

Our key technical tool is control variates, a standard statistical technique used to reduce the variance of Monte Carlo estimates (Ripley, 2009). The technique has also been used in machine learning and reinforcement learning to lower variance estimates of gradients (Greensmith et al., 2004; Paisley et al., 2012; Ranganath et al., 2014). To the best of our knowledge, we are the first to apply this technique in the context of language evaluation.

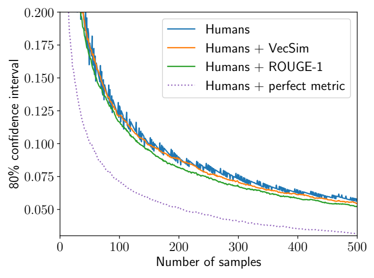
Our work also highlights the importance of human evaluation. Chaganty et al. (2017) identified a similar problem of systematic bias in evaluation metrics in the setting of knowledge base population and also propose statistical estimators that relies on human evaluation to correct bias. Unfortunately, their technique relies on having a structured output (relation triples) that are shared between



(a) `seq2seq` on CNN/Daily Mail using the `Overall`



(b) `seq2seq` on CNN/Daily Mail using `Edit`



(c) `fastqa_ext` on MS MARCO using `AnyCorrect`

Figure 6: 80% bootstrap confidence interval length as a function of the number of human judgments used when evaluating the indicated systems on their respective datasets and prompts. (a) We see a modest reduction in variance (and hence cost) relative to human evaluation by using the VecSim automatic metric with the proposed control variates estimator to estimate `Overall` scores on the CNN/Daily Mail task; the data efficiency (DE) is 1.06. (b) By improving the evaluation prompt to use `Edits` instead, it is possible to further reduce variance relative to humans (DE is 1.15). (c) Another way to reduce variance relative to humans is to improve the automatic metric evaluation; here using ROUGE-1 instead of VecSim improves the DE from 1.03 to 1.16.

systems and does not apply to evaluating natural language generation. In a similar vein, Chang et al. (2017) dynamically collect human feedback to learn better dialog policies.

7 Discussion

Prior work has shown that existing automatic metrics have poor instance-level correlation with mean human judgment and that they score many good quality responses poorly. As a result, the evaluation is systematically biased against genuine system improvements that would lead to higher human evaluation scores but not improve automatic metrics. In this paper, we have explored using an automatic metric to decrease the cost of human evaluation without introducing bias. In practice, we find that with current automatic metrics and evaluation prompts data efficiencies are only 1.08–1.15 (7–13% cost reduction). Our theory shows that further improvements are only possible by improving the correlation of the automatic metric and reducing the annotator variance of the evaluation prompt. As an example of how evaluation prompts could be improved, we found that using post-edits of summarizes decreased normalized annotator variance by a factor of three relative to using a Likert scale survey. It should be noted that changing the evaluation prompt also changes the underlying ground truth $f(z)$: it is up to us to find a prompt that still captures the essence of what we want to measure.

Without making stronger assumptions, the control variates estimator we proposed outlines the limitations of unbiased estimation. Where do we go from here? Certainly, we can try to improve the automatic metric (which is potentially as difficult as solving the task) and brainstorming alternative ways of soliciting evaluation (which has been less explored). Alternatively, we could give up on measuring absolute scores, and seek instead to find techniques stably rank methods and thus improve them. As the NLP community tackles increasingly difficult tasks, human evaluation will only become more important. We hope our work provides some clarity on to how to make it more cost effective.

Reproducibility

All code, data, and experiments for this paper are available on the CodaLab platform at `https://bit.ly/price-of-debiasing`.

Acknowledgments

We are extremely grateful to the authors of the systems we evaluated for sharing their systems' output with us. We also would like to thank Urvashi Khandelwal and Peng Qi for feedback on an earlier draft of the paper, the crowdworkers on Amazon Mechanical Turk and TurkNation for their work and feedback during the data collection process, and the anonymous reviewers for their constructive feedback.

References

A. Chaganty, A. Paranjape, P. Liang, and C. Manning. 2017. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Chang, R. Yang, L. Chen, X. Zhou, and K. Yu. 2017. Affordable on-line dialogue policy learning. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 223–231.

J. M. Conroy and H. T. Dang. 2008. Mind the gap : Dangers of divorcing evaluations of summary content from linguistic quality. In *International Conference on Computational Linguistics (COLING)*. pages 145–152.

H. T. Dang. 2006. Overview of DUC 2006. In *Document Understanding Conference*.

M. Denkowski and A. Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Workshop on Statistical Machine Translation*.

O. Dusek, J. Novikova, and V. Rieser. 2017. Referenceless quality estimation for natural language generation. *arXiv* .

E. Greensmith, P. L. Bartlett, and J. Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research (JMLR)* 5:1471–1530.

K. M. Hermann, T. Koisk, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

T. Kočisky, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. 2017. The NarrativeQA reading comprehension challenge. *arXiv preprint arXiv:1712.07040* .

A. Lavie and M. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation* 23.

C. Lin and M. Rey. 2004. Looking for a few good metrics: ROUGE and its evaluation. In *NTCIR Workshop*.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll'ar, and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. pages 740–755.

A. Liu, S. Soderland, J. Bragg, C. H. Lin, X. Ling, and D. S. Weld. 2016a. Effective crowd annotation for relation extraction. In *North American Association for Computational Linguistics (NAACL)*. pages 897–906.

C. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. 2016b. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. 2017a. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Association for Computational Linguistics (ACL)*.

R. T. Lowe, N. Pow, I. Serban, L. Charlin, C. Liu, and J. Pineau. 2017b. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue and Discourse* 8.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL system demonstrations*.

V. Mnih, C. Szepesv'ari, and J. Audibert. 2008. Empirical berstein stopping. In *International Conference on Machine Learning (ICML)*.

R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop on Cognitive Computing at NIPS*.

J. Novikova, O. Duek, A. C. Curry, and V. Rieser. 2017. Why we need new evaluation metrics for NLG. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Pagliardini, P. Gupta, and M. Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv* .

J. Paisley, D. M. Blei, and M. I. Jordan. 2012. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning (ICML)*. pages 1363–1370.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*.

R. J. Passonneau and B. Carpenter. 2014. The benefits of a model of annotation. In *Association for Computational Linguistics (ACL)*.

R. Paulus, C. Xiong, and R. Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

R. Ranganath, S. Gerrish, and D. Blei. 2014. Black box variational inference. In *Artificial Intelligence and Statistics (AISTATS)*. pages 814–822.

B. D. Ripley. 2009. *Stochastic simulation*. John Wiley & Sons.

A. See, P. J. Liu, and C. D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics (ACL)*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Association for Machine Translation in the Americas*. pages 223–231.

C. Tan, F. Wei, N. Yang, W. Lv, and M. Zhou. 2018. S-Net: From answer extraction to answer generation for machine reading comprehension. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

R. Vedantam, C. L. Zitnick, and D. Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Computer Vision and Pattern Recognition (CVPR)*. pages 4566–4575.

D. Weissenborn, G. Wiese, and L. Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Computational Natural Language Learning (CoNLL)*.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Neural Document Summarization by Jointly Learning to Score and Select Sentences

Qingyu Zhou[†][*], **Nan Yang**[‡], **Furu Wei**[‡], **Shaohan Huang**[‡], **Ming Zhou**[‡], **Tiejun Zhao**[†]
[†]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research, Beijing, China
{qyzhou,tjzhao}@hit.edu.cn
{nanya,fuwei,shaohanh,mingzhou}@microsoft.com

Abstract

Sentence scoring and sentence selection are two main steps in extractive document summarization systems. However, previous works treat them as two separated subtasks. In this paper, we present a novel end-to-end neural network framework for extractive document summarization by jointly learning to score and select sentences. It first reads the document sentences with a hierarchical encoder to obtain the representation of sentences. Then it builds the output summary by extracting sentences one by one. Different from previous methods, our approach integrates the selection strategy into the scoring model, which directly predicts the relative importance given previously selected sentences. Experiments on the CNN/Daily Mail dataset show that the proposed framework significantly outperforms the state-of-the-art extractive summarization models.

1 Introduction

Traditional approaches to automatic text summarization focus on identifying important content, usually at sentence level (Nenkova and McKeown, 2011). With the identified important sentences, a summarization system can extract them to form an output summary. In recent years, *extractive methods* for summarization have proven effective in many systems (Carbonell and Goldstein, 1998; Mihalcea and Tarau, 2004; McDonald, 2007; Cao et al., 2015a). In previous works that use extractive methods, text summarization is decomposed into two subtasks, i.e., sentence scoring and sentence selection.

Sentence scoring aims to assign an importance score to each sentence, and has been broadly studied in many previous works. Feature-based methods are popular and have proven effective, such as word probability, TF*IDF weights, sentence position and sentence length features (Luhn, 1958; Hovy and Lin, 1998; Ren et al., 2017). Graph-based methods such as TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004) measure sentence importance using weighted-graphs. In recent years, neural network has also been applied to sentence modeling and scoring (Cao et al., 2015a; Ren et al., 2017).

For the second step, *sentence selection* adopts a particular strategy to choose content sentence by sentence. Maximal Marginal Relevance (Carbonell and Goldstein, 1998) based methods select the sentence that has the maximal score and is minimally redundant with sentences already included in the summary. Integer Linear Programming based methods (McDonald, 2007) treat sentence selection as an optimization problem under some constraints such as summary length. Submodular functions (Lin and Bilmes, 2011) have also been applied to solving the optimization problem of finding the optimal subset of sentences in a document. Ren et al. (2016) train two neural networks with handcrafted features. One is used to rank sentences, and the other one is used to model redundancy during sentence selection.

In this paper, we present a neural extractive document summarization (NeuSum) framework which jointly learns to score and select sentences. Different from previous methods that treat sentence scoring and sentence selection as two tasks, our method integrates the two steps into one end-to-end trainable model. Specifically, NeuSum is a neural network model without any handcrafted features that learns to identify the relative importance of sentences. The relative importance is

[*]Contribution during internship at Microsoft Research.

measured as the gain over previously selected sentences. Therefore, each time the proposed model selects one sentence, it scores the sentences considering both sentence saliency and previously selected sentences. Through the joint learning process, the model learns to predict the relative gain given the sentence extraction state and the partial output summary.

The proposed model consists of two parts, i.e., the document encoder and the sentence extractor. The document encoder has a hierarchical architecture, which suits the compositionality of documents. The sentence extractor is built with recurrent neural networks (RNN), which provides two main functionalities. On one hand, the RNN is used to remember the partial output summary by feeding the selected sentence into it. On the other hand, it is used to provide a sentence extraction state that can be used to score sentences with their representations. At each step during extraction, the sentence extractor reads the representation of the last extracted sentence. It then produces a new sentence extraction state and uses it to score the relative importance of the rest sentences.

We conduct experiments on the *CNN/Daily Mail* dataset. The experimental results demonstrate that the proposed NEUSUM by jointly scoring and selecting sentences achieves significant improvements over separated methods. Our contributions are as follows:

- We propose a joint sentence scoring and selection model for extractive document summarization.
- The proposed model can be end-to-end trained without handcrafted features.
- The proposed model significantly outperforms state-of-the-art methods and achieves the best result on *CNN/Daily Mail* dataset.

2 Related Work

Extractive document summarization has been extensively studied for years. As an effective approach, extractive methods are popular and dominate the summarization research. Traditional extractive summarization systems use two key techniques to form the summary, sentence scoring and sentence selection. Sentence scoring is critical since it is used to measure the saliency of a sentence. Sentence selection is based on the scores of sentences to determine which sentence should be extracted, which is usually done heuristically.

Many techniques have been proposed to model and score sentences. Unsupervised methods do not require model training or data annotation. In these methods, many surface features are useful, such as term frequency (Luhn, 1958), TF*IDF weights (Erkan and Radev, 2004), sentence length (Cao et al., 2015a) and sentence positions (Ren et al., 2017). These features can be used alone or combined with weights.

Graph-based methods (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Wan and Yang, 2006) are also applied broadly to ranking sentences. In these methods, the input document is represented as a connected graph. The vertices represent the sentences, and the edges between vertices have attached weights that show the similarity of the two sentences. The score of a sentence is the importance of its corresponding vertex, which can be computed using graph algorithms.

Machine learning techniques are also widely used for better sentence modeling and importance estimation. Kupiec et al. (1995) use a Naive Bayes classifier to learn feature combinations. Conroy and O'leary (2001) further use a Hidden Markov Model in document summarization. Gillick and Favre (2009) find that using bigram features consistently yields better performance than unigrams or trigrams for ROUGE (Lin, 2004) measures.

Carbonell and Goldstein (1998) proposed the Maximal Marginal Relevance (MMR) method as a heuristic in sentence selection. Systems using MMR select the sentence which has the maximal score and is minimally redundant with previous selected sentences. McDonald (2007) treats sentence selection as an optimization problem under some constraints such as summary length. Therefore, he uses Integer Linear Programming (ILP) to solve this optimization problem. Sentence selection can also be seen as finding the optimal subset of sentences in a document. Lin and Bilmes (2011) propose using submodular functions to find the subset.

Recently, deep neural networks based approaches have become popular for extractive document summarization. Cao et al. (2015b) develop a novel summary system called PriorSum, which applies enhanced convolutional neural networks to capture the summary prior features derived from length-variable phrases. Ren et al. (2017) use a two-level attention mechanism to measure the contextual relations of sentences. Cheng and Lapata (2016) propose treating document summarization as a sequence labeling task. They first encode the sentences in the document and then classify each sentence into two classes, i.e., extraction or not. Nallapati et al. (2017) propose a system called SummaRuNNer with more features, which also treat extractive document summarization as a sequence labeling task. The two works are both in the separated paradigm, as they first assign a probability of being extracted to each sentence, and then select sentences according to the probability until reaching the length limit. Ren et al. (2016) train two neural networks with handcrafted features. One is used to rank the sentences to select the first sentence, and the other one is used to model the redundancy during sentence selection. However, their model of measuring the redundancy only considers the redundancy between the sentence that has the maximal score, which lacks the modeling of all the selection history.

3 Problem Formulation

Extractive document summarization aims to extract informative sentences to represent the important meanings of a document. Given a document $\mathcal{D} = (S_1, S_2, \ldots, S_L)$ containing L sentences, an extractive summarization system should select a subset of $\mathcal{D}$ to form the output summary $\mathcal{S} = \{\hat{S}_i | \hat{S}_i \in \mathcal{D}\}$. During the training phase, the reference summary $\mathcal{S}^*$ and the score of an output summary $\mathcal{S}$ under a given evaluation function $r(\mathcal{S}|\mathcal{S}^*)$ are available. The goal of training is to learn a scoring function $f(\mathcal{S})$ which can be used to find the best summary during testing:

$$\begin{aligned} \arg\max_{\mathcal{S}} \quad & f(\mathcal{S}) \\ \text{s.t.} \quad & \mathcal{S} = \{\hat{S}_i | \hat{S}_i \in \mathcal{D}\} \\ & |\mathcal{S}| \leq l. \end{aligned}$$

where l is length limit of the output summary. In this paper, l is the sentence number limit.

Previous state-of-the-art summarization systems search the best solution using the learned scoring function $f(\cdot)$ with two methods, MMR and ILP. In this paper, we adopt the MMR method. Since MMR tries to maximize the relative gain given previous extracted sentences, we let the model to learn to score this gain. Previous works adopt ROUGE recall as the evaluation $r(\cdot)$ considering the DUC tasks have byte length limit for summaries. In this work, we adopt the *CNN/Daily Mail* dataset to train the neural network model, which does not have this length limit. To prevent the tendency of choosing longer sentences, we use ROUGE F1 as the evaluation function $r(\cdot)$, and set the length limit l as a fixed number of sentences.

Therefore, the proposed model is trained to learn a scoring function $g(\cdot)$ of the ROUGE F1 gain, specifically:

$$g(S_t | \mathbb{S}_{t-1}) = r\left(\mathbb{S}_{t-1} \cup \{S_t\}\right) - r(\mathbb{S}_{t-1}) \quad (1)$$

where $\mathbb{S}_{t-1}$ is the set of previously selected sentences, and we omit the condition $\mathcal{S}^*$ of $r(\cdot)$ for simplicity. At each time t, the summarization system chooses the sentence with maximal ROUGE F1 gain until reaching the sentence number limit.

4 Neural Document Summarization

Figure 1 gives the overview of NEUSUM, which consists of a hierarchical document encoder, and a sentence extractor. Considering the intrinsic hierarchy nature of documents, that words form a sentence and sentences form a document, we employ a hierarchical document encoder to reflect this hierarchy structure. The sentence extractor scores the encoded sentences and extracts one of them at each step until reaching the output sentence number limit. In this section, we will first introduce the hierarchical document encoder, and then describe how the model produces summary by joint sentence scoring and selection.

4.1 Document Encoding

We employ a hierarchical document encoder to represent the sentences in the input document. We encode the document in two levels, i.e., sentence level encoding and document level encoding. Given a document $\mathcal{D} = (S_1, S_2, \ldots, S_L)$ containing L sentences. The sentence level encoder reads the j-th input sentence $S_j = (x_1^{(j)}, x_2^{(j)}, \ldots, x_{n_j}^{(j)})$ and constructs the basic sentence representation $\widetilde{s}_j$. Here we employ a bidirectional GRU (BiGRU) (Cho et al., 2014) as the recurrent unit, where GRU is defined as:

$$z_i = \sigma(\mathbf{W}_z[x_i, h_{i-1}]) \quad (2)$$

$$r_i = \sigma(\mathbf{W}_r[x_i, h_{i-1}]) \quad (3)$$

$$\widetilde{h}_i = \tanh(\mathbf{W}_h[x_i, r_i \odot h_{i-1}]) \quad (4)$$

$$h_i = (1 - z_i) \odot h_{i-1} + z_i \odot \widetilde{h}_i \quad (5)$$

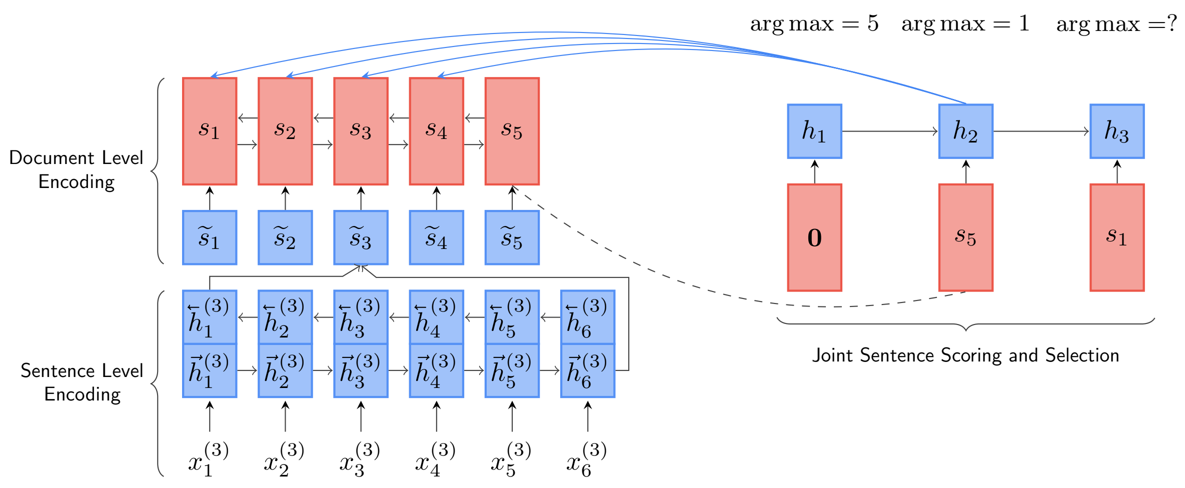


Figure 1: Overview of the NEUSUM model. The model extracts S_5 and S_1 at the first two steps. At the first step, we feed the model a zero vector $\mathbf{0}$ to represent empty partial output summary. At the second and third steps, the representations of previously selected sentences S_5 and S_1, i.e., s_5 and s_1, are fed into the extractor RNN. At the second step, the model only scores the first 4 sentences since the 5th one is already included in the partial output summary.

where $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_h$ are weight matrices.

The BiGRU consists of a forward GRU and a backward GRU. The forward GRU reads the word embeddings in sentence S_j from left to right and gets a sequence of hidden states, $(\vec{h}_1^{(j)}, \vec{h}_2^{(j)}, \ldots, \vec{h}_{n_j}^{(j)})$. The backward GRU reads the input sentence embeddings reversely, from right to left, and results in another sequence of hidden states, $(\overleftarrow{h}_1^{(j)}, \overleftarrow{h}_2^{(j)}, \ldots, \overleftarrow{h}_{n_j}^{(j)})$:

$$\vec{h}_i^{(j)} = \text{GRU}(x_i^{(j)}, \vec{h}_{i-1}^{(j)}) \tag{6}$$

$$\overleftarrow{h}_i^{(j)} = \text{GRU}(x_i^{(j)}, \overleftarrow{h}_{i+1}^{(j)}) \tag{7}$$

where the initial states of the BiGRU are set to zero vectors, i.e., $\vec{h}_1^{(j)} = 0$ and $\overleftarrow{h}_{n_j}^{(j)} = 0$.

After reading the words of the sentence S_j, we construct its sentence level representation $\widetilde{s}_j$ by concatenating the last forward and backward GRU hidden vectors:

$$\widetilde{s}_j = \begin{bmatrix} \overleftarrow{h}_1^{(j)} \\ \vec{h}_{n_j}^{(j)} \end{bmatrix} \tag{8}$$

We use another BiGRU as the document level encoder to read the sentences. With the sentence level encoded vectors $(\widetilde{s}_1, \widetilde{s}_2, \ldots, \widetilde{s}_L)$ as inputs, the document level encoder does forward and backward GRU encoding and produces two list of hidden vectors: $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_L)$ and $(\overleftarrow{s}_1, \overleftarrow{s}_2, \ldots, \overleftarrow{s}_L)$. The document level representation s_i of sentence S_i is the concatenation of the forward and backward hidden vectors:

$$s_i = \begin{bmatrix} \vec{s}_i \\ \overleftarrow{s}_i \end{bmatrix} \tag{9}$$

We then get the final sentence vectors in the given document: $D = (s_1, s_2, \ldots, s_L)$. We use sentence S_i and its representative vector s_i interchangeably in this paper.

4.2 Joint Sentence Scoring and Selection

Since the separated sentence scoring and selection cannot utilize the information of each other, the goal of our model is to make them benefit each other. We couple these two steps together so that: a) sentence scoring can be aware of previously selected sentences; b) sentence selection can be simplified since the scoring function is learned to be the ROUGE score gain as described in section 3.

Given the last extracted sentence $\hat{S}_{t-1}$, the sentence extractor decides the next sentence $\hat{S}_t$ by scoring the remaining document sentences. To score the document sentences considering both their importance and partial output summary, the model should have two key abilities: 1) remembering the information of previous selected sentences; 2) scoring the remaining document sentences based on both the previously selected sentences and the importance of remaining sentences. Therefore, we employ another GRU as the recurrent unit to remember the partial output summary, and use a Multi-Layer Perceptron (MLP) to score the document sentences. Specifically, the GRU takes the document level representation s_{t-1} of the last extracted sentence $\hat{S}_{t-1}$ as input to produce its current hidden state h_t. The sentence scorer, which is a two-layer MLP, takes two input vectors, namely the current hidden state h_t and the sentence representation vector s_i, to calculate the score $\delta(S_i)$ of sentence S_i.

$$h_t = \mathrm{GRU}(s_{t-1}, h_{t-1}) \tag{10}$$

$$\delta(S_i) = \mathbf{W}_s \tanh\left(\mathbf{W}_q h_t + \mathbf{W}_d s_i\right) \tag{11}$$

where $\mathbf{W}_s$, $\mathbf{W}_q$ and $\mathbf{W}_d$ are learnable parameters, and we omit the bias parameters for simplicity.

When extracting the first sentence, we initialize the GRU hidden state h_0 with a linear layer with tanh activation function:

$$h_0 = \tanh\left(\mathbf{W}_m \overleftarrow{s}_1 + b_m\right) \tag{12}$$

$$S_0 = \varnothing \tag{13}$$

$$s_0 = \mathbf{0} \tag{14}$$

where$\mathbf{W}_m$ and b_m are learnable parameters, and $\overleftarrow{s}_1$ is the last backward state of the document level encoder BiGRU. Since we do not have any sentences extracted yet, we use a zero vector to represent the previous extracted sentence, i.e., $s_0 = \mathbf{0}$.

With the scores of all sentences at time t, we choose the sentence with maximal gain score:

$$\hat{S}_t = \underset{S_i \in \mathcal{D}}{\arg\max}\, \delta(S_i) \tag{15}$$

4.3 Objective Function

Inspired by Inan et al. (2017), we optimize the Kullback-Leibler (KL) divergence of the model prediction P and the labeled training data distribution Q. We normalize the predicted sentence score $\delta(S_i)$ with softmax function to get the model prediction distribution P:

$$P(\hat{S}_t = S_i) = \frac{\exp\left(\delta(S_i)\right)}{\sum_{k=1}^{L} \exp\left(\delta(S_k)\right)} \tag{16}$$

During training, the model is expected to learn the relative ROUGE F1 gain at time step t with previously selected sentences $\mathbb{S}_{t-1}$. Considering that the F1 gain value might be negative in the labeled data, we follow previous works (Ren et al., 2017) to use Min-Max Normalization to rescale the gain value to $[0, 1]$:

$$g(S_i) = r(\mathbb{S}_{t-1} \cup \{S_i\}) - r(\mathbb{S}_{t-1}) \tag{17}$$

$$\widetilde{g}(S_i) = \frac{g(S_i) - \min\left(g(S)\right)}{\max\left(g(S)\right) - \min\left(g(S)\right)} \tag{18}$$

We then apply a softmax operation with temperature τ (Hinton et al., 2015) [1] to produce the labeled data distribution Q as the training target. We apply the temperature τ as a smoothing factor to produce a smoothed label distribution Q:

$$Q(S_i) = \frac{\exp\left(\tau \widetilde{g}(S_i)\right)}{\sum_{k=1}^{L} \exp\left(\tau \widetilde{g}(S_k)\right)} \tag{19}$$

Therefore, we minimize the *KL* loss function J:

$$J = D_{KL}(P \parallel Q) \tag{20}$$

5 Experiments

5.1 Dataset

A large scale dataset is essential for training neural network-based summarization models. We use the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016) as the training set in our experiments. The *CNN/Daily Mail* news contain articles and their corresponding highlights. The highlights are created by human editors and are abstractive summaries. Therefore, the highlights are not ready for training extractive systems due to the lack of supervisions.

We create an extractive summarization training set based on *CNN/Daily Mail* corpus. To determine the sentences to be extracted, we design a rule-based system to label the sentences in a given document similar to Nallapati et al. (2017). Specifically, we construct training data by maximizing the ROUGE-2 F1 score. Since it is computationally expensive to find the global optimal combination of sentences, we employ a greedy approach. Given a document with n sentences, we enumerate the candidates from 1-combination $\binom{n}{1}$ to n-combination $\binom{n}{n}$. We stop searching if the highest ROUGE-2 F1 score in $\binom{n}{k}$ is less than the best one in $\binom{n}{k-1}$. Table 1 shows the data statistics of the *CNN/Daily Mail* dataset.

We conduct data preprocessing using the same method[2] in See et al. (2017), including sentence splitting and word tokenization. Both Nallapati et al. (2016, 2017) use the *anonymized* version of the data, where the named entities are replaced by identifiers such as `entity4`. Following See et al. (2017), we use the *non-anonymized* version so we can directly operate on the original text.

[1]We set $\tau = 20$ empirically according to the model performance on the development set.

[2]https://github.com/abisee/cnn-dailymail

CNN/Daily Mail	**Training**	**Dev**	**Test**
#(Document)	287,227	13,368	11,490
#(Ref / Document)	1	1	1
Doc Len (Sentence)	31.58	26.72	27.05
Doc Len (Word)	791.36	769.26	778.24
Ref Len (Sentence)	3.79	4.11	3.88
Ref Len (Word)	55.17	61.43	58.31

Table 1: Data statistics of *CNN/Daily Mail* dataset.

5.2 Implementation Details

Model Parameters The vocabulary is collected from the *CNN/Daily Mail* training data. We lower-case the text and there are 732,304 unique word types. We use the top 100,000 words as the model vocabulary since they can cover 98.23% of the training data. The size of word embedding, sentence level encoder GRU, document level encoder GRU are set to 50, 256, and 256 respectively. We set the sentence extractor GRU hidden size to 256.

Model Training We initialize the model parameters randomly using a Gaussian distribution with Xavier scheme (Glorot and Bengio, 2010). The word embedding matrix is initialized using pre-trained 50-dimension GloVe vectors (Pennington et al., 2014)[3]. We found that larger size GloVe does not lead to improvement. Therefore, we use 50-dim word embeddings for fast training. The pre-trained GloVe vectors contain 400,000 words and cover 90.39% of our model vocabulary. We initialize the rest of the word embeddings randomly using a Gaussian distribution with Xavier scheme. The word embedding matrix is not updated during training. We use Adam (Kingma and Ba, 2015) as our optimizing algorithm. For the hyperparameters of Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. We also apply gradient clipping (Pascanu et al., 2013) with range $[-5, 5]$ during training. We use dropout (Srivastava et al., 2014) as regularization with probability $p = 0.3$ after the sentence level encoder and $p = 0.2$ after the document level encoder. We truncate each article to 80 sentences and each sentence to 100 words during both training and testing. The model is implemented with PyTorch (Paszke et al., 2017). We release the source code and related resources at `https://res.qyzhou.me`.

[3]`https://nlp.stanford.edu/projects/glove/`

Model Testing At test time, considering that LEAD3 is a commonly used and strong extractive baseline, we make NEUSUM and the baselines extract 3 sentences to make them all comparable.

5.3 Baseline

We compare NEUSUM model with the following state-of-the-art baselines:

LEAD3 The commonly used baseline by selecting the first three sentences as the summary.

TEXTRANK An unsupervised algorithm based on weighted-graphs proposed by Mihalcea and Tarau (2004). We use the implementation in Gensim (Řehůřek and Sojka, 2010).

CRSUM Ren et al. (2017) propose an extractive summarization system which considers the contextual information of a sentence. We train this baseline model with the same training data as our approach.

NN-SE Cheng and Lapata (2016) propose an extractive system which models document summarization as a sequence labeling task. We train this baseline model with the same training data as our approach.

SUMMARUNNER Nallapati et al. (2017) propose to add some interpretable features such as sentence absolute and relative positions.

PGN Pointer-Generator Network (PGN). A state-of-the-art abstractive document summarization system proposed by See et al. (2017), which incorporates copying and coverage mechanisms.

5.4 Evaluation Metric

We employ ROUGE (Lin, 2004) as our evaluation metric. ROUGE measures the quality of summary by computing overlapping lexical units, such as unigram, bigram, trigram, and longest common subsequence (LCS). It has become the standard evaluation metric for DUC shared tasks and popular for summarization evaluation. Following previous work, we use ROUGE-1 (unigram), ROUGE-2 (bigram) and ROUGE-L (LCS) as the evaluation metrics in the reported experimental results.

5.5 Results

We use the official ROUGE script[4] (version 1.5.5) to evaluate the summarization output. Table 2 summarizes the results on *CNN/Daily Mail* data set using full length ROUGE-F1[5] evaluation. It includes two unsupervised baselines, LEAD3 and TEXTRANK. The table also includes three state-of-the-art neural network based extractive models, i.e., CRSUM, NN-SE and SUMMARUNNER. In addition, we report the state-of-the-art abstractive PGN model. The result of SUMMARUNNER is on the *anonymized* dataset and not strictly comparable to our results on the *non-anonymized* version dataset. Therefore, we also include the result of LEAD3 on the *anonymized* dataset as a reference.

Models	ROUGE-1	ROUGE-2	ROUGE-L
LEAD3	40.24^{-}	17.70^{-}	36.45^{-}
TEXTRANK	40.20^{-}	17.56^{-}	36.44^{-}
CRSUM	40.52^{-}	18.08^{-}	36.81^{-}
NN-SE	41.13^{-}	18.59^{-}	37.40^{-}
PGN‡	39.53^{-}	17.28^{-}	36.38^{-}
LEAD3‡ *	39.2	15.7	35.5
SUMMARUNNER‡ *	39.6	16.2	35.3
NEUSUM	**41.59**	**19.01**	**37.98**

Table 2: Full length ROUGE F1 evaluation (%) on *CNN/Daily Mail* test set. Results with ‡ mark are taken from the corresponding papers. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most ± 0.22 as reported by the official ROUGE script. The improvement is statistically significant with respect to the results with superscript $^{-}$ mark.

NEUSUM achieves 19.01 ROUGE-2 F1 score on the *CNN/Daily Mail* dataset. Compared to the unsupervised baseline methods, NEUSUM performs better by a large margin. In terms of ROUGE-2 F1, NEUSUM outperforms the strong baseline LEAD3 by 1.31 points. NEUSUM also outperforms the neural network based models. Compared to the state-of-the-art extractive model NN-SE (Cheng and Lapata, 2016), NEUSUM performs significantly better in terms of ROUGE-1, ROUGE-2 and ROUGE-L F1 scores. Shallow features, such as sentence position, have proven effective in document summarization (Ren et al., 2017; Nallapati et al., 2017). Without any hand-crafted features, NEUSUM performs better than the CRSUM and SUMMARUNNER baseline models with features. As given by the 95% confidence interval in the official ROUGE script, our model achieves statistically significant improvements over all the baseline models. To the best of our knowledge, the proposed NEUSUM model achieves the best results on the *CNN/Daily Mail* dataset.

Models	Info	Rdnd	Overall
NN-SE	1.36	1.29	1.39
NEUSUM	**1.33**	**1.21**	**1.34**

Table 3: Rankings of NEUSUM and NN-SE in terms of informativeness (Info), redundancy (Rdnd) and overall quality by human participants (lower is better).

We also provide human evaluation results on a sample of test set. We random sample 50 documents and ask three volunteers to evaluate the output of NEUSUM and the NN-SE baseline models. They are asked to rank the output summaries from best to worst (with ties allowed) regarding informativeness, redundancy and overall quality. Table 3 shows the human evaluation results. NEUSUM performs better than the NN-SE baseline on all three aspects, especially in redundancy. This indicates that by jointly scoring and selecting sentences, NEUSUM can produce summary with less content overlap since it re-estimates the saliency of remaining sentences considering both their contents and previously selected sentences.

6 Discussion

6.1 Precision at Step-t

We analyze the accuracy of sentence selection at each step. Since we extract 3 sentences at test time, we show how NEUSUM performs when extracting each sentence. Given a document D in test set $\mathfrak{T}$, NEUSUM predicted summary $\mathcal{S}$, its reference summary $\mathcal{S}^*$, and the extractive oracle summary $\mathcal{O}$ with respect to D and $\mathcal{S}^*$ (we use the method described in section 5.1 to construct $\mathcal{O}$), we define the precision at step t as $p(@t)$:

$$p(@t) = \frac{1}{|\mathfrak{T}|} \sum_{D \in \mathfrak{T}} \mathbf{1}_{\mathcal{O}}(\mathcal{S}[t]) \qquad (21)$$

[4] http://www.berouge.com/

[5] The ROUGE evaluation option is, -m -n 2

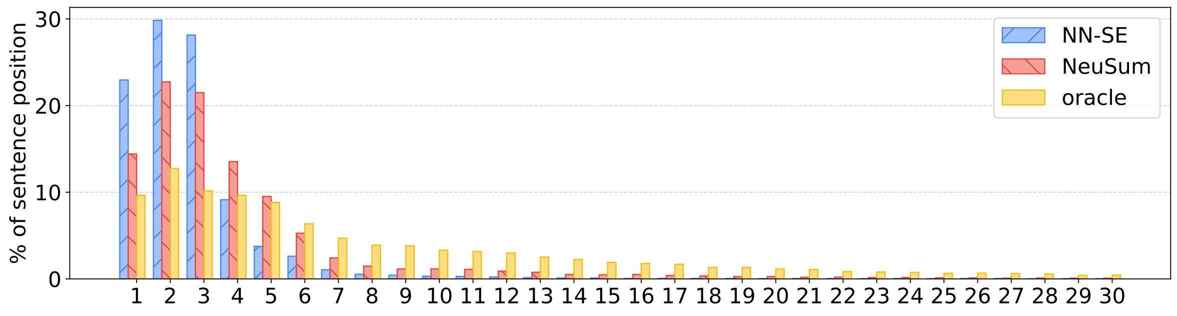


Figure 2: Position distribution of selected sentences of the NN-SE baseline, our NEUSUM model and oracle on the test set. We only draw the first 30 sentences since the average document length is 27.05.

where $\mathcal{S}[t]$ is the sentence extracted at step t, and $\mathbf{1}_{\mathcal{O}}$ is the indicator function defined as:

$$\mathbf{1}_{\mathcal{O}}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{O} \\ 0 & \text{if } x \notin \mathcal{O} \end{cases} \tag{22}$$

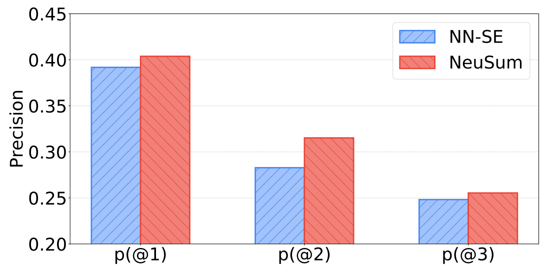


Figure 3: Precision of extracted sentence at step t of the NN-SE baseline and the NEUSUM model.

Figure 3 shows the precision at step t of NN-SE baseline and our NEUSUM. It can be observed that NEUSUM achieves better precision than the NN-SE baseline at each step. For the first sentence, both NEUSUM and NN-SE achieves good performance. The NN-SE baseline has 39.18% precision at the first step, and NEUSUM outperforms it by 1.2 points. At the second step, NEUSUM outperforms NN-SE by a large margin. In this step, the NEUSUM model extracts 31.52% sentences correctly, which is 3.24 percent higher than 28.28% of NN-SE. We think the second step selection benefits from the first step in NEUSUM since it can remember the selection history, while the separated models lack this ability.

However, we can notice the trend that the precision drops fast after each selection. We think this is due to two main reasons. First, we think that the error propagation leads to worse selection for the third selection. As shown in Figure 2, the $p(@1)$ and $p(@2)$ are 40.38% and 31.52% respectively, so the history is less reliable for the third selection. Second, intuitively, we think the later selections are more difficult compared to the previous ones since the most important sentences are already selected.

6.2 Position of Selected Sentences

Early works (Ren et al., 2017; Nallapati et al., 2017) have shown that sentence position is an important feature in extractive document summarization. Figure 2 shows the position distributions of the NN-SE baseline, our NEUSUM model and oracle on the *CNN/Daily Mail* test set. It can be seen that the NN-SE baseline model tends to extract large amount of leading sentences, especially the leading three sentences. According to the statistics, about 80.91% sentences selected by NN-SE baseline are in leading three sentences.

In the meanwhile, our NEUSUM model selects 58.64% leading three sentences. We can notice that in the oracle, the percentage of selecting leading sentences (sentence 1 to 5) is moderate, which is around 10%. Compared to NN-SE, the position of selected sentences in NEUSUM is closer to the oracle. Although NEUSUM also extracts more leading sentences than the oracle, it selects more tailing ones. For example, our NEUSUM model extracts more than 30% of sentences in the range of sentence 4 to 6. In the range of sentence 7 to 13, NN-SE barely extracts any sentences, but our NEUSUM model still extract sentences in this range. Therefore, we think this is one of the reasons why NEUSUM performs better than NN-SE.

We analyze the sentence position distribution and offer an explanation for these observations.

Intuitively, leading sentences are important for a well-organized article, especially for newswire articles. It is also well known that LEAD3 is a very strong baseline. In the training data, we found that 50.98% sentences labeled as “should be extracted” belongs to the first 5 sentences, which may cause the trained model tends to select more leading sentences. One possible situation is that one sentence in the tail of a document is more important than the leading sentences, but the margin between them is not large enough. The models which separately score and select sentences might not select sentences in the tail whose scores are not higher than the leading ones. These methods may choose the safer leading sentences as a fallback in such confusing situation because there is no direct competition between the leading and tailing candidates. In our NEUSUM model, the scoring and selection are jointly learned, and at each step the tailing candidates can compete directly with the leading ones. Therefore, NEUSUM can be more discriminating when dealing with this situation.

7 Conclusion

Conventional approaches to extractive document summarization contain two separated steps: sentence scoring and sentence selection. In this paper, we present a novel neural network framework for extractive document summarization by jointly learning to score and select sentences to address this issue. The most distinguishing feature of our approach from previous methods is that it combines sentence scoring and selection into one phase. Every time it selects a sentence, it scores the sentences according to the partial output summary and current extraction state. ROUGE evaluation results show that the proposed joint sentence scoring and selection approach significantly outperforms previous separated methods.

Acknowledgments

We thank three anonymous reviewers for their helpful comments. We also thank Danqing Huang, Chuanqi Tan, Zhirui Zhang, Shuangzhi Wu and Wei Jia for helpful discussions. The work of this paper is funded by the project of National Key Research and Development Program of China (No. 2017YFB1002102) and the project of National Natural Science Foundation of China (No. 91520204). The first author is funded by the Harbin Institute of Technology Scholarship Fund.

References

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and WANG Houfeng. 2015b. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 829–833.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP 2014*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

John M Conroy and Dianne P O’leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Eduard Hovy and Chin-Yew Lin. 1998. Automated text summarization and the summarist system. In *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, pages 197–214. Association for Computational Linguistics.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of 5th International Conference for Learning Representations*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference for Learning Representations*, San Diego.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564. Springer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 95–104, New York, NY, USA. ACM.

Pengjie Ren, Furu Wei, CHEN Zhumin, MA Jun, and Ming Zhou. 2016. A redundancy-aware sentence regression framework for extractive summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 33–43.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *Proceedings of the human language technology conference of the NAACL, Companion volume: Short papers*, pages 181–184. Association for Computational Linguistics.

Unsupervised Abstractive Meeting Summarization with Multi-Sentence Compression and Budgeted Submodular Maximization

Guokan Shang[1,2], **Wensi Ding**[1*], **Zekun Zhang**[1*], **Antoine J.-P. Tixier**[1], **Polykarpos Meladianos**[1,3], **Michalis Vazirgiannis**[1,3], **Jean-Pierre Lorré**[2]

[1]École Polytechnique, [2]Linagora, [3]AUEB

Abstract

We introduce a novel graph-based framework for abstractive meeting speech summarization that is fully unsupervised and does not rely on any annotations. Our work combines the strengths of multiple recent approaches while addressing their weaknesses. Moreover, we leverage recent advances in word embeddings and graph degeneracy applied to NLP to take exterior semantic knowledge into account, and to design custom diversity and informativeness measures. Experiments on the AMI and ICSI corpus show that our system improves on the state-of-the-art. Code and data are publicly available[1], and our system can be interactively tested[2].

1 Introduction

People spend a lot of their time in meetings. The ubiquity of web-based meeting tools and the rapid improvement and adoption of Automatic Speech Recognition (ASR) is creating pressing needs for effective meeting speech summarization mechanisms.

Spontaneous multi-party meeting speech transcriptions widely differ from traditional documents. Instead of grammatical, well-segmented *sentences*, the input is made of often ill-formed and ungrammatical text fragments called *utterances*. On top of that, ASR transcription and segmentation errors inject additional noise into the input.

In this paper, we combine the strengths of 6 approaches that had previously been applied to 3 different tasks (keyword extraction, multi-sentence compression, and summarization) into a unified, fully unsupervised end-to-end meeting speech summarization framework that can generate readable summaries despite the noise inherent to ASR transcriptions. We also introduce some novel components. Our method reaches state-of-the-art performance and can be applied to languages other than English in an almost out-of-the-box fashion.

2 Framework Overview

As illustrated in Figure 1, our system is made of 4 modules, briefly described in what follows.

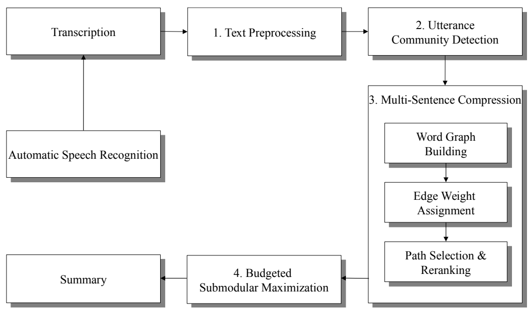

Transcription
1. Text Preprocessing
2. Utterance Community Detection
3. Multi-Sentence Compression
Word Graph Building
Edge Weight Assignment
Path Selection & Reranking
Automatic Speech Recognition
Summary
4. Budgeted Submodular Maximization

Figure 1: Overarching system pipeline.

The first module pre-processes text. The goal of the second *Community Detection* step is to group together the utterances that should be summarized by a common abstractive sentence (Murray et al., 2012). These utterances typically correspond to a topic or subtopic discussed during the meeting. A single abstractive sentence is then separately generated for each community, using an extension of the Multi-Sentence Compression Graph (MSCG) of Filippova (2010). Finally, we generate a summary by selecting the best elements from the set of abstractive sentences under a budget constraint. We cast this problem as the maximization of a custom submodular quality function.

* Work done as part of 3rd year project, with equal contribution.

[1] https://bitbucket.org/dascim/acl2018_abssumm

[2] http://datascience.open-paas.org/abs_summ_app

Note that our approach is fully unsupervised and does not rely on any annotations. Our input simply consists in a list of utterances without any metadata. All we need in addition to that is a part-of-speech tagger, a language model, a set of pre-trained word vectors, a list of stopwords and fillerwords, and optionally, access to a lexical database such as WordNet. Our system can work out-of-the-box in most languages for which such resources are available.

3 Related Work and Contributions

As detailed below, our framework combines the strengths of 6 recent works. It also includes novel components.

3.1 Multi-Sentence Compression Graph (MSCG) (Filippova, 2010)

Description: a fully unsupervised, simple approach for generating a short, self-sufficient sentence from a cluster of related, overlapping sentences. As shown in Figure 5, a word graph is constructed with special edge weights, the K-shortest weighted paths are then found and re-ranked with a scoring function, and the best path is used as the compression. The assumption is that redundancy alone is enough to ensure informativeness and grammaticality.
Limitations: despite making great strides and showing promising results, Filippova (2010) reported that 48% and 36% of the generated sentences were missing important information and were not perfectly grammatical.
Contributions: to respectively improve informativeness and grammaticality, we combine ideas found in Boudin and Morin (2013) and Mehdad et al. (2013), as described next.

3.2 More informative MSCG (Boudin and Morin, 2013)

Description: same task and approach as in Filippova (2010), except that a word co-occurrence network is built from the cluster of sentences, and that the PageRank scores of the nodes are computed in the manner of Mihalcea and Tarau (2004). The scores are then injected into the path re-ranking function to favor informative paths.
Limitations: PageRank is not state-of-the-art in capturing the importance of words in a document. Grammaticality is not considered.
Contributions: we take grammaticality into account as explained in subsection 3.4. We also follow recent evidence (Tixier et al., 2016a) that *spreading influence*, as captured by graph degeneracy-based measures, is better correlated with "keywordedness" than PageRank scores, as explained in the next subsection.

3.3 Graph-based word importance scoring (Tixier et al., 2016a)

Word co-occurrence network. As shown in Figure 2, we consider a word co-occurrence network as an undirected, weighted graph constructed by sliding a fixed-size window over text, and where edge weights represent co-occurrence counts (Tixier et al., 2016b; Mihalcea and Tarau, 2004).

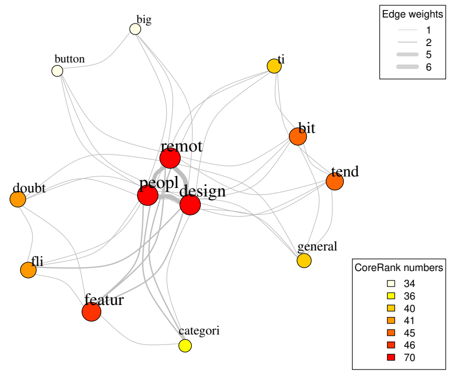


Figure 2: Word co-occurrence graph example, for the input text shown in Figure 5.

Important words are influential nodes. In social networks, it was shown that *influential spreaders*, that is, those individuals that can reach the largest part of the network in a given number of steps, are better identified via their core numbers rather than via their PageRank scores or degrees (Kitsak et al., 2010). See Figure 3 for the intuition. Similarly, in NLP, Tixier et al. (2016a) have shown that keywords are better identified via their core numbers rather than via their TextRank scores, that is, keywords are *influencers* within their word co-occurrence network.
Graph degeneracy (Seidman, 1983). Let $G(V, E)$ be an undirected, weighted graph with $n = |V|$ nodes and $m = |E|$ edges. A k-core of G is a maximal subgraph of G in which every vertex v has at least weighted degree k. As shown in Figures 3 and 4, the k-core decomposition of G forms a hierarchy of nested subgraphs whose cohesiveness and size respectively increase and decrease with k. The higher-level cores can be viewed as a *filtered version* of the graph that

excludes noise. This property is highly valuable when dealing with graphs constructed from noisy text, like utterances. The core number of a node is the highest order of a core that contains this node.

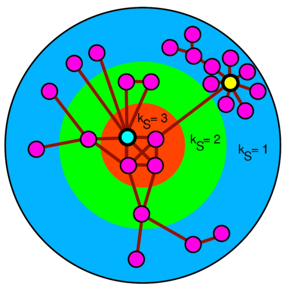
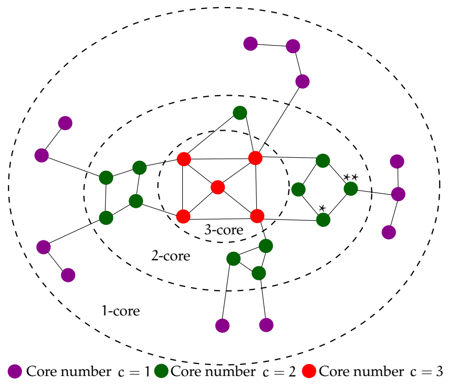


Figure 3: k-core decomposition. The blue and the yellow nodes have same degree and similar PageRank numbers. However, the blue node is a much more influential spreader as it is strategically placed in the core of the network, as captured by its higher core number.

The CoreRank number of a node (Tixier et al., 2016a; Bae and Kim, 2014) is defined as the sum of the core numbers of its neighbors. As shown in Figure 4, CoreRank more finely captures the structural position of each node in the graph than raw core numbers. Also, stabilizing scores across node neighborhoods enhances the inherent noise robustness property of graph degeneracy, which is desirable when working with noisy speech-to-text output.



$\star\star$
$\star$
3-core
2-core
1-core
● Core number $c = 1$ ● Core number $c = 2$ ● Core number $c = 3$

Figure 4: Value added by CoreRank: while nodes $\star$ and $\star\star$ have the same core number (=2), node $\star$ has a greater CoreRank score (3+2+2=7 vs 2+2+1=5), which better reflects its more central position in the graph.

Time complexity. Building a graph-of-words is $\mathcal{O}(nW)$, and computing the weighted k-core decomposition of a graph requires $\mathcal{O}(m \log(n))$ (Batagelj and Zaveršnik, 2002). For small pieces of text, this two step process is so affordable that it can be used in real-time (Meladianos et al., 2017). Finally, computing CoreRank scores can be done with only a small overhead of $\mathcal{O}(n)$, provided that the graph is stored as a hash of adjacency lists. Getting the CoreRank numbers from scratch for a community of utterances is therefore very fast, especially since typically in this context, $n \sim 10$ and $m \sim 100$.

3.4 Fluency-aware, more abstractive MSCG (Mehdad et al., 2013)

Description: a *supervised* end-to-end framework for abstractive meeting summarization. Community Detection is performed by (1) building an utterance graph with a logistic regression classifier, and (2) applying the CONGA algorithm. Then, before performing sentence compression with the MSCG, the authors also (3) build an entailment graph with a SVM classifier in order to eliminate redundant and less informative utterances. In addition, the authors propose the use of WordNet (Miller, 1995) during the MSCG building phase to capture lexical knowledge between words and thus generate more abstractive compressions, and of a language model when re-ranking the shortest paths, to favor fluent compressions.
Limitations: this effort was a significant advance, as it was the first application of the MSCG to the meeting summarization task, to the best of our knowledge. However, steps (1) and (3) above are complex, based on handcrafted features, and respectively require annotated training data in the form of links between human-written abstractive sentences and original utterances and multiple external datasets (e.g., from the Recognizing Textual Entailment Challenge). Such annotations are costly to obtain and very seldom available in practice.
Contributions: while we retain the use of WordNet and of a language model, we show that, without deteriorating the quality of the results, steps (1) and (2) above (Community Detection) can be performed in a much more simple, completely unsupervised way, and that step (3) can be removed. That is, the MSCG is powerful enough to remove redundancy and ensure informativeness, should proper edge weights and path re-ranking function be used.

In addition to the aforementioned contributions, we also introduce the following novel components into our abstractive summarization pipeline:

• we inject global exterior knowledge into the edge weights of the MSCG, by using the *Word Attraction Force* of Wang et al. (2014), based on distance in the word embedding space,

- we add a diversity term to the path re-ranking function, that measures how many unique clusters in the embedding space are visited by each path,
- rather than using all the abstractive sentences as the final summary like in Mehdad et al. (2013), we maximize a custom submodular function to select a subset of abstractive sentences that is near-optimal given a budget constraint (summary size). A brief background of submodularity in the context of summarization is provided next.

3.5 Submodularity for summarization (Lin and Bilmes, 2010; Lin, 2012)

Selecting an optimal subset of abstractive sentences from a larger set can be framed as a budgeted submodular maximization task:

$$\underset{S \subseteq \mathcal{S}}{\operatorname{argmax}}\, f(S) | \sum_{s \in S} c_s \leq \mathcal{B} \tag{1}$$

where S is a summary, c_s is the cost (word count) of sentence s, $\mathcal{B}$ is the desired summary size in words (budget), and f is a summary quality scoring set function, which assigns a single numeric score to a summary S.

This combinatorial optimization task is NP-hard. However, near-optimal performance can be guaranteed with a modified greedy algorithm (Lin and Bilmes, 2010) that iteratively selects the sentence s that maximizes the ratio of quality function gain to scaled cost $f(S \cup s) - f(S) / c_s^r$ (where S is the current summary and $r \geq 0$ is a scaling factor).

In order for the performance guarantees to hold however, f has to be *submodular* and *monotone non-decreasing*. Our proposed f is described in subsection 4.4.

4 Our Framework

We detail next each of the four modules in our architecture (shown in Figure 1).

4.1 Text preprocessing

We adopt preprocessing steps tailored to the characteristics of ASR transcriptions. Consecutive repeated unigrams and bigrams are reduced to single terms. Specific ASR tags, such as {*vocalsound*}, {*pause*}, *and* {*gap*} are filtered out. In addition, filler words, such as *uh-huh*, *okay*, *well*, and *by the way* are also discarded. Consecutive stopwords at the beginning and end of utterances are stripped. In the end, utterances that contain less than 3 non-stopwords are pruned out. The surviving utterances are used for the next steps.

4.2 Utterance community detection

The goal here is to cluster utterances into communities that should be summarized by a common abstractive sentence.

We initially experimented with techniques capitalizing on word vectors, such as k-means and hierarchical clustering based on the Euclidean distance or the Word Mover's Distance (Kusner et al., 2015). We also tried graph-based approaches, such as community detection in a complete graph where nodes are utterances and edges are weighted based on the aforementioned distances.

Best results were obtained, however, with a simple approach in which utterances are projected into the vector space and assigned standard TF-IDF weights. Then, the dimensionality of the utterance-term matrix is reduced with Latent Semantic Analysis (LSA), and finally, the k-means algorithm is applied. Note that LSA is only used here, during the utterance community detection phase, to remove noise and stabilize clustering. We do not use a topic graph in our approach.

We think using word embeddings was not effective, because in meeting speech, as opposed to traditional documents, participants tend to use the same term to refer to the same thing throughout the entire conversation, as noted by Riedhammer et al. (2010), and as verified in practice. This is probably why, for clustering utterances, capturing synonymy is counterproductive, as it artificially reduces the distance between every pair of utterances and blurs the picture.

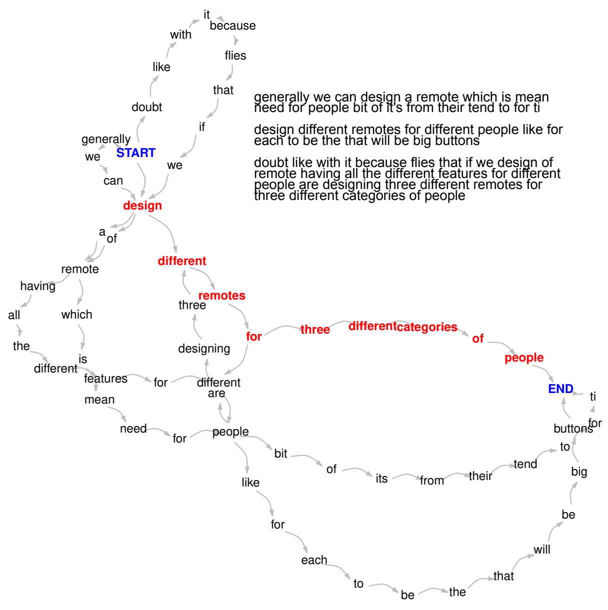
4.3 Multi-Sentence Compression

The following steps are performed separately for each community.

Word importance scoring

From a processed version of the community (stemming and stopword removal), we construct an undirected, weighted word co-occurrence network as described in subsection 3.3. We use a sliding window of size $W = 6$ not overspanning utterances. Note that stemming is performed only here, and for the sole purpose of building the word co-occurrence network.

We then compute the CoreRank numbers of the nodes as described in subsection 3.3.



generally we can design a remote which is mean need for people bit of it's from their tend to for ti

design different remotes for different people like for each to be the that will be big buttons

doubt like with it because flies that if we design of remote having all the different features for different people are designing three different remotes for three different categories of people

Figure 5: Compressed sentence (in **bold red**) generated by our multi-sentence compression graph (MSCG) for a 3-utterance community from meeting IS1009b of the AMI corpus. Using Filippova (2010)'s weighting and re-ranking scheme here would have selected another path: *design different remotes for different people bit of it's from their tend to for ti*. Note that the compressed sentence does not appear in the initial set of utterances, and is compact and grammatical, despite the redundancy, transcription and segmentation errors of the input. The *abstractive* and *robust* nature of the MSCG makes it particularly well-suited to the meeting domain.

We finally reweigh the CoreRank scores, indicative of word importance within a given community, with a quantity akin to an *Inverse Document Frequency*, where communities serve as documents and the full meeting as the collection. We thus obtain something equivalent to the TW-IDF weighting scheme of Rousseau and Vazirgiannis (2013), where the CoreRank scores are the term weights TW:

$$TW\text{-}IDF(t,d,D) = TW(t,d) \times IDF(t,D) \tag{2}$$

where t is a term belonging to community d, and D is the set of all utterance communities. We compute the IDF as $IDF(t,D) = 1 + \log|D|/D_t$, where $|D|$ is the number of communities and D_t the number of communities containing t.

The intuition behind this reweighing scheme is that a term should be considered important within a given meeting if it has a high CoreRank score within its community *and* if the number of communities in which the term appears is relatively small.

Word graph building

The backbone of the graph is laid out as a directed sequence of nodes corresponding to the words in the first utterance, with special `START` and `END` nodes at the beginning and at the end (see Figure 5). Edge direction follows the natural flow of text. Words from the remaining utterances are then iteratively added to the graph (between the `START` and `END` nodes) based on the following rules:

1) if the word is a **non-stopword**, the word is mapped onto an existing node if it has the same lowercased form and the same part-of-speech tag[3]. In case of multiple matches, we check the immediate context (the preceding and following words in the utterance and the neighboring nodes in the graph), and we pick the node with the largest context overlap or which has the greatest number of words already mapped to it (when no overlap). When there is no match, we use WordNet as described in Appendix A.

2) if the word is a **stopword** and there is a match, it is mapped only if there is an overlap of at least one non-stopword in the immediate context. Otherwise, a new node is created.

Finally, note that any two words appearing within the same utterance cannot be mapped to the same node. This ensures that every utterance is a loopless path in the graph. Of course, there are many more paths in the graphs than original utterances.

Edge Weight Assignment

Once the word graph is constructed, we assign weights to its edges as:

$$w'''(p_i, p_j) = \frac{w'(p_i, p_j)}{w''(p_i, p_j)} \tag{3}$$

where p_i and p_j are two neighbors in the MSCG. As detailed next, those weights combine *local co-occurrence statistics* (numerator) with *global exterior knowledge* (denominator). Note that the lower

[3]We used NLTK's averaged perceptron tagger, available at: http://www.nltk.org/api/nltk.tag.html#module-nltk.tag.perceptron

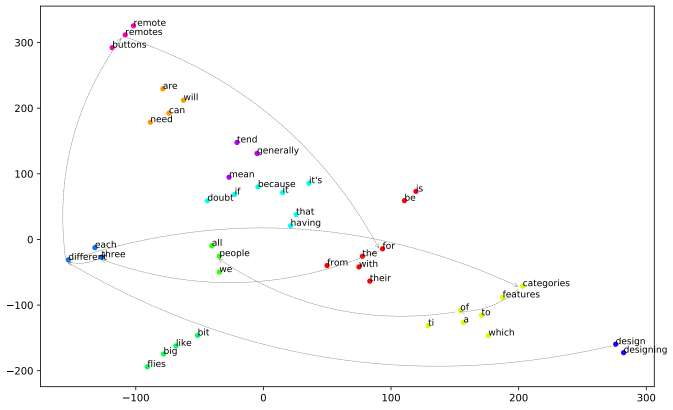


Figure 6: t-SNE visualization (Maaten and Hinton, 2008) of the Google News vectors of the words in the utterance community shown in Figure 5. Arrows join the words in the best compression path shown in Figure 5. Movements in the embedding space, as measured by the number of unique clusters covered by the path (here, 6/11), provide a sense of the diversity of the compressed sentence, as formalized in Equation 10.

the weight of an edge, the better.

Local co-occurrence statistics.
We use Filippova (2010)'s formula:

$$w'(p_i, p_j) = \frac{f(p_i) + f(p_j)}{\sum_{P \in G', p_i, p_j \in P} \text{diff}(P, p_i, p_j)^{-1}} \tag{4}$$

where $f(p_i)$ is the number of words mapped to node p_i in the MSCG G', and $\text{diff}(P, p_i, p_j)^{-1}$ is the inverse of the distance between p_i and p_j in a path P (in number of hops). This weighting function favors edges between infrequent words that frequently appear close to each other in the text (the lower, the better).

Global exterior knowledge.
We introduce a second term based on the *Word Attraction Force score* of Wang et al. (2014):

$$w''(p_i, p_j) = \frac{f(p_i) \times f(p_j)}{d^2_{p_i, p_j}} \tag{5}$$

where d_{p_i,p_j} is the Euclidean distance between the words mapped to p_i and p_j in a word embedding space[4]. This component favor paths going through salient words that have *high semantic similarity* (the higher, the better). The goal is to ensure readability of the compression, by avoiding to generate a sentence jumping from one word to a completely unrelated one.

Path re-ranking

As in Boudin and Morin (2013), we use a shortest weighted path algorithm to find the K paths between the `START` and `END` symbols having the lowest cumulative edge weight:

$$W(P) = \sum_{i=1}^{|P|-1} w'''(p_i, p_{i+1}) \tag{6}$$

Where $|P|$ is the number of nodes in the path. Paths having less than z words or that do not contain a verb are filtered out (z is a tuning parameter). However, unlike in Boudin and Morin (2013), we rerank the K best paths with the following novel weighting scheme (the lower, the better), and the path with the lowest score is used as the compression:

$$\text{score}(P) = \frac{W(P)}{|P| \times F(P) \times C(P) \times D(P)} \tag{7}$$

The denominator takes into account the length of the path, and its fluency (F), coverage (C), and diversity (D). F, C, and D are detailed in what follows.

Fluency. We estimate the grammaticality of a path with an n-gram language model. In our experiments, we used a trigram model[5]:

$$F(P) = \frac{\sum_{i=1}^{|P|} \log Pr(p_i | p_{i-n+1}^{i-1})}{\#n\text{-}gram} \tag{8}$$

where $|P|$ denote path length, and p_i and $\#n\text{-}gram$ are respectively the words and number of n-grams in the path.

Coverage. We reward the paths that visit important nouns, verbs and adjectives:

$$C(P) = \frac{\sum_{p_i \in P} \text{TW-IDF}(p_i)}{\#p_i} \tag{9}$$

where $\#p_i$ is the number of nouns, verbs and adjectives in the path. The TW-IDF scores are computed as explained in subsection 4.3.

Diversity. We cluster all words from the MSCG in the word embedding space by applying the k-means algorithm. We then measure the diversity of the vocabulary contained in a path as the number

[4]GoogleNews vectors `https://code.google.com/archive/p/word2vec`

[5]CMUSphinx English LM: `https://cmusphinx.github.io`

of unique clusters visited by the path, normalized by the length of the path:

$$D(P) = \frac{\sum_{j=1}^{k} 1_{\exists p_i \in P | p_i \in \text{cluster}_j}}{|P|} \tag{10}$$

The graphical intuition for this measure is provided in Figure 6. Note that we do not normalize D by the total number of clusters (only by path length) because k is fixed for all candidate paths.

4.4 Budgeted submodular maximization

We apply the previous steps separately for all utterance communities, which results in a set $\mathcal{S}$ of abstractive sentences (one for each community). This set of sentences can already be considered to be a summary of the meeting. However, it might exceed the maximum size allowed, and still contain some redundancy or off-topic sections unrelated to the general theme of the meeting (e.g., chit-chat).

Therefore, we design the following *submodular* and *monotone non-decreasing* objective function:

$$f(S) = \sum_{s_i \in S} n_{s_i} w_{s_i} + \lambda \sum_{j=1}^{k} 1_{\exists s_i \in S | s_i \in group_j} \tag{11}$$

where $\lambda \geq 0$ is the trade-off parameter, n_{s_i} is the number of occurrences of word s_i in S, and w_{s_i} is the CoreRank score of s_i.

Then, as explained in subsection 3.5, we obtain a near-optimal subset of abstractive sentences by maximizing f with a greedy algorithm. CoreRank scores and clusters are found as previously described, except that this time they are obtained from the full processed meeting transcription rather than from a single utterance community.

5 Experimental setup

5.1 Datasets

We conducted experiments on the widely-used AMI (McCowan et al., 2005) and ICSI (Janin et al., 2003) benchmark datasets. We used the traditional test sets of 20 and 6 meetings respectively for the AMI and ICSI corpora (Riedhammer et al., 2008). Each meeting in the AMI test set is associated with a human abstractive summary of 290 words on average, whereas each meeting in the ICSI test set is associated with 3 human abstractive summaries of respective average sizes 220, 220 and 670 words. For parameter tuning, we constructed development sets of 47 and 25 meetings, respectively for AMI and ICSI, by randomly sampling from the training sets. The word error rate of the ASR transcriptions is respectively of 36% and 37% for AMI and ICSI.

5.2 Baselines

We compared our system against 7 baselines, which are listed below and more thoroughly detailed in Appendix B. Note that preprocessing was exactly the same for our system and all baselines.

• **Random** and **Longest Greedy** are basic baselines recommended by (Riedhammer et al., 2008),
• **TextRank** (Mihalcea and Tarau, 2004),
• **ClusterRank** (Garg et al., 2009),
• **CoreRank & PageRank submodular** (Tixier et al., 2017),
• **Oracle** is the same as the random baseline, but uses the human extractive summaries as input.

In addition to the baselines above, we included in our comparison 3 variants of our system using different MSCGs: **Our System (Baseline)** uses the original MSCG of Filippova (2010), **Our System (KeyRank)** uses that of Boudin and Morin (2013), and **Our System (FluCovRank)** that of Mehdad et al. (2013). Details about each approach were given in Section 3.

5.3 Parameter tuning

For *Our System* and each of its variants, we conducted a grid search on the development sets of each corpus, for fixed summary sizes of 350 and 450 words (AMI and ICSI). We searched the following parameters:

• n: number of utterance communities (see Section 4.2). We tested values of n ranging from 20 to 60, with steps of 5. This parameter controls how much abstractive should the summary be. If all utterances are assigned to their own singleton community, the MSCG is of no utility, and our framework is extractive. It becomes more and more abstractive as the number of communities decreases.
• z: minimum path length (see Section 4.3). We searched values in the range $[6, 16]$ with steps of 2. If a path is shorter than a certain minimum number of words, it often corresponds to an invalid sentence, and should thereby be filtered out.
• λ and r, the trade-off parameter and the scaling factor (see Section 4.4). We searched $[0, 1]$ and $[0, 2]$ (respectively) with steps of 0.1. The parameter λ plays a regularization role favoring diversity.

The scaling factor makes sure the quality function gain and utterance cost are comparable.

The best parameter values for each corpus are summarized in Table 1. λ is mostly non-zero, indicating that it is necessary to include a regularization term in the submodular function. In some cases though, r is equal to zero, which means that utterance costs are not involved in the greedy decision heuristic. These observations contradict the conclusion of Lin (2012) that $r = 0$ cannot give best results.

System	AMI	ICSI
Our System	50, 8, (0.7, 0.5)	40, 14, (0.0, 0.0)
Our System (Baseline)	50, 12, (0.3, 0.5)	45, 14, (0.1, 0.0)
Our System (KeyRank)	50, 10, (0.2, 0.9)	45, 12, (0.3, 0.4)
Our System (FluCovRank)	35, 6, (0.4, 1.0)	50, 10, (0.2, 0.3)

Table 1: Optimal parameter values $n, z, (\lambda, r)$.

Apart from the tuning parameters, we set the number of LSA dimensions to 30 and 60 (resp. on AMI and ISCI). The small number of LSA dimensions retained can be explained by the fact that the AMI and ICSI transcriptions feature 532 and 1126 unique words on average, which is much smaller than traditional documents. This is due to relatively small meeting duration, and to the fact that participants tend to stick to the same terms throughout the entire conversation. For the k-means algorithm, k was set equal to the minimum path length z when doing MSCG path re-ranking (see Equation 10), and to 60 when generating the final summary (see Equation 11).

Following Boudin and Morin (2013), the number of shortest weighted paths K was set to 200, which is greater than the $K = 100$ used by Filippova (2010). Increasing K from 100 improves performance with diminishing returns, but significantly increases complexity. We empirically found 200 to be a good trade-off.

6 Results and Interpretation

Metrics. We evaluated performance with the widely-used ROUGE-1, ROUGE-2 and ROUGE-SU4 metrics (Lin, 2004). These metrics are respectively based on unigram, bigram, and unigram plus skip-bigram overlap with maximum skip distance of 4, and have been shown to be highly correlated with human evaluations (Lin, 2004). ROUGE-2 scores can be seen as a measure of summary readability (Lin and Hovy, 2003; Ganesan et al., 2010). ROUGE-SU4 does not require consecutive matches but is still sensitive to word order.

Macro-averaged results for summaries generated from automatic transcriptions can be seen in Figure 7 and Table 2. Table 2 provides detailed comparisons over the fixed budgets that we used for parameter tuning, while Figure 7 shows the performance of the models for budgets ranging from 150 to 500 words. The same information for summaries generated from manual transcriptions is available in Appendix C. Finally, summary examples are available in Appendix D.

ROUGE-1. Our systems outperform all baselines on AMI (including *Oracle*) and all baselines on ICSI (except *Oracle*). Specifically, *Our System* is best on ICSI, while *Our System (KeyRank)* is superior on AMI. We can also observe on Figure 7 that our systems are consistently better throughout the different summary sizes, even though their parameters were tuned for specific sizes only. This shows that the best parameter values are quite robust across the entire budget range.

ROUGE-2. Again, our systems (except *Our System (Baseline)*) outperform all baselines, except *Oracle*. In addition, *Our System* and *Our System (FluCovRank)* consistently improve on *Our System (Baseline)*, which proves that the novel components we introduce improve summary fluency.

ROUGE-SU4. ROUGE-SU4 was used to measure the amount of in-order word pairs overlapping. Our systems are competitive with all baselines, including *Oracle*. Like with ROUGE-1, *Our System* is better than *Our System (KeyRank)* on ICSI, whereas the opposite is true on AMI.

General remarks.

• The summaries of all systems except *Oracle* were generated from noisy ASR transcriptions, but were compared against human abstractive summaries. ROUGE being based on word overlap, it makes it very difficult to reach very high scores, because many words in the ground truth summaries do not appear in the transcriptions at all.

• The scores of all systems are lower on ICSI than on AMI. This can be explained by the fact that on ICSI, the system summaries have to jointly match 3 human abstractive summaries of different content and size, which is much more difficult than matching a single summary.

• Our framework is very competitive to *Oracle*, which is notable since the latter has direct access to the human extractive summaries. Note that *Or-*

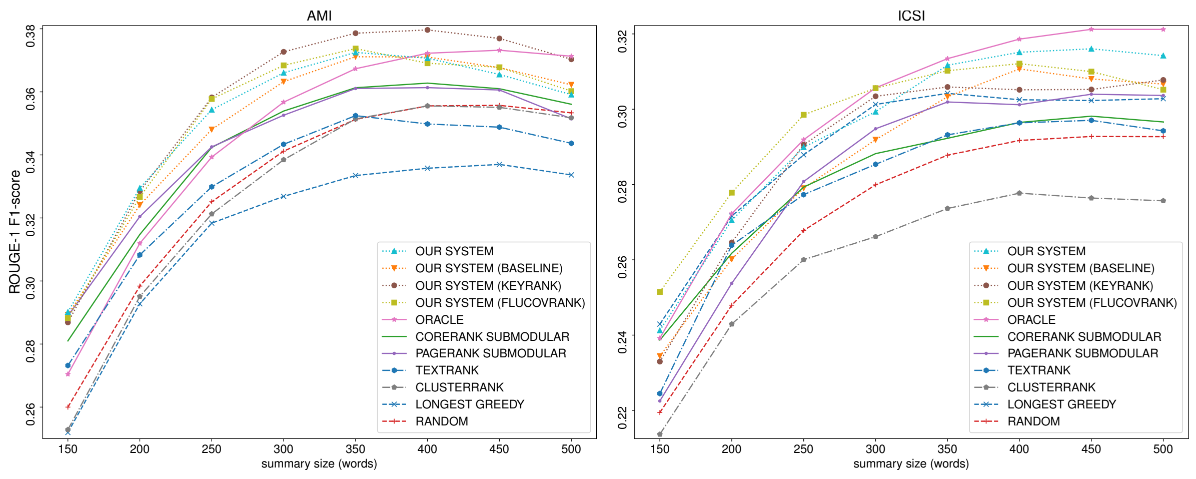


Figure 7: ROUGE-1 F-1 scores for various budgets (ASR transcriptions).

	AMI ROUGE-1			AMI ROUGE-2			AMI ROUGE-SU4			ICSI ROUGE-1			ICSI ROUGE-2			ICSI ROUGE-SU4		
	R	P	F-1	R	P	F-1	R	P	F-1	R	P	F-1	R	P	F-1	R	P	F-1
Our System	41.83	34.44	37.25	8.22	6.95	7.43	15.83	13.70	14.51	36.99	28.12	**31.60**	5.41	4.39	4.79	13.10	10.17	**11.35**
Our System (Baseline)	41.56	34.37	37.11	7.88	6.66	7.11	15.36	13.20	14.02	36.39	27.20	30.80	5.19	4.12	4.55	12.59	9.70	10.86
Our System (KeyRank)	42.43	35.01	**37.86**	8.72	7.29	**7.84**	16.19	13.76	**14.71**	35.95	27.00	30.52	4.64	3.64	4.04	12.43	9.23	10.50
Our System (FluCovRank)	41.84	34.61	37.37	8.29	6.92	7.45	16.28	13.48	14.58	36.27	27.56	31.00	5.56	4.35	**4.83**	13.47	9.85	11.29
Oracle	40.49	34.65	**36.73**	8.07	7.35	**7.55**	15.00	14.03	**14.26**	37.91	28.39	**32.12**	5.73	4.82	**5.18**	13.35	10.73	**11.80**
CoreRank Submodular	41.14	32.93	36.13	8.06	6.88	7.33	14.84	13.91	14.18	35.22	26.34	29.82	4.36	3.76	4.00	12.11	9.58	10.61
PageRank Submodular	40.84	33.08	36.10	8.27	6.88	7.42	15.37	13.71	14.32	36.05	26.69	30.40	4.82	4.16	4.42	12.19	10.39	11.14
TextRank	39.55	32.60	35.25	7.67	6.43	6.90	14.87	12.87	13.62	34.89	26.33	29.70	4.60	3.74	4.09	12.42	9.43	10.64
ClusterRank	39.36	32.53	35.14	7.14	6.05	6.46	14.34	12.80	13.35	32.63	24.44	27.64	4.03	3.44	3.68	11.04	8.88	9.77
Longest Greedy	37.31	30.93	33.35	5.77	4.71	5.11	13.79	11.11	12.15	35.57	26.74	30.23	4.84	3.88	4.27	13.09	9.46	10.90
Random	39.42	32.48	35.13	6.88	5.89	6.26	14.07	12.70	13.17	34.78	25.75	29.28	4.19	3.51	3.78	11.61	9.37	10.29

Table 2: Macro-averaged results for 350 and 450 word summaries (ASR transcriptions).

acle does not reach very high ROUGE scores because the overlap between the human extractive and abstractive summaries is low (19% and 29%, respectively on AMI and ICSI test sets).

7 Conclusion and Next Steps

Our framework combines the strengths of 6 approaches that had previously been applied to 3 different tasks (keyword extraction, multi-sentence compression, and summarization) into a unified, fully unsupervised end-to-end summarization framework, and introduces some novel components. Rigorous evaluation on the AMI and ICSI corpora shows that we reach state-of-the-art performance, and generate reasonably grammatical abstractive summaries despite taking noisy utterances as input and not relying on any annotations or training data. Finally, thanks to its fully unsupervised nature, our method is applicable to other languages than English in an almost out-of-the-box manner.

Our framework was developed for the meeting domain. Indeed, our generative component, the multi-sentence compression graph (MSCG), needs redundancy to perform well. Such redundancy is typically present in meeting speech but not in traditional documents. In addition, the MSCG is by design robust to noise, and our custom path re-ranking strategy, based on graph degeneracy, makes it even more robust to noise. As a result, our framework is advantaged on ASR input. Finally, we use a language model to favor fluent paths, which is crucial when working with (meeting) speech but not that important when dealing with well-formed input.

Future efforts should be dedicated to improving the community detection phase and generating more abstractive sentences, probably by harnessing Deep Learning. However, the lack of large training sets for the meeting domain is an obstacle to the use of neural approaches.

Acknowledgments

We are grateful to the four anonymous reviewers for their detailed and constructive feedback. This research was supported in part by the OpenPaaS::NG project.

References

Joonhyun Bae and Sangwook Kim. 2014. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Physica A: Statistical Mechanics and its Applications* 395:549–559.

Vladimir Batagelj and Matjaž Zaveršnik. 2002. Generalized cores. *arXiv preprint cs/0202039* .

Florian Boudin and Emmanuel Morin. 2013. Keyphrase extraction for n-best reranking in multi-sentence compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 298–305. http://aclweb.org/anthology/N13-1030.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, pages 322–330. http://aclweb.org/anthology/C10-1037.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, pages 340–348. http://aclweb.org/anthology/C10-1039.

Nikhil Garg, Benoit Favre, Korbinian Reidhammer, and Dilek Hakkani-Tür. 2009. Clusterrank: a graph based method for meeting summarization. In *Tenth Annual Conference of the International Speech Communication Association*.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The icsi meeting corpus. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*. volume 1, pages I–364–I–367 vol.1. https://doi.org/10.1109/ICASSP.2003.1198793.

Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. 2010. Identification of influential spreaders in complex networks. *Nature Physics* 6(11):888–893. https://doi.org/10.1038/nphys1746.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, ICML'15, pages 957–966.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. http://aclweb.org/anthology/W04-1013.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. http://aclweb.org/anthology/N03-1020.

Hui Lin. 2012. *Submodularity in natural language processing: algorithms and applications*. University of Washington.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 912–920. http://aclweb.org/anthology/N10-1134.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Iain McCowan, Jean Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, et al. 2005. The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*. volume 88.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. NG. 2013. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 136–146. http://aclweb.org/anthology/W13-2117.

Polykarpos Meladianos, Antoine Tixier, Ioannis Nikolentzos, and Michalis Vazirgiannis. 2017. Real-time keyword extraction from conversations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 462–467. http://aclweb.org/anthology/E17-2074.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. http://aclweb.org/anthology/W04-3252.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41. https://doi.org/10.1145/219717.219748.

Gabriel Murray, Giuseppe Carenini, and Raymond Ng. 2012. Using the omega index for evaluating abstractive community detection. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. Association for Computational Linguistics, pages 10–18. http://aclweb.org/anthology/W12-2602.

Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür. 2010. Long story short - global unsupervised models for keyphrase based meeting summarization. *Speech Commun.* 52(10):801–815. https://doi.org/10.1016/j.specom.2010.06.002.

Korbinian Riedhammer, Dan Gillick, Benoit Favre, and Dilek Hakkani-Tür. 2008. Packing the meeting summarization knapsack. In *Ninth Annual Conference of the International Speech Communication Association*.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*. ACM, New York, NY, USA, CIKM '13, pages 59–68. https://doi.org/10.1145/2505515.2505671.

Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5(3):269–287. https://doi.org/10.1016/0378-8733(83)90028-X.

Antoine Tixier, Fragkiskos Malliaros, and Michalis Vazirgiannis. 2016a. A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1860–1870. https://doi.org/10.18653/v1/D16-1191.

Antoine Tixier, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Combining graph degeneracy and submodularity for unsupervised extractive summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, pages 48–58. http://aclweb.org/anthology/W17-4507.

Antoine Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. 2016b. Gowvis: A web application for graph-of-words-based text visualization and summarization. In *Proceedings of ACL-2016 System Demonstrations*. Association for Computational Linguistics, pages 151–156. https://doi.org/10.18653/v1/P16-4026.

Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*. volume 39.

Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting

Yen-Chun Chen and **Mohit Bansal**
UNC Chapel Hill
{yenchun, mbansal}@cs.unc.edu

Abstract

Inspired by how humans summarize long documents, we propose an accurate and fast summarization model that first selects salient sentences and then rewrites them abstractively (i.e., compresses and paraphrases) to generate a concise overall summary. We use a novel sentence-level policy gradient method to bridge the non-differentiable computation between these two neural networks in a hierarchical way, while maintaining language fluency. Empirically, we achieve the new state-of-the-art on all metrics (including human evaluation) on the CNN/Daily Mail dataset, as well as significantly higher abstractiveness scores. Moreover, by first operating at the sentence-level and then the word-level, we enable *parallel decoding* of our neural generative model that results in substantially faster (10-20x) inference speed as well as 4x faster training convergence than previous long-paragraph encoder-decoder models. We also demonstrate the generalization of our model on the test-only DUC-2002 dataset, where we achieve higher scores than a state-of-the-art model.

1 Introduction

The task of document summarization has two main paradigms: extractive and abstractive. The former method directly chooses and outputs the salient sentences (or phrases) in the original document (Jing and McKeown, 2000; Knight and Marcu, 2000; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011). The latter abstractive approach involves rewriting the summary (Banko et al., 2000; Zajic et al., 2004), and has seen substantial recent gains due to neural sequence-to-sequence models (Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2018). Abstractive models can be more concise by performing generation from scratch, but they suffer from slow and inaccurate encoding of very long documents, with the attention model being required to look at all encoded words (in long paragraphs) for decoding each generated summary word (slow, one by one sequentially). Abstractive models also suffer from redundancy (repetitions), especially when generating multi-sentence summary.

To address both these issues and combine the advantages of both paradigms, we propose a hybrid extractive-abstractive architecture, with policy-based reinforcement learning (RL) to bridge together the two networks. Similar to how humans summarize long documents, our model first uses an *extractor agent* to select salient sentences or highlights, and then employs an *abstractor network* to rewrite (i.e., compress and paraphrase) each of these extracted sentences. To overcome the non-differentiable behavior of our extractor and train on available document-summary pairs without saliency label, we next use actor-critic policy gradient with sentence-level metric rewards to connect these two neural networks and to learn sentence saliency. We also avoid common language fluency issues (Paulus et al., 2018) by preventing the policy gradients from affecting the abstractive summarizer's word-level training, which is supported by our human evaluation study. Our sentence-level reinforcement learning takes into account the word-sentence hierarchy, which better models the language structure and makes parallelization possible. Our extractor combines reinforcement learning and pointer networks, which is inspired by Bello et al. (2017)'s attempt to solve the Traveling Salesman Problem. Our abstractor is a simple encoder-aligner-decoder model (with copying) and is trained on pseudo document-summary sentence pairs obtained via simple automatic matching criteria.

Thus, our method incorporates the abstractive paradigm's advantages of concisely rewriting sentences and generating novel words from the full vocabulary, yet it adopts intermediate extractive behavior to improve the overall model's quality, speed, and stability. Instead of encoding and attending to every word in the long input document sequentially, our model adopts a human-inspired *coarse-to-fine* approach that first extracts all the salient sentences and then decodes (rewrites) them (*in parallel*). This also avoids almost all redundancy issues because the model has already chosen non-redundant salient sentences to abstractively summarize (but adding an optional final reranker component does give additional gains by removing the fewer across-sentence repetitions).

Empirically, our approach is the new state-of-the-art on all ROUGE metrics (Lin, 2004) as well as on METEOR (Denkowski and Lavie, 2014) of the CNN/Daily Mail dataset, achieving statistically significant improvements over previous models that use complex long-encoder, copy, and coverage mechanisms (See et al., 2017). The test-only DUC-2002 improvement also shows our model's better generalization than this strong abstractive system. In addition, we surpass the popular lead-3 baseline on all ROUGE scores with an abstractive model. Moreover, our sentence-level abstractive rewriting module also produces substantially more (3x) novel N-grams that are not seen in the input document, as compared to the strong flat-structured model of See et al. (2017). This empirically justifies that our RL-guided extractor has learned sentence saliency, rather than benefiting from simply copying longer sentences. We also show that our model maintains the same level of fluency as a conventional RNN-based model because the reward does not leak to our abstractor's word-level training. Finally, our model's training is 4x and inference is more than 20x faster than the previous state-of-the-art. The optional final reranker gives further improvements while maintaining a 7x speedup.

Overall, our contribution is three fold: First we propose a novel sentence-level RL technique for the well-known task of abstractive summarization, effectively utilizing the word-then-sentence hierarchical structure without annotated matching sentence-pairs between the document and ground truth summary. Next, our model achieves the new state-of-the-art on all metrics of multiple versions of a popular summarization dataset (as well as a test-only dataset) both extractively and abstractively, without loss in language fluency (also demonstrated via human evaluation and abstractiveness scores). Finally, our parallel decoding results in a significant 10-20x speed-up over the previous best neural abstractive summarization system with even better accuracy.[1]

2 Model

In this work, we consider the task of summarizing a given long text document into several (ordered) highlights, which are then combined to form a multi-sentence summary. Formally, given a training set of document-summary pairs $\{x_i, y_i\}_{i=1}^N$, our goal is to approximate the function $h : X \to Y, X = \{x_i\}_{i=1}^N, Y = \{y_i\}_{i=1}^N$ such that $h(x_i) = y_i, 1 \leq i \leq N$. Furthermore, we assume there exists an abstracting function g defined as: $\forall s \in S_i, \exists d \in D_i$ such that $g(d) = s, 1 \leq i \leq N$, where S_i is the set of summary sentences in x_i and D_i the set of document sentences in y_i. i.e., in any given pair of document and summary, every summary sentence can be produced from some document sentence. For simplicity, we omit subscript i in the remainder of the paper. Under this assumption, we can further define another latent function $f : X \to D^n$ that satisfies $f(x) = \{d_t\}_{j=1}^n$ and $y = h(x) = [g(d_1), g(d_2), \ldots, g(d_n)]$, where $[,]$ denotes sentence concatenation. This latent function f can be seen as an extractor that chooses the salient (ordered) sentences in a given document for the abstracting function g to rewrite. Our overall model consists of these two submodules, the *extractor agent* and the *abstractor network*, to approximate the above-mentioned f and g, respectively.

2.1 Extractor Agent

The extractor agent is designed to model f, which can be thought of as extracting salient sentences from the document. We exploit a hierarchical neural model to learn the sentence representations of the document and a 'selection network' to extract sentences based on their representations.

[1]We are releasing our code, best pretrained models, as well as output summaries, to promote future research: `https://github.com/ChenRocks/fast_abs_rl`

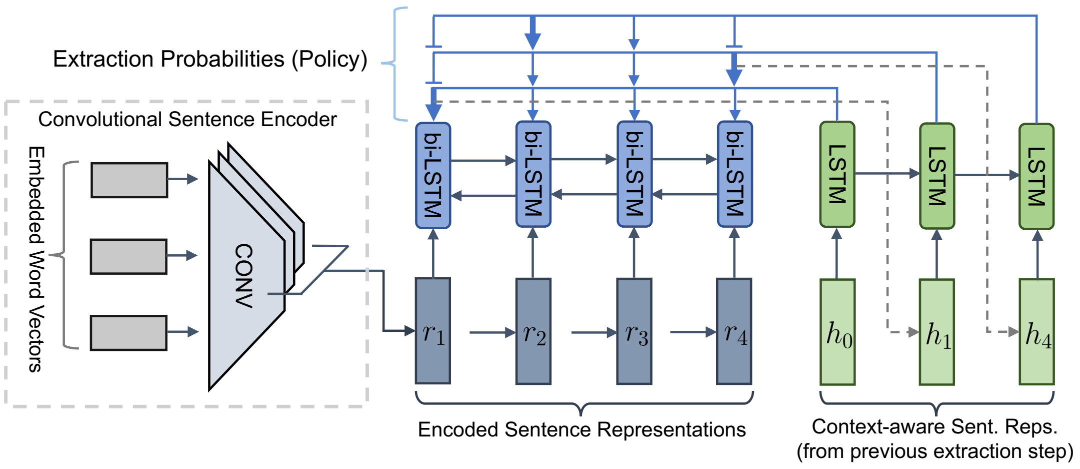

Extraction Probabilities (Policy)
Convolutional Sentence Encoder
Embedded Word Vectors
CONV
bi-LSTM
LSTM
r_1
r_2
r_3
r_4
h_0
h_1
h_4
Encoded Sentence Representations
Context-aware Sent. Reps.
(from previous extraction step)

Figure 1: Our extractor agent: the convolutional encoder computes representation r_j for each sentence. The RNN encoder (blue) computes context-aware representation h_j and then the RNN decoder (green) selects sentence j_t at time step t. With j_t selected, h_{j_t} will be fed into the decoder at time $t + 1$.

2.1.1 Hierarchical Sentence Representation

We use a temporal convolutional model (Kim, 2014) to compute r_j, the representation of each individual sentence in the documents (details in supplementary). To further incorporate global context of the document and capture the long-range semantic dependency between sentences, a bidirectional LSTM-RNN (Hochreiter and Schmidhuber, 1997; Schuster et al., 1997) is applied on the convolutional output. This enables learning a strong representation, denoted as h_j for the j-th sentence in the document, that takes into account the context of all previous and future sentences in the same document.

2.1.2 Sentence Selection

Next, to select the extracted sentences based on the above sentence representations, we add another LSTM-RNN to train a *Pointer Network* (Vinyals et al., 2015), to extract sentences recurrently. We calculate the extraction probability by:

$$u_j^t = \begin{cases} v_p^\top \tanh(W_{p1}h_j + W_{p2}e_t) & \text{if } j_t \neq j_k \\ & \forall k < t \\ -\infty & \text{otherwise} \end{cases} \tag{1}$$

$$P(j_t|j_1, \dots, j_{t-1}) = \text{softmax}(u^t) \tag{2}$$

where e_t's are the output of the *glimpse* operation (Vinyals et al., 2016):

$$a_j^t = v_g^\top \tanh(W_{g1}h_j + W_{g2}z_t) \tag{3}$$

$$\alpha^t = \text{softmax}(a^t) \tag{4}$$

$$e_t = \sum_j \alpha_j^t W_{g1}h_j \tag{5}$$

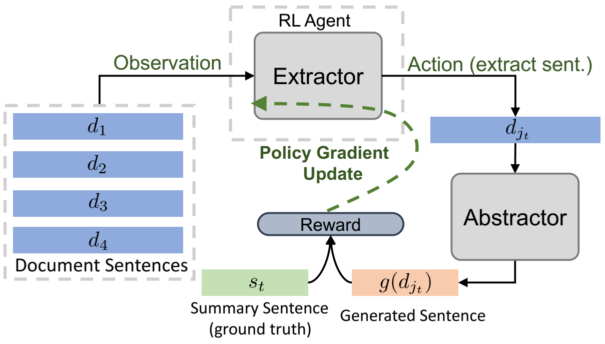


Figure 2: Reinforced training of the extractor (for one extraction step) and its interaction with the abstractor. For simplicity, the critic network is not shown. Note that all d's and s_t are raw sentences, *not* vector representations.

In Eqn. 3, z_t is the output of the added LSTM-RNN (shown in green in Fig. 1) which is referred to as the *decoder*. All the W's and v's are trainable parameters. At each time step t, the decoder performs a 2-hop attention mechanism: It first attends to h_j's to get a context vector e_t and then attends to h_j's again for the extraction probabilities.[2] This model is essentially classifying all sentences of the document at each extraction step. An illustration of the whole extractor is shown in Fig. 1.

2.2 Abstractor Network

The abstractor network approximates g, which compresses and paraphrases an extracted document sentence to a concise summary sentence. We

[2]Note that we force-zero the extraction prob. of already extracted sentences so as to prevent the model from using repeating document sentences and suffering from redundancy. This is non-differentiable and hence only done in RL training.

use the standard encoder-aligner-decoder (Bahdanau et al., 2015; Luong et al., 2015). We add the copy mechanism[3] to help directly copy some out-of-vocabulary (OOV) words (See et al., 2017). For more details, please refer to the supplementary.

3 Learning

Given that our extractor performs a non-differentiable *hard* extraction, we apply standard policy gradient methods to bridge the back-propagation and form an end-to-end trainable (stochastic) computation graph. However, simply starting from a randomly initialized network to train the whole model in an end-to-end fashion is infeasible. When randomly initialized, the extractor would often select sentences that are not relevant, so it would be difficult for the abstractor to learn to abstractively rewrite. On the other hand, without a well-trained abstractor the extractor would get noisy reward, which leads to a bad estimate of the policy gradient and a sub-optimal policy. We hence propose optimizing each sub-module separately using maximum-likelihood (ML) objectives: train the extractor to select salient sentences (fit f) and the abstractor to generate shortened summary (fit g). Finally, RL is applied to train the full model end-to-end (fit h).

3.1 Maximum-Likelihood Training for Submodules

Extractor Training: In Sec. 2.1.2, we have formulated our sentence selection as classification. However, most of the summarization datasets are end-to-end document-summary pairs without extraction (saliency) labels for each sentence. Hence, we propose a simple similarity method to provide a 'proxy' target label for the extractor. Similar to the extractive model of Nallapati et al. (2017), for each ground-truth summary sentence, we find the most similar document sentence d_{j_t} by:[4]

$$j_t = \text{argmax}_i(\text{ROUGE-L}_{recall}(d_i, s_t)) \tag{6}$$

Given these proxy training labels, the extractor is then trained to minimize the cross-entropy loss.

Abstractor Training: For the abstractor training, we create training pairs by taking each summary sentence and pairing it with its extracted document sentence (based on Eqn. 6). The network is trained as an usual sequence-to-sequence model to minimize the cross-entropy loss $L(\theta_{abs}) = -\frac{1}{M}\sum_{m=1}^{M} \log P_{\theta_{abs}}(w_m|w_{1:m-1})$ of the decoder language model at each generation step, where θ_{abs} is the set of trainable parameters of the abstractor and w_m the m^{th} generated word.

3.2 Reinforce-Guided Extraction

Here we explain how policy gradient techniques are applied to optimize the whole model. To make the extractor an RL agent, we can formulate a Markov Decision Process (MDP)[5]: at each extraction step t, the agent observes the current state $c_t = (D, d_{j_{t-1}})$, samples an action $j_t \sim \pi_{\theta_a,\omega}(c_t, j) = P(j)$ from Eqn. 2 to extract a document sentence and receive a reward[6]

$$r(t+1) = \text{ROUGE-L}_{F_1}(g(d_{j_t}), s_t) \tag{7}$$

after the abstractor summarizes the extracted sentence d_{j_t}. We denote the trainable parameters of the extractor agent by $\theta = \{\theta_a, \omega\}$ for the decoder and hierarchical encoder respectively. We can then train the extractor with policy-based RL. We illustrate this process in Fig. 2.

The vanilla policy gradient algorithm, REINFORCE (Williams, 1992), is known for high variance. To mitigate this problem, we add a critic network with trainable parameters θ_c to predict the state-value function $V^{\pi_{\theta_a,\omega}}(c)$. The predicted value of critic $b_{\theta_c,\omega}(c)$ is called the 'baseline', which is then used to estimate the *advantage function*: $A^{\pi_\theta}(c,j) = Q^{\pi_{\theta_a,\omega}}(c,j) - V^{\pi_{\theta_a,\omega}}(c)$ because the total return R_t is an estimate of action-value function $Q(c_t, j_t)$. Instead of maximizing $Q(c_t, j_t)$ as done in REINFORCE, we maximize $A^{\pi_\theta}(c,j)$ with the following policy gradient:

$$\begin{aligned} \nabla_{\theta_a,\omega} J(\theta_a,\omega) = \\ \mathbb{E}[\nabla_{\theta_a,\omega} \log\pi_\theta(c,j) A^{\pi_\theta}(c,j)] \end{aligned} \tag{8}$$

And the critic is trained to minimize the square loss: $L_c(\theta_c, \omega) = (b_{\theta_c,\omega}(c_t) - R_t)^2$. This is

[3]We use the terminology of *copy mechanism* (originally named *pointer-generator*) in order to avoid confusion with the *pointer network* (Vinyals et al., 2015).

[4]Nallapati et al. (2017) selected sentences greedily to maximize the global summary-level ROUGE, whereas we match exactly 1 document sentence for each GT summary sentence based on the *individual* sentence-level score.

[5]Strictly speaking, this is a *Partially Observable Markov Decision Process* (POMDP). We approximate it as an MDP by assuming that the RNN hidden state contains all past info.

[6]In Eqn. 6, we use ROUGE-recall because we want the extracted sentence to contain as much information as possible for rewriting. Nevertheless, for Eqn. 7, ROUGE-F_1 is more suitable because the abstractor g is supposed to rewrite the extracted sentence d to be as *concise* as the ground truth s.

known as the *Advantage Actor-Critic* (A2C), a synchronous variant of A3C (Mnih et al., 2016). For more A2C details, please refer to the supp.

Intuitively, our RL training works as follow: If the extractor chooses a good sentence, after the abstractor rewrites it the ROUGE match would be high and thus the action is encouraged. If a bad sentence is chosen, though the abstractor still produces a compressed version of it, the summary would not match the ground truth and the low ROUGE score discourages this action. Our RL with a sentence-level agent is a novel attempt in neural summarization. We use RL as a saliency guide without altering the abstractor's language model, while previous work applied RL on the word-level, which could be prone to gaming the metric at the cost of language fluency.[7]

Learning how many sentences to extract: In a typical RL setting like game playing, an episode is usually terminated by the environment. On the other hand, in text summarization, the agent does not know in advance how many summary sentence to produce for a given article (since the desired length varies for different downstream applications). We make an important yet simple, intuitive adaptation to solve this: by adding a 'stop' action to the policy action space. In the RL training phase, we add another set of trainable parameters v_{EOE} (EOE stands for 'End-Of-Extraction') with the same dimension as the sentence representation. The pointer-network decoder treats v_{EOE} as one of the extraction candidates and hence naturally results in a stop action in the stochastic policy. We set the reward for the agent performing EOE to ROUGE-$1_{F_1}([\{g(d_{j_t})\}_t], [\{s_t\}_t])$; whereas for any extraneous, unwanted extraction step, the agent receives zero reward. The model is therefore encouraged to extract when there are still remaining ground-truth summary sentences (to accumulate intermediate reward), and learn to stop by optimizing a global ROUGE and avoiding extra extraction.[8] Overall, this modification allows dynamic decisions of number-of-sentences based on the input document, eliminates the need for tuning a fixed number of steps, and enables a data-driven adaptation for any specific dataset/application.

3.3 Repetition-Avoiding Reranking

Existing abstractive summarization systems on long documents suffer from generating repeating and redundant words and phrases. To mitigate this issue, See et al. (2017) propose the coverage mechanism and Paulus et al. (2018) incorporate tri-gram avoidance during beam-search at test-time. Our model without these already performs well because the summary sentences are generated from mutually exclusive document sentences, which naturally avoids redundancy. However, we do get a small further boost to the summary quality by removing a few 'across-sentence' repetitions, via a simple reranking strategy: At sentence-level, we apply the same beam-search tri-gram avoidance (Paulus et al., 2018). We keep all k sentence candidates generated by beam search, where k is the size of the beam. Next, we then rerank all k^n combinations of the n generated summary sentence beams. The summaries are reranked by the number of repeated N-grams, the smaller the better. We also apply the diverse decoding algorithm described in Li et al. (2016) (which has almost no computation overhead) so as to get the above approach to produce useful diverse reranking lists. We show how much the redundancy affects the summarization task in Sec. 6.2.

4 Related Work

Early summarization works mostly focused on extractive and compression based methods (Jing and McKeown, 2000; Knight and Marcu, 2000; Clarke and Lapata, 2010; Berg-Kirkpatrick et al., 2011; Filippova et al., 2015). Recent large-sized corpora attracted neural methods for abstractive summarization (Rush et al., 2015; Chopra et al., 2016). Some of the recent success in neural abstractive models include hierarchical attention (Nallapati et al., 2016), coverage (Suzuki and Nagata, 2016; Chen et al., 2016; See et al., 2017), RL based metric optimization (Paulus et al., 2018), graph-based attention (Tan et al., 2017), and the copy mechanism (Miao and Blunsom, 2016; Gu et al., 2016; See et al., 2017).

[7]During this RL training of the extractor, we keep the abstractor parameters fixed. Because the input sentences for the abstractor are extracted by an intermediate stochastic policy of the extractor, it is impossible to find the correct target summary for the abstractor to fit g with ML objective. Though it is possible to optimize the abstractor with RL, in out preliminary experiments we found that this does not improve the overall ROUGE, most likely because this RL optimizes at a sentence-level and can add across-sentence redundancy. We achieve SotA results without this abstractor-level RL.

[8]We use ROUGE-1 for terminal reward because it is a better measure of bag-of-words information (i.e., has all the important information been generated); while ROUGE-L is used as intermediate rewards since it is known for better measurement of language fluency within a local sentence.

Our model shares some high-level intuition with extract-then-compress methods. Earlier attempts in this paradigm used Hidden Markov Models and rule-based systems (Jing and McKeown, 2000), statistical models based on parse trees (Knight and Marcu, 2000), and integer linear programming based methods (Martins and Smith, 2009; Gillick and Favre, 2009; Clarke and Lapata, 2010; Berg-Kirkpatrick et al., 2011). Recent approaches investigated discourse structures (Louis et al., 2010; Hirao et al., 2013; Kikuchi et al., 2014; Wang et al., 2015), graph cuts (Qian and Liu, 2013), and parse trees (Li et al., 2014; Bing et al., 2015). For neural models, Cheng and Lapata (2016) used a second neural net to select words from an extractor's output. Our abstractor does not merely 'compress' the sentences but generatively produce novel words. Moreover, our RL bridges the extractor and the abstractor for end-to-end training.

Reinforcement learning has been used to optimize the non-differential metrics of language generation and to mitigate exposure bias (Ranzato et al., 2016; Bahdanau et al., 2017). Henß et al. (2015) use Q-learning based RL for extractive summarization. Paulus et al. (2018) use RL policy gradient methods for abstractive summarization, utilizing sequence-level metric rewards with curriculum learning (Ranzato et al., 2016) or weighted ML+RL mixed loss (Paulus et al., 2018) for stability and language fluency. We use sentence-level rewards to optimize the extractor while keeping our ML trained abstractor decoder fixed, so as to achieve the best of both worlds.

Training a neural network to use another fixed network has been investigated in machine translation for better decoding (Gu et al., 2017a) and real-time translation (Gu et al., 2017b). They used a fixed pretrained *translator* and applied policy gradient techniques to train another task-specific network. In question answering (QA), Choi et al. (2017) extract one sentence and then generate the answer from the sentence's vector representation with RL bridging. Another recent work attempted a new coarse-to-fine attention approach on summarization (Ling and Rush, 2017) and found desired sharp focus properties for scaling to larger inputs (though without metric improvements). Very recently (concurrently), Narayan et al. (2018) use RL for ranking sentences in pure extraction-based summarization and Çelikyilmaz et al. (2018) investigate multiple communicating encoder agents to enhance the copying abstractive summarizer.

Finally, there are some loosely-related recent works: Zhou et al. (2017) proposed *selective gate* to improve the attention in abstractive summarization. Tan et al. (2018) used an *extract-then-synthesis* approach on QA, where an extraction model predicts the important spans in the passage and then another synthesis model generates the final answer. Swayamdipta et al. (2017) attempted cascaded non-recurrent small networks on extractive QA, resulting a scalable, parallelizable model. Fan et al. (2017) added controlling parameters to adapt the summary to length, style, and entity preferences. However, none of these used RL to bridge the non-differentiability of neural models.

5 Experimental Setup

Please refer to the supplementary for full training details (all hyperparameter tuning was performed on the validation set). We use the CNN/Daily Mail dataset (Hermann et al., 2015) modified for summarization (Nallapati et al., 2016). Because there are two versions of the dataset, original text and entity anonymized, we show results on both versions of the dataset for a fair comparison to prior works. The experiment runs training and evaluation for each version separately. Despite the fact that the 2 versions have been considered separately by the summarization community as 2 different datasets, we use same hyper-parameter values for both dataset versions to show the generalization of our model. We also show improvements on the DUC-2002 dataset in a *test-only* setup.

5.1 Evaluation Metrics

For all the datasets, we evaluate standard ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) on full-length F_1 (with stemming) following previous works (Nallapati et al., 2017; See et al., 2017; Paulus et al., 2018). Following See et al. (2017), we also evaluate on METEOR (Denkowski and Lavie, 2014) for a more thorough analysis.

5.2 Modular Extractive vs. Abstractive

Our hybrid approach is capable of both extractive and abstractive (i.e., rewriting every sentence) summarization. The extractor alone performs extractive summarization. To investigate the effect of the recurrent extractor (rnn-ext), we implement a feed-forward extractive baseline ff-ext (details in supplementary). It is also possible to apply RL

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Extractive Results				
lead-3 (See et al., 2017)	40.34	17.70	36.57	22.21
Narayan et al. (2018)	40.0	18.2	36.6	-
ff-ext	40.63	18.35	36.82	**22.91**
rnn-ext	40.17	18.11	36.41	22.81
rnn-ext + RL	**41.47**	**18.72**	**37.76**	22.35
Abstractive Results				
See et al. (2017) (w/o coverage)	36.44	15.66	33.42	16.65
See et al. (2017)	39.53	17.28	36.38	18.72
Fan et al. (2017) (controlled)	39.75	17.29	36.54	-
ff-ext + abs	39.30	17.02	36.93	20.05
rnn-ext + abs	38.38	16.12	36.04	19.39
rnn-ext + abs + RL	40.04	17.61	37.59	**21.00**
rnn-ext + abs + RL + rerank	**40.88**	**17.80**	**38.54**	20.38

Table 1: Results on the original, non-anonymized CNN/Daily Mail dataset. Adding RL gives statistically significant improvements for all metrics over non-RL rnn-ext models (and over the state-of-the-art See et al. (2017)) in both extractive and abstractive settings with $p < 0.01$. Adding the extra reranking stage yields statistically significant better results in terms of all ROUGE metrics with $p < 0.01$.

to extractor without using the abstractor (rnn-ext + RL).[9] Benefiting from the high modularity of our model, we can make our summarization system abstractive by simply applying the abstractor on the extracted sentences. Our abstractor rewrites each sentence and generates novel words from a large vocabulary, and hence every word in our overall summary is generated from scratch; making our full model categorized into the abstractive paradigm.[10] We run experiments on separately trained extractor/abstractor (ff-ext + abs, rnn-ext + abs) and the reinforced full model (rnn-ext + abs + RL) as well as the final reranking version (rnn-ext + abs + RL + rerank).

6 Results

For easier comparison, we show separate tables for the original-text vs. anonymized versions – Table 1 and Table 2, respectively. Overall, our model achieves strong improvements and the new state-of-the-art on both extractive and abstractive settings for both versions of the CNN/DM dataset (with some comparable results on the anonymized version). Moreover, Table 3 shows the generalization of our abstractive system to an out-of-domain test-only setup (DUC-2002), where our model achieves better scores than See et al. (2017).

6.1 Extractive Summarization

In the extractive paradigm, we compare our model with the extractive model from Nallapati et al. (2017) and a strong lead-3 baseline. For producing our summary, we simply concatenate the extracted sentences from the extractors. From Table 1 and Table 2, we can see that our feed-forward extractor out-performs the lead-3 baseline, empirically showing that our hierarchical sentence encoding model is capable of extracting salient sentences.[11] The reinforced extractor performs the best, because of the ability to get the summary-level reward and the reduced train-test mismatch of feeding the previous extraction decision. The improvement over lead-3 is consistent across both tables. In Table 2, it outperforms the previous best neural extractive model (Nallapati et al., 2017). In Table 1, our model also outperforms a recent, con-

Models	R-1	R-2	R-L
Extractive Results			
lead-3 (Nallapati et al., 2017)	39.2	15.7	35.5
Nallapati et al. (2017)	39.6	16.2	35.3
ff-ext	39.51	**16.85**	35.80
rnn-ext	38.97	16.65	35.32
rnn-ext + RL	**40.13**	16.58	**36.47**
Abstractive Results			
Nallapati et al. (2016)	35.46	13.30	32.65
Fan et al. (2017) (controlled)	38.68	15.40	35.47
Paulus et al. (2018) (ML)	38.30	14.81	35.49
Paulus et al. (2018) (RL+ML)	**39.87**	15.82	36.90
ff-ext + abs	38.73	15.70	36.33
rnn-ext + abs	37.58	14.68	35.24
rnn-ext + abs + RL	38.80	15.66	36.37
rnn-ext + abs + RL + rerank	39.66	**15.85**	**37.34**

Table 2: ROUGE for anonymized CNN/DM.

[9]In this case the abstractor function $g(d) = d$.

[10]Note that the abstractive CNN/DM dataset does *not* include any human-annotated extraction label, and hence our models do not receive any direct extractive supervision.

[11]The ff-ext model outperforms rnn-ext possibly because it does not predict sentence ordering; thus is easier to optimize and the n-gram based metrics do not consider sentence ordering. Also note that in our MDP formulation, we cannot apply RL on ff-ext due to its historyless nature. Even if applied naively, there is no mean for the feed-forward model to learn the EOE described in Sec. 3.2.

Models	R-1	R-2	R-L
See et al. (2017)	37.22	15.78	33.90
rnn-ext + abs + RL	39.46	17.34	36.72

Table 3: Generalization to DUC-2002 (F1).

current work by Narayan et al. (2018), showing that our pointer-network extractor and reward formulations are very effective when combined with A2C RL.

6.2 Abstractive Summarization

After applying the abstractor, the ff-ext based model still out-performs the rnn-ext model. Both combined models exceed the pointer-generator model (See et al., 2017) without coverage by a large margin for all metrics, showing the effectiveness of our 2-step hierarchical approach: our method naturally avoids repetition by extracting multiple sentences with different keypoints.[12]

Moreover, after applying reinforcement learning, our model performs better than the best model of See et al. (2017) and the best ML trained model of Paulus et al. (2018). Our reinforced model outperforms the ML trained rnn-ext + abs baseline with statistical significance of $p < 0.01$ on all metrics for both version of the dataset, indicating the effectiveness of the RL training. Also, rnn-ext + abs + RL is statistically significant better than See et al. (2017) for all metrics with $p < 0.01$.[13] In the supplementary, we show the learning curve of our RL training, where the average reward goes up quickly after the extractor learns the End-of-Extract action and then stabilizes. For all the above models, we use standard greedy decoding and find that it performs well.

Reranking and Redundancy Although the extract-then-abstract approach inherently will not generate repeating sentences like other neural-decoders do, there might still be across-sentence redundancy because the abstractor is not aware of other extracted sentences when decoding one. Hence, we incorporate an optional reranking strategy described in Sec. 3.3. The improved ROUGE scores indicate that this successfully removes some remaining redundancies and hence produces more concise summaries. Our best abstractive model (rnn-ext + abs + RL + rerank) is clearly superior than the one of See et al. (2017). We are comparable on R-1 and R-2 but a 0.4 point improvement on R-L w.r.t. Paulus et al. (2018).[14] We also outperform the results of Fan et al. (2017) on both original and anonymized dataset versions. Several previous works have pointed out that extractive baselines are very difficult to beat (in terms of ROUGE) by an abstractive system (See et al., 2017; Nallapati et al., 2017). Note that our best model is one of the first abstractive models to outperform the lead-3 baseline on the original-text CNN/DM dataset. Our extractive experiment serves as a complementary analysis of the effect of RL with extractive systems.

[12]A trivial *lead-3* + *abs* baseline obtains ROUGE of (37.37, 15.59, 34.82), which again confirms the importance of our reinforce-based sentence selection.

[13]We calculate statistical significance based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples. Output of Paulus et al. (2018) is not available so we couldn't test for statistical significance there.

	Relevance	Readability	Total
See et al. (2017)	120	128	248
rnn-ext + abs + RL + rerank	**137**	**133**	**270**
Equally good/bad	43	39	82

Table 4: Human Evaluation: pairwise comparison between our final model and See et al. (2017).

6.3 Human Evaluation

We also conduct human evaluation to ensure robustness of our training procedure. We measure *relevance* and *readability* of the summaries. Relevance is based on the summary containing important, salient information from the input article, being correct by avoiding contradictory/unrelated information, and avoiding repeated/redundant information. Readability is based on the summarys fluency, grammaticality, and coherence. To evaluate both these criteria, we design the following Amazon MTurk experiment: we randomly select 100 samples from the CNN/DM test set and ask the human testers (3 for each sample) to rank between summaries (for relevance and readability) produced by our model and that of See et al. (2017) (the models were anonymized and randomly shuffled), i.e. A is better, B is better, both are equally good/bad. Following previous work, the input article and ground truth summaries are also shown to the human participants in addition to the two model summaries.[15] From the results shown in Table 4, we can see that our model is better in both relevance and readability w.r.t. See et al. (2017).

[14]We do not list the scores of their pure RL model because they discussed its bad readability.

[15]We selected human annotators that were located in the US, had an approval rate greater than 95%, and had at least 10,000 approved HITs on record.

	Speed	
Models	total time (hr)	words / sec
(See et al., 2017)	12.9	14.8
rnn-ext + abs + RL	0.68	361.3
rnn-ext + abs + RL + rerank	2.00 (1.46 +0.54)	109.8

Table 5: Speed comparison with See et al. (2017).

6.4 Speed Comparison

Our two-stage extractive-abstractive hybrid model is not only the SotA on summary quality metrics, but more importantly also gives a significant speed-up in both train and test time over a strong neural abstractive system (See et al., 2017).[16]

Our full model is composed of a extremely fast extractor and a parallelizable abstractor, where the computation bottleneck is on the abstractor, which has to generate summaries with a large vocabulary from scratch.[17] The main advantage of our abstractor at decoding time is that we can first compute all the extracted sentences for the document, and then abstract every sentence concurrently (*in parallel*) to generate the overall summary. In Table 5, we show the substantial test-time speed-up of our model compared to See et al. (2017).[18] We calculate the total decoding time for producing all summaries for the test set.[19] Due to the fact that the main test-time speed bottleneck of RNN language generation model is that the model is constrained to generate one word at a time, the total decoding time is dependent on the number of total words generated; we hence also report the decoded words per second for a fair comparison. Our model without reranking is extremely fast. From Table 5 we can see that we achieve a speed up of 18x in time and 24x in word generation rate. Even after adding the (optional) reranker, we still maintain a 6-7x speed-up (and hence a user can choose to use the reranking component depending on their downstream application's speed requirements).[20]

[16]The only publicly available code with a pretrained model for neural summarization which we can test the speed.

[17]The time needed for extractor is negligible w.r.t. the abstractor because it does not require large matrix multiplication for generating every word. Moreover, with convolutional encoder at word-level made parallelizable by the hierarchical rnn-ext, our model is scalable for very long documents.

[18]For details of training speed-up, please see the supp.

[19]We time the model of See et al. (2017) using beam size of 4 (used for their best-reported scores). Without beam-search, it gets significantly worse ROUGE of (36.62, 15.12, 34.08), so we do not compare speed-ups w.r.t. that version.

[20]Most of the recent neural abstractive summarization systems are of similar algorithmic complexity to that of See et al. (2017). The main differences such as the training objective (ML vs. RL) and copying (soft/hard) has negligible test runtime compared to the slowest component: the long-summary attentional-decoder's sequential generation; and this is the component that we substantially speed up via our parallel sentence decoding with sentence-selection RL.

	Novel N-gram (%)			
Models	1-gm	2-gm	3-gm	4-gm
See et al. (2017)	0.1	2.2	6.0	9.7
rnn-ext + abs + RL + rerank	0.3	10.0	21.7	31.6
reference summaries	10.8	47.5	68.2	78.2

Table 6: Abstractiveness: novel n-gram counts.

7 Analysis

7.1 Abstractiveness

We compute an abstractiveness score (See et al., 2017) as the ratio of novel n-grams in the generated summary that are not present in the input document. The results are shown in Table 6: our model rewrites substantially more abstractive summaries than previous work. A potential reason for this is that when trained with individual sentence-pairs, the abstractor learns to drop more document words so as to write individual summary sentences as concise as human-written ones; thus the improvement in multi-gram novelty.

7.2 Qualitative Analysis on Output Examples

We show examples of how our best model selects sentences and then rewrites them. In the supplementary Figure 2 and Figure 3, we can see how the abstractor rewrites the extracted sentences concisely while keeping the mentioned facts. Adding the reranker makes the output more compact globally. We observe that when rewriting longer text, the abstractor would have many facts to choose from (Figure 3 sentence 2) and this is where the reranker helps avoid redundancy across sentences.

8 Conclusion

We propose a novel sentence-level RL model for abstractive summarization, which makes the model aware of the word-sentence hierarchy. Our model achieves the new state-of-the-art on both CNN/DM versions as well a better generalization on test-only DUC-2002, along with a significant speed-up in training and decoding.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by a Google Faculty Research Award, a Bloomberg Data Science Research Grant, an IBM Faculty Award, and NVidia GPU awards.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 318–325, Stroudsburg, PA, USA. Association for Computational Linguistics.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2017. Neural combinatorial optimization with reinforcement learning. *arXiv preprint 1611.09940*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 481–490, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *ACL*.

Asli Çelikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *NAACL-HLT*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *IJCAI*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–220. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint*, abs/1711.05217.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiatao Gu, Kyunghyun Cho, and Victor O. K. Li. 2017a. Trainable greedy decoding for neural machine translation. In *EMNLP*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017b. Learning to translate in real-time with neural machine translation. In *EACL*.

Sebastian Henß, Margot Mieskes, and Iryna Gurevych. 2015. A reinforcement learning approach for adaptive single- and multi-document summarization. In *International Conference of the German Society for Computational Linguistics and Language Technology (GSCL-2015)*, pages 3–12.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, Washington, USA. Association for Computational Linguistics.

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

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 178–185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 315–320, Baltimore, Maryland. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint*, abs/1611.08562.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 33–42. Association for Computational Linguistics.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 147–156, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *NAACL-HLT*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502, Seattle, Washington, USA. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Mike Schuster, Kuldip K. Paliwal, and A. General. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Jun Suzuki and Masaaki Nagata. 2016. Rnn-based encoder-decoder approach with word frequency estimation. In *EACL*.

Swabha Swayamdipta, Ankur P. Parikh, and Tom Kwiatkowski. 2017. Multi-mention learning for reading comprehension with neural cascades. *arXiv preprint*, abs/1711.00894.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer generation for machine reading comprehension. In *AAAI*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *ACL*.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Xun Wang, Yasuhisa Yoshida, Tsutomu Hirao, Katsuhito Sudoh, and Masaaki Nagata. 2015. Summarization based on task-oriented discourse parsing. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(8):1358–1367.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *HLT-NAACL 2004 Document Understanding Workshop*, pages 112–119, Boston, Massachusetts.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095–1104. Association for Computational Linguistics.

Soft Layer-Specific Multi-Task Summarization with Entailment and Question Generation

Han Guo[*] **Ramakanth Pasunuru**[*] **Mohit Bansal**
UNC Chapel Hill
{hanguo, ram, mbansal}@cs.unc.edu

Abstract

An accurate abstractive summary of a document should contain all its salient information and should be logically entailed by the input document. We improve these important aspects of abstractive summarization via multi-task learning with the auxiliary tasks of question generation and entailment generation, where the former teaches the summarization model how to look for salient questioning-worthy details, and the latter teaches the model how to rewrite a summary which is a directed-logical subset of the input document. We also propose novel multi-task architectures with high-level (semantic) layer-specific sharing across multiple encoder and decoder layers of the three tasks, as well as soft-sharing mechanisms (and show performance ablations and analysis examples of each contribution). Overall, we achieve statistically significant improvements over the state-of-the-art on both the CNN/DailyMail and Gigaword datasets, as well as on the DUC-2002 transfer setup. We also present several quantitative and qualitative analysis studies of our model's learned saliency and entailment skills.

1 Introduction

Abstractive summarization is the challenging NLG task of compressing and rewriting a document into a short, relevant, salient, and coherent summary. It has numerous applications such as summarizing storylines, event understanding, etc. As compared to extractive or compressive summarization (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015; Henß et al., 2015), abstractive summaries are based on rewriting as opposed to selecting. Recent end-to-end, neural sequence-to-sequence models and larger datasets have allowed substantial progress on the abstractive task, with ideas ranging from copy-pointer mechanism and redundancy coverage, to metric reward based reinforcement learning (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017).

Despite these strong recent advancements, there is still a lot of scope for improving the summary quality generated by these models. A good rewritten summary is one that contains all the salient information from the document, is logically followed (entailed) by it, and avoids redundant information. The redundancy aspect was addressed by coverage models (Suzuki and Nagata, 2016; Chen et al., 2016; Nallapati et al., 2016; See et al., 2017), but we still need to teach these models about how to better detect salient information from the input document, as well as about better logically-directed natural language inference skills.

In this work, we improve abstractive text summarization via soft, high-level (semantic) layer-specific multi-task learning with two relevant auxiliary tasks. The first is that of document-to-question generation, which teaches the summarization model about what are the right questions to ask, which in turn is directly related to what the salient information in the input document is. The second auxiliary task is a premise-to-entailment generation task to teach it how to rewrite a summary which is a directed-logical subset of (i.e., logically follows from) the input document, and contains no contradictory or unrelated information. For the question generation task, we use the SQuAD dataset (Rajpurkar et al., 2016), where we learn to generate a question given a sentence containing the answer, similar to the recent work

[*] Equal contribution.

by Du et al. (2017). Our entailment generation task is based on the recent SNLI classification dataset and task (Bowman et al., 2015), converted to a generation task (Pasunuru and Bansal, 2017).

Further, we also present novel multi-task learning architectures based on multi-layered encoder and decoder models, where we empirically show that it is substantially better to share the higher-level semantic layers between the three aforementioned tasks, while keeping the lower-level (lexico-syntactic) layers unshared. We also explore different ways to optimize the shared parameters and show that 'soft' parameter sharing achieves higher performance than hard sharing.

Empirically, our soft, layer-specific sharing model with the question and entailment generation auxiliary tasks achieves statistically significant improvements over the state-of-the-art on both the CNN/DailyMail and Gigaword datasets. It also performs significantly better on the DUC-2002 transfer setup, demonstrating its strong generalizability as well as the importance of auxiliary knowledge in low-resource scenarios. We also report improvements on our auxiliary question and entailment generation tasks over their respective previous state-of-the-art. Moreover, we significantly decrease the training time of the multi-task models by initializing the individual tasks from their pretrained baseline models. Finally, we present human evaluation studies as well as detailed quantitative and qualitative analysis studies of the improved saliency detection and logical inference skills learned by our multi-task model.

2 Related Work

Automatic text summarization has been progressively improving over the time, initially more focused on extractive and compressive models (Jing and McKeown, 2000; Knight and Marcu, 2002; Clarke and Lapata, 2008; Filippova et al., 2015; Kedzie et al., 2015), and moving more towards compressive and abstractive summarization based on graphs and concept maps (Giannakopoulos, 2009; Ganesan et al., 2010; Falke and Gurevych, 2017) and discourse trees (Gerani et al., 2014), syntactic parse trees (Cheung and Penn, 2014; Wang et al., 2013), and Abstract Meaning Representations (AMR) (Liu et al., 2015; Dohare and Karnick, 2017). Recent work has also adopted machine translation inspired neural seq2seq models for abstractive summarization with advances in hierarchical, distractive, saliency, and graph-attention modeling (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Chen et al., 2016; Tan et al., 2017). Paulus et al. (2018) and Henß et al. (2015) incorporated recent advances from reinforcement learning. Also, See et al. (2017) further improved results via pointer-copy mechanism and addressed the redundancy with coverage mechanism.

Multi-task learning (MTL) is a useful paradigm to improve the generalization performance of a task with related tasks while sharing some common parameters/representations (Caruana, 1998; Argyriou et al., 2007; Kumar and Daumé III, 2012). Several recent works have adopted MTL in neural models (Luong et al., 2016; Misra et al., 2016; Hashimoto et al., 2017; Pasunuru and Bansal, 2017; Ruder et al., 2017; Kaiser et al., 2017). Moreover, some of the above works have investigated the use of shared vs unshared sets of parameters. On the other hand, we investigate the importance of soft parameter sharing and high-level versus low-level layer-specific sharing.

Our previous workshop paper (Pasunuru et al., 2017) presented some preliminary results for multi-task learning of textual summarization with entailment generation. This current paper has several major differences: (1) We present question generation as an additional effective auxiliary task to enhance the important complementary aspect of saliency detection; (2) Our new high-level layer-specific sharing approach is significantly better than alternative layer-sharing approaches (including the decoder-only sharing by Pasunuru et al. (2017)); (3) Our new soft sharing parameter approach gives stat. significant improvements over hard sharing; (4) We propose a useful idea of starting multi-task models from their pretrained baselines, which significantly speeds up our experiment cycle[1]; (5) For evaluation, we show diverse improvements of our soft, layer-specific MTL model (over state-of-the-art pointer+coverage baselines) on the CNN/DailyMail, Gigaword, as well as DUC datasets; we also report human evaluation plus analysis examples of learned saliency and entailment skills; we also report improvements on the auxiliary question and entailment generation tasks over their respective previous state-of-the-art.

[1]About 4-5 days for Pasunuru et al. (2017) approach vs. only 10 hours for us. This will allow the community to try many more multi-task training and tuning ideas faster.

In our work, we use a question generation task to improve the saliency of abstractive summarization in a multi-task setting. Using the SQuAD dataset (Rajpurkar et al., 2016), we learn to generate a question given the sentence containing the answer span in the comprehension (similar to Du et al. (2017)). For the second auxiliary task of entailment generation, we use the generation version of the RTE classification task (Dagan et al., 2006; Lai and Hockenmaier, 2014; Jimenez et al., 2014; Bowman et al., 2015). Some previous work has explored the use of RTE for redundancy detection in summarization by modeling graph-based relationships between sentences to select the most non-redundant sentences (Mehdad et al., 2013; Gupta et al., 2014), whereas our approach is based on multi-task learning.

3 Models

First, we introduce our pointer+coverage baseline model and then our two auxiliary tasks: question generation and entailment generation (and finally the multi-task learning models in Sec. 4).

3.1 Baseline Pointer+Coverage Model

We use a sequence-attention-sequence model with a 2-layer bidirectional LSTM-RNN encoder and a 2-layer uni-directional LSTM-RNN decoder, along with Bahdanau et al. (2015) style attention. Let $x = \{x_1, x_2, ..., x_m\}$ be the source document and $y = \{y_1, y_2, ..., y_n\}$ be the target summary. The output summary generation vocabulary distribution conditioned over the input source document is $P_v(y|x;\theta) = \prod_{t=1}^{n} p_v(y_t|y_{1:t-1}, x; \theta)$. Let the decoder hidden state be s_t at time step t and let c_t be the context vector which is defined as a weighted combination of encoder hidden states. We concatenate the decoder's (last) RNN layer hidden state s_t and context vector c_t and apply a linear transformation, and then project to the vocabulary space by another linear transformation. Finally, the conditional vocabulary distribution at each time step t of the decoder is defined as:

$$p_v(y_t|y_{1:t-1}, x; \theta) = \text{sfm}(V_p(W_f[s_t; c_t]+b_f)+b_p) \tag{1}$$

where, W_f, V_p, b_f, b_p are trainable parameters, and $\text{sfm}(\cdot)$ is the softmax function.

Pointer-Generator Networks Pointer mechanism (Vinyals et al., 2015) helps in directly copying the words from the source sequence during target sequence generation, which is a good fit for a task like summarization. Our pointer mechanism approach is similar to See et al. (2017), who use a soft switch based on the generation probability $p_g = \sigma(W_g c_t + U_g s_t + V_g e_{w_{t-1}} + b_g)$, where $\sigma(\cdot)$ is a sigmoid function, W_g, U_g, V_g and b_g are parameters learned during training. $e_{w_{t-1}}$ is the previous time step output word embedding. The final word distribution is $P_f(y) = p_g \cdot P_v(y) + (1-p_g) \cdot P_c(y)$, where P_v vocabulary distribution is as shown in Eq. 1, and copy distribution P_c is based on the attention distribution over source document words.

Coverage Mechanism Following previous work (See et al., 2017), coverage helps alleviate the issue of word repetition while generating long summaries. We maintain a coverage vector $\hat{c}_t = \sum_{t=0}^{t-1} \alpha_t$ that sums over all of the previous time steps attention distributions α_t, and this is added as input to the attention mechanism. Coverage loss is $L_{cov}(\theta) = \sum_t \sum_i min(\alpha_{t,i}, \hat{c}_{t,i})$. Finally, the total loss is a weighted combination of cross-entropy loss and coverage loss:

$$L(\theta) = -\log P_f(y) + \lambda L_{cov}(\theta) \tag{2}$$

where λ is a tunable hyperparameter.

3.2 Two Auxiliary Tasks

Despite the strengths of the strong model described above with attention, pointer, and coverage, a good summary should also contain maximal salient information and be a directed logical entailment of the source document. We teach these skills to the abstractive summarization model via multi-task training with two related auxiliary tasks: question generation task and entailment generation.

Question Generation The task of question generation is to generate a question from a given input sentence, which in turn is related to the skill of being able to find the important salient information to ask questions about. First the model has to identify the important information present in the given sentence, then it has to frame (generate) a question based on this salient information, such that, given the sentence and the question, one has to be able to predict the correct answer (salient information in this case). A good summary should also be able to find and extract all the salient information in the given source document, and hence we incorporate such capabilities into our abstractive text summarization model by multi-task learning it with a question generation task, sharing some common parameters/representations (see more details in Sec. 4). For setting up the question generation task, we follow Du et al. (2017) and use the SQuAD dataset to extract sentence-question pairs. Next, we use the same sequence-to-sequence model architecture as our summarization model. Note that even though our question generation task is generating one question at a time[2], our multi-task framework (see Sec. 4) is set up in such a way that the sentence-level knowledge from this auxiliary task can help the document-level primary (summarization) task to generate multiple salient facts – by sharing high-level semantic layer representations. See Sec. 7 and Table 10 for a quantitative evaluation showing that the multi-task model can find multiple (and more) salient phrases in the source document. Also see Sec. 7 (and supp) for challenging qualitative examples where baseline and SotA models only recover a small subset of salient information but our multi-task model with question generation is able to detect more of the important information.

Entailment Generation The task of entailment generation is to generate a hypothesis which is entailed by (or logically follows from) the given premise as input. In summarization, the generation decoder also needs to generate a summary that is entailed by the source document, i.e., does not contain any contradictory or unrelated/extraneous information as compared to the input document. We again incorporate such inference capabilities into the summarization model via multi-task learning, sharing some common representations/parameters between our summarization and entailment generation model (more details in Sec. 4). For this task, we use the entailment-labeled pairs from the SNLI dataset (Bowman et al., 2015) and set it up as a generation task (using the same strong model architecture as our abstractive summarization model). See Sec. 7 and Table 9 for a quantitative evaluation showing that the multi-task model is better entailed by the source document and has fewer extraneous facts. Also see Sec. 7 and supplementary for qualitative examples of how our multi-task model with the entailment auxiliary task is able to generate more logically-entailed summaries than the baseline and SotA models, which instead produce extraneous, unrelated words not present (in any paraphrased form) in the source document.

[2]We also tried to generate all the questions at once from the full document, but we obtained low accuracy because of this task's challenging nature and overall less training data.

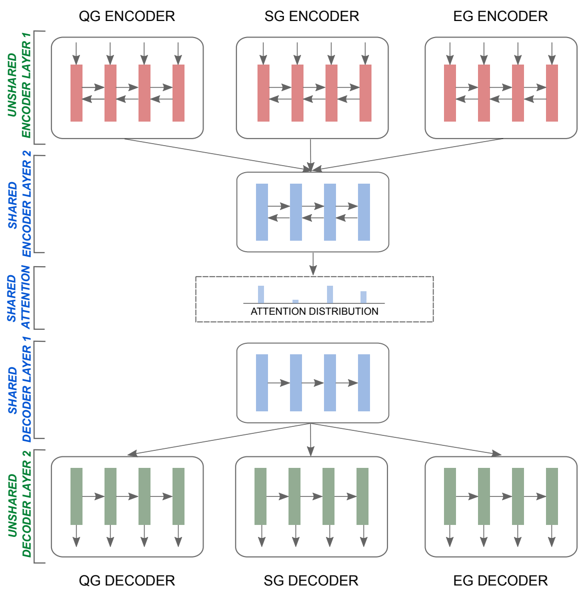


Figure 1: Overview of our multi-task model with parallel training of three tasks: abstractive summary generation (SG), question generation (QG), and entailment generation (EG). We share the 'blue' color representations across all the three tasks, i.e., second layer of encoder, attention parameters, and first layer of decoder.

4 Multi-Task Learning

We employ multi-task learning for parallel training of our three tasks: abstractive summarization, question generation, and entailment generation. In this section, we describe our novel layer-specific, soft-sharing approaches and other multi-task learning details.

4.1 Layer-Specific Sharing Mechanism

Simply sharing all parameters across the related tasks is not optimal, because models for different tasks have different input and output distributions, esp. for low-level vs. high-level parameters. Therefore, related tasks should share some common representations (e.g., high-level information), as well as need their own individual task-specific representations (esp. low-level information). To this end, we allow different components of model parameters of related tasks to be shared vs. unshared, as described next.

Encoder Layer Sharing: Belinkov et al. (2017) observed that lower layers (i.e., the layers closer to the input words) of RNN cells in a seq2seq machine translation model learn to represent word structure, while higher layers (farther from input) are more focused on high-level semantic meanings (similar to findings in the computer vision community for image features (Zeiler and Fergus, 2014)). We believe that while textual summarization, question generation, and entailment generation have different training data distributions and low-level representations, they can still benefit from sharing their models' high-level components (e.g., those that capture the skills of saliency and inference). Thus, we keep the lower-level layer (i.e., first layer closer to input words) of the 2-layer encoder of all three tasks unshared, while we share the higher layer (second layer in our model as shown in Fig. 1) across the three tasks.

Decoder Layer Sharing: Similarly for the decoder, lower layers (i.e., the layers closer to the output words) learn to represent word structure for generation, while higher layers (farther from output) are more focused on high-level semantic meaning. Hence, we again share the higher level components (first layer in the decoder far from output as show in Fig. 1), while keeping the lower layer (i.e., second layer) of decoders of all three tasks unshared.

Attention Sharing: As described in Sec. 3.1, the attention mechanism defines an attention distribution over high-level layer encoder hidden states and since we share the second, high-level (semantic) layer of all the encoders, it is intuitive to share the attention parameters as well.

4.2 Soft vs. Hard Parameter Sharing

Hard-sharing: In the most common multi-task learning hard-sharing approach, the parameters to be shared are forced to be the same. As a result, gradient information from multiple tasks will directly pass through shared parameters, hence forcing a common space representation for all the related tasks. **Soft-sharing**: In our soft-sharing approach, we encourage shared parameters to be close in representation space by penalizing their l_2 distances. Unlike hard sharing, this approach gives more flexibility for the tasks by only loosely coupling the shared space representations. We minimize the following loss function for the primary task in soft-sharing approach:

$$L(\theta) = -\log P_f(y) + \lambda L_{cov}(\theta) + \gamma \|\theta_s - \psi_s\| \quad (3)$$

where γ is a hyperparameter, θ represents the primary summarization task's full parameters, while θ_s and ψ_s represent the shared parameter subset between the primary and auxiliary tasks.

4.3 Fast Multi-Task Training

During multi-task learning, we alternate the mini-batch optimization of the three tasks, based on a tunable 'mixing ratio' $\alpha_s : \alpha_q : \alpha_e$; i.e., optimizing the summarization task for α_s mini-batches followed by optimizing the question generation task for α_q mini-batches, followed by entailment generation task for α_e mini-batches (and for 2-way versions of this, we only add one auxiliary task at a time). We continue this process until all the models converge. Also, importantly, instead of training from scratch, we start the primary task (summarization) from a 90%-converged model of its baseline to make the training process faster. We observe that starting from a fully-converged baseline makes the model stuck in a local minimum. In addition, we also start all auxiliary models from their 90%-converged baselines, as we found that starting the auxiliary models from scratch has a chance to pull the primary model's shared parameters towards randomly-initialized auxiliary model's shared parameters.

5 Experimental Setup

Datasets: We use CNN/DailyMail dataset (Hermann et al., 2015; Nallapati et al., 2016) and Gigaword (Rush et al., 2015) datasets for summarization, and the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) datasets for our entailment and question generation tasks, resp. We also show generalizability/transfer results on DUC-2002 with our CNN/DM trained models. Supplementary contains dataset details.

Evaluation Metrics: We use the standard ROUGE evaluation package (Lin, 2004) for reporting the results on all of our summarization models. Following previous work (Chopra et al., 2016; Nallapati et al., 2016), we use ROUGE full-length F1 variant for all our results. Following See et al. (2017), we also report METEOR (Denkowski and Lavie, 2014) using the MS-COCO evaluation script (Chen et al., 2015).

Human Evaluation Criteria: We used Amazon MTurk to perform human evaluation of summary *relevance* and *readability*. We selected human annotators that were located in the US, had an ap-

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
PREVIOUS WORK				
Seq2Seq(50k vocab) (See et al., 2017)	31.33	11.81	28.83	12.03
Pointer (See et al., 2017)	36.44	15.66	33.42	15.35
Pointer+Coverage (See et al., 2017) ⋆	39.53	17.28	36.38	18.72
Pointer+Coverage (See et al., 2017) †	38.82	16.81	35.71	18.14
OUR MODELS				
Two-Layer Baseline (Pointer+Coverage) ⊗	39.56	17.52	36.36	18.17
⊗ + Entailment Generation	39.84	17.63	36.54	18.61
⊗ + Question Generation	39.73	17.59	36.48	18.33
⊗ + Entailment Gen. + Question Gen.	39.81	17.64	36.54	18.54

Table 1: CNN/DailyMail summarization results. ROUGE scores are full length F-1 (as previous work). All the multi-task improvements are statistically significant over the state-of-the-art baseline.

Models	R-1	R-2	R-L
PREVIOUS WORK			
ABS+ (Rush et al., 2015)	29.76	11.88	26.96
RAS-El (Chopra et al., 2016)	33.78	15.97	31.15
lvt2k (Nallapati et al., 2016)	32.67	15.59	30.64
Pasunuru et al. (2017)	32.75	15.35	30.82
OUR MODELS			
2-Layer Pointer Baseline ⊗	34.26	16.40	32.03
⊗ + Entailment Generation	35.45	17.16	33.19
⊗ + Question Generation	35.48	17.31	32.97
⊗ + Entailment + Question	35.98	17.76	33.63

Table 2: Summarization results on Gigaword. ROUGE scores are full length F-1.

proval rate greater than 95%, and had at least 10,000 approved HITs. For the pairwise model comparisons discussed in Sec. 6.2, we showed the annotators the input article, the ground truth summary, and the two model summaries (randomly shuffled to anonymize model identities) – we then asked them to choose the better among the two model summaries or choose 'Not-Distinguishable' if both summaries are equally good/bad. Instructions for relevance were defined based on the summary containing salient/important information from the given article, being correct (i.e., avoiding contradictory/unrelated information), and avoiding redundancy. Instructions for readability were based on the summary's fluency, grammaticality, and coherence.

Training Details All our soft/hard and layer-specific sharing decisions were made on the validation/development set. Details of RNN hidden state sizes, Adam optimizer, mixing ratios, etc. are provided in the supplementary for reproducibility.

6 Results

6.1 Summarization (Primary Task) Results

Pointer+Coverage Baseline We start from the strong model of See et al. (2017).[3] Table 1 shows that our baseline model performs better than or comparable to See et al. (2017).[4] On Gigaword dataset, our baseline model (with pointer only, since coverage not needed for this single-sentence summarization task) performs better than all previous works, as shown in Table 2.

Multi-Task with Entailment Generation We first perform multi-task learning between abstractive summarization and entailment generation with soft-sharing of parameters as discussed in Sec. 4. Table 1 and Table 2 shows that this multi-task setting is better than our strong baseline models and the improvements are statistically significant on all metrics[5] on both CNN/DailyMail ($p < 0.01$ in ROUGE-1/ROUGE-L/METEOR and $p < 0.05$ in ROUGE-2) and Gigaword ($p < 0.01$ on all metrics) datasets, showing that entailment generation task is inducing useful inference skills to the summarization task (also see analysis examples in Sec. 7).

Multi-Task with Question Generation For multi-task learning with question generation, the improvements are statistically significant in ROUGE-1 ($p < 0.01$), ROUGE-L ($p < 0.05$), and METEOR ($p < 0.01$) for CNN/DailyMail and in all metrics ($p < 0.01$) for Gigaword, compared to the respective baseline models. Also, Sec. 7 presents quantitative and qualitative analysis of this model's improved saliency.[6]

[3]We use two layers so as to allow our high- versus low-level layer sharing intuition. Note that this does not increase the parameter size much (23M versus 22M for See et al. (2017)).

[4]As mentioned in the github for See et al. (2017), their publicly released pretrained model produces the lower scores that we represent by † in Table 1.

[5]Stat. significance is computed via bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples.

[6]In order to verify that our improvements were from the auxiliary tasks' specific character/capabilities and not just due to adding more data, we separately trained word embeddings on each auxiliary dataset (i.e., SNLI and SQuAD) and incorporated them into the summarization model. We found that both our 2-way multi-task models perform sig-

Models	Relevance	Readability	Total
MTL vs. Baseline			
MTL wins	43	40	83
Baseline wins	22	24	46
Non-distinguish.	35	36	71
MTL vs. See et al. (2017)			
MTL wins	39	33	72
See (2017) wins	29	38	67
Non-distinguish.	32	29	61

Table 3: CNN/DM Human Evaluation: pairwise comparison between our 3-way multi-task (MTL) model w.r.t. our baseline and See et al. (2017).

Models	Relevance	Readability	Total
MTL wins	33	32	65
Baseline wins	22	22	44
Non-distinguish.	45	46	91

Table 4: Gigaword Human Evaluation: pairwise comparison between our 3-way multi-task (MTL) model w.r.t. our baseline.

Multi-Task with Entailment and Question Generation Finally, we perform multi-task learning with all three tasks together, achieving the best of both worlds (inference skills and saliency). Table 1 and Table 2 show that our full multi-task model achieves the best scores on CNN/DailyMail and Gigaword datasets, and the improvements are statistically significant on all metrics on both CNN/DailyMail ($p < 0.01$ in ROUGE-1/ROUGE-L/METEOR and $p < 0.02$ in ROUGE-2) and Gigaword ($p < 0.01$ on all metrics). Finally, our 3-way multi-task model (with both entailment and question generation) outperforms the publicly-available pretrained result (†) of the previous SotA (See et al., 2017) with stat. significance ($p < 0.01$), as well the higher-reported results ($\star$) on ROUGE-1/ROUGE-2 ($p < 0.01$).

6.2 Human Evaluation

We also conducted a blind human evaluation on Amazon MTurk for relevance and readability, based on 100 samples, for both CNN/DailyMail and Gigaword (see instructions in Sec. 5). Table. 3 shows the CNN/DM results where we do pairwise comparison between our 3-way multi-task model's output summaries w.r.t. our baseline summaries and w.r.t. See et al. (2017) summaries. As shown, our 3-way multi-task model achieves both higher relevance and higher readability scores w.r.t. the baseline. W.r.t. See et al. (2017), our MTL model is higher in relevance scores but a bit lower in readability scores (and is higher in terms of total aggregate scores). One potential reason for this lower readability score is that our entailment generation auxiliary task encourages our summarization model to rewrite more and to be more abstractive than See et al. (2017) – see abstractiveness results in Table 11.

We also show human evaluation results on the Gigaword dataset in Table 4 (again based on pairwise comparisons for 100 samples), where we see that our MTL model is better than our state-of-the-art baseline on both relevance and readability.[7]

nificantly better than these models using the auxiliary word-embeddings, suggesting that merely adding more data in not enough.

Models	R-1	R-2	R-L
See et al. (2017)	34.30	14.25	30.82
Baseline	35.96	15.91	32.92
Multi-Task (EG + QG)	36.73	16.15	33.58

Table 5: ROUGE F1 scores on DUC-2002.

6.3 Generalizability Results (DUC-2002)

Next, we also tested our model's generalizability/transfer skills, where we take the models trained on CNN/DailyMail and directly test them on DUC-2002. We take our baseline and 3-way multi-task models, plus the pointer-coverage model from See et al. (2017).[8] We only re-tune the beam-size for each of these three models separately (based on DUC-2003 as the validation set).[9] As shown in Table 5, our multi-task model achieves statistically significant improvements over the strong baseline ($p < 0.01$ in ROUGE-1 and ROUGE-L) and the pointer-coverage model from See et al. (2017) ($p < 0.01$ in all metrics). This demonstrates that our model is able to generalize well and that the auxiliary knowledge helps more in low-resource scenarios.

6.4 Auxiliary Task Results

In this section, we discuss the individual/separated performance of our auxiliary tasks.

Entailment Generation We use the same architecture as described in Sec. 3.1 with pointer mech-

[7]Note that we did not have output files of any previous work's model on Gigaword; however, our baseline is already a strong state-of-the-art model as shown in Table 2.

[8]We use the publicly-available pretrained model from See et al. (2017)'s github for these DUC transfer results, which produces the † results in Table 1. All other comparisons and analysis in our paper are based on their higher $\star$ results.

[9]We follow previous work which has shown that larger beam values are better and feasible for DUC corpora. However, our MTL model still achieves stat. significant improvements ($p < 0.01$ in all metrics) over See et al. (2017) without beam retuning (i.e., with beam $= 4$).

Models	M	C	R	B
Pasunuru&Bansal (2017)	29.6	117.8	62.4	40.6
Our 1-layer pointer EG	32.4	139.3	65.1	43.6
Our 2-layer pointer EG	32.3	140.0	64.4	43.7

Table 6: Performance of our pointer-based entailment generation (EG) models compared with previous SotA work. M, C, R, B are short for Meteor, CIDEr-D, ROUGE-L, and BLEU-4, resp.

Models	M	C	R	B
Du et al. (2017)	15.2	-	38.0	10.8
Our 1-layer pointer QG	15.4	75.3	36.2	9.2
Our 2-layer pointer QG	17.5	95.3	40.1	13.8

Table 7: Performance of our pointer-based question generation (QG) model w.r.t. previous work.

anism, and Table 6 compares our model's performance to Pasunuru and Bansal (2017). Our pointer mechanism gives a performance boost, since the entailment generation task involves copying from the given premise sentence, whereas the 2-layer model seems comparable to the 1-layer model. Also, the supplementary shows some output examples from our entailment generation model.

Question Generation Again, we use same architecture as described in Sec. 3.1 along with pointer mechanism for the task of question generation. Table 7 compares the performance of our model w.r.t. the state-of-the-art Du et al. (2017). Also, the supplementary shows some output examples from our question generation model.

7 Ablation and Analysis Studies

Soft-sharing vs. Hard-sharing As described in Sec. 4.2, we choose soft-sharing over hard-sharing because of the more expressive parameter sharing it provides to the model. Empirical results in Table. 8 prove that soft-sharing method is statistically significantly better than hard-sharing with $p < 0.001$ in all metrics.[10]

Comparison of Different Layer-Sharing Methods We also conducted ablation studies among various layer-sharing approaches. Table 8 shows results for soft-sharing models with decoder-only sharing (D1+D2; similar to Pasunuru et al. (2017)) as well as lower-layer sharing (encoder layer 1 + decoder layer 2, with and without attention shared). As shown, our final model (high-level semantic layer sharing E2+Attn+D1) outperforms

Models	R-1	R-2	R-L	M
Final Model	**39.81**	**17.64**	**36.54**	**18.54**
SOFT-VS.-HARD SHARING				
Hard-sharing	39.51	17.44	36.33	18.21
LAYER SHARING METHODS				
D1+D2	39.62	17.49	36.44	18.34
E1+D2	39.51	17.51	36.37	18.15
E1+Attn+D2	39.32	17.36	36.11	17.88

Table 8: Ablation studies comparing our final multi-task model with hard-sharing and different alternative layer-sharing methods. Here E1, E2, D1, D2, Attn refer to parameters of the first/second layer of encoder/decoder, and attention parameters. Improvements of final model upon ablation experiments are all stat. signif. with $p < 0.05$.

Models	Average Entailment Probability
Baseline	0.907
Multi-Task (EG)	0.912

Table 9: Entailment classification results of our baseline vs. EG-multi-task model ($p < 0.001$).

these alternate sharing methods in all metrics with statistical significance ($p < 0.05$).[11]

Quantitative Improvements in Entailment We employ a state-of-the-art entailment classifier (Chen et al., 2017), and calculate the average of the entailment probability of each of the output summary's sentences being entailed by the input source document. We do this for output summaries of our baseline and 2-way-EG multi-task model (with entailment generation). As can be seen in Table 9, our multi-task model improves upon the baseline in the aspect of being entailed by the source document (with statistical significance $p < 0.001$). Further, we use the Named Entity Recognition (NER) module from CoreNLP (Manning et al., 2014) to compute the number of times the output summary contains extraneous facts (i.e., named entities as detected by the NER system) that are not present in the source documents, based on the intuition that a well-entailed summary should not contain unrelated information not followed from the input premise. We found that our 2-way MTL model with entailment generation reduces this extraneous count by 17.2% w.r.t. the baseline. The qualitative examples below further discuss this issue of generating unrelated information.

Quantitative Improvements in Saliency Detection For our saliency evaluation, we used the

[10]In the interest of space, most of the analyses are shown for CNN/DailyMail experiments, but we observed similar trends for the Gigaword experiments as well.

[11]Note that all our soft and layer sharing decisions were strictly made on the dev/validation set (see Sec. 5).

Models	Average Match Rate
Baseline	27.75 %
Multi-Task (QG)	28.06 %

Table 10: Saliency classification results of our baseline vs. QG-multi-task model ($p < 0.01$).

Models	2-gram	3-gram	4-gram
See et al. (2017)	2.24	6.03	9.72
MTL (3-way)	2.84	6.83	10.66

Table 11: Abstractiveness: novel n-gram percent.

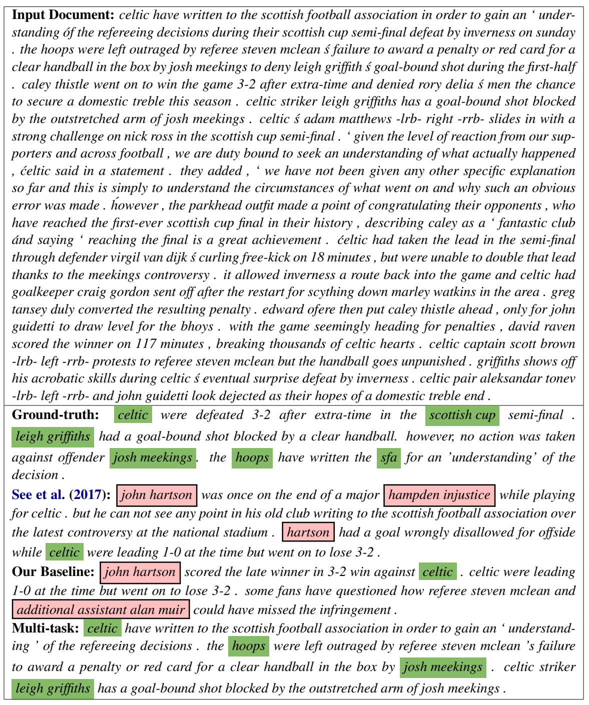
answer-span prediction classifier from Pasunuru and Bansal (2018) trained on SQuAD (Rajpurkar et al., 2016) as the keyword detection classifier. We then annotate the ground-truth and model summaries with this keyword classifier and compute the % match, i.e., how many salient words from the ground-truth summary were also generated in the model summary. The results are shown in Table 10, where the 2-way-QG MTL model (with question generation) versus baseline improvement is stat. significant ($p < 0.01$). Moreover, we found 93 more cases where our 2-way-QG MTL model detects 2 or more additional salient keywords than the pointer baseline model (as opposed to vice versa), showing that sentence-level question generation task is helping the document-level summarization task in finding more salient terms.

Qualitative Examples on Entailment and Saliency Improvements Fig. 2 presents an example of output summaries generated by See et al. (2017), our baseline, and our 3-way multi-task model. See et al. (2017) and our baseline models generate phrases like "john hartson" and "hampden injustice" that don't appear in the input document, hence they are not entailed by the input.[12] Moreover, both models missed salient information like "josh meekings", "leigh griffiths", and "hoops", that our multi-task model recovers.[13] Hence, our 3-way multi-task model generates summaries that are both better at logical entailment and contain more salient information. We refer to supplementary Fig. 3 for more details and similar examples for separated 2-way multi-task models (supplementary Fig. 1, Fig. 2).

Abstractiveness Analysis As suggested in See et al. (2017), we also compute the abstractiveness score as the number of novel n-grams between the model output summary and source document. As shown in Table 11, our multi-task model (EG + QG) is more abstractive than See et al. (2017).

[12]These extra, non-entailed unrelated/contradictory information are not present at all in any paraphrase form in the input document.

[13]We consider the fill-in-the-blank highlights annotated by human on CNN/DailyMail dataset as salient information.



Input Document: *celtic have written to the scottish football association in order to gain an ‘ understanding ́of the refereeing decisions during their scottish cup semi-final defeat by inverness on sunday . the hoops were left outraged by referee steven mclean ś failure to award a penalty or red card for a clear handball in the box by josh meekings to deny leigh griffith ś goal-bound shot during the first-half . caley thistle went on to win the game 3-2 after extra-time and denied rory delia ś men the chance to secure a domestic treble this season . celtic striker leigh griffiths has a goal-bound shot blocked by the outstretched arm of josh meekings . celtic ś adam matthews -lrb- right -rrb- slides in with a strong challenge on nick ross in the scottish cup semi-final . ‘ given the level of reaction from our supporters and across football , we are duty bound to seek an understanding of what actually happened , ćeltic said in a statement . they added , ‘ we have not been given any other specific explanation so far and this is simply to understand the circumstances of what went on and why such an obvious error was made . h́owever , the parkhead outfit made a point of congratulating their opponents , who have reached the first-ever scottish cup final in their history , describing caley as a ‘ fantastic club ánd saying ‘ reaching the final is a great achievement . ćeltic had taken the lead in the semi-final through defender virgil van dijk ś curling free-kick on 18 minutes , but were unable to double that lead thanks to the meekings controversy . it allowed inverness a route back into the game and celtic had goalkeeper craig gordon sent off after the restart for scything down marley watkins in the area . greg tansey duly converted the resulting penalty . edward ofere then put caley thistle ahead , only for john guidetti to draw level for the bhoys . with the game seemingly heading for penalties , david raven scored the winner on 117 minutes , breaking thousands of celtic hearts . celtic captain scott brown -lrb- left -rrb- protests to referee steven mclean but the handball goes unpunished . griffiths shows off his acrobatic skills during celtic ś eventual surprise defeat by inverness . celtic pair aleksandar tonev -lrb- left -rrb- and john guidetti look dejected as their hopes of a domestic treble end .*

Ground-truth: *celtic were defeated 3-2 after extra-time in the scottish cup semi-final . leigh griffiths had a goal-bound shot blocked by a clear handball. however, no action was taken against offender josh meekings. the hoops have written the sfa for an 'understanding' of the decision .*

See et al. (2017): *john hartson was once on the end of a major hampden injustice while playing for celtic . but he can not see any point in his old club writing to the scottish football association over the latest controversy at the national stadium . hartson had a goal wrongly disallowed for offside while celtic were leading 1-0 at the time but went on to lose 3-2 .*

Our Baseline: *john hartson scored the late winner in 3-2 win against celtic . celtic were leading 1-0 at the time but went on to lose 3-2 . some fans have questioned how referee steven mclean and additional assistant alan muir could have missed the infringement .*

Multi-task: *celtic have written to the scottish football association in order to gain an ‘ understanding ’ of the refereeing decisions . the hoops were left outraged by referee steven mclean 's failure to award a penalty or red card for a clear handball in the box by josh meekings . celtic striker leigh griffiths has a goal-bound shot blocked by the outstretched arm of josh meekings .*

Figure 2: Example summary from our 3-way MTL model. The boxed-red highlights are extraneously-generated words not present/paraphrased in the input document. The unboxed-green highlights show salient phrases.

8 Conclusion

We presented a multi-task learning approach to improve abstractive summarization by incorporating the ability to detect salient information and to be logically entailed by the document, via question generation and entailment generation auxiliary tasks. We propose effective soft and high-level (semantic) layer-specific parameter sharing and achieve significant improvements over the state-of-the-art on two popular datasets, as well as a generalizability/transfer DUC-2002 setup.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), Google Faculty Research Award, Bloomberg Data Science Research Grant, and NVidia GPU awards. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agency.

References

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *NIPS*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *ACL*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *IJCAI*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *EMNLP*, pages 775–786.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL*.

Shibhansh Dohare and Harish Karnick. 2017. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *ACL*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Tobias Falke and Iryna Gurevych. 2017. Bringing structure into summaries: Crowdsourcing a benchmark corpus of concept maps. In *EMNLP*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *EMNLP*, pages 360–368.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. ACL.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *EMNLP*, volume 14, pages 1602–1613.

George Giannakopoulos. 2009. Automatic summarization from multiple documents. *Ph. D. dissertation*.

Anand Gupta, Manpreet Kaur, Adarsh Singh, Aseem Goel, and Shachar Mirkin. 2014. Text summarization through entailment-based minimum vertex cover. *Lexical and Computational Semantics (* SEM 2014)*, page 75.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Stefan Henß, Margot Mieskes, and Iryna Gurevych. 2015. A reinforcement learning approach for adaptive single-and multi-document summarization. In *GSCL*, pages 3–12.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *SemEval*, pages 732–742.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 178–185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *CoRR*, abs/1706.05137.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting salient updates for disaster summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1608–1617.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Abhishek Kumar and Hal Daumé III. 2012. Learning task grouping and overlap in multi-task learning. In *ICML*.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. *Proc. SemEval*, 2:5.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*, volume 8.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *NAACL: HLT*, pages 1077–1086.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Yashar Mehdad, Giuseppe Carenini, Frank W Tompa, and Raymond T Ng. 2013. Abstractive meeting summarization with entailment and fusion. In *Proc. of the 14th European Workshop on Natural Language Generation*, pages 136–146.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *CVPR*, pages 3994–4003.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *ACL*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *NAACL*.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *NFiS@EMNLP*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Sogaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *CoRR*, abs/1705.08142.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Jun Suzuki and Masaaki Nagata. 2016. Rnn-based encoder-decoder approach with word frequency estimation. In *EACL*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *ACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *ACL*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Modeling and Prediction of Online Product Review Helpfulness: A Survey

Gerardo Ocampo Diaz and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{godiaz,vince}@hlt.utdallas.edu

Abstract

As the popularity of free-form user-generated reviews in e-commerce and review websites continues to increase, there is a growing need for automatic mechanisms that sift through the vast number of reviews and identify quality content. Online review helpfulness modeling and prediction is a task which studies the factors that determine review helpfulness and attempts to accurately predict it. This survey paper provides an overview of the most relevant work on product review helpfulness prediction and understanding in the past decade, discusses gained insights, and provides guidelines for future research.

1 Introduction

Research on the computational modeling and prediction of online review helpfulness has generally proceeded in two directions. One concerns the automatic prediction of the helpfulness of a review, where helpfulness is typically defined as the fraction of "helpful" votes it receives. Review helpfulness research in the NLP and text mining communities has largely focused on identifying textual content features of a review that are useful for automatic helpfulness prediction. The other direction concerns understanding the nature of helpfulness, where researchers seek to understand the process of human evaluation of review helpfulness and the factors that influence it.

The increasing popularity of modeling and prediction of review helpfulness since its inception more than a decade ago can be attributed to its practical significance. Nowadays, customers regularly rely on different kinds of user reviews (e.g., hotels, restaurants, products, movies) to decide what to spend their money on. Given the large number of reviews available in web platforms, a review helpfulness prediction system could substantially save people's time by allowing them to focus on the most helpful reviews. Hence, a successful review helpfulness prediction system could be as useful as a product recommender system.

Unfortunately, unlike in many key areas of research in NLP, it is by no means easy to determine the state of the art in automatic helpfulness prediction. Empirical comparisons are complicated for at least two reasons. First, historically, systems have been trained on different datasets, not all of which are publicly available. Second, researchers have not built on the successes of each other, evaluating their ideas against baselines that are not necessarily the state of the art. Worse still, new features are not always properly evaluated. This somewhat disorganized situation can be attributed in part to the lack of a common forum for researchers to discuss a long-term vision and a roadmap for research in this area.

Our goal in this survey is to present an overview of the current state of research on computational modeling and prediction of product review helpfulness. Our focus on product reviews is motivated by the fact that they are the most widely studied type of review. Despite this focus, it is by no means the case that our work is only applicable to product reviews. While online platforms differ in objectives and review domains (e.g., Amazon is an online product store, Yelp is a business review website, and TripAdvisor is a booking website for a variety of travel activities), the principles that govern the helpfulness voting process are robust across platforms and domains. This means that most, if not all, of our findings are transferable to other kinds of online reviews. We believe that this survey will be useful to researchers and developers interested in a better understanding of the mechanisms behind review helpfulness.

2 Datasets

The main source of product reviews used in past research is Amazon.com, but interesting work has been done on data from Ciao.com (a now defunct product review website). The main difference between these two sources is the metadata associated with them: Amazon.com offers anonymous voting information, whereas Ciao attaches userIDs to helpfulness votes. Ciao also uses helpfulness votes in the range of 0 to 5, whereas Amazon votes are binary. Furthermore, Ciao offers information on a social trust network, where users choose to connect to reviewers if they find their reviews consistently helpful, unlike Amazon.com, which does not offer any such social trust network. These differences have allowed researchers to make observations on Ciao.com data that cannot be made on Amazon.com.

Datasets are collected from the aforementioned sources through web scraping or APIs. When it comes to Amazon datasets, researchers can choose one of two pre-collected datasets: the Multi-Domain Sentiment Dataset[1](Blitzer et al., 2007) (MDSD) and the Amazon Review Dataset[2] (McAuley et al., 2015; He and McAuley, 2016) (ARD). These datasets have a similar number of product categories (25 and 24, respectively). However, the latest version of MDSD contains 1,422,530 reviews, while ARD contains 142.8 million reviews. Furthermore, ARD offers a variety of metadata that is not present in MDSD (e.g., product salesrank). To the best of our knowledge, there is only one pre-collected Ciao dataset[3] (302,232 reviews, 43,666 users, and 8,894,899 helpfulness votes), which was made available by Tang et al. (2013). Few researchers have used these pre-collected datasets, however. Instead, most have relied on collecting their own datasets directly from websites. As mentioned before, the general lack of testing on pre-collected datasets has made system comparisons difficult.

The majority of researchers simply use helpfulness scores (the fraction of users who vote a review as helpful) as found in websites as ground truth for system training and evaluation. Given that these scores are volatile when reviews have few votes, researchers frequently filter out reviews that do not have a minimum number of votes. Some researchers have argued that helpfulness scores might not be good indicators of actual helpfulness, and have resorted to rating or ranking reviews themselves (Liu et al., 2007; Tsur and Rappoport, 2009; Yang et al., 2015), but these approaches are not the norm.

Votes : [97, 102]
Text : I'm a much bigger fan of the Targus folding keyboard. For starters it folds into the size of a handspring. Second of all the Landware version's keys are incredibly small. The one feature benefit of landware is that it's a rigid design so it can be used on your lap - while the Targus version is very flexible and needs to be placed on a flat surface to type.

Figure 1: Example Review

Researchers have observed interesting patterns in review datasets. For instance, positive reviews are more likely to have high helpfulness scores (O'Mahony et al., 2010; Huang et al., 2015), top ranking reviews hold a disproportionate amount of votes when compared to lower-ranked reviews (Liu et al., 2007), and more recent reviews tend to get fewer votes than older reviews (Liu et al., 2007). Although some of these effects may be the consequence of website voting mechanisms (e.g., Amazon shows reviews based on their helpfulness), they should be taken in consideration when selecting and pre-processing datasets.

Perhaps the most important observation is that helpfulness scores may not be strongly correlated to review quality (Liu et al., 2007; Danescu-Niculescu-Mizil et al., 2009; Tsur and Rappoport, 2009; Ghose and Ipeirotis, 2011; Yang et al., 2015). In at least one study, independent annotators agreed more frequently (85%) with an alternate helpfulness ranking than with one based on helpfulness scores (Tsur and Rappoport, 2009). The example review in Figure 1 shows discrepancies between quality and score. While this review is relatively short and contains only a couple of judgments on its product, 97 out of 102 people voted it as helpful (0.95 score). The quality of this review does not seem to match its near-perfect score. As we will see in Section 4, these discrepancies could be explained as the consequence of several *moderating factors*, which have a direct influence on the helpfulness voting process but are largely ignored in current helpfulness prediction systems.

[1]`https://www.cs.jhu.edu/~mdredze/datasets/sentiment/`
[2]`http://jmcauley.ucsd.edu/data/amazon/`
[3]`https://www.cse.msu.edu/~tangjili/trust.html`

3 Helpfulness Prediction

Helpfulness prediction tasks include score *regression* (predicting the helpfulness score $h \in [0, 1]$ of a review), binary review *classification* (classifying a review as helpful or not), and review *ranking* (ordering a set of reviews by their helpfulness). In this section, we present the evaluation measures and approaches explored in past work.

3.1 Performance Measures

Regarding performance measures, classification tasks have used Precision, Recall, and F-measure. Regression tasks have mostly used mean squared error (MSE), which measures the average of the sum of the squared error, and root mean squared error (RMSE), which is defined as the square root of MSE. Ranking systems have used Normalized Discounted Cumulative Gain (NDCG), which is popularly used to measure the relevance of search results in information retrieval (here, helpfulness is used as a measure of relevance), and NDCG@k, a special version of NDCG that only takes into account the top k items in a ranking (this is used because users only read a limited number of reviews). Researchers have also used Pearson and Spearman correlations to measure model fit and ranking performance.

3.2 Approaches

Next, we provide a high-level overview of the approaches that have been employed to predict the helpfulness of online product reviews.

Regression has primarily been attempted through support vector regression (Kim et al., 2006; Zhang and Varadarajan, 2006; Yang et al., 2015). However, probabilistic matrix factorization (Tang et al., 2013), linear regression (Lu et al., 2010), and extended tensor factorization models (Moghaddam et al., 2012) have successfully been used to integrate sophisticated constraints into the learning process and have achieved improvements over regular regression models. Multi-layer neural networks have also been used towards this purpose (Lee and Choeh, 2014). In particular, there seems to be progress toward more sophisticated models. For instance, Mukherjee et al. (2017) used a HMM-LDA based model to jointly infer reviewer expertise, predict aspects, and review helpfulness, which showed significant improvement over simpler models. *Classification* approaches have mostly been based on SVMs (Kim et al., 2006; Hong et al., 2012; Zeng et al., 2014; Krishnamoorthy, 2015), but thresholded linear regression models (Ghose and Ipeirotis, 2011), Naive Bayes, Random Forests, J48 and JRip have also been used (O'Mahony et al., 2010; Ghose and Ipeirotis, 2011; Krishnamoorthy, 2015). Recent work has also approached this task with neural networks (Malik and Hussain, 2017; Chen et al., 2018). Regarding *ranking*, some researchers have used ranking-specific methods such as SVM ranking (Tsur and Rappoport, 2009; Hong et al., 2012), but others have attempted to recover rankings from classification (O'Mahony and Smyth, 2009, 2010) or regression (Mukherjee et al., 2017) outputs.

Table 1 provides an overview of some of the most relevant features used in helpfulness prediction systems, explains the intuition behind them and, whenever possible, their correlation to helpfulness and impact on performance. Here, we differentiate primarily between *content* and *context* features. *Content features* focus on information directly derived from the review, such as review text and star rating, whereas *context features* focus on information from outside the review, such as reviewer/user information.

Content features include *Review Length Features*, which are based on the intuition that longer reviews have more information and are thus more helpful; *Readability Features*, which are based on the conjecture that if a review is easier to read, it will be found helpful by more users; *Word-Based Features*, which are based on the idea of identifying key words whose presence indicates the importance of the information found in a review; *Word-Category Features*, which identify the presence of words belonging to specific word lists; and *Content Divergence Features*, which measure how different the contents of the review are from specific reference texts. Context features include *Reviewer Features*, which collect meaningful reviewer historical information to predict future helpfulness scores; and *User-Reviewer Idiosyncrasy Features*, which attempt to capture the similarity between users and reviewers. We also include a couple of *Miscellaneous Features*, which are based on metadata and sentiment analysis; these features are better understood in the context of the moderating factors presented in Section 4.

Researchers have managed to mostly agree on some observations regarding which features are

Feature	Description	Comments
Content Features		
Review Length Features: Measure review length using different metrics.		
Average Sentence Length	-	Used in Liu et al. (2007), Lu et al. (2010), and Yang et al. (2015) without studying its individual predictive power.
No. of Sentences	-	Used in Liu et al. (2007), Lu et al. (2010), Yang et al. (2015)
Number of Words	-	Positive correlation (Mudambi and Schuff, 2010); shown to subdue sentence features (Kim et al., 2006).
Readability Features: Measure how easy a review is to read.		
Readability	Measures how easy a text is to read	Ghose and Ipeirotis (2011) and Korfiatis et al. (2012) found a positive correlation.
Spelling Errors	-	Ghose and Ipeirotis (2011) found a negative correlation.
Paragraph Metrics	Avg. paragraph length, no. of paragraphs	Kim et al. (2006) found an insignificant difference when included in a binary classifier.
Word-Based Features: Indicate the presence of meaningful key words.		
Unigram TF-IDF	Degree of word importance in relation to all reviews for a product	Kim et al. (2006) observed a positive correlation and performance improvement when combined with review length.
Dominant Terms	Presence of particularly important terms for a specific book	Tsur and Rappoport (2009) based entire system on this metric. Tailored for book reviews: similar to UGR TF-IDF.
Word-Category Features: Indicate the presence of words of lists of semantically related words in review.		
Product features	Attempt to identify the presence of important topics	Liu et al. (2007) showed 2.89-3.22% improvement. Hong et al. (2012) presented a system which improves $\sim$ 8% accuracy over Kim et al. (2006) and Liu et al. (2007) but the individual predictive power of the feature was not analyzed. Kim et al. (2006) found it inferior to UGR TF-IDF.
Subjective Tokens	Words taken from lists of subjective adjectives and nouns	Zhang and Varadarajan (2006) found it "barely" correlated with helpfulness. No significant performance improvement.
Sentiment Words	Attempt to capture the presence of opinions, analyses, emotions etc.	Kim et al. (2006) found these features inferior to UGR TF-IDF; Yang et al. (2015) found the opposite and significant improvement over simple text features regression.
Syntactic tokens	A variety of tokens including nouns, adjectives, adverbs, wh- determiners etc.	Kim et al. (2006) found no performance gains; Hong et al. (2012) built a system with volition auxiliaries and sentence tense which showed $\sim$ 8% accuracy improvement over Kim et al. (2006) and Liu et al. (2007), but the individual predictive power of these features was not studied.
Content Divergence Features: Measure the difference between reviews and some reference text.		
Review-product descr. divergence	Helpful reviews should echo the contents of product description	Zhang and Varadarajan (2006) found no significant improvement in model correlation.
Sentiment divergence	The mainstream opinion polarity for a product and its strength are compared to those of the review	Hong et al. (2012) presented a system which improved $\sim$ 8% accuracy over Kim et al. (2006) and Liu et al. (2007) but the individual predictive power of the feature was not analyzed.
KL average review divergence	Divergence between the unigram language model of the review and aggregated product reviews	Lu et al. (2010) introduced it in their baseline model along with a variety of features; the individual predictive power of the feature was not studied.
Miscellaneous Features		
Star rating	The review-assigned product star rating	Positively correlated to helpfulness (Huang et al., 2015). Influence explained by Danescu-Niculescu-Mizil et al. (2009) and Mudambi and Schuff (2010) (see Sections 4.4, 4.2).
Subjectivity	The probability of a review and its sentences being subjective	Based on the conjecture that readers prefer subjective or objective info. based on product type. Empirical evidence found in Ghose and Ipeirotis (2011) (see Section 4.5).
Context Features		
Reviewer Features: Capture reviewer statistics.		
# Past Reviews	Previous reviews written by reviewer	No influence found by Huang et al. (2015).
# Helpful Votes	Previous votes received by reviewer	No influence found by Huang et al. (2015).
Avg. Helpfulness	Reviewer avg. past helpfulness	Positive correlation found by Huang et al. (2015). Mixed effects found by Ghose and Ipeirotis (2011).
User-Reviewer Idiosyncrasy: Capture the similarity between users and reviewers.		
Connection Strength	User-Reviewer connection strength in a social network using the metric introduced in Tang et al. (2012)	Relative performance increase of 1.15-28.38% (Lu et al., 2010; Tang et al., 2013) (see Section 4.3)
User-Reviewer Product Rating Similarity	User-Reviewer product rating history similarity	Relative performance increase of 28.38% (Tang et al., 2013) (see Section 4.3)

Table 1: Summary of Observed Features on Helpfulness

useful for helpfulness prediction[4]. *Review length* has been shown multiple times to be strongly (positively) correlated to helpfulness (Kim et al., 2006; Liu et al., 2007; Otterbacher, 2009; Mudambi and Schuff, 2010; Cao et al., 2011; Pan and Zhang, 2011; Yang et al., 2015; Bjering et al., 2015; Huang et al., 2015; Salehan and Kim, 2016) with only few researchers disagreeing on the existence of the correlation (Zhang and Varadarajan, 2006; Korfiatis et al., 2012). There is general agreement that a review's *star rating* can also be useful for helpfulness prediction. Some researchers use the extremity of the rating (positive, negative, neutral) as a feature (positive and negative reviews are seen as more useful than neutral reviews) (Ghose and Ipeirotis, 2011), while others use star ratings directly (Kim et al., 2006; Mudambi and Schuff, 2010; Pan and Zhang, 2011; Zeng et al., 2014; Huang et al., 2015; Bjering et al., 2015). Some researchers argue that star rating is useful because of the presence of *positivity bias* (i.e., reviews with positive star ratings are seen as more helpful), while few researchers disagree on the existence of a connection between star ratings and helpfulness (Otterbacher, 2009). *Review readability metrics*, which measure how "easy" it is to read a review, have been found to have a positive correlation to helpfulness (Ghose and Ipeirotis, 2011; Korfiatis et al., 2012), but have not been as thoroughly tested as other features. A recurrent idea is that of capturing *review content relevance*: unigram TF-IDF statistics (the relative importance of the words in a review when compared to other reviews of the same product) (Kim et al., 2006), dominant terms (computed using a custom metric similar to TF-IDF, but tailored for book reviews) (Tsur and Rappoport, 2009), and latent review topics (the themes present in the review) (McAuley and Leskovec, 2013; Mukherjee et al., 2017) stand out particularly.

3.3 The State of Helpfulness Prediction

The classical approach to helpfulness prediction has consisted of finding new hand-crafted features that can improve system performance. Although many interesting features continue to be found (e.g., emotion (Martin and Pu, 2014), aspect (Yang et al., 2016), and argument (Liu et al., 2017) based features), advances have been hindered by the lack of standard datasets, which are needed for performance comparisons, and feature ablation studies, which are needed to properly evaluate the contribution of newly proposed features.

Even so, as in many other areas of NLP, recent systems based on neural network architectures have shown performance increases both when using hand-crafted features (Lee and Choeh, 2014; Malik and Hussain, 2017) and when performing raw-text predictions (Chen et al., 2018). Moreover, recent systems have been shown to be able to tackle domain knowledge transfer considerably well (Chen et al., 2018). Although these systems were not compared against a robust hand-crafted feature baseline, the fact that authors are beginning to use pre-collected datasets (ARD) enables fairer comparisons. Intuitively, we expect models based on neural network architectures to be better at capturing latent semantics, as well as some of the feature interactions we will present in Section 4. In parallel, systems that have incorporated user and reviewer features, particularly those that learn from individual user votes (Tang et al., 2013), have shown large performance increases over extensive hand-crafted-only feature baselines (Lu et al., 2010; Tang et al., 2013), and more sophisticated models focused on review semantics (Mukherjee et al., 2017) have also outperformed hand-crafted-only feature baselines significantly.

4 The Helpfulness Voting Process: Entities and Moderating Factors

So far we have presented an overview of the features used in helpfulness prediction systems. With a few exceptions (Mudambi and Schuff, 2010; Ghose and Ipeirotis, 2011; Tang et al., 2013), past work on helpfulness prediction has focused exclusively on *non-moderating factors* (i.e., observable features which can contribute towards helpfulness scores, but cannot alter or influence the voting process itself). Even so, researchers have gained key insights on certain *moderating factors* (i.e., mechanisms and properties that can influence the voting process outcome). These findings are relevant not only because they can be used to enhance helpfulness prediction, but because, when put together, they constitute arguments in favor of reconsidering the helpfulness prediction task and its focus. In this section, we will present a variety of moderating factors.

[4]We do not discuss features that are not helpful since, in general, they are not as thoroughly tested as those mentioned here.

4.1 The Voting Process and its Entities

To start our discussion on moderating factors, let us provide a brief, intuitive definition of the steps involved in the helpfulness voting process and outline the entities involved in it[5]:

1. A reviewer, a, writes a review r on product p
2. A user, u, reads the review by reviewer a on product p and internally assigns it a score s using some criterion c.
3. If the score s is over some threshold t, the user votes the review as "helpful". Otherwise, the user votes it as "not helpful".

Intuitively, one can expect these four entities — reviewers, users, reviews, and products — to play a role in determining the outcome of the voting process. Moreover, it is reasonable to expect both the nature of these entities and the interactions between them to be sometimes expressed through hidden features/variables. For instance, one cannot directly observe a user's opinion of a product unless he/she writes a review, and one cannot directly observe a particular user's information needs or a product's nature, which would indicate what kind of review is most helpful for it. In the next subsections, we will discuss different moderating factors that have been discovered for each of these entities, the observable features that have been used to approximate them, and their effects on the voting process.

4.2 User-Product Predispositions

Danescu-Niculescu-Mizil et al. (2009) showed that the difference between user and reviewer opinions can influence helpfulness votes. Since user opinions are hidden, based on the assumption that star ratings are good indicators of opinion, Danescu et al. studied the interplay between review star rating deviation from the mean (the divergence between the reviewer's opinion and the average opinion of the product) and star rating variance (the level of opinion consensus for a product) for 1 million Amazon US book reviews, making the following observations:

1. When star rating variance is very low, the most helpful reviews are those with the average star rating.
2. With moderate variance, the most helpful reviews are those with a slightly-above-average star rating.
3. As variance becomes large, reviews with star ratings both above and below the average are more helpful (positive reviews still deemed somewhat more helpful).

These observations held when controlling for review text, and constitute one of the most straightforward pieces of evidence against text-only review helpfulness understanding and prediction. Although these observations show only aggregated user behavior, they have a theoretical backing by past research (Wilson and Peterson, 1989), and hint that a deeper understanding of user opinions can lead to better prediction systems.

4.3 User-Reviewer Idiosyncrasy

Tang et al. (2013) found that, by observing users' actions, user-reviewer idiosyncrasy similarity could be measured and used to enhance helpfulness prediction. They showed that the existence and strength of connections between reviewers and users in a social network, along with product rating history similarity, moderated the general user opinion of a particular reviewer's reviews. Specifically, they analyzed *social network connections* in Ciao's *circle of trust*, a social network where a user *connects* to a reviewer if they consistently find their reviews helpful, along with users' and reviewers' product rating histories, and made the following observations:

1. Users are likely to think of reviews from their connected reviewers as more helpful.
2. The more strongly users connect to a reviewer, the more helpful users consider the reviews from the reviewer.[6]
3. Users are likely to consider the reviews from reviewers with similar product ratings as more helpful.
4. The more similar the product ratings of users and reviewers, the more helpful users consider the reviews from the reviewer.

As Tang et al. proposed that differences in helpfulness scores are not necessarily a consequence of review quality, but of differences of opinion between users (if everyone thought the same way, all reviews would have a score of either 0 or 1), they were among the first to advocate for user-specific helpfulness prediction, which aims to predict how a specific user will vote, instead of predicting the

[5]Here we assume voting participation and do not attempt to reconcile it with polarity, but a deeper understanding of participation could lead to better interpretations of votes.

[6]Connection strength is measured with the metric introduced in Tang et al. (2012).

aggregated votes of the community. Under this approach, Tang et al. implemented their observations in a probabilistic matrix factorization framework and achieved a 28.38% relative improvement over a text-reviewer-based baseline that included an extensive set of text features present in other systems (Lu et al., 2010).

This suggests that the similarity between reviewers' idiosyncrasy as expressed in reviews and that of users can be approximated by studying user and reviewer actions. Further, the information used by Tang et al. (2013) towards this purpose is not the only kind that could prove useful. It could easily be extended to include the vast amount of user information stored by current day e-commerce websites such as Amazon. Users' age, gender, purchase history, location, browsing and purchase patterns, and review history (both writing and rating) could be used to define prior probabilities on some user x liking the review of a reviewer y.[7] As some of this information has already been used in recommender systems, it would be of interest to explore the extent to which techniques from this field (specifically those from collaborative filtering) can be applied to helpfulness prediction.

4.4 Product Nature

Product nature moderates users' information needs and the criteria of a helpful review. Online stores now have an astoundingly large catalog of products, which can be very different in price, use, target market, complexity, popularity, etc. Hence, it is reasonable to expect the information needs of users to depend at least somewhat on the product in question. Consider the task of *buying a house vs buying a TV*. We can easily see that the amount and nature of information needed to buy a TV or a house is considerably different. Further, the quality of these products stems from different sources: a TV's perceived quality depends mostly on its technical features, whereas the perceived quality of a house depends to some degree on the potential buyer. Therefore, it is perfectly sensible to expect helpful reviews for products of different "types" to be different. Below we show that the nature of a product moderates the effects of star ratings, review length, and subjectivity on helpfulness scores.

[7]Since a reviewer's idiosyncrasy is embodied in his/her reviews, we do not rule out the possibility that more complex text representations can also be used to approximate it. Regardless, these sources of information should still be able to complement prediction systems.

Researchers have proven the influence of product nature on the helpfulness voting process by differentiating between *search* and *experience* goods. According to Nelson (1970, 1974), the quality of search goods is derived from objective attributes (e.g., a camera), whereas the quality of experience goods is based on subjective attributes (e.g., a music CD). Mudambi and Schuff (2010) first identified that review length (word count) is positively correlated to review helpfulness, and then made the following observations:

- For experience goods, reviews with extreme star ratings (high or low) are associated with lower levels of helpfulness than reviews with moderate star ratings.
- Review depth has a greater positive effect on the helpfulness of the review for search goods than experience goods.

These observations make it clear that the nature of a product can impact the way a user will judge a review's helpfulness. However, approximating the nature of a product is not a trivial task. As stated by Mudambi and Schuff, even if these observations hold, classifying products as search or experience goods is a complicated task, since products fall at some point along a spectrum and commonly have aspects of both search and experience goods. This means that finding methods of automatically discovering product features or classifications that influence the helpfulness voting process is an important task for future research.

What other product categorizations are there that could influence helpfulness and be easily collected/computed? We propose to start by using categories already present in e-commerce websites. Intuitively, it would make sense for products under the "computers" category to be similar in their information needs. And as such, systems trained on computer reviews should learn similar parameters. As most e-commerce websites use a hierarchical product categorization system, by starting at the most specific subcategories one could potentially generalize subcategory-learned parameters into category-wide trends.

4.5 Review Nature

A review's style influences the properties that make it helpful. It is well known that when it comes to expressing opinions, the way information is presented can be almost as important as the information itself. Even if two reviewers have a similar opinion on a product, the way they frame their opinion can make a big difference when it comes to how helpful their reviews are. Consider the task of deciding whether to buy a specific car. What advice could prove useful for this decision? We could consider *regular* advice that is mostly concerned with the car itself, *comparative* advice that relates various aspects of the car with its alternatives, and *suggestive* advice, which focuses on usage recommendations.

Qazi et al. (2016) used these three types of advice to classify hotel reviews from TripAdvisor.com and made the following observations:

- For comparative reviews, longer reviews are considered more helpful.
- For suggestive and regular reviews, shorter reviews are more helpful.

Similar findings on the influence of review nature were made by Huang et al. (2015): when differentiating between reviews written by regular and top Amazon reviewers, they made the following observations:

- The influence of word count on review helpfulness is bounded (after 144 words, the effect stops) for regular reviewers.
- For top reviewers, the effect is nonexistent.

Similarly to product nature, an important research question for future work is how to identify and exploit review categories for effective helpfulness prediction. We expect more sophisticated textual features to be necessary to differentiate between meaningful styles of reviews.

4.6 Review Context

Sipos et al. (2014) found evidence that helpfulness votes are the consequence of judgments of *relative quality* (i.e., how the review compares to its neighbors) and that aggregate user voting polarity is influenced by the specific review ranking that websites display at any given point in time. To prove this, they collected daily snapshots of the top 50 reviews of 595 Amazon products over a 5 month period. Four months after the data collection period ended, they collected the full review rankings for all 595 products. This final review ranking was taken to be the "true" ranking. They studied daily changes and observed that:

- A review receives more positive votes when it is under-ranked (under its final ranking).
- A review receives more positive votes when it is superior to its neighbors.
- A review receives fewer positive votes when it is over-ranked (over its final ranking).
- A review receives fewer positive votes when it is locally inferior to its neighbors.

Sipos et al. noted that these observations are consistent with the interpretation that users vote to correct "misorderings" in the ranking. This has important consequences for user-specific helpfulness prediction systems. Recall that votes may express judgments over a set of reviews. If researchers build training sets that identify user votes and contain sufficient information to replicate context at the time of voting, systems could learn more about user preferences: a vote would no longer inform solely on a user's perceived helpfulness of a review x, but on the user's perceived helpfulness of x with respect to its neighbors. This could be particularly useful in sparsity scenarios, and could lead to better helpfulness predictions.

5 Conclusions and Recommendations

Online product review helpfulness modeling and prediction is a multi-faceted task that involves using content and context information to understand and predict helpfulness scores. Researchers now have at their disposal at least three public, pre-collected product review datasets — MDSD, ARD, and Ciao — to build and test systems. Although significant advances have been made on finding hand-crafted features for helpfulness prediction, effective comparisons between proposed approaches have been hindered by the lack of standard evaluation datasets, well-defined baselines, and feature ablation studies. However, there have been exciting developments in helpfulness prediction: systems that have attempted to exploit user and reviewer information, along with those based on sophisticated models (e.g., probabilistic matrix factorization, HMM-LDA) and neural network architectures, are promising prospects for future work. Furthermore, a variety of insightful observations have been made on moderating factors. In particular, product opinions, user idiosyncrasy, product and review nature, along with review voting context have been shown to influence the way users vote. This provides suggestive evidence that researchers should adopt a holistic view of the helpfulness voting process, which may require information not present in current datasets.

We conclude our survey with several recommendations for future work on computational modeling and prediction of review helpfulness.

Task If one acknowledges the role that users play in determining whether a review is helpful or not, it seems contradictory to insist on predicting helpfulness scores, which represent the average perception of a subset of users that (1) may not be representative of the entire population and (2) may not serve users well if their perceptions do not align with the subset of users that voted (even if the subset consisted of the entire population). This is why we consider that user-specific helpfulness prediction, first presented in Moghaddam et al. (2012) and Tang et al. (2013), should be the goal of future work, as it allows systems to tailor their predictions to users' preferences and needs (much like a recommender system). Note that pursuing user-specific helpfulness prediction is not enough. A substantial amount of work must still be done to find, approximate, and implement moderating factors in helpfulness prediction systems, as well as build models that can adequately reflect the effects of these factors.

Data Given that we recommend user-specific helpfulness prediction, we propose the development of a gold standard that contains information that can facilitate the design of user-specific models (e.g., records of who voted and how, data relevant to user-profiling recommendations such as age, location, social networks, purchase and browsing history and patterns, product reviews written, and review and product rating histories). Furthermore, as users frequently vote on reviews in a different context (scores and neighboring reviews can vary over time), this dataset should include temporal information, which would allow researchers to reconstruct the context under which votes are cast. To build this dataset, we recommend that researchers work with companies such as Amazon, which may have such information.

Features and knowledge sources While we encourage the development of user-specific helpfulness prediction, we by no means imply that a model should be trained for each user. In fact, this may not be feasible if a user has cast only a small number of votes. There are multiple ways to approach this task. One is to train a user-specific model for each *cluster* of "similar" users. Taking inspirations from collaborative filtering, we could define or learn user similarity based on their purchasing/browsing/review and product rating histories (Liu et al., 2014) as well as profiling information (Krulwich, 1997), which should be available in the aforementioned dataset. Further, "similar" reviews (i.e., reviews on which users vote similarly) could be exploited (Sarwar et al., 2001; Linden et al., 2003). Once product and user/reviewer factors are incorporated into a model, it may become feasible to use past instances to predict helpfulness votes (how similar is a test instance to past situations where a user has voted "helpful"?).

Baseline systems To design a strong baseline system, first, researchers should consider all proposed features so far, including content features, context features, and features used to approach moderating factors. Second, combinations of these features should be systematically tested on the different models proposed by researchers. As we have seen that product nature influences the voting process, these tests should be conducted over different products and product categories. We recommend identifying specific experience and search products, since the effects of product nature have already been proven for them. Although ideally, these tests would be carried out on our proposed gold-standard dataset, we believe that the Ciao dataset introduced in Tang et al. (2013) and ARD (McAuley et al., 2015) can prove useful to define a baseline in the short term. Towards this purpose, the systems proposed in Tang et al. (2013), Mukherjee et al. (2017), Malik and Hussain (2017), and Chen et al. (2018) could serve as baselines after being enriched with extra features.

Other platforms, review domains and languages While we focused on Amazon product reviews written in English, the majority of the features discussed in Section 3 are platform-, domain- and language-independent, and the existence and importance of moderating factors described in Section 4 is by no means limited to product reviews. Consequently, we encourage researchers to evaluate the usefulness of these features and study these moderating factors in different domains, platforms, and languages, possibly identifying new features and moderating factors.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of the paper. This work was supported in part by USAF Grant FA9550-15-1-0346.

References

Einar Bjering, Lars Jaakko Havro, and Oystein Moen. 2015. An empirical investigation of self-selection bias and factors influencing review helpfulness. *International Journal of Business and Management*, 10(7):16–30.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

Qing Cao, Wenjing Duan, and Qiwei Gan. 2011. Exploring determinants of voting for the helpfulness of online user reviews: A text mining approach. *Decision Support Systems*, 50(2):511–521.

Cen Chen, Yinfei Yang, Jun Zhou, Xiaolong Li, and Forrest Sheng Bao. 2018. Cross-domain review helpfulness prediction based on convolutional neural networks with auxiliary domain discriminators. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 602–607.

Cristian Danescu-Niculescu-Mizil, Gueorgi Kossinets, Jon Kleinberg, and Lillian Lee. 2009. How opinions are received by online communities: A case study on Amazon.com helpfulness votes. In *Proceedings of the 18th International Conference on World Wide Web*, pages 141–150.

Anindya Ghose and Panagiotis G. Ipeirotis. 2011. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *IEEE Transactions on Knowledge and Data Engineering*, 23(10):1498–1512.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517.

Yu Hong, Jun Lu, Jianmin Yao, Qiaoming Zhu, and Guodong Zhou. 2012. What reviews are satisfactory: Novel features for automatic helpfulness voting. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 495–504.

Albert H. Huang, Kuanchin Chen, David C. Yen, and Trang P. Tran. 2015. A study of factors that contribute to online review helpfulness. *Computers in Human Behavior*, 48:17–27.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 423–430.

Nikolaos Korfiatis, Elena García-Bariocanal, and Salvador Sánchez-Alonso. 2012. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications*, 11(3):205–217.

Srikumar Krishnamoorthy. 2015. Linguistic features for review helpfulness prediction. *Expert Systems with Applications*, 42(7):3751–3759.

Bruce Krulwich. 1997. Lifestyle finder: Intelligent user profiling using large-scale demographic data. *AI Magazine*, 18(2):37–45.

Sangjae Lee and Joon Yeon Choeh. 2014. Predicting the helpfulness of online reviews using multilayer perceptron neural networks. *Expert Systems with Applications*, 41(6):3041–3046.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian, and Xuzhen Zhu. 2014. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, 56:156–166.

Haijing Liu, Yang Gao, Pin Lv, Mengxue Li, Shiqiang Geng, Minglan Li, and Hao Wang. 2017. Using argument-based features to predict and analyse review helpfulness. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1363.

Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. 2007. Low-quality product review detection in opinion summarization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 334–342.

Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. 2010. Exploiting social context for review quality prediction. In *Proceedings of the 19th International Conference on World Wide Web*, pages 691–700.

M.S.I. Malik and Ayyaz Hussain. 2017. Helpfulness of product reviews as a function of discrete positive and negative emotions. *Computers in Human Behavior*, 73:290–302.

Lionel Martin and Pearl Pu. 2014. Prediction of helpful reviews using emotions extraction. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1551–1557.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 165–172.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52.

Samaneh Moghaddam, Mohsen Jamali, and Martin Ester. 2012. ETF: Extended tensor factorization model for personalizing prediction of review helpfulness. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 163–172.

Susan M. Mudambi and David Schuff. 2010. What makes a helpful online review? A study of customer reviews on Amazon.com. *MIS Quarterly*, 34(1):185–200.

Subhabrata Mukherjee, Kashyap Popat, and Gerhard Weikum. 2017. Exploring latent semantic factors to find useful product reviews. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 480–488.

Phillip Nelson. 1970. Information and consumer behavior. *Journal of Political Economy*, 78(2):311–329.

Phillip Nelson. 1974. Advertising as information. *Journal of Political Economy*, 82(4):729–754.

Michael P. O'Mahony, Pádraig Cunningham, and Barry Smyth. 2010. An assessment of machine learning techniques for review recommendation. In *Artificial Intelligence and Cognitive Science*, pages 241–250.

Michael P. O'Mahony and Barry Smyth. 2009. Learning to recommend helpful hotel reviews. In *Proceedings of the Third ACM Conference on Recommender Systems*, pages 305–308.

Michael P. O'Mahony and Barry Smyth. 2010. A classification-based review recommender. *Knowledge-Based Systems*, 23(4):323–329.

Jahna Otterbacher. 2009. Helpfulness in online communities: A measure of message quality. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 955–964.

Yue Pan and Jason Q. Zhang. 2011. Born unequal: A study of the helpfulness of user-generated product reviews. *Journal of Retailing*, 87(4):598–612.

Aika Qazi, Karim Bux Shah Syed, Ram Gopal Raj, Erik Cambria, Muhammad Tahir, and Daniyal Alghazzawi. 2016. A concept-level approach to the analysis of online review helpfulness. *Computers in Human Behavior*, 58:75–81.

Mohammad Salehan and Dan J. Kim. 2016. Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics. *Decision Support Systems*, 81:30–40.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295.

Ruben Sipos, Arpita Ghosh, and Thorsten Joachims. 2014. Was this review helpful to you?: It depends! Context and voting patterns in online content. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 337–348.

Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. 2013. Context-aware review helpfulness rating prediction. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 1–8.

Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 93–102.

Oren Tsur and Ari Rappoport. 2009. Revrank: A fully unsupervised algorithm for selecting the most helpful book reviews. In *International AAAI Conference on Web and Social Media*, pages 154–161.

William R. Wilson and Robert A. Peterson. 1989. Some limits on the potency of word-of-mouth information. *Advances in Consumer Research*, 16:23–29.

Yinfei Yang, Cen Chen, and Forrest Sheng Bao. 2016. Aspect-based helpfulness prediction for online product reviews. In *Proceedings of the 28th IEEE International Conference on Tools with Artificial Intelligence*, pages 836–843.

Yinfei Yang, Yaowei Yan, Minghui Qiu, and Forrest Bao. 2015. Semantic analysis and helpfulness prediction of text for online product reviews. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 38–44.

Yi-Ching Zeng, Tsun Ku, Shih-Hung Wu, Liang-Pu Chen, and Gwo-Dong Chen. 2014. Modeling the helpful opinion mining of online consumer reviews as a classification problem. *International Journal of Computational Linguistics & Chinese Language Processing*, 19(2):17–32.

Zhu Zhang and Balaji Varadarajan. 2006. Utility scoring of product reviews. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 51–57.

Mining Cross-Cultural Differences and Similarities in Social Media

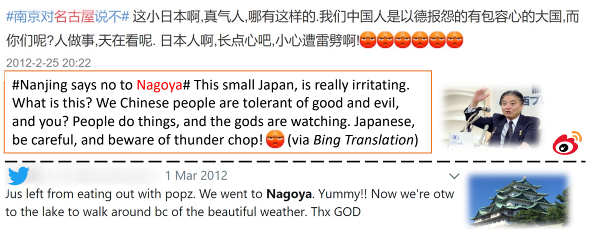
Bill Yuchen Lin[1*] **Frank F. Xu**[1*] **Kenny Q. Zhu**[1] **Seung-won Hwang**[2]
[1]Shanghai Jiao Tong University, Shanghai, China
{yuchenlin, frankxu}@sjtu.edu.cn, kzhu@cs.sjtu.edu.cn
[2]Yonsei University, Seoul, Republic of Korea
seungwonh@yonsei.ac.kr

Abstract

Cross-cultural differences and similarities are common in cross-lingual natural language understanding, especially for research in social media. For instance, people of distinct cultures often hold different opinions on a single named entity. Also, understanding slang terms across languages requires knowledge of cross-cultural similarities. In this paper, we study the problem of computing such cross-cultural differences and similarities. We present a lightweight yet effective approach, and evaluate it on two novel tasks: 1) mining cross-cultural differences of named entities and 2) finding similar terms for slang across languages. Experimental results show that our framework substantially outperforms a number of baseline methods on both tasks. The framework could be useful for machine translation applications and research in computational social science.

1 Introduction

Computing similarities between terms is one of the most fundamental computational tasks in natural language understanding. Much work has been done in this area, most notably using the distributional properties drawn from large monolingual textual corpora to train vector representations of words or other linguistic units (Pennington et al., 2014; Le and Mikolov, 2014). However, computing cross-cultural similarities of terms between different cultures is still an open research question, which is important in cross-lingual natural language understanding. In this paper, we address cross-cultural research questions such as these:


#南京对名古屋说不# 这小日本啊,真气人,哪有这样的.我们中国人是以德报怨的有包容心的大国,而你们呢?人做事,天在看呢. 日本人啊,长点心吧,小心遭雷劈啊!😡😡😡😡😡😡
2012-2-25 20:22

#Nanjing says no to Nagoya# This small Japan, is really irritating. What is this? We Chinese people are tolerant of good and evil, and you? People do things, and the gods are watching. Japanese, be careful, and beware of thunder chop! 😡 (via *Bing Translation*)

[redacted] 1 Mar 2012
Jus left from eating out with popz. We went to **Nagoya**. Yummy!! Now we're otw to the lake to walk around bc of the beautiful weather. Thx GOD

Figure 1: Two social media messages about Nagoya from different cultures in 2012

1. *Were there any cross-cultural differences between Nagoya (a city in Japan) for native English speakers and 名古屋 (Nagoya in Chinese) for Chinese people in 2012?*
2. *What English terms can be used to explain "浮云" (a Chinese slang term)?*

These kinds of questions about cross-cultural differences and similarities are important in cross-cultural social studies, multi-lingual sentiment analysis, culturally sensitive machine translation, and many other NLP tasks, especially in social media. We propose two novel tasks in mining them from social media.

The first task (Section 4) is to mine cross-cultural differences in the perception of named entities (e.g., persons, places and organizations). Back in 2012, in the case of "Nagoya", many native English speakers posted their pleasant travel experiences in Nagoya on Twitter. However, Chinese people overwhelmingly greeted the city with anger and condemnation on *Weibo* (a Chinese version of Twitter), because the city mayor denied the truthfulness of the Nanjing Massacre. Figure 1 illustrates two example microblog messages about Nagoya in Twitter and Weibo respectively.

The second task (Section 5) is to find similar terms for slang across cultures and languages. Social media is always a rich soil where slang terms emerge in many cultures. For example,

*Both authors contributed equally.

"浮云" literally means "floating clouds", but now almost equals to "nothingness" on the Chinese web. Our experiments show that well-known online machine translators such as *Google Translate* are only able to translate such slang terms to their literal meanings, even under clear contexts where slang meanings are much more appropriate.

Enabling intelligent agents to understand such cross-cultural knowledge can benefit their performances in various cross-lingual language processing tasks. Both tasks share the same core problem, which is **how to compute cross-cultural differences (or similarities) between two terms from different cultures.** A term here can be either an ordinary word, an entity name, or a slang term. We focus on names and slang in this paper for they convey more social and cultural connotations.

There are many works on cross-lingual word representation (Ruder et al., 2017) to compute general cross-lingual similarities (Camacho-Collados et al., 2017). Most existing models require bilingual supervision such as aligned parallel corpora, bilingual lexicons, or comparable documents (Sarath et al., 2014; Kočiský et al., 2014; Upadhyay et al., 2016). However, they do not purposely preserve social or cultural characteristics of named entities or slang terms, and the required parallel corpora are rare and expensive.

In this paper, we propose a lightweight yet effective approach to project two incompatible monolingual word vector spaces into a single bilingual word vector space, known as social vector space (*SocVec*). A key element of SocVec is the idea of "bilingual social lexicon", which contains bilingual mappings of selected words reflecting psychological processes, which we believe are central to capturing the socio-linguistic characteristics. Our contribution in this paper is two-fold:

(a) We present an effective approach (SocVec) to mine cross-cultural similarities and differences of terms, which could benefit research in machine translation, cross-cultural social media analysis, and other cross-lingual research in natural language processing and computational social science.

(b) We propose two novel and important tasks in cross-cultural social studies and social media analysis. Experimental results on our annotated datasets show that the proposed method outperforms many strong baseline methods.

2 The *SocVec* Framework

In this section, we first discuss the intuition behind our model, the concept of "social words" and our notations. Then, we present the overall workflow of our approach. We finally describe the *SocVec* framework in detail.

2.1 Problem Statement

We choose (English, Chinese) to be the target language pair throughout this paper for the salient cross-cultural differences between the east and the west[1]. Given an English term W and a Chinese term U, the core research question is how to compute a similarity score, $ccsim(W, U)$, to represent the *cross-cultural similarities* between them.

We cannot directly calculate the similarity between the monolingual word vectors of W and U, because they are trained separately and the semantics of dimension are not aligned. Thus, the challenge is to devise a way to compute similarities across two different vector spaces while retaining their respective cultural characteristics.

A very intuitive solution is to firstly translate the Chinese term U to its English counterpart U' through a Chinese-English bilingual lexicon, and then regard $ccsim(W, U)$ as the (cosine) similarity between W and U' with their monolingual word embeddings. However, this solution is not promising in some common cases for three reasons:

(a) if U is an OOV (Out of Vocabulary) term, e.g., a novel slang term, then there is probably no translation U' in bilingual lexicons.

(b) if W and U are names referring to the same named entity, then we have $U' = W$. Therefore, $ccsim(W, U)$ is just the similarity between W and itself, and we cannot capture any cross-cultural differences with this method.

(c) this approach does not explicitly preserve the cultural and social contexts of the terms.

To overcome the above problems, our intuition is to project both English and Chinese word vectors into a single third space, known as *SocVec*, and the projection is supposed to purposely carry cultural features of terms.

2.2 Social Words and Our Notations

Some research in psychology and sociology (Kitayama et al., 2000; Gareis and Wilkins, 2011)

[1]Nevertheless, the techniques are language independent and thus can be utilized for any language pairs so long as the necessary resources outlined in Section 2.3 are available.

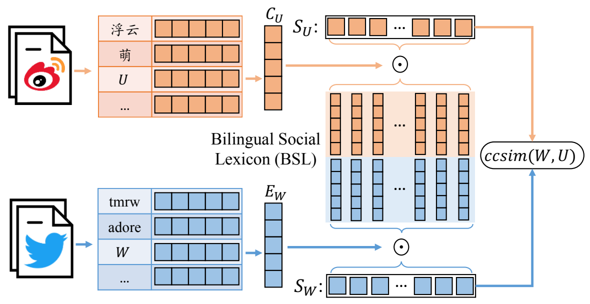


Figure 2: Workflow for computing the cross-cultural similarity between an English word *W* and a Chinese word *U*, denoted by $ccsim(W, U)$

show that culture can be highly related to emotions and opinions people express in their discussions. As suggested by Tausczik and Pennebaker (2009), we thus define the concept of "**social word**" as the words directly reflecting opinion, sentiment, cognition and other human psychological processes[2], which are important to capturing cultural and social characteristics. Both Elahi and Monachesi (2012) and Garimella et al. (2016a) find such *social words* are most effective culture/socio-linguistic features in identifying cross-cultural differences.

We use these notations throughout the paper: *CnVec* and *EnVec* denote the Chinese and English word vector space, respectively; *CSV* and *ESV* denote the Chinese and English social word vocab; *BL* means Bilingual Lexicon, and *BSL* is short for Bilingual Social Lexicon; finally, we use $\mathbf{E_x}$, $\mathbf{C_x}$ and $\mathbf{S_x}$ to denote the word vectors of the word x in *EnVec*, *CnVec* and *SocVec* spaces respectively.

2.3 Overall Workflow

Figure 2 shows the workflow of our framework to construct the *SocVec* and compute $ccsim(W, U)$. Our proposed *SocVec* model attacks the problem with the help of three low-cost external resources: (i) an English corpus and a Chinese corpus from social media; (ii) an English-to-Chinese bilingual lexicon (*BL*); (iii) an English social word vocabulary (*ESV*) and a Chinese one (*CSV*).

We train English and Chinese word embeddings (*EnVec* and *CnVec*) on the English and Chinese social media corpus respectively. Then, we build a *BSL* from the *CSV*, *ESV* and *BL* (see Section 2.4). The *BSL* further maps the previously incompatible *EnVec* and *CnVec* into a single common vector space *SocVec*, where two new vectors, S_W for W and S_U for U, are finally comparable.

2.4 Building the BSL

The process of building the *BSL* is illustrated in Figure 3. We first extract our bilingual lexicon (*BL*), where confidence score w_i represents the probability distribution on the multiple translations for each word. Afterwards, we use BL to translate each social word in the *ESV* to a set of Chinese words and then filter out all the words that are not in the *CSV*. Now, we have a set of Chinese social words for each English social word, which is denoted by a "translation set". The final step is to generate a Chinese "pseudo-word" for each English social word using their corresponding translation sets. A "pseudo-word" can be either a real word that is the most representative word in the translation set, or an imaginary word whose vector is a certain combination of the vectors of the words in the translation set.

For example, in Figure 3, the English social word "*fawn*" has three Chinese translations in the bilingual lexicon, but only two of them (underlined) are in the CSV. Thus, we only keep these two in the translation set in the filtered bilingual lexicon. The pseudo-word generator takes the word vectors of the two words (in the black box), namely 奉承 (flatter) and 谄媚 (toady), as input, and generates the pseudo-word vector denoted by "*fawn**". Note that the direction of building *BSL* can also be from Chinese to English, in the same manner. However, we find that the current direction gives better results due to the better translation quality of our *BL* in this direction.

Given an English social word, we denote $\mathbf{t_i}$ as the i^{th} Chinese word of its translation set consisting of N social words. We design four intuitive types of pseudo-word generator as follows, which are tested in the experiments:

(1) Max. Maximum of the values in each dimension, assuming dimensionality is K:

$$\text{Pseudo}(\mathbf{C_{t_1}}, ..., \mathbf{C_{t_N}}) = \begin{bmatrix} max(C_{t_1}^{(1)}, ..., C_{t_N}^{(1)}) \\ \vdots \\ max(C_{t_1}^{(K)}, ..., C_{t_N}^{(K)}) \end{bmatrix}^{\mathrm{T}}$$

(2) Avg. Average of the values in every dimension:

$$\text{Pseudo}(\mathbf{C_{t_1}}, ..., \mathbf{C_{t_N}}) = \frac{1}{N}\sum_{i}^{N} \mathbf{C_{t_i}}$$

[2]Example social words in English include *fawn, inept, tremendous, gratitude, terror, terrific, loving, traumatic*, etc. We discuss the sources of such social words in Section 3.

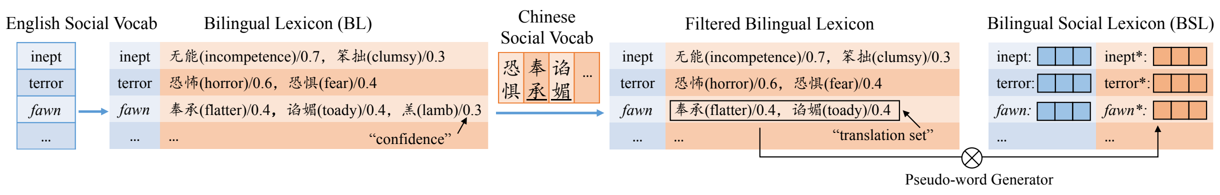

English Social Vocab
inept
terror
fawn
...
Bilingual Lexicon (BL)
inept 无能(incompetence)/0.7，笨拙(clumsy)/0.3
terror 恐怖(horror)/0.6，恐惧(fear)/0.4
fawn 奉承(flatter)/0.4，谄媚(toady)/0.4，羔(lamb)/0.3
“confidence”
Chinese Social Vocab
恐惧 奉承 谄媚 ...
Filtered Bilingual Lexicon
inept 无能(incompetence)/0.7，笨拙(clumsy)/0.3
terror 恐怖(horror)/0.6，恐惧(fear)/0.4
fawn 奉承(flatter)/0.4，谄媚(toady)/0.4
“translation set”
Bilingual Social Lexicon (BSL)
inept: inept*:
terror: terror*:
fawn: fawn*:
Pseudo-word Generator

Figure 3: Generating an entry in the BSL for “*fawn*” and its pseudo-word “*fawn**”

(3) WAvg. Weighted average value of every dimension with respect to the translation confidence:

$$\text{Pseudo}(\mathbf{C_{t_1}}, ..., \mathbf{C_{t_N}}) = \frac{1}{N}\sum_{i}^{N} w_i \mathbf{C_{t_i}}$$

(4) Top. The most confident translation:

$$\text{Pseudo}(\mathbf{C_{t_1}}, ..., \mathbf{C_{t_N}}) = \mathbf{C_{t_k}}, k = \operatorname*{argmax}_i w_i$$

Finally, the *BSL* contains a set of English-Chinese word vector pairs, where each entry represents an English social word and its Chinese pseudo-word based on its “translation set”.

2.5 Constructing the SocVec Space

Let B_i denote the English word of the i^{th} entry of the *BSL*, and its corresponding Chinese pseudo-word is denoted by B_i^*. We can project the English word vector $\mathbf{E_W}$ into the *SocVec* space by computing the cosine similarities between $\mathbf{E_W}$ and each English word vector in *BSL* as values on SocVec dimensions, effectively constructing a new vector $\mathbf{S_W}$ of size L. Similarly, we map a Chinese word vector $\mathbf{C_U}$ to be a new vector $\mathbf{S_U}$. $\mathbf{S_W}$ and $\mathbf{S_U}$ belong to the same vector space *SocVec* and are comparable. The following equation illustrates the projection, and how to compute *ccsim*[3].

$$\begin{aligned} &ccsim(W, U) := f(\mathbf{E_W}, \mathbf{C_U}) \\ &= sim\left(\begin{bmatrix} cos(\mathbf{E_W}, \mathbf{E_{B_1}}) \\ \vdots \\ cos(\mathbf{E_W}, \mathbf{E_{B_L}}) \end{bmatrix}^{\mathrm{T}}, \begin{bmatrix} cos(\mathbf{C_U}, \mathbf{C_{B_1^*}}) \\ \vdots \\ cos(\mathbf{C_U}, \mathbf{C_{B_L^*}}) \end{bmatrix}^{\mathrm{T}}\right) \\ &= sim(\mathbf{S_W}, \mathbf{S_U}) \end{aligned}$$

For example, if W is “Nagoya” and U is “名古屋”, we compute the cosine similarities between “Nagoya” and each English social word in the *BSL* with their monolingual word embeddings in English. Such similarities compose $\mathbf{S}_{\text{nagoya}}$. Similarly, we compute the cosine similarities between “名古屋” and each Chinese pseudo-word, and compose the social word vector $\mathbf{S}_{名古屋}$.

In other words, for each culture/language, the new word vectors like S_W are constructed based on the monolingual similarities of each word to the vectors of a set of task-related words (“social words” in our case). This is also a significant part of the novelty of our transformation method.

3 Experimental Setup

Prior to evaluating *SocVec* with our two proposed tasks in Section 4 and Section 5, we present our preparation steps as follows.

Social Media Corpora Our English Twitter corpus is obtained from Archive Team’s Twitter stream grab[4]. The Chinese Weibo corpus comes from Open Weiboscope Data Access[5] (Fu et al., 2013). Both corpora cover the whole year of 2012. We then randomly down-sample each corpus to 100 million messages where each message contains at least 10 characters, normalize the text (Han et al., 2012), lemmatize the text (Manning et al., 2014) and use LTP (Che et al., 2010) to perform word segmentation for the Chinese corpus.

Entity Linking and Word Embedding Entity linking is a preprocessing step which links various entity mentions (surface forms) to the identity of corresponding entities. For the Twitter corpus, we use Wikifier (Ratinov et al., 2011; Cheng and Roth, 2013), a widely used entity linker in English. Because no sophisticated tool for Chinese short text is available, we implement our own tool that is greedy for high precision. We train English and Chinese monolingual word embedding respectively using *word2vec*’s skip-gram method with a window size of 5 (Mikolov et al., 2013b).

Bilingual Lexicon Our bilingual lexicon is collected from *Microsoft Translator*[6], which translates English words to multiple Chinese words

[3]The function *sim* is a generic similarity function, for which several metrics are considered in experiments.

[4]`https://archive.org/details/twitterstream`
[5]`http://weiboscope.jmsc.hku.hk/datazip/`
[6]`http://www.bing.com/translator/api/Dictionary/Lookup?from=en&to=zh-CHS&text=<input_word>`

with confidence scores. Note that all named entities and slang terms used in the following experiments are excluded from this bilingual lexicon.

Social Word Vocabulary Our social word vocabularies come from *Empath* (Fast et al., 2016) and *OpinionFinder* (Choi et al., 2005) for English, and *TextMind* (Gao et al., 2013) for Chinese. Empath is similar to LIWC (Tausczik and Pennebaker, 2009), but has more words and more categories and is publicly available. We manually select 91 categories of words that are relevant to human perception and psychological processes following Garimella et al. (2016a). OpinionFinder consists of words relevant to opinions and sentiments, and TextMind is a Chinese counterpart for Empath. In summary, we obtain 3,343 words from Empath, 3,861 words from OpinionFinder, and 5,574 unique social words in total.

4 Task 1: Mining cross-cultural differences of named entities

Task definition: This task is to discover and quantify cross-cultural differences of concerns towards named entities. Specifically, the input in this task is a list of 700 named entities of interest and two monolingual social media corpora; the output is the scores for the 700 entities indicating the cross-cultural differences of the concerns towards them between two corpora. The ground truth is from the labels collected from human annotators.

4.1 Ground Truth Scores

Harris (1954) states that the meaning of words is evidenced by the contexts they occur with. Likewise, we assume that the cultural properties of an entity can be captured by the terms they always co-occur within a large social media corpus. Thus, for each of randomly selected 700 named entities, we present human annotators with two lists of 20 most co-occurred terms within Twitter and Weibo corpus respectively.

Our annotators are instructed to rate the topic-relatedness between the two word lists using one of following labels: “very different”, “different”, “hard to say”, “similar” and “very similar”. We do this for efficiency and avoiding subjectivity. As the word lists presented come from social media messages, the social and cultural elements are already embedded in their chances of occurrence. All four annotators are native Chinese speakers but have excellent command of English and lived in the US extensively, and they are trained with many selected examples to form shared understanding of the labels. The inter-annotator agreement is 0.67 by Cohen’s kappa coefficient, suggesting substantial correlation (Landis and Koch, 1977).

4.2 Baseline and Our Methods

We propose eight baseline methods for this novel task: **distribution-based** methods (BL-JS, E-BL-JS, and WN-WUP) compute cross-lingual relatedness between two lists of the words surrounding the input English and Chinese terms respectively ($\mathcal{L}_E$ and $\mathcal{L}_C$); **transformation-based** methods (LTrans and BLex) compute the vector representation in English and Chinese corpus respectively, and then train a transformation; MCCA, MCluster and Duong are three typical **bilingual word representation models** for computing general cross-lingual word similarities.

The $\mathcal{L}_E$ and $\mathcal{L}_C$ in the BL-JS and WN-WUP methods are the same as the lists that annotators judge. **BL-JS** (*Bilingual Lexicon Jaccard Similarity*) uses the bilingual lexicon to translate $\mathcal{L}_E$ to a Chinese word list $\mathcal{L}_E^*$ as a medium, and then calculates the Jaccard Similarity between $\mathcal{L}_E^*$ and $\mathcal{L}_C$ as J_{EC}. Similarly, we compute J_{CE}. Finally, we regard $(J_{EC} + J_{CE})/2$ as the score of this named entity. **E-BL-JS** (*Embedding-based Jaccard Similarity*) differs from BL-JS in that it instead compares the two lists of words gathered from the rankings of word embedding similarities between the name of entities and all English words and Chinese words respectively. **WN-WUP** (*WordNet Wu-Palmer Similarity*) uses Open Multilingual Wordnet (Wang and Bond, 2013) to compute the average similarities over all English-Chinese word pairs constructed from the $\mathcal{L}_E$ and $\mathcal{L}_C$.

We follow the steps of Mikolov et al. (2013a) to train a linear transformation (**LTrans**) matrix between *EnVec* and *CnVec*, using 3,000 translation pairs with maximum confidences in the bilingual lexicon. Given a named entity, this solution simply calculates the cosine similarity between the vector of its English name and the *transformed* vector of its Chinese name. **BLex** (*Bilingual Lexicon Space*) is similar to our *SocVec* but it does not use any social word vocabularies but uses bilingual lexicon entries as pivots instead.

MCCA (Ammar et al., 2016) takes two trained monolingual word embeddings with a bilingual lexicon as input, and develop a bilingual word em-

Entity	Twitter topics	Weibo topics
Maldives	coup, president Nasheed quit, political crisis	holiday, travel, honeymoon, paradise, beach
Nagoya	tour, concert, travel, attractive, Osaka	Mayor Takashi Kawamura, Nanjing Massacre, denial of history
Quebec	Conservative Party, Liberal Party, politicians, prime minister, power failure	travel, autumn, maples, study abroad, immigration, independence
Philippines	gunman attack, police, quake, tsunami	South China Sea, sovereignty dispute, confrontation, protest
Yao Ming	NBA, Chinese, good player, Asian	patriotism, collective values, Jeremy Lin, Liu Xiang, Chinese Law maker, gold medal superstar
USC	college football, baseball, Stanford, Alabama, win, lose	top destination for overseas education, Chinese student murdered, scholars, economics, Sino American politics

Table 1: Selected culturally different entities with summarized Twitter and Weibo's trending topics

bedding space. It is extended from the work of Faruqui and Dyer (2014), which performs slightly worse in the experiments. **MCluster** (Ammar et al., 2016) requires re-training the bilingual word embeddings from the two mono-lingual corpora with a bilingual lexicon. Similarly, **Duong** (Duong et al., 2016) retrains the embeddings from mono-lingual corpora with an EM-like training algorithm. We also use our BSL as the bilingual lexicon in these methods to investigate its effectiveness and generalizability. The dimensionality is tuned from $\{50, 100, 150, 200\}$ in all these bilingual word embedding methods.

With our constructed ***SocVec*** space, given a named entity with its English and Chinese names, we can simply compute the similarity between their *SocVec*s as its cross-cultural difference score. Our method is based on monolingual word embeddings and a BSL, and thus does not need the time-consuming re-training on the corpora.

4.3 Experimental Results

For qualitative evaluation, Table 1 shows some of the most culturally different entities mined by the SocVec method. The hot and trendy topics on Twitter and Weibo are manually summarized to help explain the cross-cultural differences. The perception of these entities diverges widely between English and Chinese social media, thus suggesting significant cross-cultural differences. Note that some cultural differences are time-specific. We believe such temporal variations of cultural differences can be valuable and beneficial for social studies as well. Investigating temporal factors of cross-cultural differences in social media can be an interesting future research topic in this task.

In Table 2, we evaluate the benchmark methods and our approach with three metrics: Spearman and Pearson, where correlation is computed between truth averaged scores (quantifying the labels from 1.0 to 5.0) and computed cultural difference scores from different methods; Mean Average Precision (MAP), which converts averaged scores as binary labels, by setting 3.0 as the threshold. The ***SocVec:opn*** considers only OpinionFinder as the ESV, while ***SocVec:all*** uses the union of Empath and OpinionFinder vocabularies[7].

Method	Spearman	Pearson	MAP
BL-JS	0.276	0.265	0.644
WN-WUP	0.335	0.349	0.677
E-BL-JS	0.221	0.210	0.571
LTrans	0.366	0.385	0.644
BLex	0.596	0.595	0.765
MCCA-BL(100d)	0.325	0.343	0.651
MCCA-BSL(150d)	0.357	0.376	0.671
MCluster-BL(100d)	0.365	0.388	0.693
MCluster-BSL(100d)	0.391	0.425	0.713
Duong-BL(100d)	0.618	0.627	0.785
Duong-BSL(100d)	0.625	0.631	0.791
SocVec:opn	0.668	0.662	**0.834**
SocVec:all	**0.676**	**0.671**	**0.834**
SocVec:noun	0.564	0.562	0.756
SocVec:verb	0.615	0.618	0.779
SocVec:adj.	0.636	0.639	0.800

Table 2: Comparison of Different Methods

Lexicon Ablation Test. To show the effectiveness of social words versus other type of words as the bridge between the two cultures, we also compare the results using sets of nouns (***SocVec:noun***), verbs (***SocVec:verb***) and adjectives (***SocVec:adj.***). All vocabularies under comparison are of similar sizes (around 5,000), indicating that the improvement of our method is significant. Results show that our *SocVec* models, and in particular, the *SocVec* model using the social words as cross-lingual media, performs the best.

[7]The following tuned parameters are used in *SocVec* methods: 5-word context window, 150 dimensions monolingual word vectors, cosine similarity as the *sim* function, and "*Top*" as the pseudo-word generator.

Similarity	Spearman	Pearson	MAP
PCorr.	0.631	0.625	0.806
L1 + M	0.666	0.656	0.824
Cos	**0.676**	0.669	**0.834**
L2 + E	**0.676**	**0.671**	**0.834**

Table 3: Different Similarity Functions

Generator	Spearman	Pearson	MAP
Max.	0.413	0.401	0.726
Avg.	0.667	0.625	0.831
W.Avg.	0.671	0.660	0.832
Top	**0.676**	**0.671**	**0.834**

Table 4: Different Pseudo-word Generators

Gg	Bi	Bd	CC	LT
18.24	16.38	17.11	17.38	9.14
TransBL	MCCA	MCluster	Duong	SV
18.13	17.29	17.47	20.92	**23.01**

(a) Chinese Slang to English

Gg	Bi	Bd	LT	TransBL
6.40	15.96	15.44	7.32	11.43
MCCA	MCluster	Duong	SV	
15.29	14.97	15.13	**17.31**	

(b) English Slang to Chinese

Table 5: ACS Sum Results of Slang Translation

Similarity Options. We also evaluate the effectiveness of four different similarity options in *SocVec*, namely, Pearson Correlation Coefficient (*PCorr.*), L1-normalized Manhattan distance (*L1+M*), Cosine Similarity (*Cos*) and L2-normalized Euclidean distance (*L2+E*). From Table 3, we conclude that among these four options, *Cos* and *L2+E* perform the best.

Pseudo-word Generators. Table 4 shows effect of using four pseudo-word generator functions, from which we can infer that "*Top*" generator function performs best for it reduces some noisy translation pairs.

5 Task 2: Finding most similar words for slang across languages

Task Description: This task is to find the most similar English words of a given Chinese slang term in terms of its slang meanings and sentiment, and vice versa. The input is a list of English/Chinese slang terms of interest and two monolingual social media corpora; the output is a list of Chinese/English word sets corresponding to each input slang term. Simply put, for each given slang term, we want to find a set of the words in a different language that are most similar to itself and thus can help people understand it across languages. We propose Average Cosine Similarity (Section 5.3) to evaluate a method's performance with the ground truth (presented below).

5.1 Ground Truth

Slang Terms. We collect the Chinese slang terms from an online Chinese slang glossary[8] consisting of 200 popular slang terms with English explanations. For English, we resort to a slang word list from OnlineSlangDictionary[9] with explanations and downsample the list to 200 terms.

Truth Sets. For each Chinese slang term, its truth set is a set of words extracted from its English explanation. For example, we construct the truth set of the Chinese slang term "二百五" by manually extracting significant words about its slang meanings (bold) in the glossary:

二百五: A ***foolish*** person who is lacking in sense but still ***stubborn***, ***rude***, and ***impetuous***.

Similarly, for each English slang term, its Chinese word sets are the translation of the words hand picked from its English explanation.

5.2 Baseline and Our Methods

We propose two types of baseline methods for this task. The first is based on well-known *online translators*, namely Google (Gg), Bing (Bi) and Baidu (Bd). Note that experiments using them are done in August, 2017. Another baseline method for Chinese is CC-CEDICT[10] (CC), an online public Chinese-English dictionary, which is constantly updated for popular slang terms.

Considering situations where many slang terms have literal meanings, it may be unfair to retrieve target terms from such machine translators by solely inputing slang terms without specific contexts. Thus, we utilize example sentences of their slang meanings from some websites (mainly from Urban Dictionary[11]). The following example shows how we obtain the target translation terms for the slang word "fruitcake" (an insane person):

Input sentence: *Oh man, you don't want to date that girl. She's always drunk and yelling. She is a total* ***fruitcake***.[12]

[8] https://www.chinasmack.com/glossary

[9] http://onlineslangdictionary.com/word-list/

[10] https://cc-cedict.org/wiki/

[11] http://www.urbandictionary.com/

[12] http://www.englishbaby.com/lessons/4349/slang/fruitcake

Slang	Explanation	Google	Bing	Baidu	Ours
浮云	something as ephemeral and unimportant as “passing clouds”	clouds	nothing	floating clouds	nothingness, illusion
水军	“water army”, people paid to slander competitors on the Internet and to help shape public opinion	Water army	Navy	Navy	propaganda, complicit, fraudulent
floozy	a woman with a reputation for promiscuity	N/A	劣根性 (depravity)	荡妇(slut)	骚货(slut),妖精(promiscuous)
fruitcake	a crazy person, someone who is completely insane	水果蛋糕 (fruit cake)	水果蛋糕 (fruit cake)	水果蛋糕 (fruit cake)	怪诞(bizarre),厌烦(annoying)

Table 6: Bidirectional Slang Translation Examples Produced by SocVec

Google Translation: 哦, 男人, 你不想约会那个女孩。她总是喝醉了, 大喊大叫。她是一个水果蛋糕。

Another lines of baseline methods is scoring-based. The basic idea is to score all words in our bilingual lexicon and consider the top K words as the target terms. Given a source term to be translated, the Linear Transform (LT), MCCA, MCluster and Duong methods score the candidate target terms by computing cosine similarities in their constructed bilingual vector space (with the tuned best settings in previous evaluation). A more sophisticated baseline (TransBL) leverages the bilingual lexicon: for each candidate target term w in the target language, we first obtain its translations T_w back into the source language and then calculate the average word similarities between the source term and the translations T_w as w's score.

Our *SocVec-based method* (**SV**) is also scoring-based. It simply calculates the cosine similarities between the source term and each candidate target term within *SocVec* space as their scores.

5.3 Experimental Results

To quantitatively evaluate our methods, we need to measure similarities between a produced word set and the ground truth set. Exact-matching Jaccard similarity is too strict to capture valuable relatedness between two word sets. We argue that average cosine similarity (ACS) between two sets of word vectors is a better metric for evaluating the similarity between two word sets.

$$\text{ACS}(A,B) = \frac{1}{|A||B|}\sum_{i=1}^{|A|}\sum_{j=1}^{|B|}\frac{\mathbf{A_i}\cdot\mathbf{B_j}}{\|\mathbf{A_i}\|\|\mathbf{B_j}\|}$$

The above equation illustrates such computation, where A and B are the two word sets: A is the truth set and B is a similar list produced by each method. In the previous case of “二百五” (Section 5.1), A is {foolish, stubborn, rude, impetuous} while B can be {imbecile, brainless, scumbag, imposter}. $\mathbf{A_i}$ and $\mathbf{B_j}$ denote the word vector of the i^{th} word in A and j^{th} word in B respectively. The embeddings used in ACS computations are pre-trained *GloVe* word vectors[13] and thus the computation is fair among different methods.

Chinese Slang	English Slang	Explanation
萌	adorbz, adorb, adorbs, tweeny, attractiveee	cute, adorable
二百五	shithead, stupidit, douchbag	A foolish person
鸭梨	antsy, stressy, fidgety, grouchy, badmood	stress, pressure, burden

Table 7: Slang-to-Slang Translation Examples

Experimental results of Chinese and English slang translation in terms of the sum of *ACS* over 200 terms are shown in Table 5. The performance of online translators for slang typically depends on human-set rules and supervised learning on well-annotated parallel corpora, which are rare and costly, especially for social media where slang emerges the most. This is probably the reason why they do not perform well. The Linear Transformation (LT) model is trained on highly confident translation pairs in the bilingual lexicon, which lacks OOV slang terms and social contexts around them. The TransBL method is competitive because its similarity computations are within monolingual semantic spaces and it makes great use of the bilingual lexicon, but it loses the information from the related words that are not in the bilingual lexicon. Our method (SV) outperforms baselines by directly using the distances in the *SocVec* space, which proves that the *SocVec* well captures the cross-cultural similarities between terms.

To qualitatively evaluate our model, in Table 6, we present several examples of our translations for Chinese and English slang terms as well as their

[13] https://nlp.stanford.edu/projects/glove/

explanations from the glossary. Our results are highly correlated with these explanations and capture their significant semantics, whereas most online translators just offer literal translations, even within obviously slang contexts. We take a step further to directly translate Chinese slang terms to English slang terms by filtering out ordinary (non-slang) words in the original target term lists, with examples shown in Table 7.

6 Related Work

Although social media messages have been essential resources for research in computational social science, most works based on them only focus on a single culture and language (Petrovic et al., 2010; Paul and Dredze, 2011; Rosenthal and McKeown, 2015; Wang and Yang, 2015; Zhang et al., 2015; Lin et al., 2017). Cross-cultural studies have been conducted on the basis of a questionnaire-based approach for many years. There are only a few of such studies using NLP techniques.

Nakasaki et al. (2009) present a framework to visualize the cross-cultural differences in concerns in multilingual blogs collected with a topic keyword. Elahi and Monachesi (2012) show that cross-cultural analysis through language in social media data is effective, especially using emotion terms as culture features, but the work is restricted in monolingual analysis and a single domain (love and relationship). Garimella et al. (2016a) investigate the cross-cultural differences in word usages between Australian and American English through their proposed "socio-linguistic features" (similar to our social words) in a supervised way. With the data of social network structures and user interactions, Garimella et al. (2016b) study how to quantify the controversy of topics within a culture and language. Gutiérrez et al. (2016) propose an approach to detect differences of word usage in the cross-lingual topics of multilingual topic modeling results. To the best of our knowledge, our work for Task 1 is among the first to mine and quantify the cross-cultural differences in concerns about named entities across different languages.

Existing research on slang mainly focuses on automatic discovering of slang terms (Elsahar and Elbeltagy, 2014) and normalization of noisy texts (Han et al., 2012) as well as slang formation. Ni and Wang (2017) are among the first to propose an automatic supervised framework to monolingually explain slang terms using external resources. However, research on automatic translation or cross-lingually explanation for slang terms is missing from the literature. Our work in Task 2 fills the gap by computing cross-cultural similarities with our bilingual word representations (*SocVec*) in an unsupervised way. We believe this application is useful in machine translation for social media (Ling et al., 2013).

Many existing cross-lingual word embedding models rely on expensive parallel corpora with word or sentence alignments (Klementiev et al., 2012; Kočiský et al., 2014). These works often aim to improve the performance on monolingual tasks and cross-lingual model transfer for document classification, which does not require cross-cultural signals. We position our work in a broader context of "monolingual mapping" based cross-lingual word embedding models in the survey of Ruder et al. (2017). The SocVec uses only lexicon resource and maps monolingual vector spaces into a common high-dimensional third space by incorporating social words as pivot, where orthogonality is approximated by setting clear meaning to each dimension of the *SocVec* space.

7 Conclusion

We present the SocVec method to compute cross-cultural differences and similarities, and evaluate it on two novel tasks about mining cross-cultural differences in named entities and computing cross-cultural similarities in slang terms. Through extensive experiments, we demonstrate that the proposed lightweight yet effective method outperforms a number of baselines, and can be useful in translation applications and cross-cultural studies in computational social science. Future directions include: 1) mining cross-cultural differences in general concepts other than names and slang, 2) merging the mined knowledge into existing knowledge bases, and 3) applying the SocVec in downstream tasks like machine translation.[14]

Acknowledgment

Kenny Zhu is the contact author and was supported by NSFC grants 91646205 and 61373031. Seung-won Hwang was supported by Microsoft Research Asia. Thanks to the anonymous reviewers and Hanyuan Shi for their valuable feedback.

[14]We will make our code and data available at `https://github.com/adapt-sjtu/socvec`.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

José Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proc. of SemEval@ACL*.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proc. of COLING 2010: Demonstrations*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proc. of EMNLP*.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proc. of HLT-EMNLP*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proc. of EMNLP*.

Mohammad Fazleh Elahi and Paola Monachesi. 2012. An examination of cross-cultural similarities and differences from social media data with respect to language use. In *Proc. of LREC*.

Hady Elsahar and Samhaa R Elbeltagy. 2014. A fully automated approach for arabic slang lexicon extraction from microblogs. In *Proc. of CICLing*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*.

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proc. of CHI*.

King-wa Fu, Chung-hong Chan, and Michael Chau. 2013. Assessing censorship on microblogs in china: Discriminatory keyword analysis and the real-name registration policy. *IEEE Internet Computing*, 17(3):42–50.

Rui Gao, Bibo Hao, He Li, Yusong Gao, and Tingshao Zhu. 2013. Developing simplified chinese psychological linguistic analysis dictionary for microblog. In *Proceedings of International Conference on Brain and Health Informatics*. Springer.

Elisabeth Gareis and Richard Wilkins. 2011. Love expression in the united states and germany. *International Journal of Intercultural Relations*, 35(3):307–319.

Aparna Garimella, Rada Mihalcea, and James W. Pennebaker. 2016a. Identifying cross-cultural differences in word usage. In *Proc. of COLING*.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2016b. Quantifying controversy in social media. In *Proc. of WSDM*.

E. Dario Gutiérrez, Ekaterina Shutova, Patricia Lichtenstein, Gerard de Melo, and Luca Gilardi. 2016. Detecting cross-cultural differences using a multilingual topic model. *TACL*, 4:47–60.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proc. of EMNLP-CoNLL*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Shinobu Kitayama, Hazel Rose Markus, and Masaru Kurokawa. 2000. Culture, emotion, and well-being: Good feelings in japan and the united states. *Cognition & Emotion*, 14(1):93–124.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proc. of COLING*.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *Proc. of ACL*.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33 1:159–74.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. of ICML*.

Bill Y. Lin, Frank F. Xu, Zhiyi Luo, and Kenny Q. Zhu. 2017. Multi-channel bilstm-crf model for emerging named entity recognition in social media. In *Proc. of W-NUT@EMNLP*.

Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Proc. of ACL*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proc. of ACL (System Demonstrations)*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Hiroyuki Nakasaki, Mariko Kawaba, Sayuri Yamazaki, Takehito Utsuro, and Tomohiro Fukuhara. 2009. Visualizing cross-lingual/cross-cultural differences in concerns in multilingual blogs. In *Proc. of ICWSM*.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard english words and phrases. In *Proc. of IJCNLP*.

Michael J. Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. In *Proc. of ICWSM*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proc. of HLT-NAACL*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of ACL*.

Sara Rosenthal and Kathy McKeown. 2015. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *Proc. of SIGDIAL*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *arXiv preprint arXiv:1706.04902*.

Chandar A P Sarath, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proc. in NIPS*.

Yla R. Tausczik and James W. Pennebaker. 2009. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proc. of ACL*.

Shan Wang and Francis Bond. 2013. Building the chinese open wordnet (cow): Starting from core synsets. In *Proceedings of the 11th Workshop on Asian Language Resources, a Workshop at IJCNLP*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proc. of EMNLP*.

Boliang Zhang, Hongzhao Huang, Xiaoman Pan, Sujian Li, Chin-Yew Lin, Heng Ji, Kevin Knight, Zhen Wen, Yizhou Sun, Jiawei Han, and Bülent Yener. 2015. Context-aware entity morph decoding. In *Proc. of ACL*.

Classification of Moral Foundations in Microblog Political Discourse

Kristen Johnson and Dan Goldwasser
Department of Computer Science
Purdue University, West Lafayette, IN 47907
{john1187, dgoldwas}@purdue.edu

Abstract

Previous works in computer science, as well as political and social science, have shown correlation in text between political ideologies and the moral foundations expressed within that text. Additional work has shown that policy frames, which are used by politicians to bias the public towards their stance on an issue, are also correlated with political ideology. Based on these associations, this work takes a first step towards modeling both the language and how politicians frame issues on Twitter, in order to predict the moral foundations that are used by politicians to express their stances on issues. The contributions of this work includes a dataset annotated for the moral foundations, annotation guidelines, and probabilistic graphical models which show the usefulness of jointly modeling abstract political slogans, as opposed to the unigrams of previous works, with policy frames for the prediction of the morality underlying political tweets.

1 Introduction

Social media microblogging platforms, specifically Twitter, have become highly influential and relevant to current political events. Such platforms allow politicians to communicate with the public as events are unfolding and shape public discourse on various issues. Furthermore, politicians are able to express their stances on issues and by selectively using certain political slogans, reveal their underlying political ideologies and moral views on an issue. Previous works in political and social science have shown a correlation between political ideology, stances on political issues, and the moral convictions used to justify these stances (Graham et al., 2009). For example, Figure 1 presents a tweet, by a prominent member of the U.S. Congress, which expresses concern

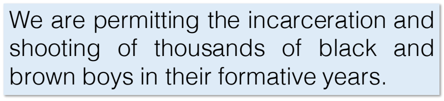

We are permitting the incarceration and shooting of thousands of black and brown boys in their formative years.

Figure 1: Example Tweet Highlighting Classification Difficulty.

about the fate of young individuals (i.e., *incarceration, shooting*), specifically for vulnerable members of minority groups. The Moral Foundations Theory (MFT) (Haidt and Joseph, 2004; Haidt and Graham, 2007) provides a theoretical framework for explaining these nuanced distinctions. The theory suggests that there are five basic moral values which underlie human moral perspectives, emerging from evolutionary, social, and cultural origins. These are referred to as the moral foundations (MF) and include *Care/Harm, Fairness/Cheating, Loyalty/Betrayal, Authority/Subversion*, and *Purity/Degradation* (Table 1 provides a more detailed explanation). The above example reflects the moral foundations that shape the author's perspective on the issue: *Harm* and *Cheating*.

Traditionally, analyzing text based on the MFT has relied on the use of a lexical resource, the Moral Foundations Dictionary (MFD) (Haidt and Graham, 2007; Graham et al., 2009). The MFD, similar to LIWC (Pennebaker et al., 2001; Tausczik and Pennebaker, 2010), associates a list of related words with each one of the moral foundations. Therefore, analyzing text equates to counting the number of occurrences of words in the text which also match the words in the MFD. Given the highly abstract and generalized nature of the moral foundations, this approach often falls short of dealing with the highly ambiguous text politicians use to express their perspectives on specific issues. The following tweet, by another prominent member of the U.S. Congress, reflects the author's use of both the *Harm* and *Cheating* moral foundations.

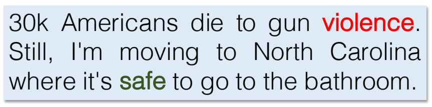

30k Americans die to gun violence. Still, I'm moving to North Carolina where it's safe to go to the bathroom.

Figure 2: Example Tweet Highlighting Classification Difficulty.

While the first foundation (*Harm*) can be directly identified using a word match to the MFD (as shown in red), the second foundation requires first identifying the sarcastic expression referring to LGBTQ rights and then using extensive world knowledge to determine the appropriate moral foundation. [1] Relying on a match of *safe* to the MFD would indicate the *Care* MF is being used instead of the *Cheating* foundation.

In this paper, we aim to solve this challenge by suggesting a data-driven approach to moral foundation identification in tweets. Previous work (Garten et al., 2016) has looked at classification-based approaches over tweets specifically related to Hurricane Sandy, augmenting the textual content with background knowledge using entity linking (Lin et al., 2017). Different from this and similar works, we look at the tweets of U.S. politicians over a long period of time, discussing a large number of events, and touching on several different political issues. Our approach is guided by the intuition that the abstract moral foundations will manifest differently in text, depending on the specific characteristics of the events discussed in the tweet. As a result, it is necessary to correctly model the relevant contextualizing information.

Specifically, we are interested in exploring how political ideology, language, and framing interact to represent morality on Twitter. We examine the interplay of political slogans (for example *"repeal and replace"* when referring to the Affordable Care Act), and policy framing techniques (Boydstun et al., 2014; Johnson et al., 2017) as features for predicting the underlying moral values which are expressed in politicians' tweets. Additionally, we identify high-level themes characterizing the main point of the tweet, which allows the model to identify the author's perspective on specific issues and generalize over the specific wording used (for example, if the tweet mentions `Religion` or `Political Maneuvering`).

This information is incorporated into global probabilistic models using Probabilistic Soft Logic (PSL), a graphical probabilistic modeling framework (Bach et al., 2013). PSL specifies high level rules over a relational representation of these features, which are compiled into a graphical model called a hinge-loss Markov random field that is used to make the final prediction. Our experiments show the importance of modeling contextualizing information, leading to significant improvements over dictionary driven approaches and purely lexical methods.

In summary, this paper makes the following contributions: (1) This work is among the first to explore jointly modeling language and political framing techniques for the classification of moral foundations used in the tweets of U.S. politicians on Twitter. (2) We provide a description of our annotation guidelines and an annotated dataset of 2,050 tweets.[2] (3) We suggest computational models which easily adapt to new policy issues, for the classification of the moral foundations present in tweets.

2 Related Works

In this paper, we explore how political ideology, language, framing, and morality interact on Twitter. Previous works have studied framing in longer texts, such as congressional speeches and news (Fulgoni et al., 2016; Tsur et al., 2015; Card et al., 2015; Baumer et al., 2015), as well as issue-independent framing on Twitter (Johnson and Goldwasser, 2016; Johnson et al., 2017). Ideology measurement (Iyyer et al., 2014; Bamman and Smith, 2015; Sim et al., 2013; Djemili et al., 2014), political sentiment analysis (Pla and Hurtado, 2014; Bakliwal et al., 2013), and polls based on Twitter political sentiment (Bermingham and Smeaton, 2011; O'Connor et al., 2010; Tumasjan et al., 2010) are also related to the study of framing. The association between Twitter and framing in molding public opinion of events and issues (Burch et al., 2015; Harlow and Johnson, 2011; Meraz and Papacharissi, 2013; Jang and

[1]The tweet refers to legislation proposed in 2016 concerning transgender bathroom access restrictions.

[2]The data will be available at `http://purduenlp.cs.purdue.edu/projects/twittermorals`.

MORAL FOUNDATION AND BRIEF DESCRIPTION
1. Care/Harm: Care for others, generosity, compassion, ability to feel pain of others, sensitivity to suffering of others, prohibiting actions that harm others.
2. Fairness/Cheating: Fairness, justice, reciprocity, reciprocal altruism, rights, autonomy, equality, proportionality, prohibiting cheating.
3. Loyalty/Betrayal: Group affiliation and solidarity, virtues of patriotism, self-sacrifice for the group, prohibiting betrayal of one's group.
4. Authority/Subversion: Fulfilling social roles, submitting to authority, respect for social hierarchy/traditions, leadership, prohibiting rebellion against authority.
5. Purity/Degradation: Associations with the sacred and holy, disgust, contamination, religious notions which guide how to live, prohibiting violating the sacred.
6. Non-moral: Does not fall under any other foundations.

Table 1: Brief Descriptions of Moral Foundations.

Hart, 2015) has also been studied.

The connection between morality and political ideology has been explored in the fields of psychology and sociology (Graham et al., 2009, 2012). Moral foundations were also used to inform downstream tasks, by using the MFD to identify the moral foundations in partisan news sources (Fulgoni et al., 2016), or to construct features for other downstream tasks (Volkova et al., 2017). Several recent works have looked into using data-driven methods that go beyond the MFD to study tweets related to Hurricane Sandy (Garten et al., 2016; Lin et al., 2017).

3 Data Annotation

The Moral Foundations Theory (Haidt and Graham, 2007) was proposed by sociologists and psychologists as a way to understand how morality develops, as well as its similarities and differences across cultures. The theory consists of the five moral foundations shown in Table 1. The goal of this work is to classify the tweets of the Congressional Tweets Dataset (Johnson et al., 2017) with the moral foundation implied in the tweet.

We first attempted to use Amazon Mechanical Turk for annotation, but found that most Mechanical Turkers would choose the *Care/Harm* or *Fairness/Cheating* label a majority of the time. Additionally, annotators preferred choosing first the foundation branch (i.e., *Care/Harm*) and then its sentiment (positive or negative) as opposed to the choice of each foundation separately, i.e., given the choice between *Harm* or *Care/Harm and Negative*, annotators preferred the latter. Based on these observations, two annotators, one liberal and one conservative (self-reported), manually annotated a subset of tweets. This subset had an inter-annotator agreement of 67.2% using Cohen's Kappa coefficient. The annotators then discussed and agreed on general guidelines which were used to label the remaining tweets of the dataset. The resulting dataset has an inter-annotator agreement of 79.2% using Cohen's Kappa statistic. The overall distribution, distributions by political party, and distributions per issue of the labeled dataset are presented in Table 2. Table 3 lists the frames that most frequently co-occured with each MF. As expected, frames concerning Morality and Sympathy are highly correlated with the *Purity* foundation, while *Subversion* is highly correlated with the Legal and Political frames.

Labeling tweets presents several challenges. First, tweets are short and thus lack the context often necessary for choosing a moral viewpoint. Tweets are often ambiguous, e.g., a tweet may express care for people who are being harmed by a policy. Another major challenge was overcoming the political bias of the annotator. For example, if a tweet discusses opposing Planned Parenthood because it provides abortion services, the liberal annotator typically viewed this as *Harm* (i.e., hurting women by taking away services from them), while the conservative annotator tended to view this as *Purity* (i.e., all life is sacred and should be protected). To overcome this bias, annotators were given the political party of the politician who wrote the tweets and instructed to choose the moral foundation *from the politician's perspective*. To further simplify the annotation process, all tweets belonging to one political party were labeled together, i.e., all Republican tweets were labeled and then all Democrat tweets were labeled. Finally, tweets present a compound problem, often expressing two thoughts which can further be contradictory. This results in one tweet having multiple moral foundations. Annotators chose a primary moral foundation whenever possible, but were allowed a secondary foundation if the tweet presented two differing thoughts.

Several recurring themes continued to appear throughout the dataset including "thoughts and prayers" for victims of gun shooting events or rhetoric against the opposing political party. The annotators agreed to use the following moral foundation labels for these repeating topics as follows: (1) The *Purity* label is used for tweets that relate to

Morals	OVERALL	PARTY		ISSUE					
		REP	DEM	ABO	ACA	GUN	IMM	LGBTQ	TER
Care	524	156	368	37	123	215	33	34	113
Harm	355	151	204	26	64	141	19	34	101
Fairness	268	55	213	41	81	19	11	86	39
Cheating	82	37	45	14	27	11	10	9	13
Loyalty	303	63	240	28	29	128	36	38	58
Betrayal	53	25	28	10	4	9	6	3	22
Authority	192	62	130	24	44	50	38	10	34
Subversion	419	251	168	34	169	75	73	25	60
Purity	174	86	88	24	3	102	5	24	41
Degradation	66	34	32	5	0	31	0	4	31
Non-moral	334	198	136	17	143	28	47	7	96

Table 2: Distributions of Moral Foundations. Overall is across the entire dataset. Party is the Republican (REP) or Democrat (DEM) specific distributions. Issue lists the six issue-specific distributions (Abortion, ACA, Guns, Immigration, LGBTQ, Terrorism).

MORAL FOUNDATION AND CO-OCCURING FRAMES
Care: Capacity & Resources, Security & Defense, Health & Safety, Quality of Life, Public Sentiment, External Regulation & Reputation
Harm: Economic, Crime & Punishment
Fairness: Fairness & Equality
Loyalty: Cultural Identity
Subversion: Legality, Constitutionality, & Jurisdiction, Political Factors & Implications, Policy Description, Prescription, & Evaluation
Purity: Morality & Ethics, Personal Sympathy & Support
Non-moral: Factual, (Self) Promotion

Table 3: Foundations and Co-occuring Frames. *Cheating, Betrayal, Authority,* and *Degradation* did not co-occur frequently with any frames.

prayers or the fight against ISIL/ISIS. (2) *Loyalty* is for tweets that discuss "stand(ing) with" others, American values, troops, or allies, or reference a demographic that the politician belongs to, e.g. if the politician tweeting is a woman and she discusses an issue in terms of its effects on women. (3) At the time the dataset was collected, the President was Barack Obama and the Republican party controlled Congress. Therefore, any tweets specifically attacking Obama or Republicans (the controlling party) were labeled as *Subversion*. (4) Tweets discussing health or welfare were labeled as *Care*. (5) Tweets which discussed limiting or restricting laws or rights were labeled as *Cheating*. (6) Sarcastic attacks, typically against the opposing political party, were labeled as *Degradation*.

4 Feature Extraction for PSL Models

For this work, we designed extraction models and PSL models that were capable of adapting to the dynamic language used on Twitter and predicting the moral foundation of a given tweet. Our approach uses weakly supervised extraction models, whose only initial supervision is a set of unigrams and the political party of the tweet's author, to extract features for each PSL model. These features are represented as PSL predicates and combined into the probabilistic rules of each model, as shown in Table 4, which successively build upon the rules of the previous model.

4.1 Global Modeling Using PSL

PSL is a declarative modeling language which can be used to specify weighted, first-order logic rules that are compiled into a hinge-loss Markov random field. This field defines a probability distribution over possible continuous value assignments to the random variables of the model (Bach et al., 2015) and is represented as:

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z} \exp\left(-\sum_{r=1}^{M} \lambda_r \phi_r(\mathbf{Y}\,,\mathbf{X})\right)$$

where Z is a normalization constant, λ is the weight vector, and

$$\phi_r(\mathbf{Y}, \mathbf{X}) = (\max\{l_r(\mathbf{Y}, \mathbf{X}), 0\})^{\rho_r}$$

is the hinge-loss potential specified by a linear function l_r. The exponent $\rho_r \in 1, 2$ is optional. Each potential represents the instantiation of a rule, which takes the following form:

$$\lambda_1 : P_1(x) \wedge P_2(x, y) \rightarrow P_3(y)$$
$$\lambda_2 : P_1(x) \wedge P_4(x, y) \rightarrow \neg P_3(y)$$

P_1, P_2, P_3, and P_4 are predicates (e.g., party, issue, and frame) and x, y are variables. Each rule has a weight λ to reflect its importance to the model. Using concrete constants *a, b* (e.g., tweets) which instantiate the variables x, y, model atoms are mapped to continuous [0,1] assignments.

MOD.	INFORMATION USED	EXAMPLE OF PSL RULE
M1	UNIGRAMS (MFD OR AR)	UNIGRAM$_M$(T, U) $\rightarrow$ MORAL(T, M)
M2	M1 + PARTY	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\rightarrow$ MORAL(T, M)
M3	M2 + ISSUE	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ ISSUE(T, I) $\rightarrow$ MORAL(T, M)
M4	M3 + PHRASE	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ PHRASE(T, PH) $\rightarrow$ MORAL(T, M)
M5	M4 + FRAME	UNIGRAM$_M$(T, U) $\wedge$ PHRASE(T, PH) $\wedge$ FRAME(T, F) $\rightarrow$ MORAL(T, M)
M6	M5 + PARTY-BIGRAMS	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ BIGRAM$_P$(T, B) $\rightarrow$ MORAL(T, M)
M7	M6 + PARTY-ISSUE-BIGRAMS	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ BIGRAM$_{PI}$(T, B) $\rightarrow$ MORAL(T, M)
M8	M7 + PHRASE	BIGRAM$_{PI}$(T, B) $\wedge$ PHRASE(T, PH) $\rightarrow$ MORAL(T, M)
M9	M8 + FRAME	BIGRAM$_{PI}$(T, B) $\wedge$ FRAME(T, F) $\rightarrow$ MORAL(T, M)
M10	M9 + PARTY-TRIGRAMS	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ TRIGRAM$_P$(T, TG) $\rightarrow$ MORAL(T, M)
M11	M10 + PARTY-ISSUE-TRIGRAMS	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ TRIGRAM$_{PI}$(T, TG) $\rightarrow$ MORAL(T, M)
M12	M11 + PHRASE	TRIGRAM$_{PI}$(T, TG) $\wedge$ PHRASE(T, PH) $\rightarrow$ MORAL(T, M)
M13	M12 + FRAME	TRIGRAM$_{PI}$(T, TG) $\wedge$ FRAME(T, F) $\rightarrow$ MORAL(T, M)

Table 4: Examples of PSL Moral Model Rules Using Gold Standard Frames. For these rules, the FRAME predicate is initialized with the known frame labels of the tweet. Each model builds successively on the rules of the previous model.

M2: UNIGRAMS + PARTY	UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ FRAME(T, F) $\rightarrow$ MORAL(T, M) UNIGRAM$_M$(T, U) $\wedge$ PARTY(T, P) $\wedge$ MORAL(T, M) $\rightarrow$ FRAME(T, F)
M13: ALL FEATURES	TRIGRAM$_{PI}$(T, TG) $\wedge$ PHRASE(T, PH) $\wedge$ FRAME(T, F) $\rightarrow$ MORAL(T, M) TRIGRAM$_{PI}$(T, TG) $\wedge$ UNIGRAM$_M$(T, U) $\wedge$ MORAL(T, M) $\rightarrow$ FRAME(T, F)

Table 5: Examples of PSL Joint Moral and Frame Model Rules. For these models, the FRAME predicate is *not initialized with known values*, but is predicted jointly with the MORAL predicate.

4.2 Feature Extraction Models

For each aspect of the tweets that composes the PSL models, scripts are written to first identify and then extract the correct information from the tweets. Once extracted, this information is formatted into PSL predicate notation and input to the PSL models. Table 4 presents the information that composes each PSL model, as well as an example of how rules in the PSL model are constructed.

Language: Works studying the Moral Foundations Theory typically assign a foundation to a body of text based on a majority match of the words in the text to the Moral Foundations Dictionary (MFD), a predefined list of unigrams associated with each foundation. These unigrams capture the conceptual idea behind each foundation. Annotators noted, however, that when choosing a foundation they typically used a small phrase or the entire tweet, not a single unigram. Based on this, we compiled all of the annotators' phrases per foundation into a unique set to create a new list of unigrams for each foundation. These unigrams are referred to as "Annotator's Rationale (AR)" throughout the remainder of this paper. The PSL predicate UNIGRAM$_M$(T, U) is used to input any unigram U from tweet T that matches the M list of unigrams (either from the MFD or AR lists) into the PSL models. An example of a rule using this predicate is shown in the first row of Table 4.

During annotation, we observed that often a tweet has only one match to a unigram, if any, and therefore a majority count approach may fail. Further, as shown in Figure 2, many tweets have one unigram that matches one foundation and another unigram that matches a different foundation. In such cases, the correct foundation cannot be determined from unigram counts alone. Based on these observations and the annotators' preference for using phrases, we incorporate the most frequent bigrams and trigrams for each political party (BIGRAM$_P$(T, B) and TRIGRAM$_P$(T, TG)) and for each party on each issue (BIGRAM$_{PI}$(T, B) and TRIGRAM$_{PI}$(T, TG)). These top 20 bigrams and trigrams contribute to a more accurate prediction than unigrams alone (Johnson et al., 2017).

Ideological Information: Previous works have shown a strong correlation between ideology and the moral foundations (Haidt and Graham, 2007), as well as between ideology and policy issues (Boydstun et al., 2014). Annotators were able to agree on labels when instructed to label from the ideological point of view of the tweet's author, even if it opposed their own views. Based on these positive correlations, we incorporate both the issue of the tweet (ISSUE(T, I)) and the political party of the author of the tweet (PARTY(T, P)) into the PSL models. Examples of how this information is represented in the PSL models are shown in rows two and three of Table 4.

Abstract Phrases: As described previously, annotators reported that phrases were more useful than unigrams in determining the moral foundation of the tweet. Due to the dynamic nature of language and trending issues on Twitter, it is impracticable to construct a list of all possible phrases one can expect to appear in tweets. However, because politicians are known for sticking to certain talking points, these phrases can be *abstracted* into higher-level phrases that are more stable and thus easier to identify and extract.

For example, a tweet discussing "President Obama's signing a bill" has two possible concrete phrases: *President Obama's signing* and *signing a bill*. Each phrase falls under two possible abstractions: political maneuvering (Obama's actions) and mentions legislation (signing of a bill). In this paper we use the following high-level abstractions: `legislation or voting`, `rights and equality`, `emotion`, `sources of danger or harm`, `positive benefits or effects`, `solidarity`, `political maneuvering`, `protection and prevention`, `American values or traditions`, `religion`, and `promotion`. For example, if a tweet mentions "civil rights" or "equal pay", then these phrases indicate that the `rights and equality` abstraction is being used to express morality. Some of these abstractions correlate with the corresponding MF or frame, e.g., the `religion` abstraction is highly correlated with the *Purity* foundation and `political maneuvering` is correlated with the Political Factors & Implications Frame.

To match phrases in tweets to these abstractions, we use the embedding-based model of Lee et al. (2017). This phrase similarity model was trained on the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) and incorporates a Convolutional Neural Network (CNN) to capture sentence structures. This model generates the embeddings of our abstract phrases and computes the cosine similarities between phrases and tweets as the scores. The input tweets and phrases are represented as the average word embeddings in the input layer, which are then projected into a convolutional layer, a max-pooling layer, and finally two fully-connected layers. The embeddings are thus represented in the final layer. The learning objective of this model is:

$$\begin{aligned}\min_{W_c,W_w}\Big(\sum_{<x_1,x_2>\in X} & max(0,\delta - cos(g(x_1),g(x_2)) \\ & + cos(g(x_1),g(t_1))) \\ & + max(0,\delta - cos(g(x_1),g(x_2))) \\ & + cos(g(x_2),g(t_2))\Big) \\ & + \lambda_c||W_c||^2 + \lambda_w||W_{init} - W_w||^2,\end{aligned}$$

where X is all the positive input pairs, δ is the margin, $g(\cdot)$ represents the network, λ_c and λ_w are the weights for L2-regularization, W_c is the network parameters, W_w is the word embeddings, W_{init} is the initial word embeddings, and t_1 and t_2 are negative examples that are randomly selected.

All tweet-phrase pairs with a cosine similarity over a given threshold are used as input to the PSL model via the predicate PHRASE(T, PH), which indicates that tweet T contains a phrase that is similar to an abstracted phrase (PH). [3] Rows four, eight, and twelve of Table 4 show examples of the phrase rules as used in our modeling procedure.

Nuanced Framing: Framing is a political strategy in which politicians carefully word their statements in order to bias public opinion towards their stance on an issue. This technique is a fine-grained view of how issues are expressed. Frames are associated with issue, political party, and ideologies. For example, if a politician emphasizes the economic burden a new bill would place on the public, then they are using the *Economic* frame. Different from this, if they emphasize how people's lives will improve because of this bill, then they are using the *Quality of Life* frame.

In this work, we explore frames in two settings: (1) where the actual frames of tweets are known and used to predict the moral foundation of the tweets and (2) when the frames are unknown and predicted jointly with the moral foundations. Using the Congressional Tweets Dataset as the true labels for 17 policy frames, this information is input to the PSL models using the FRAME(T, F) predicate as shown in Table 4. Conversely, the

[3] A threshold score of 0.45 provided the most accurate matches while minimizing noise.

same predicate can be used as a joint prediction target predicate, with no initialization, as shown in Table 5.

5 Experimental Results

In this section, we present an analysis of the results of our modeling approach. Table 6 summarizes our overall results and compares the traditional BoW SVM classifier[4] to several variations of our model. We provide an in-depth analysis, broken down by the different types of moral foundations, in Tables 7 and 8.

We also study the relationship between moral foundations, policy framing, and political ideology. Table 9 describes the results of a joint model for predicting moral foundations and policy frames. Finally, in Section 6 we discuss how moral foundations can be used for the downstream prediction of political party affiliation.

Model	MFD	AR
SVM BoW	18.70	—
PSL BoW	21.88	—
Majority Vote	12.50	10.86
M1 (Unigrams)	7.17	8.68
M3 (+ Political Info)	22.01	30.45
M5 (+ Frames)	28.94	37.44
M9 (+ Bigrams)	67.93	66.50
M13 (All features)	72.49	69.38

Table 6: Overview of Macro-weighted Average F_1 Scores of SVM and PSL Models. The top portion of the table shows the results of the three baselines. The bottom portion shows a subset of the PSL models (parentheses indicate features added onto the previous models).

Evaluation Metrics: Since each tweet can have more than one moral foundation, our prediction task is a multilabel classification task. The precision of a multilabel model is the ratio of how many predicted labels are correct:

$$Precision = \frac{1}{T}\sum_{t=1}^{T}\frac{|Y_t \cap h(x_t)|}{|h(x_t)|} \tag{1}$$

The recall of this model is the ratio of how many of the actual labels were predicted:

$$Recall = \frac{1}{T}\sum_{t=1}^{T}\frac{|Y_t \cap h(x_t)|}{|Y_t|} \tag{2}$$

[4]For this work, we used the SVM implementation provided by scikit-learn.

In both formulas, T is the number of tweets, Y_t is the true label for tweet *t*, x_t is a tweet example, and $h(x_t)$ are the predicted labels for that tweet. The F_1 score is computed as the harmonic mean of the precision and recall. Additionally, the last lines of Tables 7 and 8 provide the macro-weighted average F_1 score over all moral foundations.

Analysis of Supervised Experiments: We conducted supervised experiments using five-fold cross validation with randomly chosen splits. Table 6 shows an overview of the average results of our supervised experiments for five of the PSL models. The first column lists the SVM or PSL model. The second column presents the results of a given model when using the MFD as the source of the unigrams for the initial model (M1). The final column shows the results when the AR unigrams are used as the initial source of supervision. The first two rows show the results of predicting the morals present in tweets using a bag-of-words (BoW) approach. Both the SVM and PSL models perform poorly due to the eleven predictive classes and noisy input features. The third row shows the results when taking a majority vote over the presence of MFD unigrams, similar to previous works. This approach is simpler and less noisy than M1, the PSL model closest to this approach.

The last five lines of this table also show the overall trends of the full results shown in Tables 7 and 8. As can be seen in all three tables, as we add more information with each PSL model, the overall results continue to improve, with the final model (M13) achieving the highest F_1 score for both sources of unigrams.

An interesting trend to note is that the AR unigrams based models result in better average performance for most of the models until M9. Models M9 and above incorporate the most powerful features: bigrams and trigrams with phrases and frames. This suggests that the AR unigrams, designed specifically for the political Twitter domain, are more useful than the MFD unigrams, *when only unigrams are available*. Conversely, the MFD unigrams are designed to *conceptually* capture morality, and therefore have weaker performance in the unigram-based models, but achieve higher performance when combined with the more powerful features of the higher models. For all models, incorporating phrases and frames results in a more accurate prediction than when using unigrams alone.

Moral Fdn.	Results of Non-Joint PSL Model Predictions												
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
Care	16.61	52.51	43.34	53.24	53.38	53.59	55.64	62.40	66.00	66.48	67.32	67.59	**67.78**
Harm	12.57	47.62	42.58	50.39	57.24	55.29	60.06	67.06	71.58	71.58	72.39	**73.68**	73.54
Fairness	24.68	52.22	45.16	50.22	51.50	50.86	61.54	71.13	74.00	74.50	75.32	**75.48**	**75.48**
Cheating	0.00	0.00	0.00	0.00	0.00	0.00	0.00	21.05	51.85	51.85	56.14	**60.00**	**60.00**
Loyalty	18.29	44.53	41.49	43.87	43.59	44.22	47.65	59.15	62.82	63.75	63.75	63.95	**64.20**
Betrayal	0.00	0.00	10.00	20.00	20.00	20.00	18.18	34.78	66.67	66.67	68.42	**70.00**	**70.00**
Authority	0.00	30.93	30.19	33.10	35.53	33.96	45.52	55.29	62.50	65.91	67.78	69.23	**69.61**
Subversion	3.77	32.69	13.39	25.90	24.66	42.36	59.29	72.66	77.29	78.08	78.41	79.22	**79.61**
Purity	0.00	8.89	4.88	9.88	9.76	56.12	63.86	70.86	72.13	74.16	76.09	79.14	**80.41**
Degradation	2.99	15.38	9.52	10.00	10.00	8.00	20.69	52.94	61.54	61.54	68.09	**73.47**	**73.47**
Non-moral	0.00	0.00	1.60	3.51	12.70	12.31	54.55	71.14	80.90	81.82	82.35	82.54	**83.33**
Average	7.17	25.89	22.01	27.28	28.94	34.25	44.27	58.04	67.93	68.76	70.55	72.21	**72.49**

Table 7: F_1 Scores of PSL Models Using the Moral Foundations Dictionary (MFD). The highest prediction per moral foundation is marked in bold.

Moral Fdn.	Results of Non-Joint PSL Model Predictions												
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
Care	7.29	29.72	30.51	30.86	30.62	35.66	46.41	54.17	61.77	62.16	62.91	64.79	**64.91**
Harm	2.25	8.89	19.31	21.89	26.18	26.09	37.28	52.40	62.18	62.18	63.74	64.67	**64.86**
Fairness	9.15	26.43	27.12	28.70	30.43	31.92	53.56	69.88	72.52	72.52	74.26	**74.63**	**74.63**
Cheating	4.76	13.33	25.45	25.45	38.71	39.34	40.68	51.61	62.16	62.16	64.94	**65.82**	**65.82**
Loyalty	2.61	19.66	23.85	25.10	27.31	29.57	38.06	47.73	54.30	55.22	55.59	57.34	**57.91**
Betrayal	0.00	0.00	0.00	6.25	12.12	11.76	18.18	28.57	60.47	60.47	62.22	**65.22**	**65.22**
Authority	13.59	40.19	48.40	51.82	56.25	56.14	57.04	63.30	66.45	66.67	67.32	**67.53**	**67.53**
Subversion	4.79	40.69	42.34	43.21	43.93	44.03	47.20	55.12	56.47	56.47	57.07	57.53	**57.65**
Purity	5.62	13.64	19.78	23.16	30.00	60.38	69.66	76.67	79.35	79.35	80.21	81.82	**82.52**
Degradation	16.66	31.37	37.74	44.83	51.61	51.61	57.14	68.75	73.53	73.53	77.33	**78.95**	**78.95**
Non-moral	28.78	52.99	60.48	61.33	64.72	66.00	73.62	79.41	82.25	82.25	82.55	82.78	**83.20**
Average	8.68	25.17	30.45	32.96	37.44	41.14	48.98	58.87	66.50	66.63	68.01	69.19	**69.38**

Table 8: F_1 Scores of PSL Models Using Annotator's Rationale (AR). The highest prediction per moral foundation is marked in bold.

Analysis of Joint Experiments: In addition to studying the effects of each feature on the models' ability to predict moral foundations, we also explored jointly predicting both policy frames and moral foundations. These tasks are highly related as shown by the large increase in score between the baseline and skyline measurements in Table 9 once frames are incorporated into the models.

Both moral foundations and frame classification are challenging multilabel classification tasks, the former using 11 possible foundations and the latter consisting of 17 possible frames. Furthermore, joint learning problems are harder to learn due to a larger numbers of parameters, which in turn also affects learning and inference.

Table 9 shows the macro-weighted average F_1 scores for three different models. The Baseline model shows the results of predicting only the Moral of the tweet using the non-joint model M13, which uses all features with frames initialized. The Joint model is designed to predict both the moral foundation and frame of a tweet simultaneously (as shown in Table 5), with no frame initialization. Finally, the Skyline model is M13 with all features, where the frames are initialized with their known values.

The joint model using AR unigrams outperforms the baseline, showing that there is some benefit to modeling moral foundations and frames together, as well as using domain-specific unigrams. However, it is unable to beat the MFD-based unigrams model. This is likely due to the large amount of noise introduced by incorrect frame predictions into the joint model. As expected, the joint model does not outperform the skyline model which is able to use the known values of the frames in order to accurately classify the moral foundations associated with the tweets.

Finally, the predictions for the frames in the joint model were quite low, going from an average F_1 score of 26.09 in M1 to an average F_1 score of 27.99 in M13. This likely has two causes: (1) frame prediction is a challenging 17-label classification task, with a random baseline of 6% (which our approach is able to exceed) and (2) the lower performance is because the frames are predicted with *no initialization*. In previous works, the frame prediction models are initialized with a set of unigrams expected to occur for each frame. Different from this approach, the only information our models provide to the frames are political party, issue, associated bigrams and trigrams, and the *predicted values for the moral foundations* from using this information. The F_1 score of 27.99 with such minimal initialization indicates that there is indeed a relationship between policy frames and the moral foundations expressed in tweets worth exploring in future work.

PSL MODEL	MFD	AR
BASELINE	55.49	55.88
JOINT	51.22	58.75
SKYLINE	72.49	69.38

Table 9: Overview of Macro-weighted Average F_1 Scores of Joint PSL Model M13. BASELINE is the MORAL prediction result. JOINT is the result of jointly predicting the MORAL and uninitialized FRAME predicates. SKYLINE shows the results when using all features with initialized frames.

6 Qualitative Results

Previous works (Makazhanov and Rafiei, 2013; Preoţiuc-Pietro et al., 2017) have shown the usefulness of moral foundations for the prediction of political party preference and the political ideologies of Twitter users. The moral foundation information used in these tasks is typically represented as word-level features extracted from the MFD. Unfortunately, these dictionary-based features are often too noisy to contribute to highly accurate predictions.

Recall the example tweets shown in Figures 1 and 2. Both figures are examples of tweets that are mislabeled by the traditional MFD-based approach, but correctly labeled using PSL Model M13. Using the MFD, Figure 1 is labeled as *Authority* due to "permit", the only matching unigram, while Figure 2 is incorrectly labeled as *Care*, even though there is one matching unigram for *Harm* and one for *Care*. To further demonstrate this point we compare the dictionary features to features extracted from the MORAL predictions of our PSL model.

Table 10 shows the results of using the different feature sets for the prediction of political affiliation of the author of a given tweet. All three models use moral information for prediction, but this information is represented differently in each of the models. The MFD model (line 1) uses the MFD unigrams to directly predict the political party of the author. The PSL model (line 2) uses the MF *prediction* made by the best performing model (M13) as features. Finally, the GOLD model (line 3) uses the actual MF annotations.

The difference in performance between the GOLD and MFD results shows that directly mapping the expected MFD unigrams to politicians' tweets is not informative enough for party affiliation prediction. However, by using abstract representations of language, the PSL model is able to achieve results closer to that which can be attained when using the *actual annotations* as features.

PSL MODEL	REP	DEM
MFD	48.72	51.28
PSL	61.25	66.92
GOLD	68.57	71.43

Table 10: Accuracy of Author Political Party Prediction. REP represents Republican and DEM represents Democrat.

7 Conclusion

Moral foundations and policy frames are employed as political strategies by politicians to garner support from the public. Politicians carefully word their statements to express their moral and social positions on issues, while maximizing their base's response to their message. In this paper we present PSL models for the classification of moral foundations expressed in political discourse on the microblog, Twitter. We show the benefits and drawbacks of traditionally used MFD unigrams and domain-specific unigrams for initialization of the models. We also provide an initial approach to the joint modeling of frames and moral foundations. In future works, we will exploit the interesting connections between moral foundations and frames for the analysis of more detailed ideological leanings and stance prediction.

Acknowledgments

We thank Lyle Ungar for his insight and discussion which inspired this work. We also thank the anonymous reviewers for their thoughtful comments and suggestions. This research was partly funded by a Google Focused Research Award.

References

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proc. of UAI*.

Akshat Bakliwal, Jennifer Foster, Jennifer van der Puil, Ron O'Brien, Lamia Tounsi, and Mark Hughes. 2013. Sentiment analysis of political tweets: Towards an accurate classifier. In *Proc. of ACL*.

David Bamman and Noah A Smith. 2015. Open extraction of fine-grained political statements. In *Proc. of EMNLP*.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and comparing computational approaches for identifying the language of framing in political news. In *In Proc. of NAACL*.

Adam Bermingham and Alan F Smeaton. 2011. On using twitter to monitor political sentiment and predict election results.

Amber Boydstun, Dallas Card, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2014. Tracking the development of media frames within and across policy issues.

Lauren M. Burch, Evan L. Frederick, and Ann Pegoraro. 2015. Kissing in the carnage: An examination of framing on twitter during the vancouver riots. *Journal of Broadcasting & Electronic Media*, 59(3):399–415.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proc. of ACL*.

Sarah Djemili, Julien Longhi, Claudia Marinica, Dimitris Kotzinos, and Georges-Elia Sarfati. 2014. What does twitter have to say about ideology? In *NLP 4 CMC*.

Dean Fulgoni, Jordan Carpenter, Lyle Ungar, and Daniel Preotiuc-Pietro. 2016. An empirical exploration of moral foundations theory in partisan news sources. In *Proc. of LREC*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. The paraphrase database. In *Proc. of NAACL-HLT*.

Justin Garten, Reihane Boghrati, Joe Hoover, Kate M Johnson, and Morteza Dehghani. 2016. Morality between the lines: Detecting moral sentiment in text. In *IJCAI workshops*.

Jesse Graham, Jonathan Haidt, and Brian A Nosek. 2009. Liberals and conservatives rely on different sets of moral foundations. *Journal of personality and social psychology*, 96(5):1029.

Jesse Graham, Brian A Nosek, and Jonathan Haidt. 2012. The moral stereotypes of liberals and conservatives: Exaggeration of differences across the political spectrum. *PloS one*, 7(12):e50092.

Jonathan Haidt and Jesse Graham. 2007. When morality opposes justice: Conservatives have moral intuitions that liberals may not recognize. *Social Justice Research*, 20(1):98–116.

Jonathan Haidt and Craig Joseph. 2004. Intuitive ethics: How innately prepared intuitions generate culturally variable virtues. *Daedalus*, 133(4):55–66.

Summer Harlow and Thomas Johnson. 2011. The arab spring— overthrowing the protest paradigm? how the new york times, global voices and twitter covered the egyptian revolution. *International Journal of Communication*, 5(0).

Iyyer, Enns, Boyd-Graber, and Resnik. 2014. Political ideology detection using recursive neural networks. In *Proc. of ACL*.

S. Mo Jang and P. Sol Hart. 2015. Polarized frames on ”climate change” and ”global warming” across countries and states: Evidence from twitter big data. *Global Environmental Change*, 32:11–17.

Kristen Johnson and Dan Goldwasser. 2016. “all i know about politics is what i read in twitter”: Weakly supervised models for extracting politicians stances from twitter. In *Proceedings of COLING*.

Kristen Johnson, Di Jin, and Dan Goldwasser. 2017. Leveraging behavioral and social information for weakly supervised collective classification of political discourse on twitter. In *Proc. of ACL*.

I-Ta Lee, Mahak Goindani, Chang Li, Di Jin, Kristen Johnson, Xiao Zhang, Maria Pacheco, and Dan Goldwasser. 2017. Purduenlp at semeval-2017 task 1: Predicting semantic textual similarity with paraphrase and event embeddings. In *Proc. of SemEval*.

Ying Lin, Joe Hoover, Morteza Dehghani, Marlon Mooijman, and Heng Ji. 2017. Acquiring background knowledge to improve moral value prediction. *arXiv preprint arXiv:1709.05467*.

Aibek Makazhanov and Davood Rafiei. 2013. Predicting political preference of twitter users. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 298–305, New York, NY, USA. ACM.

Sharon Meraz and Zizi Papacharissi. 2013. Networked gatekeeping and networked framing on #egypt. *The International Journal of Press/Politics*, 18(2):138–166.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proc. of ICWSM*.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Ferran Pla and Lluís F Hurtado. 2014. Political tendency identification in twitter using sentiment analysis techniques. In *Proc. of COLING*.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 729–740.

Sim, Acree, Gross, and Smith. 2013. Measuring ideological proportions in political speeches. In *Proc. of EMNLP*.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proc. of ACL*.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *ICWSM*.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653.

Coarse-to-Fine Decoding for Neural Semantic Parsing

Li Dong and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
li.dong@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

Semantic parsing aims at mapping natural language utterances into structured meaning representations. In this work, we propose a structure-aware neural architecture which decomposes the semantic parsing process into two stages. Given an input utterance, we first generate a rough sketch of its meaning, where low-level information (such as variable names and arguments) is glossed over. Then, we fill in missing details by taking into account the natural language input and the sketch itself. Experimental results on four datasets characteristic of different domains and meaning representations show that our approach consistently improves performance, achieving competitive results despite the use of relatively simple decoders.

1 Introduction

Semantic parsing maps natural language utterances onto machine interpretable meaning representations (e.g., executable queries or logical forms). The successful application of recurrent neural networks to a variety of NLP tasks (Bahdanau et al., 2015; Vinyals et al., 2015) has provided strong impetus to treat semantic parsing as a sequence-to-sequence problem (Jia and Liang, 2016; Dong and Lapata, 2016; Ling et al., 2016). The fact that meaning representations are typically structured objects has prompted efforts to develop neural architectures which explicitly account for their structure. Examples include tree decoders (Dong and Lapata, 2016; Alvarez-Melis and Jaakkola, 2017), decoders constrained by a grammar model (Xiao et al., 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017), or modular decoders which use syntax to dynamically compose various submodels (Rabinovich et al., 2017).

In this work, we propose to decompose the decoding process into two stages. The first decoder focuses on predicting a rough *sketch* of the meaning representation, which omits low-level details, such as arguments and variable names. Example sketches for various meaning representations are shown in Table 1. Then, a second decoder fills in missing details by conditioning on the natural language input and the sketch itself. Specifically, the sketch constrains the generation process and is encoded into vectors to guide decoding.

We argue that there are at least three advantages to the proposed approach. Firstly, the decomposition disentangles high-level from low-level semantic information, which enables the decoders to model meaning at different levels of granularity. As shown in Table 1, sketches are more compact and as a result easier to generate compared to decoding the entire meaning structure in one go. Secondly, the model can explicitly share knowledge of coarse structures for the examples that have the same sketch (i.e., basic meaning), even though their actual meaning representations are different (e.g., due to different details). Thirdly, after generating the sketch, the decoder knows what the basic meaning of the utterance looks like, and the model can use it as global context to improve the prediction of the final details.

Our framework is flexible and not restricted to specific tasks or any particular model. We conduct experiments on four datasets representative of various semantic parsing tasks ranging from logical form parsing, to code generation, and SQL query generation. We adapt our architecture to these tasks and present several ways to obtain sketches from their respective meaning representations. Experimental results show that our framework achieves competitive performance compared

Dataset	Length	Example
	7.6	x : *which state has the most rivers running through it?*
GEO	13.7	y : (argmax \$0 (state:t \$0) (count \$1 (and (river:t \$1) (loc:t \$1 \$0))))
	6.9	a : (argmax#1 state:t@1 (count#1 (and river:t@1 loc:t@2)))
	11.1	x : *all flights from dallas before 10am*
ATIS	21.1	y : (lambda \$0 e (and (flight \$0) (from \$0 dallas:ci) (< (departure_time \$0) 1000:ti)))
	9.2	a : (lambda#2 (and flight@1 from@2 (< departure_time@1 ?)))
	14.4	x : *if length of bits is lesser than integer 3 or second element of bits is not equal to string 'as' ,*
DJANGO	8.7	y : if len(bits) < 3 or bits[1] != 'as':
	8.0	a : if len (NAME) < NUMBER or NAME [NUMBER] != STRING :
	17.9	Table schema: $\|$*Pianist*$\|$*Conductor*$\|$*Record Company*$\|$*Year of Recording*$\|$*Format*$\|$
WIKISQL	13.3	x : *What record company did conductor Mikhail Snitko record for after 1996?*
	13.0	y : `SELECT` *Record Company* `WHERE` (*Year of Recording* > *1996*) `AND` (*Conductor* = *Mikhail Snitko*)
	2.7	a : `WHERE` > `AND` =

Table 1: Examples of natural language expressions x, their meaning representations y, and meaning sketches a. The average number of tokens is shown in the second column.

with previous systems, despite employing relatively simple sequence decoders.

2 Related Work

Various models have been proposed over the years to learn semantic parsers from natural language expressions paired with their meaning representations (Tang and Mooney, 2000; Ge and Mooney, 2005; Zettlemoyer and Collins, 2007; Wong and Mooney, 2007; Lu et al., 2008; Kwiatkowski et al., 2011; Andreas et al., 2013; Zhao and Huang, 2015). These systems typically learn lexicalized mapping rules and scoring models to construct a meaning representation for a given input.

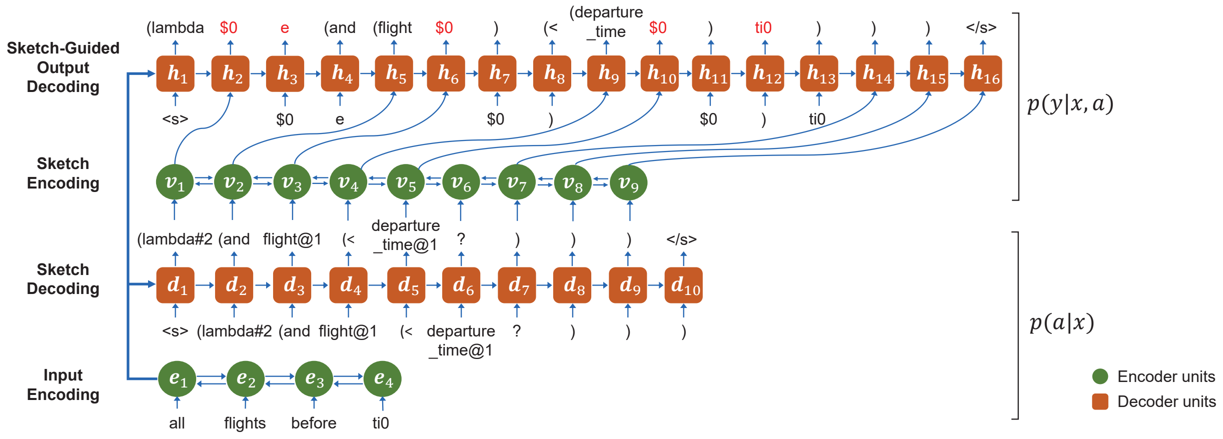
More recently, neural sequence-to-sequence models have been applied to semantic parsing with promising results (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016), eschewing the need for extensive feature engineering. Several ideas have been explored to enhance the performance of these models such as data augmentation (Kočiský et al., 2016; Jia and Liang, 2016), transfer learning (Fan et al., 2017), sharing parameters for multiple languages or meaning representations (Susanto and Lu, 2017; Herzig and Berant, 2017), and utilizing user feedback signals (Iyer et al., 2017). There are also efforts to develop structured decoders that make use of the syntax of meaning representations. Dong and Lapata (2016) and Alvarez-Melis and Jaakkola (2017) develop models which generate tree structures in a top-down fashion. Xiao et al. (2016) and Krishnamurthy et al. (2017) employ the grammar to constrain the decoding process. Cheng et al. (2017) use a transition system to generate variable-free queries. Yin and Neubig (2017) design a grammar model for the generation of abstract syntax trees (Aho et al., 2007) in depth-first, left-to-right order. Rabinovich et al. (2017) propose a modular decoder whose submodels are dynamically composed according to the generated tree structure.

Our own work also aims to model the structure of meaning representations more faithfully. The flexibility of our approach enables us to easily apply sketches to different types of meaning representations, e.g., trees or other structured objects. Coarse-to-fine methods have been popular in the NLP literature, and are perhaps best known for syntactic parsing (Charniak et al., 2006; Petrov, 2011). Artzi and Zettlemoyer (2013) and Zhang et al. (2017) use coarse lexical entries or macro grammars to reduce the search space of semantic parsers. Compared with coarse-to-fine inference for lexical induction, sketches in our case are abstractions of the final meaning representation.

The idea of using sketches as intermediate representations has also been explored in the field of program synthesis (Solar-Lezama, 2008; Zhang and Sun, 2013; Feng et al., 2017). Yaghmazadeh et al. (2017) use SEMPRE (Berant et al., 2013) to map a sentence into SQL sketches which are completed using program synthesis techniques and iteratively repaired if they are faulty.

3 Problem Formulation

Our goal is to learn semantic parsers from instances of natural language expressions paired with their structured meaning representations.



Sketch-Guided Output Decoding
Sketch Encoding
Sketch Decoding
Input Encoding
(lambda $0 e (and (flight $0) (< (departure _time $0) ti0))) </s>
<s> $0 e $0) $0) ti0
(lambda#2 (and flight@1 (< departure _time@1 ?))) </s>
<s> (lambda#2 (and flight@1 (< departure _time@1 ?)))
all flights before ti0
p(y|x, a)
p(a|x)
● Encoder units
● Decoder units

Figure 1: We first generate the meaning sketch a for natural language input x. Then, a fine meaning decoder fills in the missing details (shown in red) of meaning representation y. The coarse structure a is used to guide and constrain the output decoding.

Let $x = x_1 \cdots x_{|x|}$ denote a natural language expression, and $y = y_1 \cdots y_{|y|}$ its meaning representation. We wish to estimate $p(y|x)$, the conditional probability of meaning representation y given input x. We decompose $p(y|x)$ into a two-stage generation process:

$$p(y|x) = p(y|x, a)\, p(a|x) \tag{1}$$

where $a = a_1 \cdots a_{|a|}$ is an abstract sketch representing the meaning of y. We defer detailed description of how sketches are extracted to Section 4. Suffice it to say that the extraction amounts to stripping off arguments and variable names in logical forms, schema specific information in SQL queries, and substituting tokens with types in source code (see Table 1).

As shown in Figure 1, we first predict sketch a for input x, and then fill in missing details to generate the final meaning representation y by conditioning on both x and a. The sketch is encoded into vectors which in turn guide and constrain the decoding of y. We view the input expression x, the meaning representation y, and its sketch a as sequences. The generation probabilities are factorized as:

$$p(a|x) = \prod_{t=1}^{|a|} p(a_t | a_{<t}, x) \tag{2}$$

$$p(y|x, a) = \prod_{t=1}^{|y|} p(y_t | y_{<t}, x, a) \tag{3}$$

where $a_{<t} = a_1 \cdots a_{t-1}$, and $y_{<t} = y_1 \cdots y_{t-1}$. In the following, we will explain how $p(a|x)$ and $p(y|x, a)$ are estimated.

3.1 Sketch Generation

An *encoder* is used to encode the natural language input x into vector representations. Then, a *decoder* learns to compute $p(a|x)$ and generate the sketch a conditioned on the encoding vectors.

Input Encoder Every input word is mapped to a vector via $\mathbf{x}_t = \mathbf{W}_x \mathbf{o}(x_t)$, where $\mathbf{W}_x \in \mathbb{R}^{n \times |\mathcal{V}_x|}$ is an embedding matrix, $|\mathcal{V}_x|$ is the vocabulary size, and $\mathbf{o}(x_t)$ a one-hot vector. We use a bi-directional recurrent neural network with long short-term memory units (LSTM, Hochreiter and Schmidhuber 1997) as the input encoder. The encoder recursively computes the hidden vectors at the t-th time step via:

$$\overrightarrow{\mathbf{e}}_t = \mathrm{f}_{\mathrm{LSTM}}\left(\overrightarrow{\mathbf{e}}_{t-1}, \mathbf{x}_t\right), t = 1, \cdots, |x| \tag{4}$$

$$\overleftarrow{\mathbf{e}}_t = \mathrm{f}_{\mathrm{LSTM}}\left(\overleftarrow{\mathbf{e}}_{t+1}, \mathbf{x}_t\right), t = |x|, \cdots, 1 \tag{5}$$

$$\mathbf{e}_t = [\overrightarrow{\mathbf{e}}_t, \overleftarrow{\mathbf{e}}_t] \tag{6}$$

where $[\cdot, \cdot]$ denotes vector concatenation, $\mathbf{e}_t \in \mathbb{R}^n$, and $\mathrm{f}_{\mathrm{LSTM}}$ is the LSTM function.

Coarse Meaning Decoder The decoder's hidden vector at the t-th time step is computed by $\mathbf{d}_t = \mathrm{f}_{\mathrm{LSTM}}(\mathbf{d}_{t-1}, \mathbf{a}_{t-1})$, where $\mathbf{a}_{t-1} \in \mathbb{R}^n$ is the embedding of the previously predicted token. The hidden states of the first time step in the decoder are initialized by the concatenated encoding vectors $\mathbf{d}_0 = [\overrightarrow{\mathbf{e}}_{|x|}, \overleftarrow{\mathbf{e}}_1]$. Additionally, we use an attention mechanism (Luong et al., 2015) to learn soft alignments. We compute the attention score for the current time step t of the decoder, with the k-th hidden state in the encoder as:

$$s_{t,k} = \exp\{\mathbf{d}_t \cdot \mathbf{e}_k\} / Z_t \tag{7}$$

where $Z_t = \sum_{j=1}^{|x|} \exp\{\mathbf{d}_t \cdot \mathbf{e}_j\}$ is a normalization term. Then we compute $p(a_t|a_{<t}, x)$ via:

$$\mathbf{e}_t^d = \sum_{k=1}^{|x|} s_{t,k}\mathbf{e}_k \tag{8}$$

$$\mathbf{d}_t^{att} = \tanh\left(\mathbf{W}_1\mathbf{d}_t + \mathbf{W}_2\mathbf{e}_t^d\right) \tag{9}$$

$$p(a_t|a_{<t}, x) = \text{softmax}_{a_t}\left(\mathbf{W}_o\mathbf{d}_t^{att} + \mathbf{b}_o\right) \tag{10}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{n\times n}$, $\mathbf{W}_o \in \mathbb{R}^{|\mathcal{V}_a|\times n}$, and $\mathbf{b}_o \in \mathbb{R}^{|\mathcal{V}_a|}$ are parameters. Generation terminates once an end-of-sequence token "*</s>*" is emitted.

3.2 Meaning Representation Generation

Meaning representations are predicted by conditioning on the input x and the generated sketch a. The model uses the encoder-decoder architecture to compute $p(y|x, a)$, and decorates the sketch a with details to generate the final output.

Sketch Encoder As shown in Figure 1, a bi-directional LSTM encoder maps the sketch sequence a into vectors $\{\mathbf{v}_k\}_{k=1}^{|a|}$ as in Equation (6), where $\mathbf{v}_k$ denotes the vector of the k-th time step.

Fine Meaning Decoder The final decoder is based on recurrent neural networks with an attention mechanism, and shares the input encoder described in Section 3.1. The decoder's hidden states $\{\mathbf{h}_t\}_{t=1}^{|y|}$ are computed via:

$$\mathbf{i}_t = \begin{cases} \mathbf{v}_k & y_{t-1} \text{ is determined by } a_k \\ \mathbf{y}_{t-1} & \text{otherwise} \end{cases} \tag{11}$$

$$\mathbf{h}_t = \text{f}_{\text{LSTM}}\left(\mathbf{h}_{t-1}, \mathbf{i}_t\right)$$

where $\mathbf{h}_0 = [\overrightarrow{\mathbf{e}}_{|x|}, \overleftarrow{\mathbf{e}}_1]$, and $\mathbf{y}_{t-1}$ is the embedding of the previously predicted token. Apart from using the embeddings of previous tokens, the decoder is also fed with $\{\mathbf{v}_k\}_{k=1}^{|a|}$. If y_{t-1} is determined by a_k in the sketch (i.e., there is a one-to-one alignment between y_{t-1} and a_k), we use the corresponding token's vector $\mathbf{v}_k$ as input to the next time step.

The sketch constrains the decoding output. If the output token y_t is already in the sketch, we force y_t to conform to the sketch. In some cases, sketch tokens will indicate what information is missing (e.g., in Figure 1, token "*flight@1*" indicates that an argument is missing for the predicate "*flight*"). In other cases, sketch tokens will not reveal the number of missing tokens (e.g., "STRING" in DJANGO) but the decoder's output will indicate whether missing details have been generated (e.g., if the decoder emits a closing quote token for "STRING"). Moreover, type information in sketches can be used to constrain generation. In Table 1, sketch token "NUMBER" specifies that a numeric token should be emitted.

For the missing details, we use the hidden vector $\mathbf{h}_t$ to compute $p(y_t|y_{<t}, x, a)$, analogously to Equations (7)–(10).

3.3 Training and Inference

The model's training objective is to maximize the log likelihood of the generated meaning representations given natural language expressions:

$$\max \sum_{(x,a,y)\in\mathcal{D}} \log p(y|x, a) + \log p(a|x)$$

where $\mathcal{D}$ represents training pairs.

At test time, the prediction for input x is obtained via $\hat{a} = \arg\max_{a'} p(a'|x)$ and $\hat{y} = \arg\max_{y'} p(y'|x, \hat{a})$, where a' and y' represent coarse- and fine-grained meaning candidates. Because probabilities $p(a|x)$ and $p(y|x, a)$ are factorized as shown in Equations (2)–(3), we can obtain best results approximately by using greedy search to generate tokens one by one, rather than iterating over all candidates.

4 Semantic Parsing Tasks

In order to show that our framework applies across domains and meaning representations, we developed models for three tasks, namely parsing natural language to logical form, to Python source code, and to SQL query. For each of these tasks we describe the datasets we used, how sketches were extracted, and specify model details over and above the architecture presented in Section 3.

4.1 Natural Language to Logical Form

For our first task we used two benchmark datasets, namely GEO (880 language queries to a database of U.S. geography) and ATIS (5,410 queries to a flight booking system). Examples are shown in Table 1 (see the first and second block). We used standard splits for both datasets: 600 training and 280 test instances for GEO (Zettlemoyer and Collins, 2005); 4,480 training, 480 development, and 450 test examples for ATIS. Meaning representations in these datasets are based on λ-calculus (Kwiatkowski et al., 2011). We use brackets to linearize the hierarchical structure.

Algorithm 1 Sketch for GEO and ATIS

```
Input: t: Tree-structure λ-calculus expression
       t.pred: Predicate name, or operator name
Output: a: Meaning sketch
  ▷ (count $0 (< (fare $0) 50:do))→(count#1 (< fare@1 ?))
  function SKETCH(t)
      if t is leaf then                    ▷ No nonterminal in arguments
          return "%s@%d" % (t.pred, len(t.args))
      if t.pred is λ operator, or quantifier then ▷ e.g., count
          Omit variable information defined by t.pred
          t.pred ← "%s#%d" % (t.pred, len(variable))
      for c ← argument in t.args do
          if c is nonterminal then
              c ← SKETCH(c)
          else
              c ← "?"                      ▷ Placeholder for terminal
      return t
```

The first element between a pair of brackets is an operator or predicate name, and any remaining elements are its arguments.

Algorithm 1 shows the pseudocode used to extract sketches from λ-calculus-based meaning representations. We strip off arguments and variable names in logical forms, while keeping predicates, operators, and composition information. We use the symbol "@" to denote the number of missing arguments in a predicate. For example, we extract "*from@2*" from the expression "*(from $0 dallas:ci)*" which indicates that the predicate "*from*" has two arguments. We use "*?*" as a placeholder in cases where only partial argument information can be omitted. We also omit variable information defined by the lambda operator and quantifiers (e.g., *exists*, *count*, and *argmax*). We use the symbol "#" to denote the number of omitted tokens. For the example in Figure 1, "*lambda $0 e*" is reduced to "*lambda#2*".

The meaning representations of these two datasets are highly compositional, which motivates us to utilize the hierarchical structure of λ-calculus. A similar idea is also explored in the tree decoders proposed in Dong and Lapata (2016) and Yin and Neubig (2017) where parent hidden states are fed to the input gate of the LSTM units. On the contrary, parent hidden states serve as input to the softmax classifiers of both fine and coarse meaning decoders.

Parent Feeding Taking the meaning sketch "*(and flight@1 from@2)*" as an example, the parent of "*from@2*" is "*(and*". Let p_t denote the parent of the t-th time step in the decoder. Compared with Equation (10), we use the vector $\mathbf{d}_t^{att}$ and the hidden state of its parent $\mathbf{d}_{p_t}$ to compute the probability $p(a_t|a_{<t}, x)$ via:

$$p(a_t|a_{<t}, x) = \text{softmax}_{a_t}\left(\mathbf{W}_o[\mathbf{d}_t^{att}, \mathbf{d}_{p_t}] + \mathbf{b}_o\right)$$

where $[\cdot, \cdot]$ denotes vector concatenation. The parent feeding is used for both decoding stages.

4.2 Natural Language to Source Code

Our second semantic parsing task used DJANGO (Oda et al., 2015), a dataset built upon the Python code of the Django library. The dataset contains lines of code paired with natural language expressions (see the third block in Table 1) and exhibits a variety of use cases, such as iteration, exception handling, and string manipulation. The original split has $16,000$ training, $1,000$ development, and $1,805$ test instances.

We used the built-in lexical scanner of Python[1] to tokenize the code and obtain token types. Sketches were extracted by substituting the original tokens with their token types, except delimiters (e.g., "[", and ":"), operators (e.g., "+", and "*"), and built-in keywords (e.g., "True", and "while"). For instance, the expression "if s[:4].lower() == 'http':" becomes "if NAME [: NUMBER] . NAME ( ) == STRING :", with details about names, values, and strings being omitted.

DJANGO is a diverse dataset, spanning various real-world use cases and as a result models are often faced with out-of-vocabulary (OOV) tokens (e.g., variable names, and numbers) that are unseen during training. We handle OOV tokens with a copying mechanism (Gu et al., 2016; Gulcehre et al., 2016; Jia and Liang, 2016), which allows the fine meaning decoder (Section 3.2) to directly copy tokens from the natural language input.

Copying Mechanism Recall that we use a softmax classifier to predict the probability distribution $p(y_t|y_{<t}, x, a)$ over the pre-defined vocabulary. We also learn a copying gate $g_t \in [0, 1]$ to decide whether y_t should be copied from the input or generated from the vocabulary. We compute the modified output distribution via:

$$\begin{aligned} g_t &= \text{sigmoid}(\mathbf{w}_g \cdot \mathbf{h}_t + b_g) \\ \tilde{p}(y_t|y_{<t}, x, a) &= (1 - g_t) p(y_t|y_{<t}, x, a) \\ &\quad + \mathbb{1}_{[y_t \notin \mathcal{V}_y]} g_t \sum_{k: x_k = y_t} s_{t,k} \end{aligned}$$

[1] https://docs.python.org/3/library/tokenize

where $\mathbf{w}_g \in \mathbb{R}^n$ and $b_g \in \mathbb{R}$ are parameters, and the indicator function $\mathbb{1}_{[y_t \notin \mathcal{V}_y]}$ is 1 only if y_t is not in the target vocabulary $\mathcal{V}_y$; the attention score $s_{t,k}$ (see Equation (7)) measures how likely it is to copy y_t from the input word x_k.

4.3 Natural Language to SQL

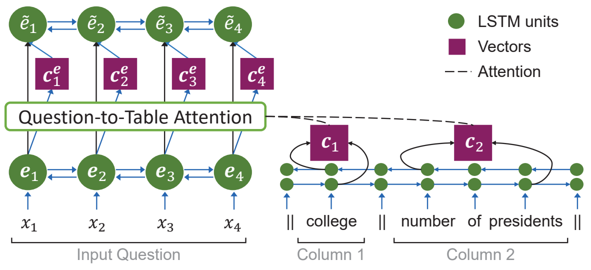
The WIKISQL (Zhong et al., 2017) dataset contains $80,654$ examples of questions and SQL queries distributed across $24,241$ tables from Wikipedia. The goal is to generate the correct SQL query for a natural language question and table schema (i.e., table column names), without using the content values of tables (see the last block in Table 1 for an example). The dataset is partitioned into a training set (70%), a development set (10%), and a test set (20%). Each table is present in one split to ensure generalization to unseen tables.

WIKISQL queries follow the format "`SELECT agg_op agg_col WHERE (cond_col cond_op cond) AND ...`", which is a subset of the SQL syntax. `SELECT` identifies the column that is to be included in the results after applying the aggregation operator `agg_op`[2] to column `agg_col`. `WHERE` can have zero or multiple conditions, which means that column `cond_col` must satisfy the constraints expressed by the operator `cond_op`[3] and the condition value `cond`. Sketches for SQL queries are simply the (sorted) sequences of condition operators `cond_op` in `WHERE` clauses. For example, in Table 1, sketch "`WHERE > AND =`" has two condition operators, namely ">" and "=".

The generation of SQL queries differs from our previous semantic parsing tasks, in that the table schema serves as input in addition to natural language. We therefore modify our input encoder in order to render it table-aware, so to speak. Furthermore, due to the formulaic nature of the SQL query, we only use our decoder to generate the `WHERE` clause (with the help of sketches). The `SELECT` clause has a fixed number of slots (i.e., aggregation operator `agg_op` and column `agg_col`), which we straightforwardly predict with softmax classifiers (conditioned on the input). We briefly explain how these components are modeled below.

Table-Aware Input Encoder Given a table schema with M columns, we employ the special token "$\|$" to concatenate its header names

[2] `agg_op` $\in \{empty,$ `COUNT`, `MIN`, `MAX`, `SUM`, `AVG`$\}$.
[3] `cond_op` $\in \{=, <, >\}$.



LSTM units
Vectors
Attention
Question-to-Table Attention
$\tilde{e}_1$ $\tilde{e}_2$ $\tilde{e}_3$ $\tilde{e}_4$
c_1^e c_2^e c_3^e c_4^e
e_1 e_2 e_3 e_4
x_1 x_2 x_3 x_4
c_1 c_2
|| college || number of presidents ||
Input Question
Column 1
Column 2

Figure 2: Table-aware input encoder (left) and table column encoder (right) used for WIKISQL.

as "$\|c_{1,1} \cdots c_{1,|c_1|}\| \cdots \|c_{M,1} \cdots c_{M,|c_M|}\|$", where the k-th column ("$c_{k,1} \cdots c_{k,|c_k|}$") has $|c_k|$ words. As shown in Figure 2, we use bi-directional LSTMs to encode the whole sequence. Next, for column c_k, the LSTM hidden states at positions $c_{k,1}$ and $c_{k,|c_k|}$ are concatenated. Finally, the concatenated vectors are used as the encoding vectors $\{\mathbf{c}_k\}_{k=1}^{M}$ for table columns.

As mentioned earlier, the meaning representations of questions are dependent on the tables. As shown in Figure 2, we encode the input question x into $\{\mathbf{e}_t\}_{t=1}^{|x|}$ using LSTM units. At each time step t, we use an attention mechanism towards table column vectors $\{\mathbf{c}_k\}_{k=1}^{M}$ to obtain the most relevant columns for $\mathbf{e}_t$. The attention score from $\mathbf{e}_t$ to $\mathbf{c}_k$ is computed via $u_{t,k} \propto \exp\{\alpha(\mathbf{e}_t) \cdot \alpha(\mathbf{c}_k)\}$, where $\alpha(\cdot)$ is a one-layer neural network, and $\sum_{k=1}^{M} u_{t,k} = 1$. Then we compute the context vector $\mathbf{c}_t^e = \sum_{k=1}^{M} u_{t,k}\mathbf{c}_k$ to summarize the relevant columns for $\mathbf{e}_t$. We feed the concatenated vectors $\{[\mathbf{e}_t, \mathbf{c}_t^e]\}_{t=1}^{|x|}$ into a bi-directional LSTM encoder, and use the new encoding vectors $\{\tilde{\mathbf{e}}_t\}_{t=1}^{|x|}$ to replace $\{\mathbf{e}_t\}_{t=1}^{|x|}$ in other model components. We define the vector representation of input x as:

$$\tilde{\mathbf{e}} = [\overrightarrow{\tilde{\mathbf{e}}}_{|x|}, \overleftarrow{\tilde{\mathbf{e}}}_1] \tag{12}$$

analogously to Equations (4)–(6).

`SELECT` Clause We feed the question vector $\tilde{\mathbf{e}}$ into a softmax classifier to obtain the aggregation operator `agg_op`. If `agg_col` is the k-th table column, its probability is computed via:

$$\sigma(\mathbf{x}) = \mathbf{w}_3 \cdot \tanh(\mathbf{W}_4\mathbf{x} + \mathbf{b}_4) \tag{13}$$

$$p(\texttt{agg\_col} = k|x) \propto \exp\{\sigma([\tilde{\mathbf{e}}, \mathbf{c}_k])\} \tag{14}$$

where $\sum_{j=1}^{M} p(\texttt{agg\_col} = j|x) = 1$, $\sigma(\cdot)$ is a scoring network, and $\mathbf{W}_4 \in \mathbb{R}^{2n \times m}$, $\mathbf{w}_3, \mathbf{b}_4 \in \mathbb{R}^m$ are parameters.

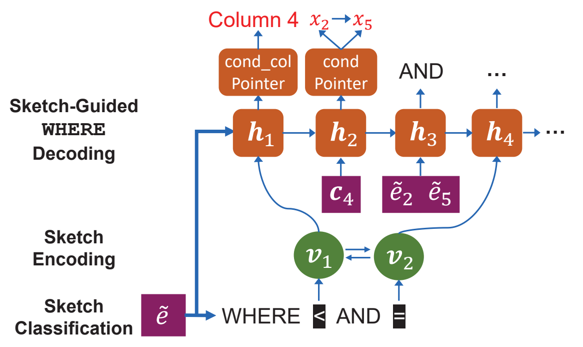


Figure 3: Fine meaning decoder of the WHERE clause used for WIKISQL.

WHERE Clause We first generate sketches whose details are subsequently decorated by the fine meaning decoder described in Section 3.2. As the number of sketches in the training set is small (35 in total), we model sketch generation as a classification problem. We treat each sketch a as a category, and use a softmax classifier to compute $p(a|x)$:

$$p(a|x) = \text{softmax}_a (\mathbf{W}_a \tilde{\mathbf{e}} + \mathbf{b}_a)$$

where $\mathbf{W}_a \in \mathbb{R}^{|\mathcal{V}_a| \times n}$, $\mathbf{b}_a \in \mathbb{R}^{|\mathcal{V}_a|}$ are parameters, and $\tilde{\mathbf{e}}$ is the table-aware input representation defined in Equation (12).

Once the sketch is predicted, we know the condition operators and number of conditions in the WHERE clause which follows the format "WHERE (cond_op cond_col cond) AND ...". As shown in Figure 3, our generation task now amounts to populating the sketch with condition columns cond_col and their values cond.

Let $\{\mathbf{h}_t\}_{t=1}^{|y|}$ denote the LSTM hidden states of the fine meaning decoder, and $\{\mathbf{h}_t^{att}\}_{t=1}^{|y|}$ the vectors obtained by the attention mechanism as in Equation (9). The condition column cond_col_{y_t} is selected from the table's headers. For the k-th column in the table, we compute $p(\text{cond\_col}_{y_t} = k | y_{<t}, x, a)$ as in Equation (14), but use different parameters and compute the score via $\sigma([\mathbf{h}_t^{att}, \mathbf{c}_k])$. If the k-th table column is selected, we use $\mathbf{c}_k$ for the input of the next LSTM unit in the decoder.

Condition values are typically mentioned in the input questions. These values are often phrases with multiple tokens (e.g., *Mikhail Snitko* in Table 1). We therefore propose to select a *text span* from input x for each condition value cond_{y_t} rather than copying tokens one by one. Let $x_l \cdots x_r$ denote the text span from which cond_{y_t} is copied. We factorize its probability as:

$$\begin{aligned} & p(\text{cond}_{y_t} = x_l \cdots x_r | y_{<t}, x, a) \\ & \quad = p\left([\![l]\!]_{y_t}^L | y_{<t}, x, a\right) p\left([\![r]\!]_{y_t}^R | y_{<t}, x, a, [\![l]\!]_{y_t}^L\right) \\ & p\left([\![l]\!]_{y_t}^L | y_{<t}, x, a\right) \propto \exp\{\sigma([\mathbf{h}_t^{att}, \tilde{\mathbf{e}}_l])\} \\ & p\left([\![r]\!]_{y_t}^R | y_{<t}, x, a, [\![l]\!]_{y_t}^L\right) \propto \exp\{\sigma([\mathbf{h}_t^{att}, \tilde{\mathbf{e}}_l, \tilde{\mathbf{e}}_r])\} \end{aligned}$$

where $[\![l]\!]_{y_t}^L / [\![r]\!]_{y_t}^R$ represents the first/last copying index of cond_{y_t} is l/r, the probabilities are normalized to 1, and $\sigma(\cdot)$ is the scoring network defined in Equation (13). Notice that we use different parameters for the scoring networks $\sigma(\cdot)$. The copied span is represented by the concatenated vector $[\tilde{\mathbf{e}}_l, \tilde{\mathbf{e}}_r]$, which is fed into a one-layer neural network and then used as the input to the next LSTM unit in the decoder.

5 Experiments

We present results on the three semantic parsing tasks discussed in Section 4. Our implementation and pretrained models are available at https://github.com/donglixp/coarse2fine.

5.1 Experimental Setup

Preprocessing For GEO and ATIS, we used the preprocessed versions provided by Dong and Lapata (2016), where natural language expressions are lowercased and stemmed with NLTK (Bird et al., 2009), and entity mentions are replaced by numbered markers. We combined predicates and left brackets that indicate hierarchical structures to make meaning representations compact. We employed the preprocessed DJANGO data provided by Yin and Neubig (2017), where input expressions are tokenized by NLTK, and quoted strings in the input are replaced with place holders. WIKISQL was preprocessed by the script provided by Zhong et al. (2017), where inputs were lowercased and tokenized by Stanford CoreNLP (Manning et al., 2014).

Configuration Model hyperparameters were cross-validated on the training set for GEO, and were validated on the development split for the other datasets. Dimensions of hidden vectors and word embeddings were selected from $\{250, 300\}$ and $\{150, 200, 250, 300\}$, respectively. The dropout rate was selected from $\{0.3, 0.5\}$. Label smoothing (Szegedy et al., 2016) was employed for GEO and ATIS. The smoothing parameter was set to 0.1. For WIKISQL, the hidden size of $\sigma(\cdot)$

Method	GEO	ATIS
ZC07 (Zettlemoyer and Collins, 2007)	86.1	84.6
UBL (Kwiatkowksi et al., 2010)	87.9	71.4
FUBL (Kwiatkowski et al., 2011)	88.6	82.8
GUSP++ (Poon, 2013)	—	83.5
KCAZ13 (Kwiatkowski et al., 2013)	89.0	—
DCS+L (Liang et al., 2013)	87.9	—
TISP (Zhao and Huang, 2015)	88.9	84.2
SEQ2SEQ (Dong and Lapata, 2016)	84.6	84.2
SEQ2TREE (Dong and Lapata, 2016)	87.1	84.6
ASN (Rabinovich et al., 2017)	85.7	85.3
ASN+SUPATT (Rabinovich et al., 2017)	87.1	85.9
ONESTAGE	85.0	85.3
COARSE2FINE	88.2	87.7
− sketch encoder	87.1	86.9
+ oracle sketch	93.9	95.1

Table 2: Accuracies on GEO and ATIS.

Method	Accuracy
Retrieval System	14.7
Phrasal SMT	31.5
Hierarchical SMT	9.5
SEQ2SEQ+UNK replacement	45.1
SEQ2TREE+UNK replacement	39.4
LPN+COPY (Ling et al., 2016)	62.3
SNM+COPY (Yin and Neubig, 2017)	71.6
ONESTAGE	69.5
COARSE2FINE	74.1
− sketch encoder	72.1
+ oracle sketch	83.0

Table 3: DJANGO results. Accuracies in the first and second block are taken from Ling et al. (2016) and Yin and Neubig (2017).

and $\alpha(\cdot)$ in Equation (13) was set to 64. Word embeddings were initialized by GloVe (Pennington et al., 2014), and were shared by table encoder and input encoder in Section 4.3. We appended 10-dimensional part-of-speech tag vectors to embeddings of the question words in WIKISQL. The part-of-speech tags were obtained by the spaCy toolkit. We used the RMSProp optimizer (Tieleman and Hinton, 2012) to train the models. The learning rate was selected from $\{0.002, 0.005\}$. The batch size was 200 for WIKISQL, and was 64 for other datasets. Early stopping was used to determine the number of epochs.

Evaluation We use accuracy as the evaluation metric, i.e., the percentage of the examples that are correctly parsed to their gold standard meaning representations. For WIKISQL, we also execute generated SQL queries on their corresponding tables, and report the execution accuracy which is defined as the proportion of correct answers.

5.2 Results and Analysis

We compare our model (COARSE2FINE) against several previously published systems as well as various baselines. Specifically, we report results with a model which decodes meaning representations in one stage (ONESTAGE) without leveraging sketches. We also report the results of several ablation models, i.e., without a sketch encoder and without a table-aware input encoder.

Table 2 presents our results on GEO and ATIS. Overall, we observe that COARSE2FINE outperforms ONESTAGE, which suggests that disentangling high-level from low-level information during decoding is beneficial. The results also show that removing the sketch encoder harms performance since the decoder loses access to additional contextual information. Compared with previous neural models that utilize syntax or grammatical information (SEQ2TREE, ASN; the second block in Table 2), our method performs competitively despite the use of relatively simple decoders. As an upper bound, we report model accuracy when gold meaning sketches are given to the fine meaning decoder (+oracle sketch). As can be seen, predicting the sketch correctly boosts performance. The oracle results also indicate the accuracy of the fine meaning decoder.

Table 3 reports results on DJANGO where we observe similar tendencies. COARSE2FINE outperforms ONESTAGE by a wide margin. It is also superior to the best reported result in the literature (SNM+COPY; see the second block in the table). Again we observe that the sketch encoder is beneficial and that there is an 8.9 point difference in accuracy between COARSE2FINE and the oracle.

Results on WIKISQL are shown in Table 4. Our model is superior to ONESTAGE as well as to previous best performing systems. COARSE2FINE's accuracies on aggregation `agg_op` and `agg_col` are 90.2% and 92.0%, respectively, which is comparable to SQLNET (Xu et al., 2017). So the most gain is obtained by the improved decoder of the `WHERE` clause. We also find that a table-aware input encoder is critical for doing well on this task, since the same question might lead to different SQL queries depending on the table schemas. Consider the question "*how many presidents are graduated from A*". The SQL query over table "‖*President*‖*College*‖" is "`SELECT`

Method	Accuracy	Execution Accuracy
SEQ2SEQ	23.4	35.9
Aug Ptr Network	43.3	53.3
SEQ2SQL (Zhong et al., 2017)	48.3	59.4
SQLNET (Xu et al., 2017)	61.3	68.0
ONESTAGE	68.8	75.9
COARSE2FINE	71.7	78.5
− sketch encoder	70.8	77.7
− table-aware input encoder	68.6	75.6
+ oracle sketch	73.0	79.6

Table 4: Evaluation results on WIKISQL. Accuracies in the first block are taken from Zhong et al. (2017) and Xu et al. (2017).

Method	GEO	ATIS	DJANGO	WIKISQL
ONESTAGE	85.4	85.9	73.2	95.4
COARSE2FINE	89.3	88.0	77.4	95.9

Table 5: Sketch accuracy. For ONESTAGE, sketches are extracted from the meaning representations it generates.

COUNT(*President*) WHERE (*College* = *A*)", but the query over table "‖*College*‖*Number of Presidents*‖" would be "SELECT *Number of Presidents* WHERE (*College* = *A*)".

We also examine the predicted sketches themselves in Table 5. We compare sketches generated by COARSE2FINE against ONESTAGE. The latter model generates meaning representations without an intermediate sketch generation stage. Nevertheless, we can extract sketches from the output of ONESTAGE following the procedures described in Section 4. Sketches produced by COARSE2FINE are more accurate across the board. This is not surprising because our model is trained explicitly to generate compact meaning sketches. Taken together (Tables 2–4), our results show that better sketches bring accuracy gains on GEO, ATIS, and DJANGO. On WIKISQL, the sketches predicted by COARSE2FINE are marginally better compared with ONESTAGE. Performance improvements on this task are mainly due to the fine meaning decoder. We conjecture that by decomposing decoding into two stages, COARSE2FINE can better match table columns and extract condition values without interference from the prediction of condition operators. Moreover, the sketch provides a canonical order of condition operators, which is beneficial for the decoding process (Vinyals et al., 2016; Xu et al., 2017).

6 Conclusions

In this paper we presented a coarse-to-fine decoding framework for neural semantic parsing. We first generate meaning sketches which abstract away from low-level information such as arguments and variable names and then predict missing details in order to obtain full meaning representations. The proposed framework can be easily adapted to different domains and meaning representations. Experimental results show that coarse-to-fine decoding improves performance across tasks. In the future, we would like to apply the framework in a weakly supervised setting, i.e., to learn semantic parsers from question-answer pairs and to explore alternative ways of defining meaning sketches.

Acknowledgments We would like to thank Pengcheng Yin for sharing with us the preprocessed version of the DJANGO dataset. We gratefully acknowledge the financial support of the European Research Council (award number 681760; Dong, Lapata) and the AdeptMind Scholar Fellowship program (Dong).

References

Alfred V Aho, Ravi Sethi, and Jeffrey D Ullman. 2007. *Compilers: principles, techniques, and tools*, volume 2. Addison-wesley Reading.

David Alvarez-Melis and Tommi S Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 47–52, Sofia, Bulgaria.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013*

Conference on Empirical Methods in Natural Language Processing, pages 1533–1544, Seattle, Washington. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 168–175, New York, NY.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 44–55. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 33–43, Berlin, Germany.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 48–56, Vancouver, Canada.

Yu Feng, Ruben Martins, Yuepeng Wang, Isil Dillig, and Thomas Reps. 2017. Component-based synthesis for complex apis. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, pages 599–612, New York, NY.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16, Ann Arbor, Michigan.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640, Berlin, Germany.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 623–628, Vancouver, Canada.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 963–973, Vancouver, Canada.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 12–22, Berlin, Germany.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1517–1527, Copenhagen, Denmark.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 599–609, Berlin, Germany.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics System Demonstrations*, pages 55–60, Baltimore, Maryland.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation. In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering*, pages 574–584, Washington, DC.

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

Slav Petrov. 2011. *Coarse-to-fine natural language processing*. Springer Science & Business Media.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 933–943, Sofia, Bulgaria.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1139–1149, Vancouver, Canada.

Armando Solar-Lezama. 2008. *Program Synthesis by Sketching*. Ph.D. thesis, University of California at Berkeley, Berkeley, CA.

Raymond Hendy Susanto and Wei Lu. 2017. Neural architectures for multilingual semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 38–44, Vancouver, Canada.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, Hong Kong, China.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2773–2781, Montreal, Canada.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1341–1350, Berlin, Germany.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: Query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1:63:1–63:26.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 440–450, Vancouver, Canada.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 678–687, Prague, Czech Republic.

Sai Zhang and Yuyin Sun. 2013. Automatically synthesizing SQL queries from input-output examples. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, pages 224–234, Piscataway, NJ.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223. Association for Computational Linguistics.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421, Denver, Colorado.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Confidence Modeling for Neural Semantic Parsing

Li Dong[†][*] and **Chris Quirk**[‡] and **Mirella Lapata**[†]
[†] School of Informatics, University of Edinburgh
[‡] Microsoft Research, Redmond
li.dong@ed.ac.uk chrisq@microsoft.com mlap@inf.ed.ac.uk

Abstract

In this work we focus on confidence modeling for neural semantic parsers which are built upon sequence-to-sequence models. We outline three major causes of uncertainty, and design various metrics to quantify these factors. These metrics are then used to estimate confidence scores that indicate whether model predictions are likely to be correct. Beyond confidence estimation, we identify which parts of the input contribute to uncertain predictions allowing users to interpret their model, and verify or refine its input. Experimental results show that our confidence model significantly outperforms a widely used method that relies on posterior probability, and improves the quality of interpretation compared to simply relying on attention scores.

1 Introduction

Semantic parsing aims to map natural language text to a formal meaning representation (e.g., logical forms or SQL queries). The neural sequence-to-sequence architecture (Sutskever et al., 2014; Bahdanau et al., 2015) has been widely adopted in a variety of natural language processing tasks, and semantic parsing is no exception. However, despite achieving promising results (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016), neural semantic parsers remain difficult to interpret, acting in most cases as a black box, not providing any information about what made them arrive at a particular decision. In this work, we explore ways to estimate and interpret the model's confidence in its predictions, which we argue can provide users with immediate and meaningful feedback regarding uncertain outputs.

An explicit framework for confidence modeling would benefit the development cycle of neural semantic parsers which, contrary to more traditional methods, do not make use of lexicons or templates and as a result the sources of errors and inconsistencies are difficult to trace. Moreover, from the perspective of application, semantic parsing is often used to build natural language interfaces, such as dialogue systems. In this case it is important to know whether the system understands the input queries with high confidence in order to make decisions more reliably. For example, knowing that some of the predictions are uncertain would allow the system to generate clarification questions, prompting users to verify the results before triggering unwanted actions. In addition, the training data used for semantic parsing can be small and noisy, and as a result, models do indeed produce uncertain outputs, which we would like our framework to identify.

A widely-used confidence scoring method is based on posterior probabilities $p(y|x)$ where x is the input and y the model's prediction. For a linear model, this method makes sense: as more positive evidence is gathered, the score becomes larger. Neural models, in contrast, learn a complicated function that often overfits the training data. Posterior probability is effective when making decisions about model output, but is no longer a good indicator of confidence due in part to the nonlinearity of neural networks (Johansen and Socher, 2017). This observation motivates us to develop a confidence modeling framework for sequence-to-sequence models. We categorize the causes of uncertainty into three types, namely *model uncertainty*, *data uncertainty*, and *input uncertainty* and design different metrics to characterize them.

[*] Work carried out during an internship at Microsoft Research.

We compute these confidence metrics for a given prediction and use them as features in a regression model which is trained on held-out data to fit prediction F1 scores. At test time, the regression model's outputs are used as confidence scores. Our approach does not interfere with the training of the model, and can be thus applied to various architectures, without sacrificing test accuracy. Furthermore, we propose a method based on backpropagation which allows to interpret model behavior by identifying which parts of the input contribute to uncertain predictions.

Experimental results on two semantic parsing datasets (IFTTT, Quirk et al. 2015; and DJANGO, Oda et al. 2015) show that our model is superior to a method based on posterior probability. We also demonstrate that thresholding confidence scores achieves a good trade-off between coverage and accuracy. Moreover, the proposed uncertainty backpropagation method yields results which are qualitatively more interpretable compared to those based on attention scores.

2 Related Work

Confidence Estimation Confidence estimation has been studied in the context of a few NLP tasks, such as statistical machine translation (Blatz et al., 2004; Ueffing and Ney, 2005; Soricut and Echihabi, 2010), and question answering (Gondek et al., 2012). To the best of our knowledge, confidence modeling for semantic parsing remains largely unexplored. A common scheme for modeling uncertainty in neural networks is to place distributions over the network's weights (Denker and Lecun, 1991; MacKay, 1992; Neal, 1996; Blundell et al., 2015; Gan et al., 2017). But the resulting models often contain more parameters, and the training process has to be accordingly changed, which makes these approaches difficult to work with. Gal and Ghahramani (2016) develop a theoretical framework which shows that the use of dropout in neural networks can be interpreted as a Bayesian approximation of Gaussian Process. We adapt their framework so as to represent uncertainty in the encoder-decoder architectures, and extend it by adding Gaussian noise to weights.

Semantic Parsing Various methods have been developed to learn a semantic parser from natural language descriptions paired with meaning representations (Tang and Mooney, 2000; Zettlemoyer and Collins, 2007; Lu et al., 2008; Kwiatkowski et al., 2011; Andreas et al., 2013; Zhao and Huang, 2015). More recently, a few sequence-to-sequence models have been proposed for semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016) and shown to perform competitively whilst eschewing the use of templates or manually designed features. There have been several efforts to improve these models including the use of a tree decoder (Dong and Lapata, 2016), data augmentation (Jia and Liang, 2016; Kočiský et al., 2016), the use of a grammar model (Xiao et al., 2016; Rabinovich et al., 2017; Yin and Neubig, 2017; Krishnamurthy et al., 2017), coarse-to-fine decoding (Dong and Lapata, 2018), network sharing (Susanto and Lu, 2017; Herzig and Berant, 2017), user feedback (Iyer et al., 2017), and transfer learning (Fan et al., 2017). Current semantic parsers will by default generate some output for a given input even if this is just a random guess. System results can thus be somewhat unexpected inadvertently affecting user experience. Our goal is to mitigate these issues with a confidence scoring model that can estimate how likely the prediction is correct.

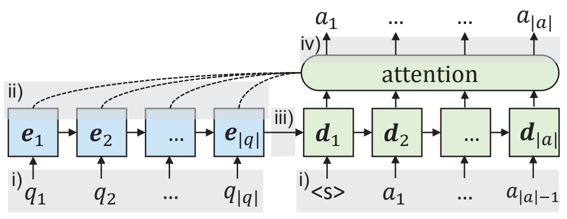
3 Neural Semantic Parsing Model

In the following section we describe the neural semantic parsing model (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016) we assume throughout this paper. The model is built upon the sequence-to-sequence architecture and is illustrated in Figure 1. An *encoder* is used to encode natural language input $q = q_1 \cdots q_{|q|}$ into a vector representation, and a *decoder* learns to generate a logical form representation of its meaning $a = a_1 \cdots a_{|a|}$ conditioned on the encoding vectors. The encoder and decoder are two different recurrent neural networks with long short-term memory units (LSTMs; Hochreiter and Schmidhuber 1997) which process tokens sequentially. The probability of generating the whole sequence $p(a|q)$ is factorized as:

$$p(a|q) = \prod_{t=1}^{|a|} p(a_t|a_{<t}, q) \tag{1}$$

where $a_{<t} = a_1 \cdots a_{t-1}$.

Let $\mathbf{e}_t \in \mathbb{R}^n$ denote the hidden vector of the encoder at time step t. It is computed via $\mathbf{e}_t = \mathrm{f}_{\mathrm{LSTM}}(\mathbf{e}_{t-1}, \mathbf{q}_t)$, where $\mathrm{f}_{\mathrm{LSTM}}$ refers to the LSTM unit, and $\mathbf{q}_t \in \mathbb{R}^n$ is the word embedding



a_1 ... ... $a_{|a|}$
iv)
attention
ii)
e_1 e_2 ... $e_{|q|}$
iii)
d_1 d_2 ... $d_{|a|}$
i) q_1 q_2 ... $q_{|q|}$
i) <s> a_1 ... $a_{|a|-1}$

Figure 1: We use dropout as approximate Bayesian inference to obtain model uncertainty. The dropout layers are applied to i) token vectors; ii) the encoder's output vectors; iii) bridge vectors; and iv) decoding vectors.

of q_t. Once the tokens of the input sequence are encoded into vectors, $\mathbf{e}_{|q|}$ is used to initialize the hidden states of the first time step in the decoder.

Similarly, the hidden vector of the decoder at time step t is computed by $\mathbf{d}_t = \mathrm{f}_{\mathrm{LSTM}}(\mathbf{d}_{t-1}, \mathbf{a}_{t-1})$, where $\mathbf{a}_{t-1} \in \mathbb{R}^n$ is the word vector of the previously predicted token. Additionally, we use an attention mechanism (Luong et al., 2015a) to utilize relevant encoder-side context. For the current time step t of the decoder, we compute its attention score with the k-th hidden state in the encoder as:

$$r_{t,k} \propto \exp\{\mathbf{d}_t \cdot \mathbf{e}_k\} \tag{2}$$

where $\sum_{j=1}^{|q|} r_{t,j} = 1$. The probability of generating a_t is computed via:

$$\mathbf{c}_t = \sum_{k=1}^{|q|} r_{t,k}\mathbf{e}_k \tag{3}$$

$$\mathbf{d}_t^{att} = \tanh(\mathbf{W}_1\mathbf{d}_t + \mathbf{W}_2\mathbf{c}_t) \tag{4}$$

$$p(a_t|a_{<t}, q) = \mathrm{softmax}_{a_t}\left(\mathbf{W}_o\mathbf{d}_t^{att}\right) \tag{5}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{n\times n}$ and $\mathbf{W}_o \in \mathbb{R}^{|V_a|\times n}$ are three parameter matrices.

The training objective is to maximize the likelihood of the generated meaning representation a given input q, i.e., maximize $\sum_{(q,a)\in\mathcal{D}} \log p(a|q)$, where $\mathcal{D}$ represents training pairs. At test time, the model's prediction for input q is obtained via $\hat{a} = \arg\max_{a'} p(a'|q)$, where a' represents candidate outputs. Because $p(a|q)$ is factorized as shown in Equation (1), we can use beam search to generate tokens one by one rather than iterating over all possible results.

4 Confidence Estimation

Given input q and its predicted meaning representation a, the confidence model estimates score $s(q,a) \in (0,1)$. A large score indicates the model is confident that its prediction is correct. In order to gauge confidence, we need to estimate "what we do not know". To this end, we identify three causes of uncertainty, and design various metrics characterizing each one of them. We then feed these metrics into a regression model in order to predict $s(q,a)$.

Algorithm 1 Dropout Perturbation

```
Input: $q, a$: Input and its prediction
       $\mathcal{M}$: Model parameters
1: for $i \leftarrow 1, \cdots, F$ do
2:     $\hat{\mathcal{M}}^i \leftarrow$ Apply dropout layers to $\mathcal{M}$        ▷ Figure 1
3:     Run forward pass and compute $\hat{p}(a|q; \hat{\mathcal{M}}^i)$
4: Compute variance of $\{\hat{p}(a|q; \hat{\mathcal{M}}^i)\}_{i=1}^F$        ▷ Equation (6)
```

4.1 Model Uncertainty

The model's parameters or structures contain uncertainty, which makes the model less confident about the values of $p(a|q)$. For example, noise in the training data and the stochastic learning algorithm itself can result in model uncertainty. We describe metrics for capturing uncertainty below:

Dropout Perturbation Our first metric uses dropout (Srivastava et al., 2014) as approximate Bayesian inference to estimate model uncertainty (Gal and Ghahramani, 2016). Dropout is a widely used regularization technique during training, which relieves overfitting by randomly masking some input neurons to zero according to a Bernoulli distribution. In our work, we use dropout at *test time*, instead. As shown in Algorithm 1, we perform F forward passes through the network, and collect the results $\{\hat{p}(a|q; \hat{\mathcal{M}}^i)\}_{i=1}^F$ where $\hat{\mathcal{M}}^i$ represents the perturbed parameters. Then, the uncertainty metric is computed by the variance of results. We define the metric on the sequence level as:

$$\mathrm{var}\{\hat{p}(a|q; \hat{\mathcal{M}}^i)\}_{i=1}^F. \tag{6}$$

In addition, we compute uncertainty u_{a_t} at the token-level a_t via:

$$u_{a_t} = \mathrm{var}\{\hat{p}(a_t|a_{<t}, q; \hat{\mathcal{M}}^i)\}_{i=1}^F \tag{7}$$

where $\hat{p}(a_t|a_{<t}, q; \hat{\mathcal{M}}^i)$ is the probability of generating token a_t (Equation (5)) using perturbed model $\hat{\mathcal{M}}^i$. We operationalize token-level uncertainty in two ways, as the average score $\mathrm{avg}\{u_{a_t}\}_{t=1}^{|a|}$ and the maximum score

$\max\{u_{a_t}\}_{t=1}^{|a|}$ (since the uncertainty of a sequence is often determined by the most uncertain token). As shown in Figure 1, we add dropout layers in i) the word vectors of the encoder and decoder $\mathbf{q}_t, \mathbf{a}_t$; ii) the output vectors of the encoder $\mathbf{e}_t$; iii) bridge vectors $\mathbf{e}_{|q|}$ used to initialize the hidden states of the first time step in the decoder; and iv) decoding vectors $\mathbf{d}_t^{att}$ (Equation (4)).

Gaussian Noise Standard dropout can be viewed as applying noise sampled from a Bernoulli distribution to the network parameters. We instead use Gaussian noise, and apply the metrics in the same way discussed above. Let $\mathbf{v}$ denote a vector perturbed by noise, and $\mathbf{g}$ a vector sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. We use $\hat{\mathbf{v}} = \mathbf{v} + \mathbf{g}$ and $\hat{\mathbf{v}} = \mathbf{v} + \mathbf{v} \odot \mathbf{g}$ as two noise injection methods. Intuitively, if the model is more confident in an example, it should be more robust to perturbations.

Posterior Probability Our last class of metrics is based on posterior probability. We use the log probability $\log p(a|q)$ as a sequence-level metric. The token-level metric $\min\{p(a_t|a_{<t}, q)\}_{t=1}^{|a|}$ can identify the most uncertain predicted token. The perplexity per token $-\frac{1}{|a|}\sum_{t=1}^{|a|} \log p\left(a_t|a_{<t}, q\right)$ is also employed.

4.2 Data Uncertainty

The coverage of training data also affects the uncertainty of predictions. If the input q does not match the training distribution or contains unknown words, it is difficult to predict $p\left(a|q\right)$ reliably. We define two metrics:

Probability of Input We train a language model on the training data, and use it to estimate the probability of input $p(q|\mathcal{D})$ where $\mathcal{D}$ represents the training data.

Number of Unknown Tokens Tokens that do not appear in the training data harm robustness, and lead to uncertainty. So, we use the number of unknown tokens in the input q as a metric.

4.3 Input Uncertainty

Even if the model can estimate $p\left(a|q\right)$ reliably, the input itself may be ambiguous. For instance, the input *the flight is at 9 o'clock* can be interpreted as either `flight_time(9am)` or `flight_time(9pm)`. Selecting between these predictions is difficult, especially if they are both highly likely. We use the following metrics to measure uncertainty caused by ambiguous inputs.

Variance of Top Candidates We use the variance of the probability of the top candidates to indicate whether these are similar. The sequence-level metric is computed by:

$$\mathrm{var}\{p(a^i|q)\}_{i=1}^{K}$$

where $a^1 \dots a^K$ are the K-best predictions obtained by the beam search during inference (Section 3).

Entropy of Decoding The sequence-level entropy of the decoding process is computed via:

$$H[a|q] = -\sum_{a'} p(a'|q) \log p(a'|q)$$

which we approximate by Monte Carlo sampling rather than iterating over all candidate predictions. The token-level metrics of decoding entropy are computed by $\mathrm{avg}\{H[a_t|a_{<t}, q]\}_{t=1}^{|a|}$ and $\max\{H[a_t|a_{<t}, q]\}_{t=1}^{|a|}$.

4.4 Confidence Scoring

The sentence- and token-level confidence metrics defined in Section 4 are fed into a gradient tree boosting model (Chen and Guestrin, 2016) in order to predict the overall confidence score $s\left(q, a\right)$. The model is wrapped with a logistic function so that confidence scores are in the range of $(0, 1)$.

Because the confidence score indicates whether the prediction is likely to be correct, we can use the prediction's F1 (see Section 6.2) as target value. The training loss is defined as:

$$\sum_{(q,a)\in\mathcal{D}} \ln(1+e^{-\hat{s}(q,a)})^{y_{q,a}} + \ln(1+e^{\hat{s}(q,a)})^{(1-y_{q,a})}$$

where $\mathcal{D}$ represents the data, $y_{q,a}$ is the target F1 score, and $\hat{s}(q, a)$ the predicted confidence score. We refer readers to Chen and Guestrin (2016) for mathematical details of how the gradient tree boosting model is trained. Notice that we learn the confidence scoring model on the held-out set (rather than on the training data of the semantic parser) to avoid overfitting.

5 Uncertainty Interpretation

Confidence scores are useful in so far they can be traced back to the inputs causing the uncertainty in the first place. For semantic parsing, identifying

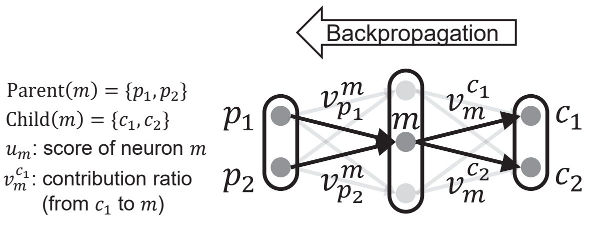

Backpropagation
Parent(m) = {p_1, p_2}
Child(m) = {c_1, c_2}
u_m: score of neuron m
$v_m^{c_1}$: contribution ratio (from c_1 to m)
p_1
p_2
$v_{p_1}^m$
$v_{p_2}^m$
m
$v_m^{c_1}$
$v_m^{c_2}$
c_1
c_2

Figure 2: Uncertainty backpropagation at the neuron level. Neuron m's score u_m is collected from child neurons c_1 and c_2 by $u_m = v_m^{c_1} u_{c_1} + v_m^{c_2} u_{c_2}$. The score u_m is then redistributed to its parent neurons p_1 and p_2, which satisfies $v_{p_1}^m + v_{p_2}^m = 1$.

which input words contribute to uncertainty would be of value, e.g., these could be treated explicitly as special cases or refined if they represent noise.

In this section, we introduce an algorithm that backpropagates token-level uncertainty scores (see Equation (7)) from predictions to input tokens, following the ideas of Bach et al. (2015) and Zhang et al. (2016). Let u_m denote neuron m's uncertainty score, which indicates the degree to which it contributes to uncertainty. As shown in Figure 2, u_m is computed by the summation of the scores backpropagated from its child neurons:

$$u_m = \sum_{c \in \text{Child}(m)} v_m^c u_c$$

where Child(m) is the set of m's child neurons, and the non-negative contribution ratio v_m^c indicates how much we backpropagate u_c to neuron m. Intuitively, if neuron m contributes more to c's value, ratio v_m^c should be larger.

After obtaining score u_m, we redistribute it to its parent neurons in the same way. Contribution ratios from m to its parent neurons are normalized to 1:

$$\sum_{p \in \text{Parent}(m)} v_p^m = 1$$

where Parent(m) is the set of m's parent neurons.

Given the above constraints, we now define different backpropagation rules for the operators used in neural networks. We first describe the rules used for fully-connected layers. Let $\mathbf{x}$ denote the input. The output is computed by $\mathbf{z} = \sigma(\mathbf{W}\mathbf{x}+\mathbf{b})$, where σ is a nonlinear function, $\mathbf{W} \in \mathbb{R}^{|\mathbf{z}|*|\mathbf{x}|}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^{|\mathbf{z}|}$ is the bias, and neuron $\mathbf{z}_i$ is computed via $\mathbf{z}_i = \sigma(\sum_{j=1}^{|\mathbf{x}|} \mathbf{W}_{i,j}\mathbf{x}_j + \mathbf{b}_i)$. Neuron $\mathbf{x}_k$'s uncertainty score u_{x_k} is gathered from the next layer:

Algorithm 2 Uncertainty Interpretation

```
Input: $q, a$: Input and its prediction
Output: $\{\hat{u}_{q_t}\}_{t=1}^{|q|}$: Interpretation scores for input tokens
Function: TokenUnc: Get token-level uncertainty
 1: ▷ Get token-level uncertainty for predicted tokens
 2: $\{u_{a_t}\}_{t=1}^{|a|} \leftarrow$ TokenUnc$(q, a)$
 3: ▷ Initialize uncertainty scores for backpropagation
 4: for $t \leftarrow 1, \cdots, |a|$ do
 5:     Decoder classifier's output neuron $\leftarrow u_{a_t}$
 6: ▷ Run backpropagation
 7: for $m \leftarrow$ neuron in backward topological order do
 8:     ▷ Gather scores from child neurons
 9:     $u_m \leftarrow \sum_{c \in \text{Child}(m)} v_m^c u_c$
10: ▷ Summarize scores for input words
11: for $t \leftarrow 1, \cdots, |q|$ do
12:     $u_{q_t} \leftarrow \sum_{c \in \mathbf{q}_t} u_c$
13: $\{\hat{u}_{q_t}\}_{t=1}^{|q|} \leftarrow$ normalize $\{u_{q_t}\}_{t=1}^{|q|}$
```

$$u_{x_k} = \sum_{i=1}^{|\mathbf{z}|} v_{x_k}^{z_i} u_{z_i} = \sum_{i=1}^{|\mathbf{z}|} \frac{|\mathbf{W}_{i,k}\mathbf{x}_k|}{\sum_{j=1}^{|\mathbf{x}|} |\mathbf{W}_{i,j}\mathbf{x}_j|} u_{z_i}$$

ignoring the nonlinear function σ and the bias $\mathbf{b}$. The ratio $v_{x_k}^{z_i}$ is proportional to the contribution of $\mathbf{x}_k$ to the value of $\mathbf{z}_i$.

We define backpropagation rules for element-wise vector operators. For $\mathbf{z} = \mathbf{x} \pm \mathbf{y}$, these are:

$$u_{x_k} = \frac{|\mathbf{x}_k|}{|\mathbf{x}_k|+|\mathbf{y}_k|} u_{z_k} \quad u_{y_k} = \frac{|\mathbf{y}_k|}{|\mathbf{x}_k|+|\mathbf{y}_k|} u_{z_k}$$

where the contribution ratios $v_{x_k}^{z_k}$ and $v_{y_k}^{z_k}$ are determined by $|\mathbf{x}_k|$ and $|\mathbf{y}_k|$. For multiplication, the contribution of two elements in $\frac{1}{3} * 3$ should be the same. So, the propagation rules for $\mathbf{z} = \mathbf{x} \odot \mathbf{y}$ are:

$$u_{x_k} = \frac{|\log|\mathbf{x}_k||}{|\log|\mathbf{x}_k||+|\log|\mathbf{y}_k||} u_{z_k} \; u_{y_k} = \frac{|\log|\mathbf{y}_k||}{|\log|\mathbf{x}_k||+|\log|\mathbf{y}_k||} u_{z_k}$$

where the contribution ratios are determined by $|\log|\mathbf{x}_k||$ and $|\log|\mathbf{y}_k||$.

For scalar multiplication, $\mathbf{z} = \lambda\mathbf{x}$ where λ denotes a constant. We directly assign $\mathbf{z}$'s uncertainty scores to $\mathbf{x}$ and the backpropagation rule is $u_{x_k} = u_{z_k}$.

As shown in Algorithm 2, we first initialize uncertainty backpropagation in the decoder (lines 1–5). For each predicted token a_t, we compute its uncertainty score u_{a_t} as in Equation (7). Next, we find the dimension of a_t in the decoder's softmax classifier (Equation (5)), and initialize the neuron with the uncertainty score u_{a_t}. We then backpropagate these uncertainty scores through

Dataset	Example
IFTTT	*turn android phone to full volume at 7am monday to friday* `date_time−every_day_of_the_week_at−((time_of_day (07)(:)(00)) (days_of_the_week (1)(2)(3)(4)(5))) THEN android_device−set_ringtone_volume−(volume ({'volume_level':1.0,'name':'100%'}))`
DJANGO	*for every key in sorted list of user_settings* `for key in sorted(user_settings):`

Table 1: Natural language descriptions and their meaning representations from IFTTT and DJANGO.

the network (lines 6–9), and finally into the neurons of the input words. We summarize them and compute the token-level scores for interpreting the results (line 10–13). For input word vector $\mathbf{q}_t$, we use the summation of its neuron-level scores as the token-level score:

$$\hat{u}_{q_t} \propto \sum_{c \in \mathbf{q}_t} u_c$$

where $c \in \mathbf{q}_t$ represents the neurons of word vector $\mathbf{q}_t$, and $\sum_{t=1}^{|q|} \hat{u}_{q_t} = 1$. We use the normalized score $\hat{u}_{q_t}$ to indicate token q_t's contribution to prediction uncertainty.

6 Experiments

In this section we describe the datasets used in our experiments and various details concerning our models. We present our experimental results and analysis of model behavior. Our code is publicly available at https://github.com/donglixp/confidence.

6.1 Datasets

We trained the neural semantic parser introduced in Section 3 on two datasets covering different domains and meaning representations. Examples are shown in Table 1.

IFTTT This dataset (Quirk et al., 2015) contains a large number of if-this-then-that programs crawled from the IFTTT website. The programs are written for various applications, such as home security (e.g., "*email me if the window opens*"), and task automation (e.g., "*save instagram photos to dropbox*"). Whenever a program's trigger is satisfied, an action is performed. Triggers and actions represent functions with arguments; they are selected from different channels (160 in total) representing various services (e.g., Android). There are 552 trigger functions and 229 action functions. The original split contains 77,495 training, 5,171 development, and 4,294 test instances. The subset that removes non-English descriptions was used in our experiments.

DJANGO This dataset (Oda et al., 2015) is built upon the code of the Django web framework. Each line of Python code has a manually annotated natural language description. Our goal is to map the English pseudo-code to Python statements. This dataset contains diverse use cases, such as iteration, exception handling, and string manipulation. The original split has 16,000 training, 1,000 development, and 1,805 test examples.

6.2 Settings

We followed the data preprocessing used in previous work (Dong and Lapata, 2016; Yin and Neubig, 2017). Input sentences were tokenized using NLTK (Bird et al., 2009) and lowercased. We filtered words that appeared less than four times in the training set. Numbers and URLs in IFTTT and quoted strings in DJANGO were replaced with place holders. Hyperparameters of the semantic parsers were validated on the development set. The learning rate and the smoothing constant of RMSProp (Tieleman and Hinton, 2012) were 0.002 and 0.95, respectively. The dropout rate was 0.25. A two-layer LSTM was used for IFTTT, while a one-layer LSTM was employed for DJANGO. Dimensions for the word embedding and hidden vector were selected from $\{150, 250\}$. The beam size during decoding was 5.

For IFTTT, we view the predicted trees as a set of productions, and use balanced F1 as evaluation metric (Quirk et al., 2015). We do not measure accuracy because the dataset is very noisy and there rarely is an exact match between the predicted output and the gold standard. The F1 score of our neural semantic parser is 50.1%, which is comparable to Dong and Lapata (2016). For DJANGO, we measure the fraction of exact matches, where F1 score is equal to accuracy. Because there are unseen variable names at test time, we use attention scores as alignments to replace unknown to-

Method	IFTTT	DJANGO
POSTERIOR	0.477	0.694
CONF	**0.625**	**0.793**
− MODEL	0.595	0.759
− DATA	0.610	0.787
− INPUT	0.608	0.785

Table 2: Spearman ρ correlation between confidence scores and F1. Best results are shown in **bold**. All correlations are significant at $p < 0.01$.

	F1	Dout	Noise	PR	PPL	LM	#UNK	Var
Dout	**0.59**							
Noise	**0.59**	0.90						
PR	**0.52**	0.84	0.82					
PPL	0.48	0.78	0.78	0.89				
LM	0.30	0.26	0.32	0.27	0.25			
#UNK	0.27	0.31	0.33	0.29	0.25	0.32		
Var	0.49	0.83	0.78	0.88	0.79	0.25	0.27	
Ent	0.53	0.78	0.78	0.80	0.75	0.27	0.30	0.76

Table 3: Correlation matrix for F1 and individual confidence metrics on the IFTTT dataset. All correlations are significant at $p < 0.01$. Best predictors are shown in **bold**. Dout is short for dropout, PR for posterior probability, PPL for perplexity, LM for probability based on a language model, #UNK for number of unknown tokens, Var for variance of top candidates, and Ent for Entropy.

	F1	Dout	Noise	PR	PPL	LM	#UNK	Var
Dout	**0.76**							
Noise	**0.78**	0.94						
PR	**0.73**	0.89	0.90					
PPL	0.64	0.80	0.81	0.84				
LM	0.32	0.41	0.40	0.38	0.30			
#UNK	0.27	0.28	0.28	0.26	0.19	0.35		
Var	0.70	0.87	0.87	0.89	0.87	0.37	0.23	
Ent	0.72	0.89	0.90	0.92	0.86	0.38	0.26	0.90

Table 4: Correlation matrix for F1 and individual confidence metrics on the DJANGO dataset. All correlations are significant at $p < 0.01$. Best predictors are shown in **bold**. Same shorthands apply as in Table 3.

kens in the prediction with the input words they align to (Luong et al., 2015b). The accuracy of our parser is 53.7%, which is better than the result (45.1%) of the sequence-to-sequence model reported in Yin and Neubig (2017).

To estimate model uncertainty, we set dropout rate to 0.1, and performed 30 inference passes. The standard deviation of Gaussian noise was 0.05. The language model was estimated using KenLM (Heafield et al., 2013). For input uncertainty, we computed variance for the 10-best candidates. The confidence metrics were implemented in batch mode, to take full advantage of GPUs. Hyperparameters of the confidence scoring model were cross-validated. The number of boosted trees was selected from $\{20, 50\}$. The maximum tree depth was selected from $\{3, 4, 5\}$. We set the subsample ratio to 0.8. All other hyperparameters in XGBoost (Chen and Guestrin, 2016) were left with their default values.

6.3 Results

Confidence Estimation We compare our approach (CONF) against confidence scores based on posterior probability $p(a|q)$ (POSTERIOR). We also report the results of three ablation variants (−MODEL, −DATA, −INPUT) by removing each group of confidence metrics described in Section 4. We measure the relationship between confidence scores and F1 using Spearman's ρ correlation coefficient which varies between -1 and 1 (0 implies there is no correlation). High ρ indicates that the confidence scores are high for correct predictions and low otherwise.

As shown in Table 2, our method CONF outperforms POSTERIOR by a large margin. The ablation results indicate that model uncertainty plays the most important role among the confidence metrics. In contrast, removing the metrics of data uncertainty affects performance less, because most examples in the datasets are in-domain. Improvements for each group of metrics are significant with $p < 0.05$ according to bootstrap hypothesis testing (Efron and Tibshirani, 1994).

Tables 3 and 4 show the correlation matrix for F1 and individual confidence metrics on the IFTTT and DJANGO datasets, respectively. As can be seen, metrics representing model uncertainty and input uncertainty are more correlated to each other compared with metrics capturing data uncertainty. Perhaps unsurprisingly metrics of the same group are highly inter-correlated since they model the same type of uncertainty. Table 5 shows the relative importance of individual metrics in the regression model. As importance score we use the average gain (i.e., loss reduction) brought by the confidence metric once added as feature to the branch of the decision tree (Chen and Guestrin, 2016). The results indicate that model uncertainty (Noise/Dropout/Posterior/Perplexity) plays

Metric	Dout	Noise	PR	PPL	LM	#UNK	Var	Ent
IFTTT	0.39	**1.00**	**0.89**	0.27	0.26	0.46	0.43	0.34
DJANGO	**1.00**	**0.59**	0.22	**0.58**	0.49	0.14	0.24	0.25

Table 5: Importance scores of confidence metrics (normalized by maximum value on each dataset). Best results are shown in **bold**. Same shorthands apply as in Table 3.

the most important role. On IFTTT, the number of unknown tokens (#UNK) and the variance of top candidates (var(K-best)) are also very helpful because this dataset is relatively noisy and contains many ambiguous inputs.

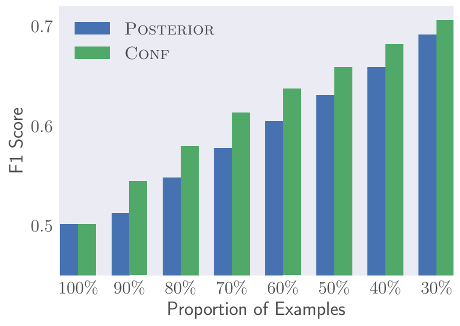
Finally, in real-world applications, confidence scores are often used as a threshold to trade-off precision for coverage. Figure 3 shows how F1 score varies as we increase the confidence threshold, i.e., reduce the proportion of examples that we return answers for. F1 score improves monotonically for POSTERIOR and our method, which, however, achieves better performance when coverage is the same.

Uncertainty Interpretation We next evaluate how our backpropagation method (see Section 5) allows us to identify input tokens contributing to uncertainty. We compare against a method that interprets uncertainty based on the attention mechanism (ATTENTION). As shown in Equation (2), attention scores $r_{t,k}$ can be used as soft alignments between the time step t of the decoder and the k-th input token. We compute the normalized uncertainty score $\hat{u}_{q_t}$ for a token q_t via:

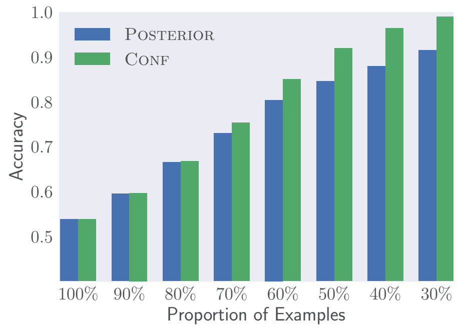
$$\hat{u}_{q_t} \propto \sum_{t=1}^{|a|} r_{t,k} u_{a_t} \tag{8}$$

where u_{a_t} is the uncertainty score of the predicted token a_t (Equation (7)), and $\sum_{t=1}^{|q|} \hat{u}_{q_t} = 1$.

Unfortunately, the evaluation of uncertainty interpretation methods is problematic. For our semantic parsing task, we do not a priori know which tokens in the natural language input contribute to uncertainty and these may vary depending on the architecture used, model parameters, and so on. We work around this problem by creating a proxy gold standard. We inject noise to the vectors representing tokens in the encoder (see Section 4.1) and then estimate the uncertainty caused by each token q_t (Equation (6)) under the assumption that



(a) IFTTT



(b) DJANGO

Figure 3: Confidence scores are used as threshold to filter out uncertain test examples. As the threshold increases, performance improves. The horizontal axis shows the proportion of examples beyond the threshold.

addition of noise should only affect genuinely uncertain tokens. Notice that here we inject noise to one token at a time[1] instead of all parameters (see Figure 1). Tokens identified as uncertain by the above procedure are considered gold standard and compared to those identified by our method. We use Gaussian noise to perturb vectors in our experiments (dropout obtained similar results).

We define an evaluation metric based on the overlap ($overlap@K$) among tokens identified as uncertain by the model and the gold standard. Given an example, we first compute the interpretation scores of the input tokens according to our method, and obtain a list τ_1 of K tokens with highest scores. We also obtain a list τ_2 of K tokens with highest ground-truth scores and measure the degree of overlap between these two lists:

$$\text{overlap}@K = \frac{|\tau_1 \cap \tau_2|}{K}$$

[1]Noise injection as described above is used for evaluation purposes only since we need to perform forward passes multiple times (see Section 4.1) for each token, and the running time increases linearly with the input length.

Method	IFTTT		DJANGO	
	@2	@4	@2	@4
ATTENTION	0.525	0.737	0.637	0.684
BACKPROP	**0.608**	**0.791**	**0.770**	**0.788**

Table 6: Uncertainty interpretation against inferred ground truth; we compute the overlap between tokens identified as contributing to uncertainty by our method and those found in the gold standard. Overlap is shown for top 2 and 4 tokens. Best results are in **bold**.

	google_calendar−any_event_starts THEN facebook −create_a_status_message−(status_message ({description}))
ATT	post calendar event to facebook
BP	post calendar event to facebook
	feed−new_feed_item−(feed_url(_url_sports.espn.go.com)) THEN ...
ATT	espn mlb headline to readability
BP	espn mlb headline to readability
	weather−tomorrow's_low_drops_below−((temperature(0)) (degrees_in(c))) THEN ...
ATT	warn me when it's going to be freezing tomorrow
BP	warn me when it's going to be freezing tomorrow
	if str_number[0] == '_STR_':
ATT	if first element of str_number equals a string _STR_ .
BP	if first element of str_number equals a string _STR_ .
	start = 0
ATT	start is an integer 0 .
BP	start is an integer 0 .
	if name.startswith('_STR_'):
ATT	if name starts with an string _STR_ ,
BP	if name starts with an string _STR_ ,

Table 7: Uncertainty interpretation for ATTENTION (ATT) and BACKPROP (BP) . The first line in each group is the model prediction. Predicted tokens and input words with large scores are shown in red and blue, respectively.

where $K \in \{2, 4\}$ in our experiments. For example, the overlap@4 metric of the lists $\tau_1 = [q_7, q_8, q_2, q_3]$ and $\tau_2 = [q_7, q_8, q_3, q_4]$ is $3/4$, because there are three overlapping tokens.

Table 6 reports results with overlap@2 and overlap@4. Overall, BACKPROP achieves better interpretation quality than the attention mechanism. On both datasets, about 80% of the top-4 tokens identified as uncertain agree with the ground truth. Table 7 shows examples where our method has identified input tokens contributing to the uncertainty of the output. We highlight token a_t if its uncertainty score u_{a_t} is greater than $0.5 * \text{avg}\{u_{a_{t'}}\}_{t'=1}^{|a|}$. The results illustrate that the parser tends to be uncertain about tokens which are function arguments (e.g., URLs, and message content), and ambiguous inputs. The examples show that BACKPROP is qualitatively better compared to ATTENTION; attention scores often produce inaccurate alignments while BACKPROP can utilize information flowing through the LSTMs rather than only relying on the attention mechanism.

7 Conclusions

In this paper we presented a confidence estimation model and an uncertainty interpretation method for neural semantic parsing. Experimental results show that our method achieves better performance than competitive baselines on two datasets. Directions for future work are many and varied. The proposed framework could be applied to a variety of tasks (Bahdanau et al., 2015; Schmaltz et al., 2017) employing sequence-to-sequence architectures. We could also utilize the confidence estimation model within an active learning framework for neural semantic parsing.

Acknowledgments

We would like to thank Pengcheng Yin for sharing with us the preprocessed version of the DJANGO dataset. We gratefully acknowledge the financial support of the European Research Council (award number 681760; Dong, Lapata) and the Adept-Mind Scholar Fellowship program (Dong).

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 47–52, Sofia, Bulgaria.

Sebastian Bach, Alexander Binder, Grgoire Montavon, Frederick Klauschen, Klaus-Robert Mller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto

Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 315–321, Geneva, Switzerland.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pages 1613–1622, Lille, France.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco, California.

John S Denker and Yann Lecun. 1991. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859, Denver, Colorado.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 33–43, Berlin, Germany.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Bradley Efron and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC press.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 48–56, Vancouver, Canada.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059, New York City, NY.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. 2017. Scalable bayesian learning of recurrent neural networks for language modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 321–331, Vancouver, Canada.

D. C. Gondek, A. Lally, A. Kalyanpur, J. W. Murdock, P. A. Duboue, L. Zhang, Y. Pan, Z. M. Qiu, and C. Welty. 2012. A framework for merging and ranking of answers in DeepQA. *IBM Journal of Research and Development*, 56(3.4):14:1–14:12.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 623–628, Vancouver, Canada.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 963–973, Vancouver, Canada.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 12–22, Berlin, Germany.

Alexander Johansen and Richard Socher. 2017. Learning when to skim and when to read. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 257–264, Vancouver, Canada.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1517–1527, Copenhagen, Denmark.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 599–609, Berlin, Germany.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 11–19, Beijing, China.

David J. C. MacKay. 1992. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

Radford M Neal. 1996. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation. In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering*, pages 574–584, Washington, DC.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 878–888, Beijing, China.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1139–1149, Vancouver, Canada.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2797–2803, Copenhagen, Denmark.

Radu Soricut and Abdessamad Echihabi. 2010. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621, Uppsala, Sweden.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Raymond Hendy Susanto and Wei Lu. 2017. Neural architectures for multilingual semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 38–44, Vancouver, Canada.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, Montreal, Canada.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, Hong Kong, China.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Nicola Ueffing and Hermann Ney. 2005. Word-level confidence estimation for machine translation using phrase-based translation models. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 763–770, Vancouver, Canada.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1341–1350, Berlin, Germany.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 440–450, Vancouver, Canada.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 678–687, Prague, Czech Republic.

Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. 2016. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559, Amsterdam, Netherlands.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421, Denver, Colorado.

STRUCTVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing

Pengcheng Yin, Chunting Zhou, Junxian He, Graham Neubig
Language Technologies Institute
Carnegie Mellon University
{pcyin,ctzhou,junxianh,gneubig}@cs.cmu.edu

Abstract

Semantic parsing is the task of transducing natural language (NL) utterances into formal meaning representations (MRs), commonly represented as tree structures. Annotating NL utterances with their corresponding MRs is expensive and time-consuming, and thus the limited availability of labeled data often becomes the bottleneck of data-driven, supervised models. We introduce STRUCTVAE, a variational auto-encoding model for semi-supervised semantic parsing, which learns both from limited amounts of parallel data, and readily-available unlabeled NL utterances. STRUCTVAE models latent MRs not observed in the unlabeled data as tree-structured latent variables. Experiments on semantic parsing on the ATIS domain and Python code generation show that with extra unlabeled data, STRUCTVAE outperforms strong supervised models.[1]

1 Introduction

Semantic parsing tackles the task of mapping natural language (NL) utterances into structured formal meaning representations (MRs). This includes parsing to general-purpose logical forms such as λ-calculus (Zettlemoyer and Collins, 2005, 2007) and the abstract meaning representation (AMR, Banarescu et al. (2013); Misra and Artzi (2016)), as well as parsing to computer-executable programs to solve problems such as question answering (Berant et al., 2013; Yih et al., 2015; Liang et al., 2017), or generation of domain-specific (*e.g.*, SQL) or general purpose programming languages (*e.g.*, Python) (Quirk et al., 2015; Yin and Neubig, 2017; Rabinovich et al., 2017).

[1]Code available at `http://pcyin.me/struct_vae`

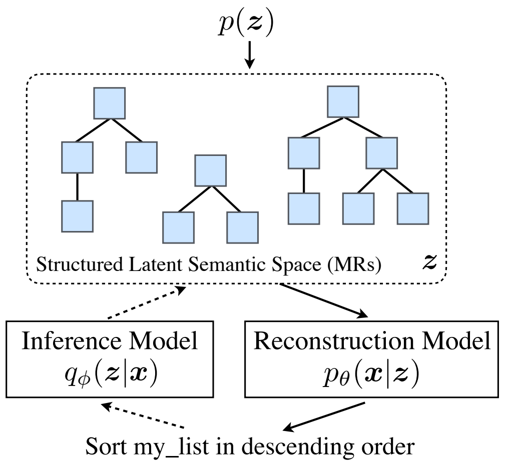

$p(\boldsymbol{z})$
Structured Latent Semantic Space (MRs)
$\boldsymbol{z}$
Inference Model
$q_\phi(\boldsymbol{z}|\boldsymbol{x})$
Reconstruction Model
$p_\theta(\boldsymbol{x}|\boldsymbol{z})$
Sort my_list in descending order

Figure 1: Graphical Representation of STRUCTVAE

While these models have a long history (Zelle and Mooney, 1996; Tang and Mooney, 2001), recent advances are largely attributed to the success of neural network models (Xiao et al., 2016; Ling et al., 2016; Dong and Lapata, 2016; Iyer et al., 2017; Zhong et al., 2017). However, these models are also extremely *data hungry*: optimization of such models requires large amounts of training data of parallel NL utterances and manually annotated MRs, the creation of which can be expensive, cumbersome, and time-consuming. Therefore, the limited availability of parallel data has become the bottleneck of existing, purely supervised-based models. These data requirements can be alleviated with *weakly-supervised* learning, where the denotations (*e.g.*, answers in question answering) of MRs (*e.g.*, logical form queries) are used as indirect supervision (Clarke et al. (2010); Liang et al. (2011); Berant et al. (2013), *inter alia*), or *data-augmentation techniques* that automatically generate pseudo-parallel corpora using hand-crafted or induced grammars (Jia and Liang, 2016; Wang et al., 2015).

In this work, we focus on *semi-supervised* learning, aiming to learn from both limited amounts of parallel NL-MR corpora, and *unlabeled* but readily-available NL utterances. We draw inspiration from recent success in applying variational auto-encoding (VAE) models in semi-supervised sequence-to-sequence learning (Miao and Blunsom, 2016; Kociský et al., 2016), and propose STRUCTVAE — a principled deep generative approach for semi-supervised learning with tree-structured latent variables (Fig. 1). STRUCTVAE is based on a generative story where the surface NL utterances are generated from tree-structured latent MRs following the standard VAE architecture: (1) an off-the-shelf semantic parser functions as the *inference model*, parsing an observed NL utterance into latent meaning representations (§ 3.2); (2) a *reconstruction model* decodes the latent MR into the original observed utterance (§ 3.1). This formulation enables our model to perform both standard supervised learning by optimizing the inference model (*i.e.*, the parser) using parallel corpora, and unsupervised learning by maximizing the variational lower bound of the likelihood of the unlabeled utterances (§ 3.3).

In addition to these contributions to semi-supervised semantic parsing, STRUCTVAE contributes to generative model research as a whole, providing a recipe for training VAEs with *structured* latent variables. Such a structural latent space is contrast to existing VAE research using *flat* representations, such as continuous distributed representations (Kingma and Welling, 2013), discrete symbols (Miao and Blunsom, 2016), or hybrids of the two (Zhou and Neubig, 2017).

We apply STRUCTVAE to semantic parsing on the ATIS domain and Python code generation. As an auxiliary contribution, we implement a transition-based semantic parser, which uses Abstract Syntax Trees (ASTs, § 3.2) as intermediate MRs and achieves strong results on the two tasks. We then apply this parser as the inference model for semi-supervised learning, and show that with extra unlabeled data, STRUCTVAE outperforms its supervised counterpart. We also demonstrate that STRUCTVAE is compatible with different structured latent representations, applying it to a simple sequence-to-sequence parser which uses λ-calculus logical forms as MRs.

2 Semi-supervised Semantic Parsing

In this section we introduce the objectives for semi-supervised semantic parsing, and present high-level intuition in applying VAEs for this task.

2.1 Supervised and Semi-supervised Training

Formally, semantic parsing is the task of mapping utterance $\boldsymbol{x}$ to a meaning representation $\boldsymbol{z}$. As noted above, there are many varieties of MRs that can be represented as either graph structures (*e.g.*, AMR) or tree structures (*e.g.*, λ-calculus and ASTs for programming languages). In this work we specifically focus on tree-structured MRs (see Fig. 2 for a running example Python AST), although application of a similar framework to graph-structured representations is also feasible.

Traditionally, purely supervised semantic parsers train a probabilistic model $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ using parallel data $\mathbb{L}$ of NL utterances and annotated MRs (*i.e.*, $\mathbb{L} = \{\langle \boldsymbol{x}, \boldsymbol{z} \rangle\}$). As noted in the introduction, one major bottleneck in this approach is the lack of such parallel data. Hence, we turn to semi-supervised learning, where the model additionally has access to a relatively large amount of unlabeled NL utterances $\mathbb{U} = \{\boldsymbol{x}\}$. Semi-supervised learning then aims to maximize the log-likelihood of examples in both $\mathbb{L}$ and $\mathbb{U}$:

$$\mathcal{J} = \underbrace{\sum_{\langle \boldsymbol{x}, \boldsymbol{z} \rangle \in \mathbb{L}} \log p_\phi(\boldsymbol{z}|\boldsymbol{x})}_{\text{supervised obj. } \mathcal{J}_s} + \alpha \underbrace{\sum_{\boldsymbol{x} \in \mathbb{U}} \log p(\boldsymbol{x})}_{\text{unsupervised obj. } \mathcal{J}_u} \quad (1)$$

The joint objective consists of two terms: (1) a supervised objective $\mathcal{J}_s$ that maximizes the conditional likelihood of annotated MRs, as in standard supervised training of semantic parsers; and (2) a unsupervised objective $\mathcal{J}_u$, which maximizes the marginal likelihood $p(\boldsymbol{x})$ of unlabeled NL utterances $\mathbb{U}$, controlled by a tuning parameter α. Intuitively, if the modeling of $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{x})$ is coupled (*e.g.*, they share parameters), then optimizing the marginal likelihood $p(\boldsymbol{x})$ using the unsupervised objective $\mathcal{J}_u$ would help the learning of the semantic parser $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ (Zhu, 2005). STRUCTVAE uses the variational auto-encoding framework to jointly optimize $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{x})$, as outlined in § 2.2 and detailed in § 3.

2.2 VAEs for Semi-supervised Learning

From Eq. (1), our semi-supervised model must be able to calculate the probability $p(\boldsymbol{x})$ of unlabeled NL utterances. To model $p(\boldsymbol{x})$, we use VAEs, which provide a principled framework for generative models using neural networks (Kingma and Welling, 2013). As shown in Fig. 1, VAEs define a *generative story* (bold arrows in Fig. 1, explained in § 3.1) to model $p(\boldsymbol{x})$, where a latent MR $\boldsymbol{z}$ is sampled from a prior, and then passed to the *reconstruction* model to decode into the surface utterance $\boldsymbol{x}$. There is also an *inference* model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ that allows us to infer the most probable latent MR $\boldsymbol{z}$ given the input $\boldsymbol{x}$ (dashed arrows in Fig. 1, explained in § 3.2). In our case, the inference process is equivalent to the task of semantic parsing if we set $q_\phi(\cdot) \triangleq p_\phi(\cdot)$. VAEs also provide a framework to compute an approximation of $p(\boldsymbol{x})$ using the inference and reconstruction models, allowing us to effectively optimize the unsupervised and supervised objectives in Eq. (1) in a joint fashion (Kingma et al. (2014), explained in § 3.3).

3 StructVAE: VAEs with Tree-structured Latent Variables

3.1 Generative Story

StructVAE follows the standard VAE architecture, and defines a generative story that explains how an NL utterance is generated: a latent meaning representation $\boldsymbol{z}$ is sampled from a prior distribution $p(\boldsymbol{z})$ over MRs, which encodes the latent semantics of the utterance. A *reconstruction* model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ then decodes the sampled MR $\boldsymbol{z}$ into the observed NL utterance $\boldsymbol{x}$.

Both the prior $p(\boldsymbol{z})$ and the reconstruction model $p(\boldsymbol{x}|\boldsymbol{z})$ takes tree-structured MRs as inputs. To model such inputs with rich internal structures, we follow Konstas et al. (2017), and model the distribution over a sequential surface representation of $\boldsymbol{z}$, $\boldsymbol{z}^s$ instead. Specifically, we have $p(\boldsymbol{z}) \triangleq p(\boldsymbol{z}^s)$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z}) \triangleq p_\theta(\boldsymbol{x}|\boldsymbol{z}^s)$[2]. For code generation, $\boldsymbol{z}^s$ is simply the surface source code of the AST $\boldsymbol{z}$. For semantic parsing, $\boldsymbol{z}^s$ is the linearized s-expression of the logical form. Linearization allows us to use standard sequence-to-sequence networks to model $p(\boldsymbol{z})$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. As we will explain in § 4.3, we find these two components perform well with linearization.

Specifically, the prior is parameterized by a Long Short-Term Memory (LSTM) language model over $\boldsymbol{z}^s$. The reconstruction model is an attentional sequence-to-sequence network (Luong et al., 2015), augmented with a copying mechanism (Gu et al., 2016), allowing an out-of-vocabulary (OOV) entity in $\boldsymbol{z}^s$ to be copied to $\boldsymbol{x}$ (*e.g.*, the variable name `my_list` in Fig. 1 and its AST in Fig. 2). We refer readers to Appendix B for details of the neural network architecture.

[2]Linearizion is used by the prior and the reconstruction model only, and not by the inference model.

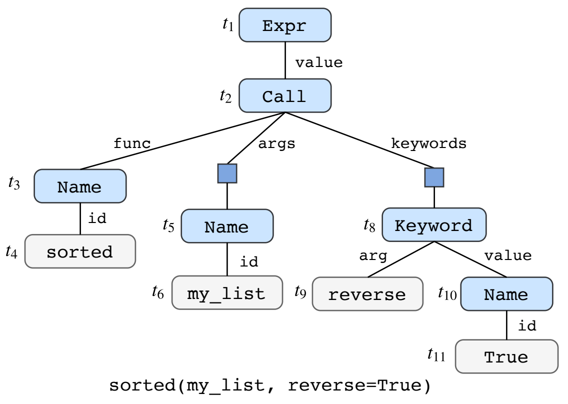
3.2 Inference Model

StructVAE models the semantic parser $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ as the inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ in VAE (§ 2.2), which maps NL utterances $\boldsymbol{x}$ into tree-structured meaning representations $\boldsymbol{z}$. $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ can be any trainable semantic parser, with the corresponding MRs forming the structured latent semantic space. In this work, we primarily use a semantic parser based on the Abstract Syntax Description Language (ASDL) framework (Wang et al., 1997) as the inference model. The parser encodes $\boldsymbol{x}$ into ASTs (Fig. 2). ASTs are the native meaning representation scheme of source code in modern programming languages, and can also be adapted to represent other semantic structures, like λ-calculus logical forms (see § 4.2 for details). We remark that StructVAE works with other semantic parsers with different meaning representations as well (*e.g.*, using λ-calculus logical forms for semantic parsing on Atis, explained in § 4.3).

Our inference model is a transition-based parser inspired by recent work in neural semantic parsing and code generation. The transition system is an adaptation of Yin and Neubig (2017) (hereafter YN17), which decomposes the generation process of an AST into sequential applications of tree-construction actions following the ASDL grammar, thus ensuring the syntactic well-formedness of generated ASTs. Different from YN17, where ASTs are represented as a Context Free Grammar learned from a parsed corpus, we follow Rabinovich et al. (2017) and use ASTs defined under the ASDL formalism (§ 3.2.1).

3.2.1 Generating ASTs with ASDL Grammar

First, we present a brief introduction to ASDL. An AST can be generated by applying typed *constructors* in an ASDL grammar, such as those in Fig. 3 for the Python ASDL grammar. Each constructor specifies a language construct, and is assigned to a particular *composite type*. For example, the constructor `Call` has type `expr` (expression), and it denotes function calls. Constructors are associated with multiple *fields*. For instance, the `Call` constructor and has three fields: *func*, *args* and *keywords*. Like constructors, fields are also strongly typed. For example, the *func* field of `Call` has `expr` type. Fields with composite types are instantiated by constructors of the same type, while fields with *primitive* types store values (*e.g.*, identifier names or string literals). Each field also has



t	Frontier Field	Action
t_1	`stmt root`	`Expr(expr value)`
t_2	`expr value`	`Call(expr func, expr* args, keyword* keywords)`
t_3	`expr func`	`Name(identifier id)`
t_4	`identifier id`	GENTOKEN[*sorted*]
t_5	`expr* args`	`Name(identifier id)`
t_6	`identifier id`	GENTOKEN[*my_list*]
t_7	`expr* args`	REDUCE (close the frontier field)
t_8	`keyword* keywords`	`keyword(identifier arg, expr value)`
t_9	`identifier arg`	GENTOKEN[*reverse*]
t_{10}	`expr value`	`Name(identifier id)`
t_{11}	`identifier id`	GENTOKEN[*True*]
t_{12}	`keyword* keywords`	REDUCE (close the frontier field)

Figure 2: ***Left*** An example ASDL AST with its surface source code. Field names are labeled on upper arcs. Blue squares denote fields with *sequential* cardinality. Grey nodes denote primitive identifier fields, with annotated values. Fields are labeled with time steps at which they are generated. ***Right*** Action sequences used to construct the example AST. Frontier fields are denoted by their signature `(type name)`. Each constructor in the Action column refers to an APPLYCONSTR action.

```
stmt  ↦ FunctionDef(identifier name,
                             arguments args, stmt* body)
      | ClassDef(identifier name, expr* bases, stmt* body)
      | Expr(expr value)
      | Return(expr? value)

expr  ↦ Call(expr func, expr* args, keyword* keywords)
      | Name(identifier id)
      | Str(string s)
```

Figure 3: Excerpt of the python abstract syntax grammar (Python Software Foundation, 2016)

a cardinality (single, optional ?, and sequential $*$), specifying the number of values the field has.

Each node in an AST corresponds to a typed field in a constructor (except for the root node). Depending on the cardinality of the field, an AST node can be instantiated with one or multiple constructors. For instance, the *func* field in the example AST has single cardinality, and is instantiated with a `Name` constructor; while the *args* field with sequential cardinality could have multiple constructors (only one shown in this example).

Our parser employs a transition system to generate an AST using three types of actions. Fig. 2 (Right) lists the sequence of actions used to generate the example AST. The generation process starts from an initial derivation with only a root node of type `stmt` (statement), and proceeds according to the top-down, left-to-right traversal of the AST. At each time step, the parser applies an action to the *frontier field* of the derivation:

APPLYCONSTR$[c]$ actions apply a constructor c to the frontier composite field, expanding the derivation using the fields of c. For fields with single or optional cardinality, an APPLYCONSTR action instantiates the empty frontier field using the constructor, while for fields with sequential cardinality, it appends the constructor to the frontier field. For example, at t_2 the `Call` constructor is applied to the *value* field of `Expr`, and the derivation is expanded using its three child fields.

REDUCE actions complete generation of a field with optional or multiple cardinalities. For instance, the *args* field is instantiated by `Name` at t_5, and then closed by a REDUCE action at t_7.

GENTOKEN$[v]$ actions populate an empty primitive frontier field with token v. A primitive field whose value is a single token (*e.g.*, identifier fields) can be populated with a single GENTOKEN action. Fields of `string` type can be instantiated using multiple such actions, with a final GENTOKEN[`</f>`] action to terminate the generation of field values.

3.2.2 Modeling $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

The probability of generating an AST $\boldsymbol{z}$ is naturally decomposed into the probabilities of the actions $\{a_t\}$ used to construct $\boldsymbol{z}$:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \prod_t p(a_t|a_{<t}, \boldsymbol{x}).$$

Following YN17, we parameterize $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ using a sequence-to-sequence network with auxiliary recurrent connections following the topology of the AST. Interested readers are referred to Appendix B and Yin and Neubig (2017) for details of the neural network architecture.

3.3 Semi-supervised Learning

In this section we explain how to optimize the semi-supervised learning objective Eq. (1) in STRUCTVAE.

Supervised Learning For the supervised learning objective, we modify $\mathcal{J}_s$, and use the labeled data to optimize both the inference model (the semantic parser) and the reconstruction model:

$$\mathcal{J}_s \triangleq \sum_{(\boldsymbol{x},\boldsymbol{z})\in\mathbb{L}} \big(\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) + \log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big) \quad (2)$$

Unsupervised Learning To optimize the unsupervised learning objective $\mathcal{J}_u$ in Eq. (1), we maximize the variational lower-bound of $\log p(\boldsymbol{x})$:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big(\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big) - \lambda \cdot \mathrm{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})] = \mathcal{L} \quad (3)$$

where $\mathrm{KL}[q_\phi||p]$ is the Kullback-Leibler (KL) divergence. Following common practice in optimizing VAEs, we introduce λ as a tuning parameter of the KL divergence to control the impact of the prior (Miao and Blunsom, 2016; Bowman et al., 2016).

To optimize the parameters of our model in the face of non-differentiable discrete latent variables, we follow Miao and Blunsom (2016), and approximate $\frac{\partial\mathcal{L}}{\partial\phi}$ using the score function estimator (a.k.a. REINFORCE, Williams (1992)):

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\phi} &= \frac{\partial}{\partial\phi}\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \\ &\quad \underbrace{\Big(\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) - \lambda\big(\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{z})\big)\Big)}_{\text{learning signal}} \\ &= \frac{\partial}{\partial\phi}\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} l'(\boldsymbol{x},\boldsymbol{z}) \\ &\approx \frac{1}{|\mathcal{S}(\boldsymbol{x})|}\sum_{\boldsymbol{z}_i\in\mathcal{S}(\boldsymbol{x})} l'(\boldsymbol{x},\boldsymbol{z}_i)\frac{\partial \log q_\phi(\boldsymbol{z}_i|\boldsymbol{x})}{\partial\phi} \qquad (4)\end{aligned}$$

where we approximate the gradient using a set of samples $\mathcal{S}(\boldsymbol{x})$ drawn from $q_\phi(\cdot|\boldsymbol{x})$. To ensure the quality of sampled latent MRs, we follow Guu et al. (2017) and use beam search. The term $l'(\boldsymbol{x},\boldsymbol{z})$ is defined as the *learning signal* (Miao and Blunsom, 2016). The learning signal weights the gradient for each latent sample $\boldsymbol{z}$. In REINFORCE, to cope with the high variance of the learning signal, it is common to use a baseline $b(\boldsymbol{x})$ to stabilize learning, and re-define the learning signal as

$$l(\boldsymbol{x},\boldsymbol{z}) \triangleq l'(\boldsymbol{x},\boldsymbol{z}) - b(\boldsymbol{x}). \quad (5)$$

Specifically, in STRUCTVAE, we define

$$b(\boldsymbol{x}) = a \cdot \log p(\boldsymbol{x}) + c, \quad (6)$$

where $\log p(\boldsymbol{x})$ is a pre-trained LSTM language model. This is motivated by the empirical observation that $\log p(\boldsymbol{x})$ correlates well with the reconstruction score $\log p_\theta(\boldsymbol{x}|\boldsymbol{z})$, hence with $l'(\boldsymbol{x},\boldsymbol{z})$.

Finally, for the reconstruction model, its gradient can be easily computed:

$$\frac{\partial\mathcal{L}}{\partial\theta} \approx \frac{1}{|\mathcal{S}(\boldsymbol{x})|}\sum_{\boldsymbol{z}_i\in\mathcal{S}(\boldsymbol{x})}\frac{\partial \log p_\theta(\boldsymbol{x}|\boldsymbol{z}_i)}{\partial\theta}.$$

Discussion Perhaps the most intriguing question here is why semi-supervised learning could improve semantic parsing performance. While the underlying theoretical exposition still remains an active research problem (Singh et al., 2008), in this paper we try to empirically test some likely hypotheses. In Eq. (4), the gradient received by the inference model from each latent sample $\boldsymbol{z}$ is weighed by the learning signal $l(\boldsymbol{x},\boldsymbol{z})$. $l(\boldsymbol{x},\boldsymbol{z})$ can be viewed as the reward function in REINFORCE learning. It can also be viewed as weights associated with pseudo-training examples $\{\langle\boldsymbol{x},\boldsymbol{z}\rangle : \boldsymbol{z} \in \mathcal{S}(\boldsymbol{x})\}$ sampled from the inference model. Intuitively, a sample $\boldsymbol{z}$ with higher rewards should: (1) have $\boldsymbol{z}$ adequately encode the input, leading to high reconstruction score $\log p_\theta(\boldsymbol{x}|\boldsymbol{z})$; and (2) have $\boldsymbol{z}$ be succinct and natural, yielding high prior probability. Let $\boldsymbol{z}^*$ denote the gold-standard MR of $\boldsymbol{x}$. Consider the ideal case where $\boldsymbol{z}^* \in \mathcal{S}(\boldsymbol{x})$ and $l(\boldsymbol{x},\boldsymbol{z}^*)$ is positive, while $l(\boldsymbol{x},\boldsymbol{z}')$ is negative for other imperfect samples $\boldsymbol{z}' \in \mathcal{S}(\boldsymbol{x})$, $\boldsymbol{z}' \neq \boldsymbol{z}^*$. In this ideal case, $\langle\boldsymbol{x},\boldsymbol{z}^*\rangle$ would serve as a positive training example and other samples $\langle\boldsymbol{x},\boldsymbol{z}'\rangle$ would be treated as negative examples. Therefore, the inference model would receive informative gradient updates, and learn to discriminate between gold and imperfect MRs. This intuition is similar in spirit to recent efforts in interpreting gradient update rules in reinforcement learning (Guu et al., 2017). We will present more empirical statistics and observations in § 4.3.

4 Experiments

4.1 Datasets

In our semi-supervised semantic parsing experiments, it is of interest how STRUCTVAE could further improve upon a supervised parser with extra unlabeled data. We evaluate on two datasets:

Semantic Parsing We use the ATIS dataset, a collection of 5,410 telephone inquiries of flight booking (*e.g.*, *"Show me flights from ci0 to ci1"*). The target MRs are defined using λ-calculus logical forms (*e.g.*, "`lambda $0 e (and (flight $0) (from $ci0) (to $ci1))`"). We use the pre-processed dataset released by Dong and Lapata (2016), where entities (*e.g.*, cities) are canonicalized using typed slots (*e.g.*, `ci0`). To predict λ-

calculus logical forms using our transition-based parser, we use the ASDL grammar defined by Rabinovich et al. (2017) to convert between logical forms and ASTs (see Appendix C for details).

Code Generation The DJANGO dataset (Oda et al., 2015) contains 18,805 lines of Python source code extracted from the Django web framework. Each line of code is annotated with an NL utterance. Source code in the DJANGO dataset exhibits a wide variety of real-world use cases of Python, including IO operation, data structure manipulation, class/function definition, *etc*. We use the pre-processed version released by Yin and Neubig (2017) and use the `astor` package to convert ASDL ASTs into Python source code.

4.2 Setup

Labeled and Unlabeled Data STRUCTVAE requires access to extra unlabeled NL utterances for semi-supervised learning. However, the datasets we use do not accompany with such data. We therefore simulate the semi-supervised learning scenario by randomly sub-sampling K examples from the training split of each dataset as the labeled set $\mathbb{L}$. To make the most use of the NL utterances in the dataset, we construct the unlabeled set $\mathbb{U}$ using all NL utterances in the training set[3,4].

Training Procedure Optimizing the unsupervised learning objective Eq. (3) requires sampling structured MRs from the inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. Due to the complexity of the semantic parsing problem, we cannot expect any valid samples from randomly initialized $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. We therefore pre-train the inference and reconstruction models using the supervised objective Eq. (2) until convergence, and then optimize using the semi-supervised learning objective Eq. (1). Throughout all experiments we set α (Eq. (1)) and λ (Eq. (3)) to 0.1. The sample size $|\mathcal{S}(\boldsymbol{x})|$ is 5. We observe that the variance of the learning signal could still be high when low-quality samples are drawn from the inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. We therefore clip all learning signals lower than $k = -20.0$. Early-stopping is used to avoid over-fitting. We also pre-train the prior $p(\boldsymbol{z})$ (§ 3.3) and the baseline function Eq. (6). Readers are referred to Appendix D for more detail of the configurations.

[3]We also tried constructing $\mathbb{U}$ using the disjoint portion of the NL utterances not presented in the labeled set $\mathbb{L}$, but found this yields slightly worse performance, probably due to lacking enough unlabeled data. Interpreting these results would be an interesting avenue for future work.

[4]While it might be relatively easy to acquire additional unlabeled utterances in practical settings (*e.g.*, through query logs of a search engine), unfortunately most academic semantic parsing datasets, like the ones used in this work, do not feature large sets of in-domain unlabeled data. We therefore perform simulated experiments instead.

$\vert\mathbb{L}\vert$	SUP.	SELFTRAIN	STRUCTVAE
500	63.2	65.3	**66.0**
1,000	74.6	74.2	**75.7**
2,000	80.4	**83.3**	82.4
3,000	82.8	**83.6**	**83.6**
4,434 (All)	**85.3**	–	84.5

Previous Methods	ACC.
ZC07 (Zettlemoyer and Collins, 2007)	84.6
WKZ14 (Wang et al., 2014)	**91.3**
SEQ2TREE (Dong and Lapata, 2016)†	84.6
ASN (Rabinovich et al., 2017)†	85.3
+ supervised attention	85.9

Table 1: Performance on ATIS w.r.t. the size of labeled training data $\mathbb{L}$. †Existing neural network-based methods

$\vert\mathbb{L}\vert$	SUP.	SELFTRAIN	STRUCTVAE
1,000	49.9	49.5	**52.0**
2,000	56.6	55.8	**59.0**
3,000	61.0	61.4	**62.4**
5,000	63.2	64.5	**65.6**
8,000	70.3	69.6	**71.5**
12,000	71.1	71.6	**72.0**
16,000 (All)	**73.7**	–	72.3

Previous Method	ACC.
YN17 (Yin and Neubig, 2017)	71.6

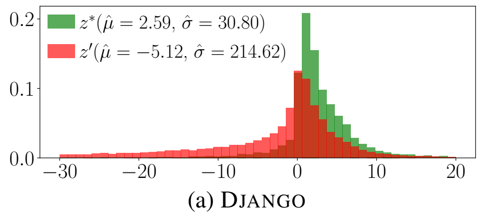
Table 2: Performance on DJANGO w.r.t. the size of labeled training data $\mathbb{L}$

Metric As standard in semantic parsing research, we evaluate by exact-match **accuracy**.

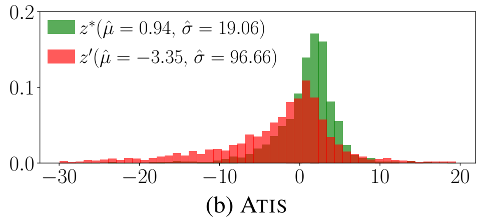
4.3 Main Results

Tab. 1 and Tab. 2 list the results on ATIS and DJANGO, resp, with varying amounts of labeled data $\mathbb{L}$. We also present results of training the transition-based parser using only the supervised objective (**SUP.**, Eq. (2)). We also compare STRUCTVAE with self-training (**SELFTRAIN**), a semi-supervised learning baseline which uses the supervised parser to predict MRs for unlabeled utterances in $\mathbb{U} - \mathbb{L}$, and adds the predicted examples to the training set to fine-tune the supervised model. Results for STRUCTVAE are averaged over four runs to account for the additional fluctuation caused by REINFORCE training.

Supervised System Comparison First, to highlight the effectiveness of our transition parser based on ASDL grammar (hence the reliability of



(a) DJANGO



(b) ATIS

Figure 4: Histograms of learning signals on DJANGO ($|\mathbb{L}| = 5000$) and ATIS ($|\mathbb{L}| = 2000$). Difference in sample means is statistically significant ($p < 0.05$).

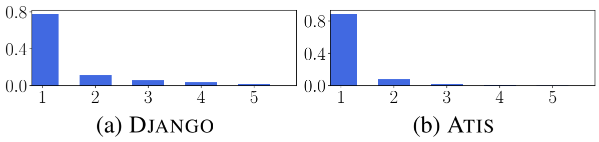


Figure 5: Distribution of the rank of $l(\boldsymbol{x}, \boldsymbol{z}^*)$ in sampled set

our supervised baseline), we compare the supervised version of our parser with existing parsing models. On ATIS, our supervised parser trained on the full data is competitive with existing neural network based models, surpassing the SEQ2TREE model, and on par with the Abstract Syntax Network (ASN) without using extra supervision. On DJANGO, our model significantly outperforms the YN17 system, probably because the transition system used by our parser is defined natively to construct ASDL ASTs, reducing the number of actions for generating each example. On DJANGO, the average number of actions is 14.3, compared with 20.3 reported in YN17.

Semi-supervised Learning Next, we discuss our main comparison between STRUCTVAE with the supervised version of the parser (recall that the supervised parser is used as the inference model in STRUCTVAE, § 3.2). First, comparing our proposed STRUCTVAE with the supervised parser when there are extra unlabeled data (*i.e.*, $|\mathbb{L}| < 4,434$ for ATIS and $|\mathbb{L}| < 16,000$ for DJANGO), semi-supervised learning with STRUCTVAE consistently achieves better performance. Notably, on DJANGO, our model registers results as competitive as previous state-of-the-art method (YN17) using only *half* the training data (71.5 when $|\mathbb{L}| = 8000$ v.s. 71.6 for YN17). This demonstrates that STRUCTVAE is capable of learning from unlabeled NL utterances by inferring high quality, structurally rich latent meaning representations, further improving the performance of its supervised counterpart that is already competitive. Second, comparing STRUCTVAE with self-training, we find STRUCTVAE outperforms SELFTRAIN in eight out of ten settings, while SELFTRAIN under-performs the supervised parser in four out of ten settings. This shows self-training does not necessarily yield stable gains while STRUCTVAE does. Intuitively, STRUCTVAE would perform better since it benefits from the additional signal of the quality of MRs from the reconstruction model (§ 3.3), for which we present more analysis in our next set of experiments.

For the sake of completeness, we also report the results of STRUCTVAE when $\mathbb{L}$ is the full training set. Note that in this scenario there is no extra unlabeled data disjoint with the labeled set, and not surprisingly, STRUCTVAE does not outperform the supervised parser. In addition to the supervised objective Eq. (2) used by the supervised parser, STRUCTVAE has the extra unsupervised objective Eq. (3), which uses sampled (probably incorrect) MRs to update the model. When there is no extra unlabeled data, those sampled (incorrect) MRs add noise to the optimization process, causing STRUCTVAE to under-perform.

Study of Learning Signals As discussed in § 3.3, in semi-supervised learning, the gradient received by the inference model from each sampled latent MR is weighted by the learning signal. Empirically, we would expect that on average, the learning signals of gold-standard samples $\boldsymbol{z}^*$, $l(\boldsymbol{x}, \boldsymbol{z}^*)$, are positive, larger than those of other (imperfect) samples $\boldsymbol{z}'$, $l(\boldsymbol{x}, \boldsymbol{z}')$. We therefore study the statistics of $l(\boldsymbol{x}, \boldsymbol{z}^*)$ and $l(\boldsymbol{x}, \boldsymbol{z}')$ for all utterances $\boldsymbol{x} \in \mathbb{U} - \mathbb{L}$, *i.e.*, the set of utterances which are not included in the labeled set.[5] The statistics are obtained by performing inference using trained models. Figures 4a and 4b depict the histograms of learning signals on DJANGO and ATIS, resp. We observe that the learning signals for gold samples concentrate on positive intervals. We also show the mean and variance of the learning signals. On average, we have $l(\boldsymbol{x}, \boldsymbol{z}^*)$ being positive and $l(\boldsymbol{x}, \boldsymbol{z})$ negative. Also note that the distribution of $l(\boldsymbol{x}, \boldsymbol{z}^*)$ has smaller variance and is more concentrated. Therefore the inference model receives informative gradient updates to discriminate between gold and imperfect

[5]We focus on cases where $\boldsymbol{z}^*$ is in the sample set $\mathcal{S}(\boldsymbol{x})$.

NL	*join p and cmd into a file path, substitute it for f*	
$\boldsymbol{z}_1^s$	`f = os.path.join(p, cmd)` ✓	
	$\log q(\boldsymbol{z} \mid \boldsymbol{x}) = -1.00$	$\log p(\boldsymbol{x} \mid \boldsymbol{z}) = -2.00$
	$\log p(\boldsymbol{z}) = -24.33$	$l(\boldsymbol{x}, \boldsymbol{z}) = 9.14$
$\boldsymbol{z}_2^s$	`p = path.join(p, cmd)` ✗	
	$\log q(\boldsymbol{z} \mid \boldsymbol{x}) = -8.12$	$\log p(\boldsymbol{x} \mid \boldsymbol{z}) = -20.96$
	$\log p(\boldsymbol{z}) = -27.89$	$l(\boldsymbol{x}, \boldsymbol{z}) = -9.47$
NL	*append i-th element of existing to child_loggers*	
$\boldsymbol{z}_1^s$	`child_loggers.append(existing[i])` ✓	
	$\log q(\boldsymbol{z} \mid \boldsymbol{x}) = -2.38$	$\log p(\boldsymbol{x} \mid \boldsymbol{z}) = -9.66$
	$\log p(\boldsymbol{z}) = -13.52$	$l(\boldsymbol{x}, \boldsymbol{z}) = 1.32$
$\boldsymbol{z}_2^s$	`child_loggers.append(existing[existing])` ✗	
	$\log q(\boldsymbol{z} \mid \boldsymbol{x}) = -1.83$	$\log p(\boldsymbol{x} \mid \boldsymbol{z}) = -16.11$
	$\log p(\boldsymbol{z}) = -12.43$	$l(\boldsymbol{x}, \boldsymbol{z}) = -5.08$
NL	*split string pks by ',', substitute the result for primary_keys*	
$\boldsymbol{z}_1^s$	`primary_keys = pks.split(',')` ✓	
	$\log q(\boldsymbol{z} \mid \boldsymbol{x}) = -2.38$	$\log p(\boldsymbol{x} \mid \boldsymbol{z}) = -11.39$
	$\log p(\boldsymbol{z}) = -10.24$	$l(\boldsymbol{x}, \boldsymbol{z}) = 2.05$
$\boldsymbol{z}_2^s$	`primary_keys = pks.split + ','` ✗	
	$\log q(\boldsymbol{z} \mid \boldsymbol{x}) = -0.84$	$\log p(\boldsymbol{x} \mid \boldsymbol{z}) = -14.87$
	$\log p(\boldsymbol{z}) = -20.41$	$l(\boldsymbol{x}, \boldsymbol{z}) = -2.60$

Table 3: Inferred latent MRs on DJANGO ($|\mathbb{L}| = 5000$). For simplicity we show the surface representation of MRs ($\boldsymbol{z}^s$, source code) instead.

samples. Next, we plot the distribution of the rank of $l(\boldsymbol{x}, \boldsymbol{z}^*)$, among the learning signals of all samples of $\boldsymbol{x}$, $\{l(\boldsymbol{x}, \boldsymbol{z}_i) : \boldsymbol{z}_i \in \mathcal{S}(\boldsymbol{x})\}$. Results are shown in Fig. 5. We observe that the gold samples $\boldsymbol{z}^*$ have the largest learning signals in around 80% cases. We also find that when $\boldsymbol{z}^*$ has the largest learning signal, its average difference with the learning signal of the highest-scoring incorrect sample is 1.27 and 0.96 on DJANGO and ATIS, respectively.

Finally, to study the relative contribution of the reconstruction score $\log p(\boldsymbol{x}|\boldsymbol{z})$ and the prior $\log p(\boldsymbol{z})$ to the learning signal, we present examples of inferred latent MRs during training (Tab. 3). Examples 1&2 show that the reconstruction score serves as an informative quality measure of the latent MR, assigning the correct samples $\boldsymbol{z}_1^s$ with high $\log p(\boldsymbol{x}|\boldsymbol{z})$, leading to positive learning signals. This is in line with our assumption that a good latent MR should adequately encode the semantics of the utterance. Example 3 shows that the prior is also effective in identifying "unnatural" MRs (*e.g.*, it is rare to add a function and a string literal, as in $\boldsymbol{z}_2^s$). These results also suggest that the prior and the reconstruction model perform well with linearization of MRs. Finally, note that in Examples 2&3 the learning signals for the correct samples $\boldsymbol{z}_1^s$ are positive even if their inference scores $q(\boldsymbol{z}|\boldsymbol{x})$ are lower than those of $\boldsymbol{z}_2^s$.

$\lvert\mathbb{L}\rvert$	SUPERVISED	STRUCTVAE-SEQ
500	47.3	**55.6**
1,000	62.5	**73.1**
2,000	73.9	**74.8**
3,000	80.6	**81.3**
4,434 (All)	**84.6**	84.2

Table 4: Performance of the STRUCTVAE-SEQ on ATIS w.r.t. the size of labeled training data $\mathbb{L}$

ATIS				DJANGO			
$\lvert\mathbb{L}\rvert$	SUP.	MLP	LM	$\lvert\mathbb{L}\rvert$	SUP.	MLP	LM
500	63.2	*61.5*†	**66.0**	1,000	49.9	*47.0*†	**52.0**
1,000	74.6	**76.3**	75.7	5,000	63.2	*62.5*†	**65.6**
2,000	80.4	**82.9**	82.4	8,000	70.3	*67.6*†	**71.5**
3,000	82.8	*81.4*†	**83.6**	12,000	71.1	71.6	**72.0**

Table 5: Comparison of STRUCTVAE with different baseline functions $b(\boldsymbol{x})$, *italic*†: semi-supervised learning with the MLP baseline is worse than supervised results.

This result further demonstrates that learning signals provide informative gradient weights for optimizing the inference model.

Generalizing to Other Latent MRs Our main results are obtained using a strong AST-based semantic parser as the inference model, with copy-augmented reconstruction model and an LSTM language model as the prior. However, there are many other ways to represent and infer structure in semantic parsing (Carpenter, 1998; Steedman, 2000), and thus it is of interest whether our basic STRUCTVAE framework generalizes to other semantic representations. To examine this, we test STRUCTVAE using λ-calculus logical forms as latent MRs for semantic parsing on the ATIS domain. We use standard sequence-to-sequence networks with attention (Luong et al., 2015) as inference and reconstruction models. The inference model is trained to construct a tree-structured logical form using the transition actions defined in Cheng et al. (2017). We use a classical tri-gram Kneser-Ney language model as the prior. Tab. 4 lists the results for this STRUCTVAE-SEQ model.

We can see that even with this very different model structure STRUCTVAE still provides significant gains, demonstrating its compatibility with different inference/reconstruction networks and priors. Interestingly, compared with the results in Tab. 1, we found that the gains are especially larger with few labeled examples — STRUCTVAE-SEQ achieves improvements of 8-10 points when $|\mathbb{L}| < 1000$. These results suggest that semi-supervision is especially useful in improving a mediocre parser in low resource settings.

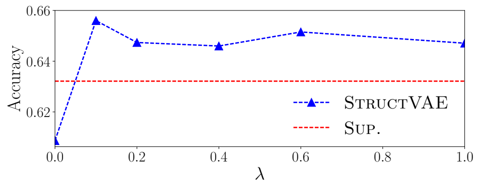


Figure 6: Performance on Django ($|\mathbb{L}| = 5000$) w.r.t. the KL weight λ

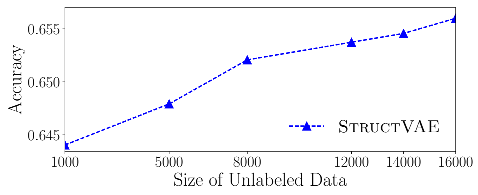


Figure 7: Performance on Django ($|\mathbb{L}| = 5000$) w.r.t. the size of unlabeled data $\mathbb{U}$

Impact of Baseline Functions In § 3.3 we discussed our design of the baseline function $b(\boldsymbol{x})$ incorporated in the learning signal (Eq. (4)) to stabilize learning, which is based on a language model (LM) over utterances (Eq. (6)). We compare this baseline with a commonly used one in REINFORCE training: the multi-layer perceptron (MLP). The MLP takes as input the last hidden state of the utterance given by the encoding LSTM of the inference model. Tab. 5 lists the results over sampled settings. We found that although StructVAE with the MLP baseline sometimes registers better performance on Atis, in most settings it is worse than our LM baseline, and could be even worse than the supervised parser. On the other hand, our LM baseline correlates well with the learning signal, yielding stable improvements over the supervised parser. This suggests the importance of using carefully designed baselines in REINFORCE learning, especially when the reward signal has large range (*e.g.*, log-likelihoods).

Impact of the Prior $p(\boldsymbol{z})$ Fig. 6 depicts the performance of StructVAE as a function of the KL term weight λ in Eq. (3). When StructVAE degenerates to a vanilla auto-encoder without the prior distribution (*i.e.*, $\lambda = 0$), it under-performs the supervised baseline. This is in line with our observation in Tab. 3 showing that the prior helps identify unnatural samples. The performance of the model also drops when $\lambda > 0.1$, suggesting that empirically controlling the influence of the prior to the inference model is important.

Impact of Unlabeled Data Size Fig. 7 illustrates the accuracies w.r.t. the size of unlabeled data. StructVAE yields consistent gains as the size of the unlabeled data increases.

5 Related Works

Semi-supervised Learning for NLP Semi-supervised learning comes with a long history (Zhu, 2005), with applications in NLP from early work of self-training (Yarowsky, 1995), and graph-based methods (Das and Smith, 2011), to recent advances in auto-encoders (Cheng et al., 2016; Socher et al., 2011; Zhang et al., 2017) and deep generative methods (Xu et al., 2017). Our work follows the line of neural variational inference for text processing (Miao et al., 2016), and resembles Miao and Blunsom (2016), which uses VAEs to model summaries as discrete latent variables for semi-supervised summarization, while we extend the VAE architecture for more complex, tree-structured latent variables.

Semantic Parsing Most existing works alleviate issues of limited parallel data through weakly-supervised learning, using the denotations of MRs as indirect supervision (Reddy et al., 2014; Krishnamurthy et al., 2016; Neelakantan et al., 2016; Pasupat and Liang, 2015; Yin et al., 2016). For semi-supervised learning of semantic parsing, Kate and Mooney (2007) first explore using transductive SVMs to learn from a semantic parser's predictions. Konstas et al. (2017) apply self-training to bootstrap an existing parser for AMR parsing. Kociský et al. (2016) employ VAEs for semantic parsing, but in contrast to StructVAE's structured representation of MRs, they model NL utterances as flat latent variables, and learn from unlabeled MR data. There have also been efforts in unsupervised semantic parsing, which exploits external linguistic analysis of utterances (*e.g.*, dependency trees) and the schema of target knowledge bases to infer the latent MRs (Poon and Domingos, 2009; Poon, 2013). Another line of research is domain adaptation, which seeks to transfer a semantic parser learned from a source domain to the target domain of interest, therefore alleviating the need of parallel data from the target domain (Su and Yan, 2017; Fan et al., 2017; Herzig and Berant, 2018).

6 Conclusion

We propose StructVAE, a deep generative model with tree-structured latent variables for semi-supervised semantic parsing. We apply StructVAE to semantic parsing and code generation tasks, and show it outperforms a strong supervised parser using extra unlabeled data.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of LAW-ID@ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the SIGNLL*.

Bob Carpenter. 1998. *Type-logical Semantics*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of ACL*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of ACL*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*.

Dipanjan Das and Noah A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of HLT*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of ACL*.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. *arXiv preprint arXiv:1804.07918* .

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of ACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Rohit J. Kate and Raymond J. Mooney. 2007. Semi-supervised learning for semantic parsing using support vector machines. In *Proceedings of NAACL-HLT*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Proceedings of NIPS*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Tomás Kociský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of EMNLP*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*.

Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. 2016. Semantic parsing to probabilistic programs for situated question answering. In *Proceedings of EMNLP*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of EMNLP*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of ICML*.

Dipendra K. Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of EMNLP*.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of ICLR*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (T). In *Proceedings of ASE*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of ACL*.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of EMNLP*.

Python Software Foundation. 2016. Python abstract grammar. https://docs.python.org/2/library/ast.html.

Chris Quirk, Raymond J. Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of ACL*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of ACL*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of ACL* .

Aarti Singh, Robert D. Nowak, and Xiaojin Zhu. 2008. Unlabeled data: Now it helps, now it doesn't. In *Proceedings of NIPS*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

Mark Steedman. 2000. *The Syntactic Process*.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of EMNLP*.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of ECML*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for ccg semantic parsing. In *Proceedings of EMNLP*.

Daniel C. Wang, Andrew W. Appel, Jeffrey L. Korn, and Christopher S. Serra. 1997. The zephyr abstract syntax description language. In *Proceedings of DSL*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* .

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of ACL*.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Proceedings of AAAI*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In *Proceedings of IJCAI*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of ACL*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI*.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*.

Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. 2017. Semi-supervised structured prediction with neural crf autoencoder. In *Proceedings of EMNLP*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* .

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of ACL*.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Sequence-to-Action: End-to-End Semantic Graph Generation for Semantic Parsing

Bo Chen[†‡], **Le Sun**[†], **Xianpei Han**[†]
[†]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[‡]University of Chinese Academy of Sciences, Beijing, China
{chenbo,sunle,xianpei}@iscas.ac.cn

Abstract

This paper proposes a neural semantic parsing approach – Sequence-to-Action, which models semantic parsing as an end-to-end semantic graph generation process. Our method simultaneously leverages the advantages from two recent promising directions of semantic parsing. Firstly, our model uses a semantic graph to represent the meaning of a sentence, which has a tight-coupling with knowledge bases. Secondly, by leveraging the powerful representation learning and prediction ability of neural network models, we propose a RNN model which can effectively map sentences to action sequences for semantic graph generation. Experiments show that our method achieves state-of-the-art performance on OVERNIGHT dataset and gets competitive performance on GEO and ATIS datasets.

1 Introduction

Semantic parsing aims to map natural language sentences to logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Lu et al., 2008; Kwiatkowski et al., 2013). For example, the sentence "*Which states border Texas?*" will be mapped to `answer (A, (state (A), next_to (A, stateid ( texas ))))`.

A semantic parser needs two functions, one for structure prediction and the other for semantic grounding. Traditional semantic parsers are usually based on compositional grammar, such as CCG (Zettlemoyer and Collins, 2005, 2007), DCS (Liang et al., 2011), etc. These parsers compose structure using manually designed grammars, use lexicons for semantic grounding, and exploit features for candidate logical forms ranking. Unfortunately, it is challenging to design grammars and learn accurate lexicons, especially in wide-open domains. Moreover, it is often hard to design effective features, and its learning process is not end-to-end. To resolve the above problems, two promising lines of work have been proposed: Semantic graph-based methods and Seq2Seq methods.

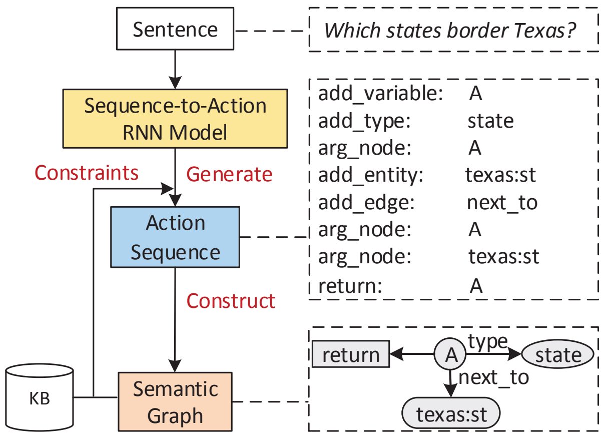


Figure 1: Overview of our method, with a demonstration example.

Semantic graph-based methods (Reddy et al., 2014, 2016; Bast and Haussmann, 2015; Yih et al., 2015) represent the meaning of a sentence as a semantic graph (i.e., a sub-graph of a knowledge base, see example in Figure 1) and treat semantic parsing as a semantic graph matching/generation process. Compared with logical forms, semantic graphs have a tight-coupling with knowledge bases (Yih et al., 2015), and share many commonalities with syntactic structures (Reddy et al., 2014). Therefore both the structure and semantic constraints from knowledge bases can be easily exploited during parsing (Yih et al., 2015). The main challenge of semantic graph-based parsing is how to effectively construct the semantic graph of a sentence. Currently, semantic graphs are either constructed by matching with patterns (Bast and Haussmann, 2015), transforming from dependency tree (Reddy et al., 2014, 2016), or via a staged heuristic search algorithm (Yih et al., 2015). These methods are all based on manually-designed, heuristic construction processes, making them hard to handle open/complex situations.

In recent years, RNN models have achieved success in sequence-to-sequence problems due to its strong ability on both representation learning and prediction, e.g., in machine translation (Cho et al., 2014). A lot of Seq2Seq models have also been employed for semantic parsing (Xiao et al., 2016; Dong and Lapata, 2016; Jia and Liang, 2016), where a sentence is parsed by translating it to linearized logical form using RNN models. There is no need for high-quality lexicons, manually-built grammars, and hand-crafted features. These models are trained end-to-end, and can leverage attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) to learn soft alignments between sentences and logical forms.

In this paper, we propose a new neural semantic parsing framework – Sequence-to-Action, which can simultaneously leverage the advantages of semantic graph representation and the strong prediction ability of Seq2Seq models. Specifically, we model semantic parsing as an end-to-end semantic graph generation process. For example in Figure 1, our model will parse the sentence "*Which states border Texas*" by generating a sequence of actions [`add_variable:A`, `add_type:state`, ...]. To achieve the above goal, we first design an action set which can encode the generation process of semantic graph (including node actions such as `add_variable`, `add_entity`, `add_type`, edge actions such as `add_edge`, and operation actions such as `argmin`, `argmax`, `count`, `sum`, etc.). And then we design a RNN model which can generate the action sequence for constructing the semantic graph of a sentence. Finally we further enhance parsing by incorporating both structure and semantic constraints during decoding.

Compared with the manually-designed, heuristic generation algorithms used in traditional semantic graph-based methods, our sequence-to-action method generates semantic graphs using a RNN model, which is learned end-to-end from training data. Such a learnable, end-to-end generation makes our approach more effective and can fit to different situations.

Compared with the previous Seq2Seq semantic parsing methods, our sequence-to-action model predicts a sequence of semantic graph generation actions, rather than linearized logical forms. We find that the action sequence encoding can better capture structure and semantic information, and is more compact. And the parsing can be enhanced by exploiting structure and semantic constraints. For example, in GEO dataset, the action `add_edge:next_to` must subject to the semantic constraint that its arguments must be of type `state` and `state`, and the structure constraint that the edge `next_to` must connect two nodes to form a valid graph.

We evaluate our approach on three standard datasets: GEO (Zelle and Mooney, 1996), ATIS (He and Young, 2005) and OVERNIGHT (Wang et al., 2015b). The results show that our method achieves state-of-the-art performance on OVERNIGHT dataset and gets competitive performance on GEO and ATIS datasets.

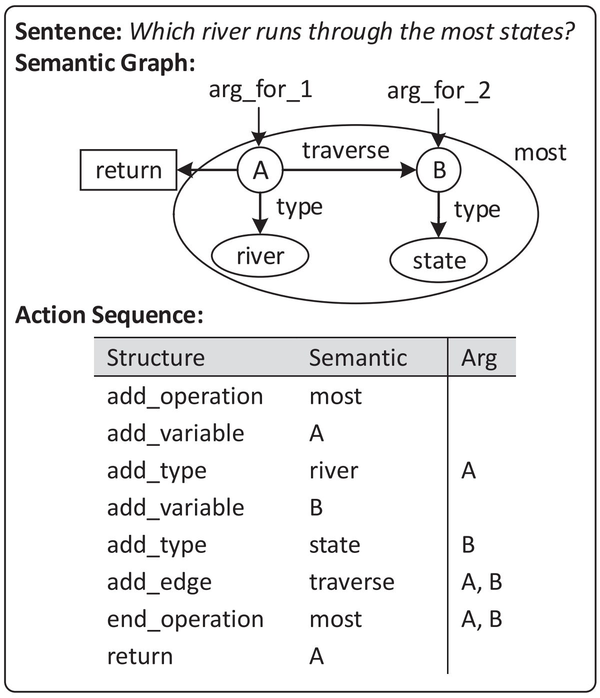
The main contributions of this paper are summarized as follows:

- We propose a new semantic parsing framework – Sequence-to-Action, which models semantic parsing as an end-to-end semantic graph generation process. This new framework can synthesize the advantages of semantic graph representation and the prediction ability of Seq2Seq models.

- We design a sequence-to-action model, including an action set encoding for semantic graph generation and a Seq2Seq RNN model for action sequence prediction. We further enhance the parsing by exploiting structure and semantic constraints during decoding. Experiments validate the effectiveness of our method.

2 Sequence-to-Action Model for End-to-End Semantic Graph Generation

Given a sentence $X = x_1, ..., x_{|X|}$, our sequence-to-action model generates a sequence of actions $Y = y_1, ..., y_{|Y|}$ for constructing the correct semantic graph. Figure 2 shows an example. The conditional probability $P(Y|X)$ used in our


Sentence: *Which river runs through the most states?*

Semantic Graph:

Action Sequence:

Structure	Semantic	Arg
add_operation	most	
add_variable	A	
add_type	river	A
add_variable	B	
add_type	state	B
add_edge	traverse	A, B
end_operation	most	A, B
return	A	

Figure 2: An example of a sentence paired with its semantic graph, together with the action sequence for semantic graph generation.

model is decomposed as follows:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|y_{<t}, X) \tag{1}$$

where $y_{<t} = y_1, ..., y_{t-1}$.

To achieve the above goal, we need: 1) an action set which can encode semantic graph generation process; 2) an encoder which encodes natural language input X into a vector representation, and a decoder which generates $y_1, ..., y_{|Y|}$ conditioned on the encoding vector. In following we describe them in detail.

2.1 Actions for Semantic Graph Generation

Generally, a semantic graph consists of nodes (including variables, entities, types) and edges (semantic relations), with some universal operations (e.g., `argmax`, `argmin`, `count`, `sum`, and `not`). To generate a semantic graph, we define six types of actions as follows:

Add Variable Node: This kind of actions denotes adding a variable node to semantic graph. In most cases a variable node is a return node (e.g., *which*, *what*), but can also be an intermediate variable node. We represent this kind of action as `add_variable:A`, where `A` is the identifier of the variable node.

Add Entity Node: This kind of actions denotes adding an entity node (e.g., *Texas*, *New York*) and is represented as `add_entity_node:texas`. An entity node corresponds to an entity in knowledge bases.

Add Type Node: This kind of actions denotes adding a type node (e.g., *state*, *city*). We represent them as `add_type_node:state`.

Add Edge: This kind of actions denotes adding an edge between two nodes. An edge is a binary relation in knowledge bases. This kind of actions is represented as `add_edge:next_to`.

Operation Action: This kind of actions denotes adding an operation. An operation can be `argmax`, `argmin`, `count`, `sum`, `not`, et al. Because each operation has a scope, we define two actions for an operation, one is operation start action, represented as `start_operation:most`, and the other is operation end action, represented as `end_operation:most`. The subgraph within the start and end operation actions is its scope.

Argument Action: Some above actions need argument information. For example, which nodes the `add_edge:next_to` action should connect to. In this paper, we design argument actions for `add_type`, `add_edge` and `operation` actions, and the argument actions should be put directly after its main action.

For `add_type` actions, we put an argument action to indicate which node this type node should constrain. The argument can be a variable node or an entity node. An argument action for a type node is represented as `arg:A`.

For `add_edge` action, we use two argument actions: `arg1_node` and `arg2_node`, and they are represented as `arg1_node:A` and `arg2_node:B`.

We design argument actions for different operations. For `operation:sum`, there are three arguments: `arg-for`, `arg-in` and `arg-return`. For `operation:count`, they are `arg-for` and `arg-return`. There are two `arg-for` arguments for `operation:most`.

We can see that each action encodes both structure and semantic information, which makes it easy to capture more information for parsing and can be tightly coupled with knowledge base. Furthermore, we find that action sequence encoding is more compact than linearized logical form (See Section 4.4 for more details).

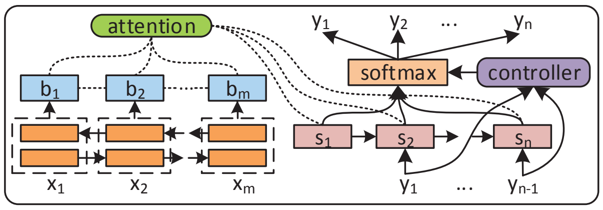


Figure 3: Our attention-based Sequence-to-Action RNN model, with a controller for incorporating constraints.

2.2 Neural Sequence-to-Action Model

Based on the above action encoding mechanism, this section describes our encoder-decoder model for mapping sentence to action sequence. Specifically, similar to the RNN model in Jia and Liang (2016), this paper employs the attention-based sequence-to-sequence RNN model. Figure 3 presents the overall structure.

Encoder: The encoder converts the input sequence $x_1, ..., x_m$ to a sequence of context-sensitive vectors $b_1, ..., b_m$ using a bidirectional RNN (Bahdanau et al., 2014). Firstly each word x_i is mapped to its embedding vector, then these vectors are fed into a forward RNN and a backward RNN. The sequence of hidden states $h_1, ..., h_m$ are generated by recurrently applying the recurrence:

$$h_i = LSTM(\phi^{(x)}(x_i), h_{i-1}). \tag{2}$$

The recurrence takes the form of LSTM (Hochreiter and Schmidhuber, 1997). Finally, for each input position i, we define its context-sensitive embedding as $b_i = [h_i^F, h_i^B]$.

Decoder: This paper uses the classical attention-based decoder (Bahdanau et al., 2014), which generates action sequence $y_1, ..., y_n$, one action at a time. At each time step j, it writes y_j based on the current hidden state s_j, then updates the hidden state to s_{j+1} based on s_j and y_j. The decoder is formally defined by the following equations:

$$s_1 = \tanh(W^{(s)}[h_m^F, h_1^B]) \tag{3}$$

$$e_{ji} = s_j^T W^{(a)} b_i \tag{4}$$

$$a_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})} \tag{5}$$

$$c_j = \sum_{i=1}^{m} a_{ji} b_i \tag{6}$$

$$P(y_j = w | x, y_{1:j-1}) \propto \exp(U_w[s_j, c_j]) \tag{7}$$

$$s_{j+1} = LSTM([\phi^{(y)}(y_j), c_j], s_j) \tag{8}$$

where the normalized attention scores a_{ji} defines the probability distribution over input words, indicating the attention probability on input word i at time j; e_{ji} is un-normalized attention score. To incorporate constraints during decoding, an extra controller component is added and its details will be described in Section 3.3.

Action Embedding. The above decoder needs the embedding of each action. As described above, each action has two parts, one for structure (e.g., `add_edge`), and the other for semantic (e.g., `next_to`). As a result, actions may share the same structure or semantic part, e.g., `add_edge:next_to` and `add_edge:loc` have the same structure part, and `add_node:A` and `arg_node:A` have the same semantic part. To make parameters more compact, we first embed the structure part and the semantic part independently, then concatenate them to get the final embedding. For instance, $\phi^{(y)}($`add_edge:next_to` $) = [\ \phi^{(y)}_{strut}($ `add_edge` $), \phi^{(y)}_{sem}($ `next_to` $)]$. The action embeddings $\phi^{(y)}$ are learned during training.

3 Constrained Semantic Parsing using Sequence-to-Action Model

In this section, we describe how to build a neural semantic parser using sequence-to-action model. We first describe the training and the inference of our model, and then introduce how to incorporate structure and semantic constraints during decoding.

3.1 Training

Parameter Estimation. The parameters of our model include RNN parameters $W^{(s)}$, $W^{(a)}$, U_w, word embeddings $\phi^{(x)}$, and action embeddings $\phi^{(y)}$. We estimate these parameters from training data. Given a training example with a sentence X and its action sequence Y, we maximize the likelihood of the generated sequence of actions given X. The objective function is:

$$\sum_{i=1}^{n} \log P(Y_i|X_i) \tag{9}$$

Standard stochastic gradient descent algorithm is employed to update parameters.

Logical Form to Action Sequence. Currently, most datasets of semantic parsing are labeled with logical forms. In order to train our model, we

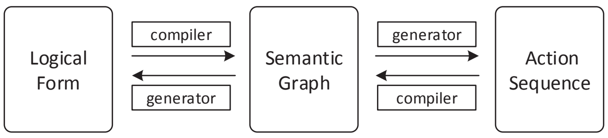


Figure 4: The procedure of converting between logical form and action sequence.

convert logical forms to action sequences using semantic graph as an intermediate representation (See Figure 4 for an overview). Concretely, we transform logical forms into semantic graphs using a depth-first-search algorithm from root, and then generate the action sequence using the same order. Specifically, entities, variables and types are nodes; relations are edges. Conversely we can convert action sequence to logical form similarly. Based on the above algorithm, action sequences can be transformed into logical forms in a deterministic way, and the same for logical forms to action sequences.

Mechanisms for Handling Entities. Entities play an important role in semantic parsing (Yih et al., 2015). In Dong and Lapata (2016), entities are replaced with their types and unique IDs. In Jia and Liang (2016), entities are generated via attention-based copying mechanism helped with a lexicon. This paper implements both mechanisms and compares them in experiments.

3.2 Inference

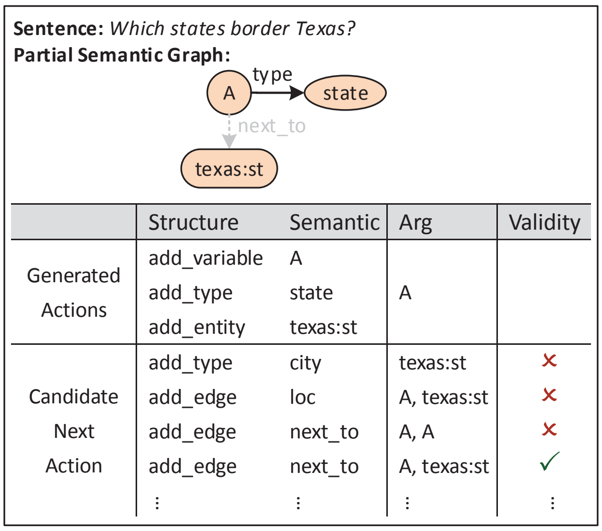
Given a new sentence X, we predict action sequence by:

$$Y^* = \underset{Y}{\text{argmax}} P(Y|X) \qquad (10)$$

where Y represents action sequence, and $P(Y|X)$ is computed using Formula (1). Beam search is used for best action sequence decoding. Semantic graph and logical form can be derived from Y^* as described in above.

3.3 Incorporating Constraints in Decoding

For decoding, we generate action sequentially. It is obviously that the next action has a strong correlation with the partial semantic graph generated to current, and illegal actions can be filtered using structure and semantic constraints. Specifically, we incorporate constraints in decoding using a controller. This procedure has two steps: 1) the controller constructs partial semantic graph using the actions generated to current; 2) the controller checks whether a new generated action can meet



	Structure	Semantic	Arg	Validity
Generated Actions	add_variable	A		
	add_type	state	A	
	add_entity	texas:st		
Candidate Next Action	add_type	city	texas:st	✗
	add_edge	loc	A, texas:st	✗
	add_edge	next_to	A, A	✗
	add_edge	next_to	A, texas:st	✓
	⋮	⋮	⋮	⋮

Figure 5: A demonstration of illegal action filtering using constraints. The graph in color is the constructed semantic graph to current.

all structure/semantic constraints using the partial semantic graph.

Structure Constraints. The structure constraints ensure action sequence will form a connected acyclic graph. For example, there must be two argument nodes for an edge, and the two argument nodes should be different (The third candidate next action in Figure 5 violates this constraint). This kind of constraints are domain-independent. The controller encodes structure constraints as a set of rules.

Semantic Constraints. The semantic constraints ensure the constructed graph must follow the schema of knowledge bases. Specifically, we model two types of semantic constraints. One is selectional preference constraints where the argument types of a relation should follow knowledge base schemas. For example, in GEO dataset, relation `next_to`'s `arg1` and `arg2` should both be a `state`. The second is type conflict constraints, i.e., an entity/variable node's type must be consistent, i.e., a node cannot be both of type `city` and `state`. Semantic constraints are domain-specific and are automatically extracted from knowledge base schemas. The controller encodes semantic constraints as a set of rules.

4 Experiments

In this section, we assess the performance of our method and compare it with previous methods.

4.1 Datasets

We conduct experiments on three standard datasets: GEO, ATIS and OVERNIGHT.

GEO contains natural language questions about US geography paired with corresponding Prolog database queries. Following Zettlemoyer and Collins (2005), we use the standard 600/280 instance splits for training/test.

ATIS contains natural language questions of a flight database, with each question is annotated with a lambda calculus query. Following Zettlemoyer and Collins (2007), we use the standard 4473/448 instance splits for training/test.

OVERNIGHT contains natural language paraphrases paired with logical forms across eight domains. We evaluate on the standard train/test splits as Wang et al. (2015b).

4.2 Experimental Settings

Following the experimental setup of Jia and Liang (2016): we use 200 hidden units and 100-dimensional word vectors for sentence encoding. The dimensions of action embedding are tuned on validation datasets for each corpus. We initialize all parameters by uniformly sampling within the interval [-0.1, 0.1]. We train our model for a total of 30 epochs with an initial learning rate of 0.1, and halve the learning rate every 5 epochs after epoch 15. We replace word vectors for words occurring only once with an universal word vector. The beam size is set as 5. Our model is implemented in Theano (Bergstra et al., 2010), and the codes and settings are released on Github: https://github.com/dongpobeyond/Seq2Act.

We evaluate different systems using the standard accuracy metric, and the accuracies on different datasets are obtained as same as Jia and Liang (2016).

4.3 Overall Results

We compare our method with state-of-the-art systems on all three datasets. Because all systems using the same training/test splits, we directly use the reported best performances from their original papers for fair comparison.

For our method, we train our model with three settings: the first one is the basic sequence-to-action model without constraints – Seq2Act; the second one adds structure constraints in decoding – Seq2Act (+C1); the third one is the full model which adds both structure and semantic constraints – Seq2Act (+C1+C2). Semantic constraints (C2) are stricter than structure constraints (C1). Therefore we set that C1 should be first met for C2 to be met. So in our experiments we add constraints incrementally. The overall results are shown in Table 1-2. From the overall results, we can see that:

	GEO	ATIS
Previous Work		
Zettlemoyer and Collins (2005)	79.3	–
Zettlemoyer and Collins (2007)	86.1	84.6
Kwiatkowksi et al. (2010)	88.9	–
Kwiatkowski et al. (2011)	88.6	82.8
Liang et al. (2011)* (+lexicon)	**91.1**	–
Poon (2013)	–	83.5
Zhao et al. (2015)	88.9	84.2
Rabinovich et al. (2017)	87.1	**85.9**
Seq2Seq Models		
Jia and Liang (2016)	85.0	76.3
Jia and Liang (2016)* (+data)	89.3	83.3
Dong and Lapata (2016): 2Seq	84.6	84.2
Dong and Lapata (2016): 2Tree	87.1	84.6
Our Models		
Seq2Act	87.5	84.6
Seq2Act (+C1)	88.2	85.0
Seq2Act (+C1+C2)	88.9	85.5

Table 1: Test accuracies on GEO and ATIS datasets, where * indicates systems with extra-resources are used.

1) By synthetizing the advantages of semantic graph representation and the prediction ability of Seq2Seq model, our method achieves state-of-the-art performance on OVERNIGHT dataset, and gets competitive performance on GEO and ATIS dataset. In fact, on GEO our full model (Seq2Act+C1+C2) also gets the best test accuracy of 88.9 if under the same settings, which only falls behind Liang et al. (2011)* which uses extra handcrafted lexicons and Jia and Liang (2016)* which uses extra augmented training data. On ATIS our full model gets the second best test accuracy of 85.5, which only falls behind Rabinovich et al. (2017) which uses a supervised attention strategy. On OVERNIGHT, our full model gets state-of-the-art accuracy of 79.0, which even outperforms Jia and Liang (2016)* with extra augmented training data.

2) Compared with the linearized logical form representation used in previous Seq2Seq baselines, our action sequence encoding is more effective for semantic parsing. On all three datasets,

	Soc.	Blo.	Bas.	Res.	Cal.	Hou.	Pub.	Rec.	Avg.
Previous Work									
Wang et al. (2015b)	48.2	41.9	46.3	75.9	74.4	54.0	59.0	70.8	58.8
Seq2Seq Models									
Xiao et al. (2016)	80.0	55.6	80.5	80.1	75.0	61.9	75.8	–	72.7
Jia and Liang (2016)	81.4	58.1	85.2	76.2	78.0	71.4	76.4	79.6	75.8
Jia and Liang (2016)* (+data)	79.6	60.2	87.5	79.5	81.0	72.5	78.3	81.0	77.5
Our Models									
Seq2Act	81.4	60.4	87.5	79.8	81.0	73.0	79.5	81.5	78.0
Seq2Act (+C1)	81.8	60.9	88.0	80.1	81.0	73.5	80.1	82.0	78.4
Seq2Act (+C1+C2)	**82.1**	**61.4**	**88.2**	**80.7**	**81.5**	**74.1**	**80.7**	**82.9**	**79.0**

Table 2: Test accuracies on OVERNIGHT dataset, which includes eight domains: Social, Blocks, Basketball, Restaurants, Calendar, Housing, Publications, and Recipes.

our basic Seq2Act model gets better results than all Seq2Seq baselines. On GEO, the Seq2Act model achieve test accuracy of 87.5, better than the best accuracy 87.1 of Seq2Seq baseline. On ATIS, the Seq2Act model obtains a test accuracy of 84.6, the same as the best Seq2Seq baseline. On OVERNGIHT, the Seq2Act model gets a test accuracy of 78.0, better than the best Seq2Seq baseline gets 77.5. We argue that this is because our action sequence encoding is more compact and can capture more information.

3) Structure constraints can enhance semantic parsing by ensuring the validity of graph using the generated action sequence. In all three datasets, Seq2Act (+C1) outperforms the basic Seq2Act model. This is because a part of illegal actions will be filtered during decoding.

4) By leveraging knowledge base schemas during decoding, semantic constraints are effective for semantic parsing. Compared to Seq2Act and Seq2Act (+C1), the Seq2Act (+C1+C2) gets the best performance on all three datasets. This is because semantic constraints can further filter semantic illegal actions using selectional preference and consistency between types.

4.4 Detailed Analysis

Effect of Entity Handling Mechanisms. This paper implements two entity handling mechanisms – *Replacing* (Dong and Lapata, 2016) which identifies entities and then replaces them with their types and IDs, and attention-based *Copying* (Jia and Liang, 2016). To compare the above two mechanisms, we train and test with our full model and the results are shown in Table 3. We can see that, *Replacing* mechanism outperforms *Copying* in all three datasets. This is because *Replacing* is done in preprocessing, while attention-based *Copying* is done during parsing and needs additional copy mechanism.

	Replacing	*Copying*
GEO	88.9	88.2
ATIS	85.5	84.0
OVERNIGHT	79.0	77.9

Table 3: Test accuracies of Seq2Act (+C1+C2) on GEO, ATIS, and OVERNIGHT of two entity handling mechanisms.

	Logical Form	Action Sequence
GEO	28.2	18.2
ATIS	28.4	25.8
OVERNIGHT	46.6	33.3

Table 4: Average length of logical forms and action sequences on three datasets. On OVERNIGHT, we average across all eight domains.

Linearized Logical Form vs. Action Sequence. Table 4 shows the average length of linearized logical forms used in previous Seq2Seq models and the action sequences of our model on all three datasets. As we can see, action sequence encoding is more compact than linearized logical form encoding: action sequence is shorter on all three datasets, 35.5%, 9.2% and 28.5% reduction in length respectively. The main advantage of a shorter/compact encoding is that it will reduce the influence of long distance dependency problem.

4.5 Error Analysis

We perform error analysis on results and find there are mainly two types of errors.

Unseen/Informal Sentence Structure. Some test sentences have unseen syntactic structures. For example, the first case in Table 5 has an unseen

Error Types	Examples
Un-covered Sentence Structure	Sentence: *Iowa borders how many states?* (Formal Form: *How many states does Iowa border?*) Gold Parse: `answer(A, count(B, (const (C, stateid(iowa)), next_to(C, B), state (B)), A))` Predicted Parse: `answer (A, count(B, state(B), A))`
Under-Mapping	Sentence: *Please show me **first class** flights from indianapolis to memphis one way leaving before 10am* Gold Parse: `(lambda x (and (flight x) (oneway x) (class_type x first:cl) (< (departure_time x) 1000:ti) (from x indianapolis:ci) (to x memphis:ci)))` Predicted Parse: `(lambda x (and (flight x) (oneway x) (< (departure_time x) 1000:ti) (from x indianapolis:ci) (to x memphis:ci)))`

Table 5: Some examples for error analysis. Each example includes the sentence for parsing, with gold parse and predicted parse from our model.

and informal structure, where entity word "*Iowa*" and relation word "*borders*" appear ahead of the question words "*how many*". For this problem, we can employ sentence rewriting or paraphrasing techniques (Chen et al., 2016; Dong et al., 2017) to transform unseen sentence structures into normal ones.

Under-Mapping. As Dong and Lapata (2016) discussed, the attention model does not take the alignment history into consideration, makes some words are ignored during parsing. For example in the second case in Table 5, "*first class*" is ignored during the decoding process. This problem can be further solved using explicit word coverage models used in neural machine translation (Tu et al., 2016; Cohn et al., 2016)

5 Related Work

Semantic parsing has received significant attention for a long time (Kate and Mooney, 2006; Clarke et al., 2010; Krishnamurthy and Mitchell, 2012; Artzi and Zettlemoyer, 2013; Berant and Liang, 2014; Quirk et al., 2015; Artzi et al., 2015; Reddy et al., 2017). Traditional methods are mostly based on the principle of compositional semantics, which first trigger predicates using lexicons and then compose them using grammars. The prominent grammars include SCFG (Wong and Mooney, 2007; Li et al., 2015), CCG (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2011; Cai and Yates, 2013), DCS (Liang et al., 2011; Berant et al., 2013), etc. As discussed above, the main drawback of grammar-based methods is that they rely on high-quality lexicons, manually-built grammars, and hand-crafted features.

In recent years, one promising direction of semantic parsing is to use semantic graph as representation. Thus semantic parsing is modeled as a semantic graph generation process. Ge and Mooney (2009) build semantic graph by transforming syntactic tree. Bast and Haussmann (2015) identify the structure of a semantic query using three pre-defined patterns. Reddy et al. (2014, 2016) use Freebase-based semantic graph representation, and convert sentences to semantic graphs using CCG or dependency tree. Yih et al. (2015) generate semantic graphs using a staged heuristic search algorithm. These methods are all based on manually-designed, heuristic generation process, which may suffer from syntactic parse errors (Ge and Mooney, 2009; Reddy et al., 2014, 2016), structure mismatch (Chen et al., 2016), and are hard to deal with complex sentences (Yih et al., 2015).

One other direction is to employ neural Seq2Seq models, which models semantic parsing as an end-to-end, sentence to logical form machine translation problem. Dong and Lapata (2016), Jia and Liang (2016) and Xiao et al. (2016) transform word sequence to linearized logical forms. One main drawback of these methods is that it is hard to capture and exploit structure and semantic constraints using linearized logical forms. Dong and Lapata (2016) propose a Seq2Tree model to capture the hierarchical structure of logical forms.

It has been shown that structure and semantic constraints are effective for enhancing semantic parsing. Krishnamurthy et al. (2017) use type constraints to filter illegal tokens. Liang et al. (2017) adopt a Lisp interpreter with pre-defined functions to produce valid tokens. Iyyer et al. (2017) adopt type constraints to generate valid actions. Inspired by these approaches, we also incorporate both structure and semantic constraints in our neural sequence-to-action model.

Transition-based approaches are important in both dependency parsing (Nivre, 2008; Henderson et al., 2013) and AMR parsing (Wang et al., 2015a). In semantic parsing, our method has a tight-coupling with knowledge bases, and constraints can be exploited for more accurate decoding. We believe this can also be used to enhance previous transition based methods and may also be used in other parsing tasks, e.g., AMR parsing.

6 Conclusions

This paper proposes Sequence-to-Action, a method which models semantic parsing as an end-to-end semantic graph generation process. By leveraging the advantages of semantic graph representation and exploiting the representation learning and prediction ability of Seq2Seq models, our method achieved significant performance improvements on three datasets. Furthermore, structure and semantic constraints can be easily incorporated in decoding to enhance semantic parsing.

For future work, to solve the problem of the lack of training data, we want to design weakly supervised learning algorithm using denotations (QA pairs) as supervision. Furthermore, we want to collect labeled data by designing an interactive UI for annotation assist like (Yih et al., 2016), which uses semantic graphs to annotate the meaning of sentences, since semantic graph is more natural and can be easily annotated without the need of expert knowledge.

Acknowledgments

This research work is supported by the National Key Research and Development Program of China under Grant No.2017YFB1002104; and the National Natural Science Foundation of China under Grants no. 61572477 and 61772505. Moreover, we sincerely thank the reviewers for their valuable comments.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1699–1710. http://aclweb.org/anthology/D15-1198.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1(1):49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. pages 1431–1440. https://doi.org/10.1145/2806416.2806472.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1533–1544. http://www.aclweb.org/anthology/D13-1160.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1415–1425. http://www.aclweb.org/anthology/P14-1133.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*. pages 1–7.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 423–433. http://www.aclweb.org/anthology/P13-1042.

Bo Chen, Le Sun, Xianpei Han, and Bo An. 2016. Sentence rewriting for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 766–777. http://www.aclweb.org/anthology/P16-1073.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Uppsala, Sweden, pages 18–27. http://www.aclweb.org/anthology/W10-2903.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 876–885. http://www.aclweb.org/anthology/N16-1102.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 33–43. http://www.aclweb.org/anthology/P16-1004.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 875–886. https://www.aclweb.org/anthology/D17-1091.

Ruifang Ge and Raymond Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 611–619. http://www.aclweb.org/anthology/P/P09/P09-1069.

Yulan He and Steve Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech Language* 19(1):85 – 106. https://doi.org/https://doi.org/10.1016/j.csl.2004.03.001.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Comput. Linguist.* 39(4):949–998. http://dx.doi.org/10.1162/COLI$_{a0}$0158.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1821–1831. http://aclweb.org/anthology/P17-1167.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 12–22. http://www.aclweb.org/anthology/P16-1002.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 913–920. https://doi.org/10.3115/1220175.1220290.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1516–1526. https://www.aclweb.org/anthology/D17-1160.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 754–765. http://www.aclweb.org/anthology/D12-1069.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 1223–1233. http://www.aclweb.org/anthology/D10-1119.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1545–1556. http://www.aclweb.org/anthology/D13-1161.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 1512–1523. http://www.aclweb.org/anthology/D11-1140.

Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2015. Improving semantic parsing with enriched synchronous context-free grammar. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1455–1465. http://aclweb.org/anthology/D15-1170.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 23–33. http://aclweb.org/anthology/P17-1003.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 590–599. http://www.aclweb.org/anthology/P11-1060.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, pages 783–792. http://www.aclweb.org/anthology/D08-1082.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.* 34(4):513–553. http://dx.doi.org/10.1162/coli.07-056-R1-07-027.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 933–943. http://www.aclweb.org/anthology/P13-1092.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 878–888. http://www.aclweb.org/anthology/P15-1085.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1139–1149. http://aclweb.org/anthology/P17-1105.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392. http://aclweb.org/anthology/Q14-1030.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics* 4:127–140. http://sivareddy.in/papers/reddy2016transforming.pdf.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 89–101. https://www.aclweb.org/anthology/D17-1009.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 76–85. http://www.aclweb.org/anthology/P16-1008.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 366–375. http://www.aclweb.org/anthology/N15-1040.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015b. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1332–1342. http://www.aclweb.org/anthology/P15-1129.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 960–967. http://www.aclweb.org/anthology/P07-1121.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1341–1350. http://www.aclweb.org/anthology/P16-1127.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1321–1331. http://www.aclweb.org/anthology/P15-1128.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 201–206. http://anthology.aclweb.org/P16-2033.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*. AAAI Press/MIT Press, Portland, OR, pages 1050–1055. http://www.cs.utexas.edu/users/ai-lab/?zelle:aaai96.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, pages 678–687. http://www.aclweb.org/anthology/D/D07/D07-1071.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*. pages 658–666.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1527–1536. http://www.aclweb.org/anthology/N15-1176.

On the Limitations of Unsupervised Bilingual Dictionary Induction

Anders Søgaard[♡] **Sebastian Ruder**[♠♣] **Ivan Vulić**[◇]
[♡]University of Copenhagen, Copenhagen, Denmark
[♠]Insight Research Centre, National University of Ireland, Galway, Ireland
[♣]Aylien Ltd., Dublin, Ireland
[◇]Language Technology Lab, University of Cambridge, UK
soegaard@di.ku.dk,sebastian@ruder.io,iv250@cam.ac.uk

Abstract

Unsupervised machine translation—i.e., not assuming *any* cross-lingual supervision signal, whether a dictionary, translations, or comparable corpora—seems impossible, but nevertheless, Lample et al. (2018a) recently proposed a fully unsupervised machine translation (MT) model. The model relies heavily on an adversarial, unsupervised alignment of word embedding spaces for *bilingual dictionary induction* (Conneau et al., 2018), which we examine here. Our results identify the limitations of current unsupervised MT: unsupervised bilingual dictionary induction performs much worse on morphologically rich languages that are not dependent marking, when monolingual corpora from different domains or different embedding algorithms are used. We show that a simple trick, exploiting a weak supervision signal from identical words, enables more robust induction, and establish a near-perfect correlation between unsupervised bilingual dictionary induction performance and a previously unexplored graph similarity metric.

1 Introduction

Cross-lingual word representations enable us to reason about word meaning in multilingual contexts and facilitate cross-lingual transfer (Ruder et al., 2018). Early cross-lingual word embedding models relied on large amounts of parallel data (Klementiev et al., 2012; Mikolov et al., 2013a), but more recent approaches have tried to minimize the amount of supervision necessary (Vulić and Korhonen, 2016; Levy et al., 2017; Artetxe et al., 2017). Some researchers have even presented *unsupervised* methods that do not rely on any form of cross-lingual supervision at all (Barone, 2016; Conneau et al., 2018; Zhang et al., 2017).

Unsupervised cross-lingual word embeddings hold promise to induce bilingual lexicons and machine translation models in the absence of dictionaries and translations (Barone, 2016; Zhang et al., 2017; Lample et al., 2018a), and would therefore be a major step toward machine translation to, from, or even between low-resource languages.

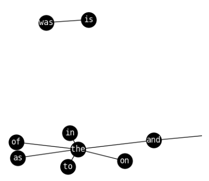
Unsupervised approaches to learning cross-lingual word embeddings are based on the assumption that monolingual word embedding graphs are approximately isomorphic, that is, after removing a small set of vertices (words) (Mikolov et al., 2013b; Barone, 2016; Zhang et al., 2017; Conneau et al., 2018). In the words of Barone (2016):

> *. . . we hypothesize that, if languages are used to convey thematically similar information in similar contexts, these random processes should be approximately isomorphic between languages, and that this isomorphism can be learned from the statistics of the realizations of these processes, the monolingual corpora, in principle without any form of explicit alignment.*

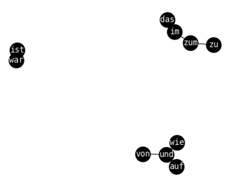
Our results indicate this assumption is not true in general, and that approaches based on this assumption have important limitations.

Contributions We focus on the recent state-of-the-art unsupervised model of Conneau et al. (2018).[1] Our contributions are: (a) In §2, we show that the monolingual word embeddings used in Conneau et al. (2018) are *not* approximately isomorphic, using the VF2 algorithm (Cordella et al., 2001) and we therefore introduce a metric for quantifying the similarity of word embeddings, based on Laplacian eigenvalues. (b) In §3, we identify circumstances under which the unsupervised bilingual

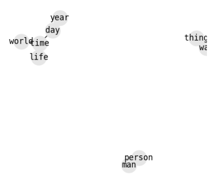
[1]Our motivation for this is that Artetxe et al. (2017) use small dictionary seeds for supervision, and Barone (2016) seems to obtain worse performance than Conneau et al. (2018). Our results should extend to Barone (2016) and Zhang et al. (2017), which rely on very similar methodology.



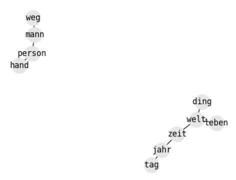
(a) Top 10 most frequent English words



(b) German translations of top 10 most frequent English words



(c) Top 10 most frequent English nouns



(d) German translations of top 10 most frequent English nouns

Figure 1: Nearest neighbor graphs.

dictionary induction (BDI) algorithm proposed in Conneau et al. (2018) does not lead to good performance. (c) We show that a simple trick, exploiting a weak supervision signal from words that are identical across languages, makes the algorithm much more robust. Our main finding is that the performance of unsupervised BDI depends heavily on all three factors: the language pair, the comparability of the monolingual corpora, and the parameters of the word embedding algorithms.

2 How similar are embeddings across languages?

As mentioned, recent work focused on unsupervised BDI assumes that monolingual word embedding spaces (or at least the subgraphs formed by the most frequent words) are approximately isomorphic. In this section, we show, by investigating the nearest neighbor graphs of word embedding spaces, that word embeddings are far from isomorphic. We therefore introduce a method for computing the similarity of non-isomorphic graphs. In §4.7, we correlate our similarity metric with performance on unsupervised BDI.

Isomorphism To motivate our study, we first establish that word embeddings are far from graph isomorphic[2]—even for two closely related languages, English and German, and using embeddings induced from comparable corpora (Wikipedia) with the same hyper-parameters.

If we take the top k most frequent words in English, and the top k most frequent words in German, and build nearest neighbor graphs for English and German using the monolingual word embeddings used in Conneau et al. (2018), the graphs are of course very different. This is, among other things, due to German case and the fact that *the* translates into *der*, *die*, and *das*, but unsupervised alignment does not have access to this kind of information. *Even* if we consider the top k most frequent English words *and their translations* into German, the nearest neighbor graphs are not isomorphic. Figure 1a-b shows the nearest neighbor graphs of the top 10 most frequent English words on Wikipedia, and their German translations.

Word embeddings are particularly good at capturing relations between nouns, but even if we consider the top k most frequent English *nouns* and their translations, the graphs are not isomorphic; see Figure 1c-d. We take this as evidence that word embeddings are not approximately isomorphic across languages. We also ran graph isomorphism checks on 10 random samples of frequent English nouns and their translations into Spanish, and only in 1/10 of the samples were the corresponding nearest neighbor graphs isomorphic.

Eigenvector similarity Since the nearest neighbor graphs are not isomorphic, even for frequent translation pairs in neighboring languages, we want to quantify the potential for unsupervised BDI using a metric that captures varying degrees of graph similarity. Eigenvalues are compact representations of global properties of graphs, and we introduce a spectral metric based on Laplacian eigenvalues (Shigehalli and Shettar, 2011) that quantifies the extent to which the nearest neighbor graphs are *isospectral*. Note that (approximately) isospectral graphs need not be (approximately) isomorphic, but (approximately) isomorphic graphs are always (approximately) isospectral (Gordon et al., 1992). Let A_1 and A_2 be the adjacency matrices of the nearest neighbor graphs G_1 and G_2 of our two word embeddings, respectively. Let $L_1 = D_1 - A_1$ and $L_2 = D_2 - A_2$ be the Laplacians of the nearest neighbor graphs, where D_1 and D_2 are the corresponding diagonal matrices of degrees. We now

[2]Two graphs that contain the same number of graph vertices connected in the same way are said to be isomorphic. In the context of weighted graphs such as word embeddings, a weak version of this is to require that the underlying nearest neighbor graphs for the most frequent k words are isomorphic.

compute the eigensimilarity of the Laplacians of the nearest neighbor graphs, L_1 and L_2. For each graph, we find the smallest k such that the sum of the k largest Laplacian eigenvalues is <90% of the Laplacian eigenvalues. We take the smallest k of the two, and use the sum of the squared differences between the largest k Laplacian eigenvalues Δ as our similarity metric.

$$\Delta = \sum_{i=1}^{k} (\lambda_{1_i} - \lambda_{2_i})^2$$

where k is chosen s.t.

$$\min_j \{ \frac{\sum_{i=1}^{k} \lambda_{ji}}{\sum_{i=1}^{n} \lambda_{ji}} > 0.9 \}$$

Note that $\Delta = 0$ means the graphs are isospectral, and the metric goes to infinite. Thus, the higher Δ is, the *less* similar the graphs (i.e., their Laplacian spectra). We discuss the correlation between unsupervised BDI performance and approximate isospectrality or eigenvector similarity in §4.7.

3 Unsupervised cross-lingual learning

3.1 Learning scenarios

Unsupervised neural machine translation relies on BDI using cross-lingual embeddings (Lample et al., 2018a; Artetxe et al., 2018), which in turn relies on the assumption that word embedding graphs are approximately isomorphic. The work of Conneau et al. (2018), which we focus on here, also makes several implicit assumptions that may or may not be necessary to achieve such isomorphism, and which may or may not scale to low-resource languages. The algorithms are not intended to be limited to learning scenarios where these assumptions hold, but since they do in the reported experiments, it is important to see to what extent these assumptions are necessary for the algorithms to produce useful embeddings or dictionaries.

We focus on the work of Conneau et al. (2018), who present a fully unsupervised approach to aligning monolingual word embeddings, induced using *fastText* (Bojanowski et al., 2017). We describe the learning algorithm in §3.2. Conneau et al. (2018) consider a specific set of learning scenarios:

(a) The authors work with the following **languages**: English-{French, German, Chinese, Russian, Spanish}. These languages, except French, are dependent marking (Table 1).[3] We evaluate Conneau et al. (2018) on (English to) Estonian (ET), Finnish (FI), Greek (EL), Hungarian (HU), Polish (PL), and Turkish (TR) in §4.2, to test whether the selection of languages in the original study introduces a bias.

(b) The monolingual corpora in their experiments are comparable; Wikipedia corpora are used, except for an experiment in which they include Google Gigawords. We evaluate across different **domains**, i.e., on all combinations of Wikipedia, EuroParl, and the EMEA medical corpus, in §4.3. We believe such scenarios are more realistic for low-resource languages.

(c) The monolingual embedding models are induced using the same **algorithms** with the same **hyper-parameters**. We evaluate Conneau et al. (2018) on pairs of embeddings induced with different hyper-parameters in §4.4. While keeping hyper-parameters fixed is always possible, it is of practical interest to know whether the unsupervised methods work on any set of pre-trained word embeddings.

We also investigate the sensitivity of unsupervised BDI to the **dimensionality** of the monolingual word embeddings in §4.5. The motivation for this is that dimensionality reduction will alter the geometric shape and remove characteristics of the embedding graphs that are important for alignment; but on the other hand, lower dimensionality introduces regularization, which will make the graphs more similar. Finally, in §4.6, we investigate the impact of different types of query **test words** on performance, including how performance varies across part-of-speech word classes and on shared vocabulary items.

3.2 Summary of Conneau et al. (2018)

We now introduce the method of Conneau et al. (2018).[4] The approach builds on existing work on learning a mapping between monolingual word embeddings (Mikolov et al., 2013b; Xing et al., 2015) and consists of the following steps: 1) **Monolingual word embeddings:** An off-the-shelf word embedding algorithm (Bojanowski et al., 2017) is used to learn source and target language spaces X

[3] A dependent-marking language marks agreement and case more commonly on dependents than on heads.

[4] https://github.com/facebookresearch/MUSE

and Y. 2) **Adversarial mapping:** A translation matrix W is learned between the spaces X and Y using adversarial techniques (Ganin et al., 2016). A discriminator is trained to discriminate samples from the translated source space WX from the target space Y, while W is trained to prevent this. This, again, is motivated by the assumption that source and target language word embeddings are approximately isomorphic. 3) **Refinement (Procrustes analysis):** W is used to build a small bilingual dictionary of frequent words, which is pruned such that only bidirectional translations are kept (Vulić and Korhonen, 2016). A new translation matrix W that translates between the spaces X and Y of these frequent word pairs is then induced by solving the Orthogonal Procrustes problem:

$$W^* = \text{argmin}_W \|WX - Y\|_\text{F} = UV^\top \quad \text{s.t. } U\Sigma V^\top = \text{SVD}(YX^\top) \tag{1}$$

This step can be used iteratively by using the new matrix W to create new seed translation pairs. It requires frequent words to serve as reliable anchors for learning a translation matrix. In the experiments in Conneau et al. (2018), as well as in ours, the iterative Procrustes refinement improves performance across the board. 4) **Cross-domain similarity local scaling** (CSLS) is used to expand high-density areas and condense low-density ones, for more accurate nearest neighbor calculation, CSLS reduces the hubness problem in high-dimensional spaces (Radovanović et al., 2010; Dinu et al., 2015). It relies on the mean similarity of a source language embedding x to its K target language nearest neighbours ($K = 10$ suggested) $nn_1, \ldots, nn_K$:

$$mnn_T(x) = \frac{1}{K}\sum_{i=1}^{K} cos(x, nn_i) \tag{2}$$

where cos is the cosine similarity. $mnn_S(y)$ is defined in an analogous manner for any target language embedding y. $CSLS(x, y)$ is then calculated as follows:

$$2cos(x, y) - mnn_T(x) - mnn_S(y) \tag{3}$$

3.3 A simple supervised method

Instead of learning cross-lingual embeddings completely without supervision, we can extract inexpensive supervision signals by harvesting identically spelled words as in, e.g. (Artetxe et al., 2017; Smith et al., 2017). Specifically, we use identically spelled words that occur in the vocabularies of both languages as bilingual seeds, without employing any additional transliteration or lemmatization/normalization methods. Using this seed dictionary, we then run the refinement step using Procrustes analysis of Conneau et al. (2018).

4 Experiments

In the following experiments, we investigate the robustness of unsupervised cross-lingual word embedding learning, varying the language pairs, monolingual corpora, hyper-parameters, etc., to obtain a better understanding of when and why unsupervised BDI works.

Task: Bilingual dictionary induction After the shared cross-lingual space is induced, given a list of N source language words $x_{u,1}, \ldots, x_{u,N}$, the task is to find a target language word t for each *query word* x_u relying on the representations in the space. t_i is the target language word closest to the source language word $x_{u,i}$ in the induced cross-lingual space, also known as the *cross-lingual nearest neighbor*. The set of learned N $(x_{u,i}, t_i)$ pairs is then run against a gold standard dictionary.

We use bilingual dictionaries compiled by Conneau et al. (2018) as gold standard, and adopt their evaluation procedure: each test set in each language consists of 1500 gold translation pairs. We rely on CSLS for retrieving the nearest neighbors, as it consistently outperformed the cosine similarity in all our experiments. Following a standard evaluation practice (Vulić and Moens, 2013; Mikolov et al., 2013b; Conneau et al., 2018), we report *Precision at 1* scores (P@1): how many times one of the correct translations of a source word w is retrieved as the nearest neighbor of w in the target language.

4.1 Experimental setup

Our default experimental setup closely follows the setup of Conneau et al. (2018). For each language we induce monolingual word embeddings for all languages from their respective tokenized and lowercased Polyglot Wikipedias (Al-Rfou et al., 2013) using *fastText* (Bojanowski et al., 2017). Only words with more than 5 occurrences are retained for training. Our *fastText* setup relies on skip-gram with negative sampling (Mikolov et al., 2013a) with standard hyper-parameters: bag-of-words contexts with the window size 2, 15 negative samples, subsampling rate 10^{-4}, and character n-gram length

	Marking	**Type**	**# Cases**
English (EN)	dependent	isolating	None
French (FR)	mixed	fusional	None
German (DE)	dependent	fusional	4
Chinese (ZH)	dependent	isolating	None
Russian (RU)	dependent	fusional	6–7
Spanish (ES)	dependent	fusional	None
Estonian (ET)	mixed	agglutinative	10+
Finnish (FI)	mixed	agglutinative	10+
Greek (EL)	double	fusional	3
Hungarian (HU)	dependent	agglutinative	10+
Polish (PL)	dependent	fusional	6–7
Turkish (TR)	dependent	agglutinative	6–7

Table 1: Languages in Conneau et al. (2018) and in our experiments (lower half)

	Unsupervised (Adversarial)	**Supervised** (Identical)	**Similarity** (Eigenvectors)
EN-ES	81.89	**82.62**	2.07
EN-ET	00.00	**31.45**	6.61
EN-FI	00.09	**28.01**	7.33
EN-EL	00.07	**42.96**	5.01
EN-HU	45.06	**46.56**	3.27
EN-PL	46.83	**52.63**	2.56
EN-TR	32.71	**39.22**	3.14
ET-FI	**29.62**	24.35	3.98

Table 2: Bilingual dictionary induction scores (P@1$\times$100%) using **a)** the unsupervised method with adversarial training; **b)** the supervised method with a bilingual seed dictionary consisting of identical words (shared between the two languages). The third columns lists eigenvector similarities between 10 randomly sampled source language nearest neighbor subgraphs of 10 nodes and the subgraphs of their translations, all from the benchmark dictionaries in Conneau et al. (2018).

3-6. All embeddings are 300-dimensional.

As we analyze the impact of various modeling assumptions in the following sections (e.g., domain differences, algorithm choices, hyper-parameters), we also train monolingual word embeddings using other corpora and different hyper-parameter choices. Quick summaries of each experimental setup are provided in the respective subsections.

4.2 Impact of language similarity

Conneau et al. (2018) present results for several target languages: Spanish, French, German, Russian, Chinese, and Esperanto. All languages but Esperanto are isolating or exclusively concatenating languages from a morphological point of view. All languages but French are dependent-marking. Table 1 lists three important morphological properties of the languages involved in their/our experiments.

Agglutinative languages with mixed or double marking show more morphological variance with content words, and we speculate whether unsupervised BDI is challenged by this kind of morphological complexity. To evaluate this, we experiment with Estonian and Finnish, and we include Greek, Hungarian, Polish, and Turkish to see how their approach fares on combinations of these two morphological traits.

We show results in the left column of Table 2. The results are quite dramatic. The approach achieves impressive performance for Spanish, one of the languages Conneau et al. (2018) include in their paper. For the languages we add here, performance is less impressive. For the languages with dependent marking (Hungarian, Polish, and Turkish), P@1 scores are still reasonable, with Turkish being slightly lower (0.327) than the others. However, for Estonian and Finnish, the method fails completely. Only in less than 1/1000 cases does a nearest neighbor search in the induced embeddings return a correct translation of a query word.[5]

The sizes of Wikipedias naturally vary across languages: e.g., *fastText* trains on approximately 16M sentences and 363M word tokens for Spanish, while it trains on 1M sentences and 12M words for Finnish. However, the difference in performance cannot be explained by the difference in training data sizes. To verify that near-zero performance in Finnish is not a result of insufficient training data, we have conducted another experiment using the large **Finnish WaC corpus** (Ljubešić et al., 2016) containing 1.7B words in total (this is similar in size to the English Polyglot Wikipedia). However, even with this large Finnish corpus, the model does not induce anything useful: P@1 equals 0.0.

We note that while languages with mixed marking may be harder to align, it seems unsupervised BDI is possible between similar, mixed marking languages. So while unsupervised learning fails for English-Finnish and English-Estonian, performance is reasonable and stable for the more similar Estonian-Finnish pair (Table 2). In general, unsupervised BDI, using the approach in Conneau et al. (2018), seems challenged when pairing En-

[5]We note, though, that varying our random seed, performance for Estonian, Finnish, and Greek is sometimes (approximately 1 out of 10 runs) *on par* with Turkish. Detecting main causes and remedies for the inherent instability of adversarial training is one the most important avenues for future research.

glish with languages that are not isolating and do not have dependent marking.[6]

The promise of zero-supervision models is that we can learn cross-lingual embeddings even for low-resource languages. On the other hand, a similar distribution of embeddings requires languages to be similar. This raises the question whether we need fully unsupervised methods at all. In fact, our supervised method that relies on very naive supervision in the form of identically spelled words leads to competitive performance for similar language pairs and better results for dissimilar pairs. The fact that we can reach competitive and more robust performance with such a simple heuristic questions the true applicability of fully unsupervised approaches and suggests that it might often be better to rely on available weak supervision.

4.3 Impact of domain differences

Monolingual word embeddings used in Conneau et al. (2018) are induced from Wikipedia, a near-parallel corpus. In order to assess the sensitivity of unsupervised BDI to the comparability and domain similarity of the monolingual corpora, we replicate the experiments in Conneau et al. (2018) using combinations of word embeddings extracted from three different domains: **1)** parliamentary proceedings from EuroParl.v7 (Koehn, 2005), **2)** Wikipedia (Al-Rfou et al., 2013), and **3)** the EMEA corpus in the medical domain (Tiedemann, 2009). We report experiments with three language pairs: English-{Spanish, Finnish, Hungarian}.

To control for the corpus size, we restrict each corpus in each language to 1.1M sentences in total (i.e., the number of sentences in the smallest, EMEA corpus). 300-dim *fastText* vectors are induced as in §4.1, retaining all words with more than 5 occurrences in the training data. For each pair of monolingual corpora, we compute their domain (dis)similarity by calculating the Jensen-Shannon divergence (El-Gamal, 1991), based on term distributions.[7] The domain similarities are displayed in Figures 2a–c.[8]

We show the results of unsupervised BDI in Figures 2g–i. For Spanish, we see good performance in all three cases where the English and Spanish corpora are from the same domain. *When the two corpora are from different domains, performance is close to zero.* For Finnish and Hungarian, performance is always poor, suggesting that more data is needed, even when domains are similar. This is in sharp contrast with the results of our minimally supervised approach (Figures 2d–f) based on identical words, which achieves decent performance in many set-ups.

We also observe a strong decrease in P@1 for English-Spanish (from 81.19% to 46.52%) when using the smaller Wikipedia corpora. This result indicates the importance of procuring large monolingual corpora from similar domains in order to enable unsupervised dictionary induction. However, resource-lean languages, for which the unsupervised method was designed in the first place, cannot be guaranteed to have as large monolingual training corpora as available for English, Spanish or other major resource-rich languages.

4.4 Impact of hyper-parameters

Conneau et al. (2018) use the same hyper-parameters for inducing embeddings for all languages. This is of course always practically possible, but we are interested in seeing whether their approach works on pre-trained embeddings induced with possibly very different hyper-parameters. We focus on two hyper-parameters: context window-size (*win*) and the parameter controlling the number of n-gram features in the *fastText* model (*chn*), while at the same time varying the underlying algorithm: *skip-gram* vs. *cbow*. The results for English-Spanish are listed in Table 3.

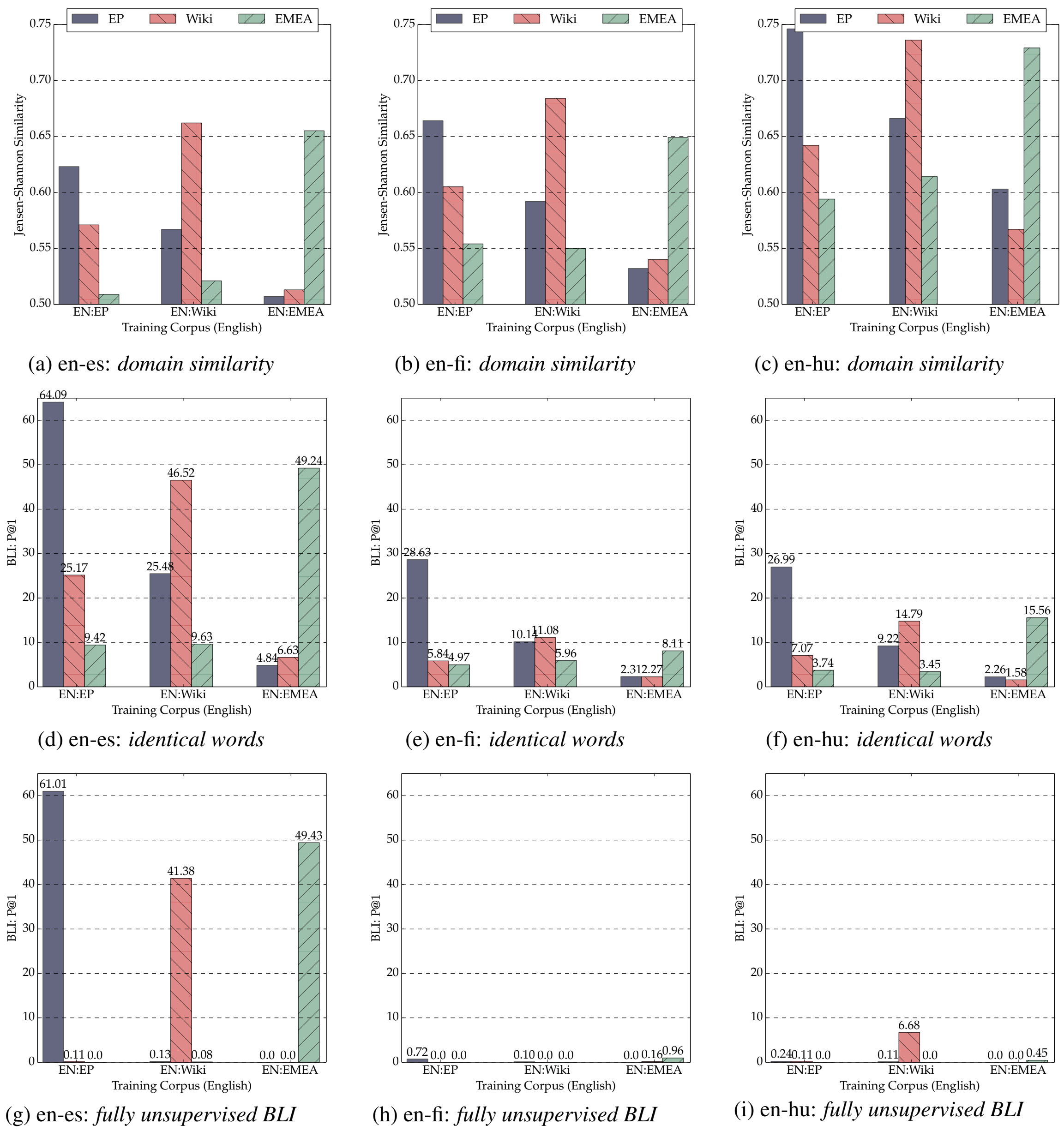
The small variations in the hyper-parameters with the same underlying algorithm (i.e., using *skip-gram* or *cbow* for both EN and ES) yield only slight drops in the final scores. Still, the best scores are obtained with the same configuration on both sides. Our main finding here is that unsupervised BDI fails (even) for EN-ES when the two monolingual embedding spaces are induced by two different algorithms (see the results of the entire Spanish *cbow* column).[9] In sum, this means that *the unsupervised approach is unlikely to work on pre-trained word embeddings unless they are induced on same-*

[6]One exception here is French, which they include in their paper, but French arguably has a relatively simple morphology.

[7]In order to get comparable term distributions, we translate the source language to the target language using the bilingual dictionaries provided by Conneau et al. (2018).

[8]We also computed $\mathcal{A}$-distances (Blitzer et al., 2007) and confirmed that trends were similar.

[9]We also checked if this result might be due to a lower-quality monolingual ES space. However, monolingual word similarity scores on available datasets in Spanish show performance comparable to that of Spanish *skip-gram* vectors: e.g., Spearman's ρ correlation is ≈ 0.7 on the ES evaluation set from SemEval-2017 Task 2 (Camacho-Collados et al., 2017).



(a) en-es: *domain similarity* (b) en-fi: *domain similarity* (c) en-hu: *domain similarity*

(d) en-es: *identical words* (e) en-fi: *identical words* (f) en-hu: *identical words*

(g) en-es: *fully unsupervised BLI* (h) en-fi: *fully unsupervised BLI* (i) en-hu: *fully unsupervised BLI*

Figure 2: Influence of language-pair *and* domain similarity on BLI performance, with three language pairs (en-es/fi/hu). **Top row, (a)-(c)**: Domain similarity (higher is more similar) computed as $dsim = 1 - JS$, where JS is Jensen-Shannon divergence; **Middle row, (d)-(f)**: baseline BLI model which learns a linear mapping between two monolingual spaces based on a set of identical (i.e., shared) words; **Bottom row, (g)-(i)**: fully unsupervised BLI model relying on the distribution-level alignment and adversarial training. Both BLI models apply the Procrustes analysis and use CSLS to retrieve nearest neighbours.

or comparable-domain, reasonably-sized training data using the same underlying algorithm.

4.5 Impact of dimensionality

We also perform an experiment on 40-dimensional monolingual word embeddings. This leads to reduced expressivity, and can potentially make the geometric shapes of embedding spaces harder to align; on the other hand, reduced dimensionality may also lead to less overfitting. We generally see worse performance (P@1 is 50.33 for Spanish, 21.81 for Hungarian, 20.11 for Polish, and 22.03 for Turkish) – but, very *interestingly*, we obtain *better performance* for Estonian (13.53), Finnish (15.33), and Greek (24.17) than we did with 300 dimensions. We hypothesize this indicates monolingual word embedding algorithms over-fit to some of the rarer peculiarities of these languages.

	English (skipgram, win=2, chn=3-6)	
	Spanish *(skipgram)*	Spanish *(cbow)*
==	81.89	00.00
≠ win=10	81.28	00.07
≠ chn=2-7	80.74	00.00
≠ win=10, chn=2-7	80.15	00.13

Table 3: Varying the underlying *fastText* algorithm and hyper-parameters. The first column lists differences in training configurations between English and Spanish monolingual embeddings.

	en-es	en-hu	en-fi
Noun	80.94	26.87	00.00
Verb	66.05	25.44	00.00
Adjective	85.53	53.28	00.00
Adverb	80.00	51.57	00.00
Other	73.00	53.40	00.00

Table 4: P@1 $\times$ 100% scores for query words with different parts-of-speech.

4.6 Impact of evaluation procedure

BDI models are evaluated on a held-out set of query words. Here, we analyze the performance of the unsupervised approach across different parts-of-speech, frequency bins, and with respect to query words that have orthographically identical counterparts in the target language with the same or a different meaning.

Part-of-speech We show the impact of the part-of-speech of the query words in Table 4; again on a representative subset of our languages. The results indicate that performance on verbs is lowest across the board. This is consistent with research on distributional semantics and verb meaning (Schwartz et al., 2015; Gerz et al., 2016).

Frequency We also investigate the impact of the frequency of query words. We calculate the word frequency of English words based on Google's Trillion Word Corpus: query words are divided in groups based on their rank – i.e., the first group contains the top 100 most frequent words, the second one contains the 101th-1000th most frequent words, etc. – and plot performance (P@1) relative to rank in Figure 3. For EN-FI, P@1 was 0 across all frequency ranks. The plot shows sensitivity to frequency for HU, but less so for ES.

Homographs Since we use identical word forms (homographs) for supervision, we investigated whether these are representative or harder to align than other words. Table 5 lists performance for three sets of query words: (a) source words that have homographs (words that are spelled the same way) with the same meaning (homonyms) in the target language, e.g., many proper names; (b) source words that have homographs that are not homonyms in the target language, e.g., many short words; and (c) other words. Somewhat surprisingly, words which have translations that are homographs, are associated with *lower* precision than other words. This is probably due to loan words and proper names, but note that using homographs as supervision for alignment, we achieve high precision for this part of the vocabulary *for free*.

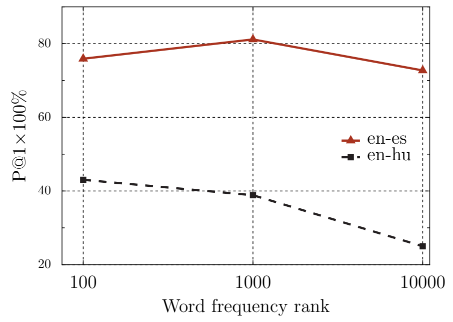


Figure 3: P@1 scores for EN-ES and EN-HU for queries with different frequency ranks.

Spelling	Meaning	en-es	en-hu	en-fi
Same	Same	45.94	18.07	00.00
Same	Diff	39.66	29.97	00.00
Diff	Diff	62.42	34.45	00.00

Table 5: Scores (P@1 $\times$ 100%) for query words with same and different spellings and meanings.

4.7 Evaluating eigenvector similarity

Finally, in order to get a better understanding of the limitations of unsupervised BDI, we correlate the graph similarity metric described in §2 (right column of Table 2) with performance across languages (left column). Since we already established that the monolingual word embeddings are far from isomorphic—in contrast with the intuitions motivating previous work (Mikolov et al., 2013b; Barone, 2016; Zhang et al., 2017; Conneau et al., 2018)—we would like to establish another diagnostic metric that identifies embedding spaces for which the approach in Conneau et al. (2018) is likely to work. Differences in morphology, domain, or embedding parameters seem to be predictive of poor performance, but a metric that is independent of linguistic

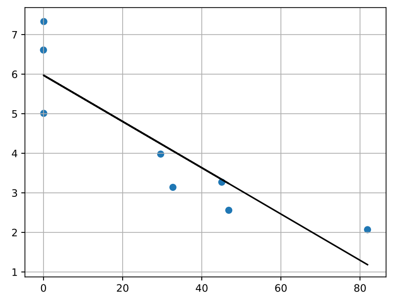


Figure 4: Strong correlation ($\rho = 0.89$) between BDI performance (x) and graph similarity (y)

categorizations and the characteristics of the monolingual corpora would be more widely applicable. We plot the values in Table 2 in Figure 4. Recall that our graph similarity metric returns a value in the half-open interval $[0, \infty)$. The correlation between BDI performance and graph similarity is strong ($\rho \sim 0.89$).

5 Related work

Cross-lingual word embeddings Cross-lingual word embedding models typically, unlike Conneau et al. (2018), require aligned words, sentences, or documents (Levy et al., 2017). Most approaches based on word alignments learn an explicit mapping between the two embedding spaces (Mikolov et al., 2013b; Xing et al., 2015). Recent approaches try to minimize the amount of supervision needed (Vulić and Korhonen, 2016; Artetxe et al., 2017; Smith et al., 2017). See Upadhyay et al. (2016) and Ruder et al. (2018) for surveys.

Unsupervised cross-lingual learning Haghighi et al. (2008) were first to explore unsupervised BDI, using features such as context counts and orthographic substrings, and canonical correlation analysis. Recent approaches use adversarial learning (Goodfellow et al., 2014) and employ a discriminator, trained to distinguish between the translated source and the target language space, and a generator learning a translation matrix (Barone, 2016). Zhang et al. (2017), in addition, use different forms of regularization for convergence, while Conneau et al. (2018) uses additional steps to refine the induced embedding space.

Unsupervised machine translation Research on unsupervised machine translation (Lample et al., 2018a; Artetxe et al., 2018; Lample et al., 2018b) has generated a lot of interest recently with a promise to support the construction of MT systems for and between resource-poor languages. All unsupervised NMT methods critically rely on accurate unsupervised BDI and back-translation. Models are trained to reconstruct a corrupted version of the source sentence and to translate its translated version back to the source language. Since the crucial input to these systems are indeed cross-lingual word embedding spaces induced in an unsupervised fashion, in this paper we also implicitly investigate one core limitation of such unsupervised MT techniques.

6 Conclusion

We investigated when unsupervised BDI (Conneau et al., 2018) is possible and found that differences in morphology, domains or word embedding algorithms may challenge this approach. Further, we found eigenvector similarity of sampled nearest neighbor subgraphs to be predictive of unsupervised BDI performance. We hope that this work will guide further developments in this new and exciting field.

Acknowledgments

We thank the anonymous reviewers, as well as Hinrich Schütze and Yova Kementchedjhieva, for their valuable feedback. Anders is supported by the ERC Starting Grant LOWLANDS No. 313695 and a Google Focused Research Award. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289. Ivan is supported by the ERC Consolidator Grant LEXICAL No. 648909.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*, pages 451–462.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of ICLR (Conference Track)*.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*, 1, pages 440–447.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:125–136.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 Task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of SEMEVAL*, pages 15–26.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. *Proceedings of ICLR*.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. 2001. An improved algorithm for matching large graphs. *Proceedings of the 3rd IAPR TC-15 Workshop on Graphbased Representations in Pattern Recognition*, 17:1–35.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR (Workshop Papers)*.

M. A El-Gamal. 1991. The role of priors in active Bayesian learning in the sequential statistical decision framework. In *Maximum Entropy and Bayesian Methods*, pages 33–38. Springer Netherlands.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:1–35.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*, pages 2672–2680.

Carolyn Gordon, David L. Webb, and Scott Wolpert. 1992. One cannot hear the shape of a drum. *Bulletin of the American Mathematical Society*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, pages 771–779.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING*, pages 1459–1474.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit (MT SUMMIT)*, pages 79–86.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *Proceedings of ICLR (Conference Papers)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. *CoRR*, abs/1804.07755.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of EACL*, pages 765–774.

Nikola Ljubešić, Tommi Pirinen, and Antonio Toral. 2016. Finnish Web corpus fiWaC 1.0. Slovenian language resource repository CLARIN.SI.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation.

Milos Radovanović, Alexandros Nanopoulos, and Mirjana Ivanovic. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2018. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*, pages 258–267.

Vijayalaxmi Shigehalli and Vidya Shettar. 2011. Spectral technique using normalized adjacency matrices for graph matching. *International Journal of Computational Science and Mathematics*, 3:371–378.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR (Conference Papers)*.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Proceedings of RANLP*, pages 237–248.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*, pages 1661–1670.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*, pages 247–257.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of EMNLP*, pages 1613–1624.

Chao Xing, Chao Liu, Dong Wang, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of NAACL-HLT*, pages 1005–1010.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of ACL*, pages 1959–1970.

A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings

Mikel Artetxe and **Gorka Labaka** and **Eneko Agirre**
IXA NLP Group
University of the Basque Country (UPV/EHU)
{mikel.artetxe,gorka.labaka,e.agirre}@ehu.eus

Abstract

Recent work has managed to learn cross-lingual word embeddings without parallel data by mapping monolingual embeddings to a shared space through adversarial training. However, their evaluation has focused on favorable conditions, using comparable corpora or closely-related languages, and we show that they often fail in more realistic scenarios. This work proposes an alternative approach based on a fully unsupervised initialization that explicitly exploits the structural similarity of the embeddings, and a robust self-learning algorithm that iteratively improves this solution. Our method succeeds in all tested scenarios and obtains the best published results in standard datasets, even surpassing previous supervised systems. Our implementation is released as an open source project at `https://github.com/artetxem/vecmap`.

1 Introduction

Cross-lingual embedding mappings have shown to be an effective way to learn bilingual word embeddings (Mikolov et al., 2013; Lazaridou et al., 2015). The underlying idea is to independently train the embeddings in different languages using monolingual corpora, and then map them to a shared space through a linear transformation. This allows to learn high-quality cross-lingual representations without expensive supervision, opening new research avenues like unsupervised neural machine translation (Artetxe et al., 2018b; Lample et al., 2018).

While most embedding mapping methods rely on a small seed dictionary, adversarial training has recently produced exciting results in fully unsupervised settings (Zhang et al., 2017a,b; Conneau et al., 2018). However, their evaluation has focused on particularly favorable conditions, limited to closely-related languages or comparable Wikipedia corpora. When tested on more realistic scenarios, we find that they often fail to produce meaningful results. For instance, none of the existing methods works in the standard English-Finnish dataset from Artetxe et al. (2017), obtaining translation accuracies below 2% in all cases (see Section 5).

On another strand of work, Artetxe et al. (2017) showed that an iterative self-learning method is able to bootstrap a high quality mapping from very small seed dictionaries (as little as 25 pairs of words). However, their analysis reveals that the self-learning method gets stuck in poor local optima when the initial solution is not good enough, thus failing for smaller training dictionaries.

In this paper, we follow this second approach and propose a new unsupervised method to build an initial solution without the need of a seed dictionary, based on the observation that, given the similarity matrix of all words in the vocabulary, each word has a different distribution of similarity values. Two equivalent words in different languages should have a similar distribution, and we can use this fact to induce the initial set of word pairings (see Figure 1). We combine this initialization with a more robust self-learning method, which is able to start from the weak initial solution and iteratively improve the mapping. Coupled together, we provide a fully unsupervised cross-lingual mapping method that is effective in realistic settings, converges to a good solution in all cases tested, and sets a new state-of-the-art in bilingual lexicon extraction, even surpassing previous supervised methods.

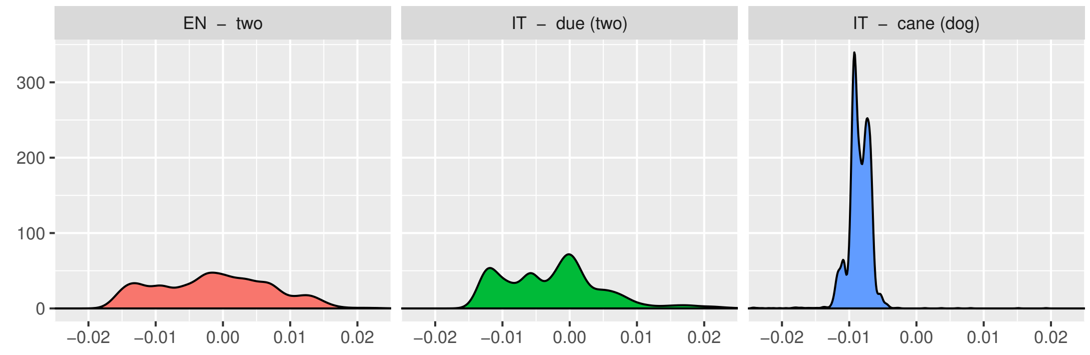


Figure 1: Motivating example for our unsupervised initialization method, showing the similarity distributions of three words (corresponding to the smoothed density estimates from the normalized square root of the similarity matrices as defined in Section 3.2). Equivalent translations (two and *due*) have more similar distributions than non-related words (two and *cane* - meaning dog). This observation is used to build an initial solution that is later improved through self-learning.

2 Related work

Cross-lingual embedding mapping methods work by independently training word embeddings in two languages, and then mapping them to a shared space using a linear transformation.

Most of these methods are **supervised**, and use a bilingual dictionary of a few thousand entries to learn the mapping. Existing approaches can be classified into regression methods, which map the embeddings in one language using a least-squares objective (Mikolov et al., 2013; Shigeto et al., 2015; Dinu et al., 2015), canonical methods, which map the embeddings in both languages to a shared space using canonical correlation analysis and extensions of it (Faruqui and Dyer, 2014; Lu et al., 2015), orthogonal methods, which map the embeddings in one or both languages under the constraint of the transformation being orthogonal (Xing et al., 2015; Artetxe et al., 2016; Zhang et al., 2016; Smith et al., 2017), and margin methods, which map the embeddings in one language to maximize the margin between the correct translations and the rest of the candidates (Lazaridou et al., 2015). Artetxe et al. (2018a) showed that many of them could be generalized as part of a multi-step framework of linear transformations.

A related research line is to adapt these methods to the **semi-supervised** scenario, where the training dictionary is much smaller and used as part of a bootstrapping process. While similar ideas where already explored for traditional count-based vector space models (Peirsman and Padó, 2010; Vulić and Moens, 2013), Artetxe et al. (2017) brought this approach to pre-trained low-dimensional word embeddings, which are more widely used nowadays. More concretely, they proposed a self-learning approach that alternates the mapping and dictionary induction steps iteratively, obtaining results that are comparable to those of supervised methods when starting with only 25 word pairs.

A practical approach for reducing the need of bilingual supervision is to design **heuristics to build the seed dictionary**. The role of the seed lexicon in learning cross-lingual embedding mappings is analyzed in depth by Vulić and Korhonen (2016), who propose using document-aligned corpora to extract the training dictionary. A more common approach is to rely on shared words and cognates (Peirsman and Padó, 2010; Smith et al., 2017), while Artetxe et al. (2017) go further and restrict themselves to shared numerals. However, while these approaches are meant to eliminate the need of bilingual data in practice, they also make strong assumptions on the writing systems of languages (e.g. that they all use a common alphabet or Arabic numerals). Closer to our work, a recent line of **fully unsupervised** approaches drops these assumptions completely, and attempts to learn cross-lingual embedding mappings based on distributional information alone. For that purpose, existing methods rely on adversarial training. This was first proposed by Miceli Barone (2016), who combine an encoder that maps source language embeddings into the target language, a decoder that reconstructs the source language embeddings from the mapped embeddings, and a discriminator that discriminates between the mapped embeddings and the true target language embeddings. Despite promising, they conclude that their model "is not competitive with other cross-lingual representation approaches". Zhang et al. (2017a) use a very similar architecture, but incorporate additional techniques like noise injection to aid training and report competitive results on bilingual lexicon extraction. Conneau et al. (2018) drop the reconstruction component, regularize the mapping to be orthogonal, and incorporate an iterative refinement process akin to self-learning, reporting very strong results on a large bilingual lexicon extraction dataset. Finally, Zhang et al. (2017b) adopt the earth mover's distance for training, optimized through a Wasserstein generative adversarial network followed by an alternating optimization procedure. However, all this previous work used comparable Wikipedia corpora in most experiments and, as shown in Section 5, face difficulties in more challenging settings.

3 Proposed method

Let X and Z be the word embedding matrices in two languages, so that their ith row X_{i*} and Z_{i*} denote the embeddings of the ith word in their respective vocabularies. Our goal is to learn the linear transformation matrices W_X and W_Z so the mapped embeddings XW_X and ZW_Z are in the same cross-lingual space. At the same time, we aim to build a dictionary between both languages, encoded as a sparse matrix D where $D_{ij} = 1$ if the jth word in the target language is a translation of the ith word in the source language.

Our proposed method consists of four sequential steps: a pre-processing that normalizes the embeddings (§3.1), a fully unsupervised initialization scheme that creates an initial solution (§3.2), a robust self-learning procedure that iteratively improves this solution (§3.3), and a final refinement step that further improves the resulting mapping through symmetric re-weighting (§3.4).

3.1 Embedding normalization

Our method starts with a pre-processing that length normalizes the embeddings, then mean centers each dimension, and then length normalizes them again. The first two steps have been shown to be beneficial in previous work (Artetxe et al., 2016), while the second length normalization guarantees the final embeddings to have a unit length. As a result, the dot product of any two embeddings is equivalent to their cosine similarity and directly related to their Euclidean distance[1], and can be taken as a measure of their similarity.

3.2 Fully unsupervised initialization

The underlying difficulty of the mapping problem in its unsupervised variant is that the word embedding matrices X and Z are unaligned across both axes: neither the ith vocabulary item X_{i*} and Z_{i*} nor the jth dimension of the embeddings X_{*j} and Z_{*j} are aligned, so there is no direct correspondence between both languages. In order to overcome this challenge and build an initial solution, we propose to first construct two alternative representations X' and Z' that are aligned across their jth dimension X'_{*j} and Z'_{*j}, which can later be used to build an initial dictionary that aligns their respective vocabularies.

Our approach is based on a simple idea: while the axes of the original embeddings X and Z are different in nature, both axes of their corresponding similarity matrices $M_X = XX^T$ and $M_Z = ZZ^T$ correspond to words, which can be exploited to reduce the mismatch to a single axis. More concretely, assuming that the embedding spaces are perfectly isometric, the similarity matrices M_X and M_Z would be equivalent up to a permutation of their rows and columns, where the permutation in question defines the dictionary across both languages. In practice, the isometry requirement will not hold exactly, but it can be assumed to hold approximately, as the very same problem of mapping two embedding spaces without supervision would otherwise be hopeless. Based on that, one could try every possible permutation of row and column indices to find the best match between M_X and M_Z, but the resulting combinatorial explosion makes this approach intractable.

In order to overcome this problem, we propose to first sort the values in each row of M_X and M_Z, resulting in matrices $\text{sorted}(M_X)$ and $\text{sorted}(M_Z)$[2]. Under the strict isometry condition, equivalent words would get the exact same vector across languages, and thus, given a word and its row in $\text{sorted}(M_X)$, one could apply nearest neighbor retrieval over the rows of $\text{sorted}(M_Z)$ to find its corresponding translation.

On a final note, given the singular value decomposition $X = USV^T$, the similarity matrix

[1]Given two length normalized vectors u and v, $u \cdot v = \cos(u, v) = 1 - ||u - v||^2/2$.

[2]Note that the values in each row are sorted independently from other rows.

is $M_X = US^2U^T$. As such, its square root $\sqrt{M_X} = USU^T$ is closer in nature to the original embeddings, and we also find it to work better in practice. We thus compute $\mathrm{sorted}(\sqrt{M_X})$ and $\mathrm{sorted}(\sqrt{M_Z})$ and normalize them as described in Section 3.1, yielding the two matrices X' and Z' that are later used to build the initial solution for self-learning (see Section 3.3).

In practice, the isometry assumption is strong enough so the above procedure captures some cross-lingual signal. In our English-Italian experiments, the average cosine similarity across the gold standard translation pairs is 0.009 for a random solution, 0.582 for the optimal supervised solution, and 0.112 for the mapping resulting from this initialization. While the latter is far from being useful on its own (the accuracy of the resulting dictionary is only 0.52%), it is substantially better than chance, and it works well as an initial solution for the self-learning method described next.

3.3 Robust self-learning

Previous work has shown that self-learning can learn high-quality bilingual embedding mappings starting with as little as 25 word pairs (Artetxe et al., 2017). In this method, training iterates through the following two steps until convergence:

1. Compute the optimal orthogonal mapping maximizing the similarities for the current dictionary D:

$$\arg\max_{W_X, W_Z} \sum_i \sum_j D_{ij}((X_{i*}W_X) \cdot (Z_{j*}W_Z))$$

   An optimal solution is given by $W_X = U$ and $W_Z = V$, where $USV^T = X^TDZ$ is the singular value decomposition of X^TDZ.

2. Compute the optimal dictionary over the similarity matrix of the mapped embeddings $XW_XW_Z^TZ^T$. This typically uses nearest neighbor retrieval from the source language into the target language, so $D_{ij} = 1$ if $j = \mathrm{argmax}_k\,(X_{i*}W_X) \cdot (Z_{k*}W_Z)$ and $D_{ij} = 0$ otherwise.

The underlying optimization objective is independent from the initial dictionary, and the algorithm is guaranteed to converge to a local optimum of it. However, the method does not work if starting from a completely random solution, as it tends to get stuck in poor local optima in that case. For that reason, we use the unsupervised initialization procedure at Section 3.2 to build an initial solution. However, simply plugging in both methods did not work in our preliminary experiments, as the quality of this initial method is not good enough to avoid poor local optima. For that reason, we next propose some key improvements in the dictionary induction step to make self-learning more robust and learn better mappings:

- **Stochastic dictionary induction**. In order to encourage a wider exploration of the search space, we make the dictionary induction stochastic by randomly keeping some elements in the similarity matrix with probability p and setting the remaining ones to 0. As a consequence, the smaller the value of p is, the more the induced dictionary will vary from iteration to iteration, thus enabling to escape poor local optima. So as to find a fine-grained solution once the algorithm gets into a good region, we increase this value during training akin to simulated annealing, starting with $p = 0.1$ and doubling this value every time the objective function at step 1 above does not improve more than $\epsilon = 10^{-6}$ for 50 iterations.

- **Frequency-based vocabulary cutoff**. The size of the similarity matrix grows quadratically with respect to that of the vocabularies. This does not only increase the cost of computing it, but it also makes the number of possible solutions grow exponentially[3], presumably making the optimization problem harder. Given that less frequent words can be expected to be noisier, we propose to restrict the dictionary induction process to the k most frequent words in each language, where we find $k = 20{,}000$ to work well in practice.

- **CSLS retrieval**. Dinu et al. (2015) showed that nearest neighbor suffers from the hubness problem. This phenomenon is known to occur as an effect of the curse of dimensionality, and causes a few points (known as *hubs*) to be nearest neighbors of many other points (Radovanović et al., 2010a,b). Among the existing solutions to penalize the similarity score of hubs, we adopt the Cross-domain

[3]There are m^n possible combinations that go from a source vocabulary of n entries to a target vocabulary of m entries.

Similarity Local Scaling (CSLS) from Conneau et al. (2018). Given two mapped embeddings x and y, the idea of CSLS is to compute $\mathrm{r_T}(x)$ and $\mathrm{r_S}(y)$, the average cosine similarity of x and y for their k nearest neighbors in the other language, respectively. Having done that, the corrected score $\mathrm{CSLS}(x, y) = 2\cos(x, y) - \mathrm{r_T}(x) - \mathrm{r_S}(y)$. Following the authors, we set $k = 10$.

- **Bidirectional dictionary induction**. When the dictionary is induced from the source into the target language, not all target language words will be present in it, and some will occur multiple times. We argue that this might accentuate the problem of local optima, as repeated words might act as strong attractors from which it is difficult to escape. In order to mitigate this issue and encourage diversity, we propose inducing the dictionary in both directions and taking their corresponding concatenation, so $D = D_{X \to Z} + D_{Z \to X}$.

In order to build the **initial dictionary**, we compute X' and Z' as detailed in Section 3.2 and apply the above procedure over them. As the only difference, this first solution does not use the stochastic zeroing in the similarity matrix, as there is no need to encourage diversity (X' and Z' are only used once), and the threshold for vocabulary cutoff is set to $k = 4,000$, so X' and Z' can fit in memory. Having computed the initial dictionary, X' and Z' are discarded, and the remaining iterations are performed over the original embeddings X and Z.

3.4 Symmetric re-weighting

As part of their multi-step framework, Artetxe et al. (2018a) showed that re-weighting the target language embeddings according to the cross-correlation in each component greatly improved the quality of the induced dictionary. Given the singular value decomposition $USV^T = X^T DZ$, this is equivalent to taking $W_X = U$ and $W_Z = VS$, where X and Z are previously whitened applying the linear transformations $(X^T X)^{-\frac{1}{2}}$ and $(Z^T Z)^{-\frac{1}{2}}$, and later de-whitened applying $U^T (X^T X)^{\frac{1}{2}} U$ and $V^T (Z^T Z)^{\frac{1}{2}} V$.

However, re-weighting also accentuates the problem of local optima when incorporated into self-learning as, by increasing the relevance of dimensions that best match for the current solution, it discourages to explore other regions of the search space. For that reason, we propose using it as a final step once self-learning has converged to a good solution. Unlike Artetxe et al. (2018a), we apply re-weighting symmetrically in both languages, taking $W_X = US^{\frac{1}{2}}$ and $W_Z = VS^{\frac{1}{2}}$. This approach is neutral in the direction of the mapping, and gives good results as shown in our experiments.

4 Experimental settings

Following common practice, we evaluate our method on **bilingual lexicon extraction**, which measures the accuracy of the induced dictionary in comparison to a gold standard.

As discussed before, **previous evaluation** has focused on favorable conditions. In particular, existing unsupervised methods have almost exclusively been tested on Wikipedia corpora, which is comparable rather than monolingual, exposing a strong cross-lingual signal that is not available in strictly unsupervised settings. In addition to that, some datasets comprise unusually small embeddings, with only 50 dimensions and around 5,000-10,000 vocabulary items (Zhang et al., 2017a,b). As the only exception, Conneau et al. (2018) report positive results on the English-Italian dataset of Dinu et al. (2015) in addition to their main experiments, which are carried out in Wikipedia. While this dataset does use strictly monolingual corpora, it still corresponds to a pair of two relatively close indo-european languages.

In order to get a wider picture of how our method compares to previous work in different conditions, including more challenging settings, we carry out our experiments in the widely used **dataset** of Dinu et al. (2015) and the subsequent extensions of Artetxe et al. (2017, 2018a), which together comprise English-Italian, English-German, English-Finnish and English-Spanish. More concretely, the dataset consists of 300-dimensional CBOW embeddings trained on WacKy crawling corpora (English, Italian, German), Common Crawl (Finnish) and WMT News Crawl (Spanish). The gold standards were derived from dictionaries built from Europarl word alignments and available at OPUS (Tiedemann, 2012), split in a test set of 1,500 entries and a training set of 5,000 that we do not use in our experiments. The datasets are freely available. As a non-european agglutinative language, the English-Finnish pair is particularly challeng-

	ES-EN				IT-EN				TR-EN			
	best	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**
Zhang et al. (2017a), $\lambda = 1$	71.43	68.18	**10**	13.2	60.38	56.45	**10**	12.3	0.00	0.00	0	13.0
Zhang et al. (2017a), $\lambda = 10$	70.24	66.37	**10**	13.0	57.64	52.60	**10**	12.6	21.07	17.95	**10**	13.2
Conneau et al. (2018), code	76.18	75.82	**10**	25.1	**67.32**	**67.00**	**10**	25.9	32.64	14.34	5	25.3
Conneau et al. (2018), paper	76.15	75.81	**10**	25.1	67.21	60.22	9	25.5	29.79	16.48	7	25.5
Proposed method	**76.43**	**76.28**	**10**	**0.6**	66.96	66.92	**10**	**0.9**	**36.10**	**35.93**	**10**	**1.7**

Table 1: Results of unsupervised methods on the dataset of Zhang et al. (2017a). We perform 10 runs for each method and report the best and average accuracies (%), the number of successful runs (those with >5% accuracy) and the average runtime (minutes).

	EN-IT				EN-DE				EN-FI				EN-ES			
	best	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**
Zhang et al. (2017a), $\lambda = 1$	0.00	0.00	0	47.0	0.00	0.00	0	47.0	0.00	0.00	0	45.4	0.00	0.00	0	44.3
Zhang et al. (2017a), $\lambda = 10$	0.00	0.00	0	46.6	0.00	0.00	0	46.0	0.07	0.01	0	44.9	0.07	0.01	0	43.0
Conneau et al. (2018), code	45.40	13.55	3	46.1	47.27	42.15	9	45.4	1.62	0.38	0	44.4	36.20	21.23	6	45.3
Conneau et al. (2018), paper	45.27	9.10	2	45.4	0.07	0.01	0	45.0	0.07	0.01	0	44.7	35.47	7.09	2	44.9
Proposed method	**48.53**	**48.13**	**10**	**8.9**	**48.47**	**48.19**	**10**	**7.3**	**33.50**	**32.63**	**10**	**12.9**	**37.60**	**37.33**	**10**	**9.1**

Table 2: Results of unsupervised methods on the dataset of Dinu et al. (2015) and the extensions of Artetxe et al. (2017, 2018a). We perform 10 runs for each method and report the best and average accuracies (%), the number of successful runs (those with >5% accuracy) and the average runtime (minutes).

ing due to the linguistic distance between them. For completeness, we also test our method in the Spanish-English, Italian-English and Turkish-English datasets of Zhang et al. (2017a), which consist of 50-dimensional CBOW embeddings trained on Wikipedia, as well as gold standard dictionaries[4] from Open Multilingual WordNet (Spanish-English and Italian-English) and Google Translate (Turkish-English). The lower dimensionality and comparable corpora make an easier scenario, although it also contains a challenging pair of distant languages (Turkish-English).

Our method is implemented in Python using NumPy and CuPy. Together with it, we also test the **methods** of Zhang et al. (2017a) and Conneau et al. (2018) using the publicly available implementations from the authors[5]. Given that Zhang et al. (2017a) report using a different value of their hyperparameter λ for different language pairs ($\lambda = 10$ for English-Turkish and $\lambda = 1$ for the rest), we test both values in all our experiments to better understand its effect. In the case of Conneau et al. (2018), we test both the default hyperparameters in the source code as well as those reported in the paper, with iterative refinement activated in both cases. Given the instability of these methods, we perform 10 runs for each, and report the best and average accuracies, the number of successful runs (those with >5% accuracy) and the average runtime. All the experiments were run in a single Nvidia Titan Xp.

5 Results and discussion

We first present the main results (§5.1), then the comparison to the state-of-the-art (§5.2), and finally ablation tests to measure the contribution of each component (§5.3).

5.1 Main results

We report the results in the dataset of Zhang et al. (2017a) at Table 1. As it can be seen, the proposed method performs at par with that of Conneau et al. (2018) both in Spanish-English and Italian-English, but gets substantially better results in the more challenging Turkish-English pair. While we are able to reproduce the results reported by Zhang et al. (2017a), their method gets the worst results of all by a large margin. Another disadvantage of that model is that different

[4]The test dictionaries were obtained through personal communication with the authors. The rest of the language pairs were left out due to licensing issues.

[5]Despite our efforts, Zhang et al. (2017b) was left out because: 1) it does not create a one-to-one dictionary, thus difficulting direct comparison, 2) it depends on expensive proprietary software 3) its computational cost is orders of magnitude higher (running the experiments would have taken several months).

Supervision	Method	EN-IT	EN-DE	EN-FI	EN-ES
5k dict.	Mikolov et al. (2013)	34.93[†]	35.00[†]	25.91[†]	27.73[†]
	Faruqui and Dyer (2014)	38.40[*]	37.13[*]	27.60[*]	26.80[*]
	Shigeto et al. (2015)	41.53[†]	43.07[†]	31.04[†]	33.73[†]
	Dinu et al. (2015)	37.7	38.93[*]	29.14[*]	30.40[*]
	Lazaridou et al. (2015)	40.2	-	-	-
	Xing et al. (2015)	36.87[†]	41.27[†]	28.23[†]	31.20[†]
	Zhang et al. (2016)	36.73[†]	40.80[†]	28.16[†]	31.07[†]
	Artetxe et al. (2016)	39.27	41.87[*]	30.62[*]	31.40[*]
	Artetxe et al. (2017)	39.67	40.87	28.72	-
	Smith et al. (2017)	43.1	43.33[†]	29.42[†]	35.13[†]
	Artetxe et al. (2018a)	45.27	44.13	**32.94**	36.60
25 dict.	Artetxe et al. (2017)	37.27	39.60	28.16	-
Init. heurist.	Smith et al. (2017), cognates	39.9	-	-	-
	Artetxe et al. (2017), num.	39.40	40.27	26.47	-
None	Zhang et al. (2017a), $\lambda = 1$	0.00[*]	0.00[*]	0.00[*]	0.00[*]
	Zhang et al. (2017a), $\lambda = 10$	0.00[*]	0.00[*]	0.01[*]	0.01[*]
	Conneau et al. (2018), code[‡]	45.15[*]	46.83[*]	0.38[*]	35.38[*]
	Conneau et al. (2018), paper[‡]	45.1	0.01[*]	0.01[*]	35.44[*]
	Proposed method	**48.13**	**48.19**	32.63	**37.33**

Table 3: Accuracy (%) of the proposed method in comparison with previous work. [*]Results obtained with the official implementation from the authors. [†]Results obtained with the framework from Artetxe et al. (2018a). The remaining results were reported in the original papers. For methods that do not require supervision, we report the average accuracy across 10 runs. [‡]For meaningful comparison, runs with <5% accuracy are excluded when computing the average, but note that, unlike ours, their method often gives a degenerated solution (see Table 2).

language pairs require different hyperparameters: $\lambda = 1$ works substantially better for Spanish-English and Italian-English, but only $\lambda = 10$ works for Turkish-English.

The results for the more challenging dataset from Dinu et al. (2015) and the extensions of Artetxe et al. (2017, 2018a) are given in Table 2. In this case, our proposed method obtains the best results in all metrics for all the four language pairs tested. The method of Zhang et al. (2017a) does not work at all in this more challenging scenario, which is in line with the negative results reported by the authors themselves for similar conditions (only %2.53 accuracy in their large Gigaword dataset). The method of Conneau et al. (2018) also fails for English-Finnish (only 1.62% in the best run), although it is able to get positive results in some runs for the rest of language pairs. Between the two configurations tested, the default hyperparameters in the code show a more stable behavior.

These results confirm the robustness of the proposed method. While the other systems succeed in some runs and fail in others, our method converges to a good solution in all runs without exception and, in fact, it is the only one getting positive results for English-Finnish. In addition to being more robust, our method also obtains substantially better accuracies, surpassing previous methods by at least 1-3 points in all but the easiest pairs. Moreover, our method is not sensitive to hyperparameters that are difficult to tune without a development set, which is critical in realistic unsupervised conditions.

At the same time, our method is significantly faster than the rest. In relation to that, it is interesting that, while previous methods perform a fixed number of iterations and take practically the same time for all the different language pairs, the runtime of our method adapts to the difficulty of the task thanks to the dynamic convergence criterion of our stochastic approach. This way, our method tends to take longer for more challenging language pairs (1.7 vs 0.6 minutes for es-en and tr-en in one dataset, and 12.9 vs 7.3 minutes for en-fi and en-de in the other) and, in fact, our (relative) execution times correlate surprisingly well with the linguistic distance with English (closest/fastest is German, followed by Italian/Spanish, followed by Turkish/Finnish).

	EN-IT				EN-DE				EN-FI				EN-ES			
	best	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**	**best**	**avg**	**s**	**t**
Full system	48.53	48.13	10	8.9	48.47	48.19	10	7.3	33.50	32.63	10	12.9	37.60	37.33	10	9.1
- Unsup. init.	0.07	0.02	0	16.5	0.00	0.00	0	17.3	0.07	0.01	0	13.8	0.13	0.02	0	15.9
- Stochastic	48.20	48.20	10	2.7	48.13	48.13	10	2.5	0.28	0.28	0	4.3	37.80	37.80	10	2.6
- Cutoff (k=100k)	46.87	46.46	10	114.5	48.27	48.12	10	105.3	31.95	30.78	10	162.5	35.47	34.88	10	185.2
- CSLS	0.00	0.00	0	15.0	0.00	0.00	0	13.8	0.00	0.00	0	13.1	0.00	0.00	0	14.1
- Bidirectional	46.00	45.37	10	5.6	48.27	48.03	10	5.5	31.39	24.86	8	7.8	36.20	35.77	10	7.3
- Re-weighting	46.07	45.61	10	8.4	48.13	47.41	10	7.0	32.94	31.77	10	11.2	36.00	35.45	10	9.1

Table 4: Ablation test on the dataset of Dinu et al. (2015) and the extensions of Artetxe et al. (2017, 2018a). We perform 10 runs for each method and report the best and average accuracies (%), the number of successful runs (those with >5% accuracy) and the average runtime (minutes).

5.2 Comparison with the state-of-the-art

Table 3 shows the results of the proposed method in comparison to previous systems, including those with different degrees of supervision. We focus on the widely used English-Italian dataset of Dinu et al. (2015) and its extensions. Despite being fully unsupervised, our method achieves the best results in all language pairs but one, even surpassing previous supervised approaches. The only exception is English-Finnish, where Artetxe et al. (2018a) gets marginally better results with a difference of 0.3 points, yet ours is the only unsupervised system that works for this pair. At the same time, it is remarkable that the proposed system gets substantially better results than Artetxe et al. (2017), the only other system based on self-learning, with the additional advantage of being fully unsupervised.

5.3 Ablation test

In order to better understand the role of different aspects in the proposed system, we perform an ablation test, where we separately analyze the effect of initialization, the different components of our robust self-learning algorithm, and the final symmetric re-weighting. The obtained results are reported in Table 4.

In concordance with previous work, our results show that self-learning does not work with random initialization. However, the proposed unsupervised initialization is able to overcome this issue without the need of any additional information, performing at par with other character-level heuristics that we tested (e.g. shared numerals).

As for the different self-learning components, we observe that the stochastic dictionary induction is necessary to overcome the problem of poor local optima for English-Finnish, although it does not make any difference for the rest of easier language pairs. The frequency-based vocabulary cutoff also has a positive effect, yielding to slightly better accuracies and much faster runtimes. At the same time, CSLS plays a critical role in the system, as hubness severely accentuates the problem of local optima in its absence. The bidirectional dictionary induction is also beneficial, contributing to the robustness of the system as shown by English-Finnish and yielding to better accuracies in all cases.

Finally, these results also show that symmetric re-weighting contributes positively, bringing an improvement of around 1-2 points without any cost in the execution time.

6 Conclusions

In this paper, we show that previous unsupervised mapping methods (Zhang et al., 2017a; Conneau et al., 2018) often fail on realistic scenarios involving non-comparable corpora and/or distant languages. In contrast to adversarial methods, we propose to use an initial weak mapping that exploits the structure of the embedding spaces in combination with a robust self-learning approach. The results show that our method succeeds in all cases, providing the best results with respect to all previous work on unsupervised and supervised mappings.

The ablation analysis shows that our initial solution is instrumental for making self-learning work without supervision. In order to make self-learning robust, we also added stochasticity to dictionary induction, used CSLS instead of nearest neighbor, and produced bidirectional dictionaries. Results also improved using smaller intermediate vocabularies and re-weighting the final solution. Our implementation is available as an open source project at `https://github.com/artetxem/vecmap`.

In the future, we would like to extend the method from the bilingual to the multilingual scenario, and go beyond the word level by incorporating embeddings of longer phrases.

Acknowledgments

This research was partially supported by the Spanish MINECO (TUNER TIN2015-65308-C5-1-R, MUSTER PCIN-2015-226 and TADEEP TIN2015-70214-P, cofunded by EU FEDER), the UPV/EHU (excellence research group), and the NVIDIA GPU grant program. Mikel Artetxe enjoys a doctoral grant from the Spanish MECD.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), workshop track*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280, Beijing, China. Association for Computational Linguistics.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256, Denver, Colorado. Association for Computational Linguistics.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126, Berlin, Germany. Association for Computational Linguistics.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Yves Peirsman and Sebastian Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 921–929, Los Angeles, California. Association for Computational Linguistics.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010a. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531.

Milos Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010b. On the existence of obstinate results in vector space models. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193.

Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. 2015. Ridge regression, hubness, and zero-shot learning. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Proceedings, Part I*, pages 135–151.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 247–257, Berlin, Germany. Association for Computational Linguistics.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1613–1624, Seattle, Washington, USA. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945, Copenhagen, Denmark. Association for Computational Linguistics.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual pos tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1307–1317, San Diego, California. Association for Computational Linguistics.

A Multi-lingual Multi-task Architecture for Low-resource Sequence Labeling

Ying Lin[1] [*], **Shengqi Yang**[2], **Veselin Stoyanov**[3], **Heng Ji**[1]
[1] Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY, USA
{liny9,jih}@rpi.edu
[2] Intelligent Advertising Lab, JD.com, Santa Clara, CA, USA
sheqiyan@gmail.com
[3] Applied Machine Learning, Facebook, Menlo Park, CA, USA
vesko.st@gmail.com

Abstract

We propose a multi-lingual multi-task architecture to develop supervised models with a minimal amount of labeled data for sequence labeling. In this new architecture, we combine various transfer models using two layers of parameter sharing. On the first layer, we construct the basis of the architecture to provide universal word representation and feature extraction capability for all models. On the second level, we adopt different parameter sharing strategies for different transfer schemes. This architecture proves to be particularly effective for low-resource settings, when there are less than 200 training sentences for the target task. Using Name Tagging as a target task, our approach achieved 4.3%-50.5% absolute F-score gains compared to the mono-lingual single-task baseline model. [1]

1 Introduction

When we use supervised learning to solve Natural Language Processing (NLP) problems, we typically train an individual model for each task with task-specific labeled data. However, our target task may be intrinsically linked to other tasks. For example, Part-of-speech (POS) tagging and Name Tagging can both be considered as sequence labeling; Machine Translation (MT) and Abstractive Text Summarization both require the ability to understand the source text and generate natural language sentences. Therefore, it is valuable to transfer knowledge from related tasks to the target task. Multi-task Learning (MTL) is one of the most effective solutions for knowledge transfer across tasks. In the context of neural network architectures, we usually perform MTL by sharing parameters across models (Ruder, 2017).

Previous studies (Collobert and Weston, 2008; Dong et al., 2015; Luong et al., 2016; Liu et al., 2018; Yang et al., 2017) have proven that MTL is an effective approach to boost the performance of related tasks such as MT and parsing. However, most of these previous efforts focused on tasks and languages which have sufficient labeled data but hit a performance ceiling on each task alone. Most NLP tasks, including some well-studied ones such as POS tagging, still suffer from the lack of training data for many low-resource languages. According to Ethnologue[2], there are 7,099 living languages in the world. It is an unattainable goal to annotate data in all languages, especially for tasks with complicated annotation requirements. Furthermore, some special applications (e.g., disaster response and recovery) require rapid development of NLP systems for extremely low-resource languages. Therefore, in this paper, we concentrate on enhancing supervised models in low-resource settings by borrowing knowledge learned from related high-resource languages and tasks.

In (Yang et al., 2017), the authors simulated a low-resource setting for English and Spanish by downsampling the training data for the target task. However, for most low-resource languages, the data sparsity problem also lies in related tasks and languages. Under such circumstances, a single transfer model can only bring limited improvement. To tackle this issue, we propose a *multi-lingual multi-task* architecture which combines different transfer models within a unified architecture through two levels of parameter sharing. In the first level, we share character embeddings,

[*] Part of this work was done when the first author was on an internship at Facebook.

[1] The code of our model is available at https://github.com/limteng-rpi/mlmt

[2] https://www.ethnologue.com/guides/how-many-languages

character-level convolutional neural networks, and word-level long-short term memory layer across all models. These components serve as a basis to connect multiple models and transfer universal knowledge among them. In the second level, we adopt different sharing strategies for different transfer schemes. For example, we use the same output layer for all Name Tagging tasks to share task-specific knowledge (e.g., I-PER[3] should not be assigned to the first word in a sentence).

To illustrate our idea, we take *sequence labeling* as a case study. In the NLP context, the goal of sequence labeling is to assign a categorical label (e.g., POS tag) to each token in a sentence. It underlies a range of fundamental NLP tasks, including POS Tagging, Name Tagging, and chunking.

Experiments show that our model can effectively transfer various types of knowledge from different auxiliary tasks and obtains up to 50.5% absolute F-score gains on Name Tagging compared to the mono-lingual single-task baseline. Additionally, our approach does not rely on a large amount of auxiliary task data to achieve the improvement. Using merely 1% auxiliary data, we already obtain up to 9.7% absolute gains in F-score.

2 Model

2.1 Basic Architecture

The goal of sequence labeling is to assign a categorical label to each token in a given sentence. Though traditional methods such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) (Lafferty et al., 2001; Ratinov and Roth, 2009; Passos et al., 2014) achieved high performance on sequence labeling tasks, they typically relied on hand-crafted features, therefore it is difficult to adapt them to new tasks or languages. To avoid task-specific engineering, (Collobert et al., 2011) proposed a feed-forward neural network model that only requires word embeddings trained on a large scale corpus as features. After that, several neural models based on the combination of long-short term memory (LSTM) and CRFs (Ma and Hovy, 2016; Lample et al., 2016; Chiu and Nichols, 2016) were proposed and achieved better performance on sequence labeling tasks.

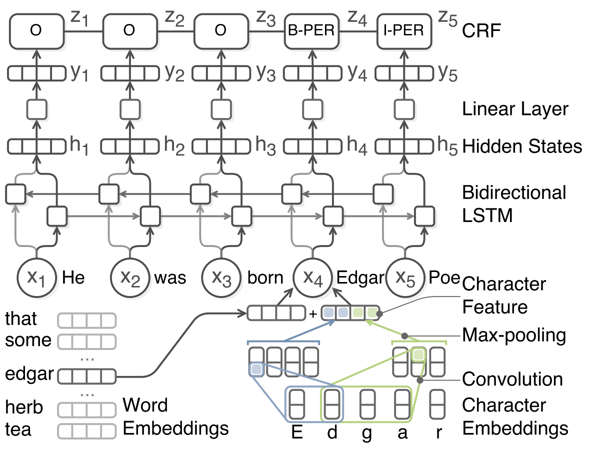


Figure 1: LSTM-CNNs: an LSTM-CRFs-based model for Sequence Labeling

LSTM-CRFs-based models are well-suited for multi-lingual multi-task learning for three reasons: (1) They learn features from word and character embeddings and therefore require little feature engineering; (2) As the input and output of each layer in a neural network are abstracted as vectors, it is fairly straightforward to share components between neural models; (3) Character embeddings can serve as a bridge to transfer morphological and semantic information between languages with identical or similar scripts, without requiring cross-lingual dictionaries or parallel sentences.

Therefore, we design our multi-task multi-lingual architecture based on the LSTM-CNNs model proposed in (Chiu and Nichols, 2016). The overall framework is illustrated in Figure 1. First, each word w_i is represented as the combination x_i of two parts, word embedding and character feature vector, which is extracted from character embeddings of the characters in w_i using convolutional neural networks (CharCNN). On top of that, a bidirectional LSTM processes the sequence $\boldsymbol{x} = \{x_1, x_2, ...\}$ in both directions and encodes each word and its context into a fixed-size vector h_i. Next, a linear layer converts h_i to a score vector y_i, in which each component represents the predicted score of a target tag. In order to model correlations between tags, a CRFs layer is added at the top to generate the best tagging path for the whole sequence. In the CRFs layer, given an input sentence $\boldsymbol{x}$ of length L and the output of the linear layer $\boldsymbol{y}$, the score of a sequence of tags $\boldsymbol{z}$ is

[3]We adopt the BIOES annotation scheme. Prefixes B-, I-, E-, and S- represent the beginning of a mention, inside of a mention, the end of a mention and a single-token mention respectively. The O tag is assigned to a word which is not part of any mention.

defined as:

$$S(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{t=1}^{L} (\boldsymbol{A}_{z_{t-1}, z_t} + \boldsymbol{y}_{t, z_t}),$$

where $\boldsymbol{A}$ is a transition matrix in which $\boldsymbol{A}_{p,q}$ represents the binary score of transitioning from tag p to tag q, and $\boldsymbol{y}_{t,z}$ represents the unary score of assigning tag z to the t-th word. Given the ground truth sequence of tags $\boldsymbol{z}$, we maximize the following objective function during the training phase:

$$\begin{aligned}\mathcal{O} &= \log P(\boldsymbol{z}|\boldsymbol{x}) \\ &= S(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) - \log \sum_{\tilde{\boldsymbol{z}} \in \boldsymbol{Z}} e^{S(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{z}})},\end{aligned}$$

where $\boldsymbol{Z}$ is the set of all possible tagging paths.

We emphasize that our actual implementation differs slightly from the LSTM-CNNs model. We do not use additional word- and character-level explicit symbolic features (e.g., capitalization and lexicon) as they may require additional language-specific knowledge. Additionally, we transform character feature vectors using highway networks (Srivastava et al., 2015), which is reported to enhance the overall performance by (Kim et al., 2016) and (Liu et al., 2018). Highway networks is a type of neural network that can smoothly switch its behavior between transforming and carrying information.

2.2 Multi-task Multi-lingual Architecture

MTL can be employed to improve performance on multiple tasks at the same time, such as MT and parsing in (Luong et al., 2016). However, in our scenario, we only focused on enhancing the performance of a low-resource task, which is our target task or ***main task***. Our proposed architecture aims to transfer knowledge from a set of ***auxiliary tasks*** to the main task. For simplicity, we refer to a model of a main (auxiliary) task as a main (auxiliary) model.

To jointly train multiple models, we perform multi-task learning using parameter sharing. Let Θ_i be the set of parameters for model m_i and $\Theta_{i,j} = \Theta_i \cap \Theta_j$ be the shared parameters between m_i and m_j. When optimizing model m_i, we update Θ_i and hence $\Theta_{i,j}$. In this way, we can partially train model m_j as $\Theta_{i,j} \subseteq \Theta_j$. Previously, each MTL model generally uses a single transfer scheme. In order to merge different transfer models into a unified architecture, we employ two levels of parameter sharing as follows.

On the first level, we construct the basis of the architecture by sharing character embeddings, CharCNN and bidirectional LSTM among all models. This level of parameter sharing aims to provide universal word representation and feature extraction capability for all tasks and languages.

Character Embeddings and Character-level CNNs. Character features can represent morphological and semantic information; e.g., the English morpheme `dis-` usually indicates negation and reversal as in "*disagree*" and "*disapproval*". For low-resource languages lacking in data to suffice the training of high-quality word embeddings, character embeddings learned from other languages may provide crucial information for labeling, especially for rare and out-of-vocabulary words. Take the English word "*overflying*" (flying over) as an example. Even if it is rare or absent in the corpus, we can still infer the word meaning from its suffix `over-` (above), root `fly`, and prefix `-ing` (present participle form). In our architecture, we share character embeddings and the CharCNN between languages with identical or similar scripts to enhance word representation for low-resource languages.

Bidirectional LSTM. The bidirectional LSTM layer is essential to extract character, word, and contextual information from a sentence. However, with a large number of parameters, it cannot be fully trained only using the low-resource task data. To tackle this issue, we share the bidirectional LSTM layer across all models. Bear in mind that because our architecture does not require aligned cross-lingual word embeddings, sharing this layer across languages may confuse the model as it equally handles embeddings in different spaces. Nevertheless, under low-resource circumstances, data sparsity is the most critical factor that affects the performance.

On top of this basis, we adopt different parameter sharing strategies for different transfer schemes. For cross-task transfer, we use the same word embedding matrix across tasks so that they can mutually enhance word representations. For cross-lingual transfer, we share the linear layer and CRFs layer among languages to transfer task-specific knowledge, such as the transition score between two tags.

Word Embeddings. For most words, in addition to character embeddings, word embeddings are still crucial to represent semantic informa-

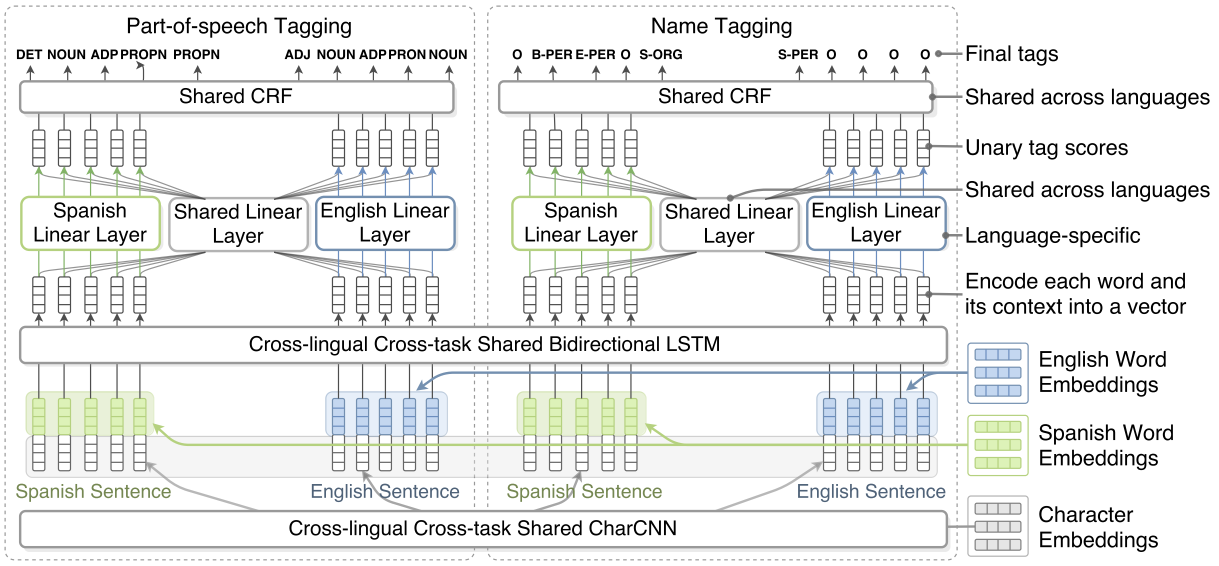


Figure 2: Multi-task Multi-lingual Architecture

tion. We use the same word embedding matrix for tasks in the same language. The matrix is initialized with pre-trained embeddings and optimized as parameters during training. Thus, task-specific knowledge can be encoded into the word embeddings by one task and subsequently utilized by another one. For a low-resource language even without sufficient raw text, we mix its data with a related high-resource language to train word embeddings. In this way, we merge both corpora and hence their vocabularies.

Recently, Conneau et al. (2017) proposed a domain-adversarial method to align two monolingual word embedding matrices without cross-lingual supervision such as a bilingual dictionary. Although cross-lingual word embeddings are not required, we evaluate our framework with aligned embeddings generated using this method. Experiment results show that the incorporation of cross-lingual embeddings substantially boosts the performance under low-resource settings.

Linear Layer and CRFs. As the tag set varies from task to task, the linear layer and CRFs can only be shared across languages. We share these layers to transfer task-specific knowledge to the main model. For example, our model corrects [S-PER Charles] [S-PER Picqué] to [B-PER Charles] [E-PER Picqué] because the CRFs layer fully trained on other languages assigns a low score to the rare transition S-PER→S-PER and promotes B-PER→E-PER. In addition to the shared linear layer, we add an unshared language-specific linear layer to allow the model to behave differently toward some features for different languages. For example, the suffix `-ment` usually indicates nouns in English whereas indicates adverbs in French.

We combine the output of the shared linear layer $\boldsymbol{y}^u$ and the output of the language-specific linear layer $\boldsymbol{y}^s$ using:

$$\boldsymbol{y} = \boldsymbol{g} \odot \boldsymbol{y}^s + (1 - \boldsymbol{g}) \odot \boldsymbol{y}^u,$$

where $\boldsymbol{g} = \sigma(\boldsymbol{W}_g \boldsymbol{h} + \boldsymbol{b}_g)$. $\boldsymbol{W}_g$ and $\boldsymbol{b}_g$ are optimized during training. $\boldsymbol{h}$ is the LSTM hidden states. As $\boldsymbol{W}_g$ is a square matrix, $\boldsymbol{y}$, $\boldsymbol{y}^s$, and $\boldsymbol{y}^u$ have the same dimension.

Although we only focus on sequence labeling in this work, our architecture can be adapted for many NLP tasks with slight modification. For example, for text classification tasks, we can take the last hidden state of the forward LSTM as the sentence representation and replace the CRFs layer with a Softmax layer.

In our model, each task has a separate object function. To optimize multiple tasks within one model, we adopt the alternating training approach in (Luong et al., 2016). At each training step, we sample a task d_i with probability $\frac{r_i}{\sum_j r_j}$, where r_i is the *mixing rate value* assigned to d_i. In our experiments, instead of tuning r_i, we estimate it by:

$$r_i = \mu_i \zeta_i \sqrt{N_i}\ ,$$

where μ_i is the task coefficient, ζ_i is the language coefficient, and N_i is the number of training examples. μ_i (or ζ_i) takes the value 1 if the task (or language) of d_i is the same as that of the target task; Otherwise it takes the value 0.1. For example, given English Name Tagging as the target task, the task coefficient μ and language coefficient ζ of Spanish Name Tagging are 0.1 and 1 respectively.

While assigning lower mixing rate values to auxiliary tasks, this formula also takes the amount of data into consideration. Thus, auxiliary tasks receive higher probabilities to reduce overfitting when we have a smaller amount of main task data.

3 Experiments

3.1 Data Sets

For Name Tagging, we use the following data sets: Dutch (NLD) and Spanish (ESP) data from the CoNLL 2002 shared task (Tjong Kim Sang, 2002), English (ENG) data from the CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003), Russian (RUS) data from LDC2016E95 (Russian Representative Language Pack), and Chechen (CHE) data from TAC KBP 2017 10-Language EDL Pilot Evaluation Source Corpus[4]. We select Chechen as another target language in addition to Dutch and Spanish because it is a truly under-resourced language and its related language, Russian, also lacks NLP resources.

Code	Train	Dev	Test
NLD	202,931 (13,344)	37,761 (2,616)	68,994 (3,941)
ESP	207,484 (18,797)	51,645 (4,351)	52,098 (3,558)
ENG	204,567 (23,499)	51,578 (5,942)	46,666 (5,648)
RUS	66,333 (3,143)	8,819 (413)	7,771 (407)
CHE	98,355 (2,674)	12,265 (312)	11,933 (366)

Table 1: Name Tagging data set statistics: #token and #name (between parentheses).

For POS Tagging, we use English, Dutch, Spanish, and Russian data from the CoNLL 2017 shared task (Zeman et al., 2017; Nivre et al., 2017). In this data set, each token is annotated with two POS tags, UPOS (universal POS tag) and XPOS (language-specific POS tag). We use UPOS because it is consistent throughout all languages.

3.2 Experimental Setup

We use 50-dimensional pre-trained word embeddings and 50-dimensional randomly initialized character embeddings. We train word embeddings using the word2vec package[5]. English, Spanish, and Dutch embeddings are trained on corresponding Wikipedia articles (2017-12-20 dumps). Russian embeddings are trained on documents in LDC2016E95. Chechen embeddings are trained on documents in TAC KBP 2017 10-Language EDL Pilot Evaluation Source Corpus. To learn a mapping between mono-lingual word embeddings and obtain cross-lingual embeddings, we use the unsupervised model in the MUSE library[6] (Conneau et al., 2017). Although word embeddings are fine-tuned during training, we update the embedding matrix in a sparse way and thus do not have to update a large number of parameters.

We optimize parameters using Stochastic Gradient Descent with momentum, gradient clipping and exponential learning rate decay. At step t, the learning rate α_t is updated using $\alpha_t = \alpha_0 * \rho^{t/T}$, where α_0 is the initial learning rate, ρ is the decay rate, and T is the decay step.[7] To reduce overfitting, we apply Dropout (Srivastava et al., 2014) to the output of the LSTM layer.

We conduct hyper-parameter optimization by exploring the space of parameters shown in Table 2 using random search (Bergstra and Bengio, 2012). Due to time constraints, we only perform parameter sweeping on the Dutch Name Tagging task with 200 training examples. We select the set of parameters that achieves the best performance on the development set and apply it to all models.

Layer	Range	Final
CharCNN Filter Number	$[10, 30]$	20
Highway Layer Number	$[1, 2]$	2
Highway Activation Function	ReLU, SeLU	SeLU
LSTM Hidden State Size	$[50, 200]$	171
LSTM Dropout Rate	$[0.3, 0.8]$	0.6
Learning Rate	$[0.01, 0.2]$	0.02
Batch Size	$[5, 25]$	19

Table 2: Hyper-parameter search space.

3.3 Comparison of Different Models

In Figure 3, 4, and 5, we compare our model with the mono-lingual single-task LSTM-CNNs model (denoted as *baseline*), cross-task transfer model, and cross-lingual transfer model in low-resource settings with Dutch, Spanish, and Chechen Name Tagging as the main task respectively. We use English as the related language for Dutch and Spanish, and use Russian as the related language for

[4]`https://tac.nist.gov/2017/KBP/data.html`

[5]`https://github.com/tmikolov/word2vec`

[6]`https://github.com/facebookresearch/MUSE`

[7]Momentum β, gradient clipping threshold, ρ, and T are set to 0.9, 5.0, 0.9, and 10000 in the experiments.

Chechen. For cross-task transfer, we take POS Tagging as the auxiliary task. Because the CoNLL 2017 data does not include Chechen, we only use Russian POS Tagging and Russian Name Tagging as auxiliary tasks for Chechen Name Tagging.

We take Name Tagging as the target task for three reasons: (1) POS Tagging has a much lower requirement for the amount of training data. For example, using only 10 training sentences, our baseline model achieves 75.5% and 82.9% prediction accuracy on Dutch and Spanish; (2) Compared to POS Tagging, Name Tagging has been considered as a more challenging task; (3) Existing POS Tagging resources are relatively richer than Name Tagging ones; e.g., the CoNLL 2017 data set provides POS Tagging training data for 45 languages. Name Tagging also has a higher annotation cost as its annotation guidelines are usually more complicated.

We can see that our model substantially outperforms the mono-lingual single-task baseline model and obtains visible gains over single transfer models. When trained with less than 50 main tasks training sentences, cross-lingual transfer consistently surpasses cross-task transfer, which is not surprising because in the latter scheme, the linear layer and CRFs layer of the main model are not shared with other models and thus cannot be fully trained with little data.

Because there are only 20,400 sentences in Chechen documents, we also experiment with the data augmentation method described in Section 2.2 by training word embeddings on a mixture of Russian and Chechen data. This method yields additional 3.5%-10.0% absolute F-score gains. We also experiment with transferring from English to Chechen. Because Chechen uses Cyrillic alphabet , we convert its data set to Latin script. Surprisingly, although these two languages are not close, we get more improvement by using English as the auxiliary language.

In Table 3, we compare our model with state-of-the-art models using all Dutch or Spanish Name Tagging data. Results show that although we design this architecture for low-resource settings, it also achieves good performance in high-resource settings. In this experiment, with sufficient training data for the target task, we perform another round of parameter sweeping. We increase the embedding sizes and LSTM hidden state size to 100 and 225 respectively.

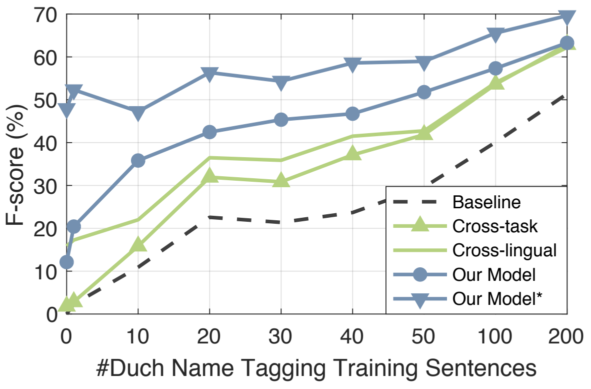


Figure 3: Performance on Dutch Name Tagging. We scale the horizontal axis to show more details under 100 sentences. Our Model*: our model with MUSE cross-lingual embeddings.

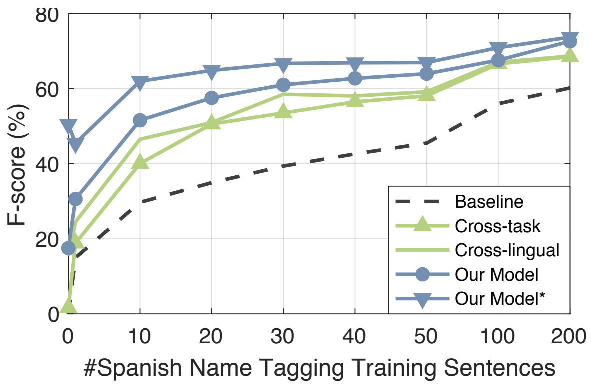


Figure 4: Performance on Spanish Name Tagging.

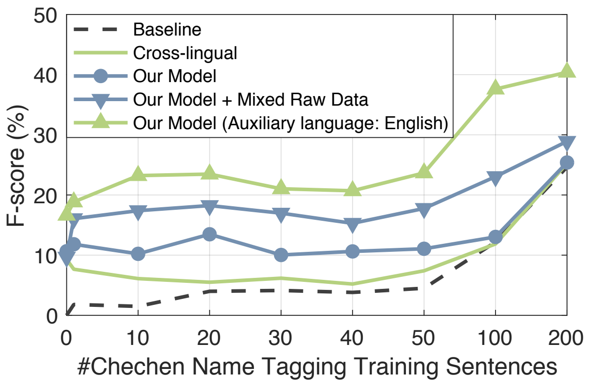


Figure 5: Performance on Chechen Name Tagging.

3.4 Qualitative Analysis

In Table 4, we compare Name Tagging results from the baseline model and our model, both trained with 100 main task sentences.

The first three examples show that shared character-level networks can transfer different levels of morphological and semantic information.

Language	Model	F-score
Dutch	Gillick et al. (2016)	82.84
	Lample et al. (2016)	81.74
	Yang et al. (2017)	85.19
	Baseline	85.14
	Cross-task	85.69
	Cross-lingual	85.71
	Our Model	**86.55**
Spanish	Gillick et al. (2016)	82.95
	Lample et al. (2016)	85.75
	Yang et al. (2017)	85.77
	Baseline	85.44
	Cross-task	85.37
	Cross-lingual	85.02
	Our Model	**85.88**

Table 3: Comparison with state-of-the-art models.

In example #1, the baseline model fails to identify "*Palestijnen*", an unseen word in the Dutch data, while our model can recognize it because the shared CharCNN represents it in a way similar to its corresponding English word "*Palestinians*", which occurs 20 times. In addition to mentions, the shared CharCNN can also improve representations of context words, such as "*staat*" (state) in the example. For some words dissimilar to corresponding English words, the CharCNN may enhance their word representations by transferring morpheme-level knowledge. For example, in sentence #2, our model is able to identify "*Rusland*" (Russia) as the suffix `-land` is usually associated with location names in the English data; e.g., Finland. Furthermore, the CharCNN is capable of capturing some word-level patterns, such as capitalized hyphenated compound and acronym as example #3 shows. In this sentence, neither "*PMS-centra*" nor "*MST*" can be found in auxiliary task data, while we observe a number of similar expressions, such as American-style and LDP.

The transferred knowledge also helps reduce overfitting. For example, in sentence #4, the baseline model mistakenly tags "*sección*" (section) and "*consellería*" (department) as organizations because their capitalized forms usually appear in Spanish organization names. With knowledge learned in auxiliary tasks that a lowercased word is rarely tagged as a proper noun, our model is able to avoid overfitting and correct these errors. Sentence #5 shows an opposite situation, where the capitalized word "*campesinos*" (farm worker) never appears in Spanish names.

In Table 5, we show differences between cross-lingual transfer and cross-task transfer. Although the cross-task transfer model recognizes "*Ingeborg Marx*" missed by the baseline model, it mistakenly assigns an S-PER tag to "*Marx*". Instead, from English Name Tagging, the cross-lingual transfer model borrows task-specific knowledge through the shared CRFs layer that (1) B-PER→S-PER is an invalid transition, and (2) even if we assign S-PER to "*Ingeborg*", it is rare to have continuous person names without any conjunction or punctuation. Thus, the cross-lingual model promotes the sequence B-PER→E-PER.

In Figure 6, we depict the change of tag distribution with the number of training sentences. When trained with less than 100 sentences, the baseline model only correctly predicts a few tags dominated by frequent types. By contrast, our model has a visibly higher recall and better predicts infrequent tags, which can be attributed to the implicit data augmentation and inductive bias introduced by MTL (Ruder, 2017). For example, if all location names in the Dutch training data are single-token ones, the baseline model will inevitably overfit to the tag S-LOC and possibly label "*Caldera de Taburiente*" as [S-LOC Caldera] [S-LOC de] [S-LOC Taburiente], whereas with the shared CRFs layer fully trained on English Name Tagging, our model prefers B-LOC→I-LOC→E-LOC, which receives a higher transition score.

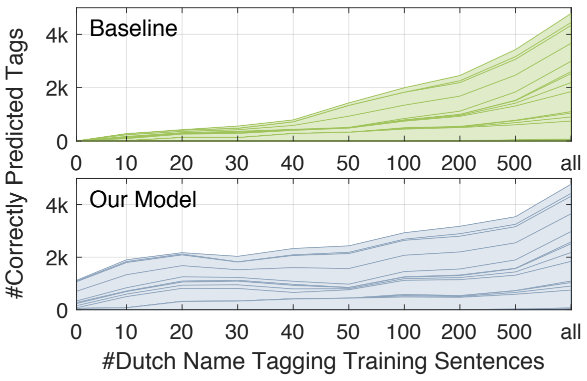


Figure 6: The distribution of correctly predicted tags on Dutch Name Tagging. The height of each stack indicates the number of a certain tag.

3.5 Ablation Studies

In order to quantify the contributions of individual components, we conduct ablation studies on Dutch Name Tagging with different numbers of training sentences for the target task. For the basic model, we we use separate LSTM layers and

#1 [DUTCH]: *If a Palestinian State is, however, the first thing the Palestinians will do.*

⋆ [B] Als er een Palestijnse staat komt, is dat echter het eerste wat de Palestijnen zullen doen

⋆ [A] Als er een [S-MISC Palestijnse] staat komt, is dat echter het eerste wat de [S-MISC Palestijnen] zullen doen

#2 [DUTCH]: *That also frustrates the Muscovites, who still live in the proud capital of Russia but can not look at the soaps that the stupid farmers can see on the outside.*

⋆ [B] Ook dat frustreert de Moskovieten , die toch in de fiere hoofdstad van Rusland wonen maar niet naar de soaps kunnen kijken die de domme boeren op de buiten wel kunnen zien

⋆ [A] Ook dat frustreert de [S-MISC Moskovieten] , die toch in de fiere hoofdstad van [S-LOC Rusland] wonen maar niet naar de soaps kunnen kijken die de domme boeren op de buiten wel kunnen zien

#3 [DUTCH]: *And the PMS centers are merging with the centers for school supervision, the MSTs.*

⋆ [B] En smelten de PMS-centra samen met de centra voor schooltoezicht, de MST's .

⋆ [A] En smelten de [S-MISC PMS-centra] samen met de centra voor schooltoezicht, de [S-MISC MST's] .

#4 [SPANISH]: *The trade union section of CC.OO. in the Department of Justice has today denounced more attacks of students to educators in centers dependent on this department ...*

⋆ [B] La [B-ORG sección] [I-ORG sindical] [I-ORG de] [S-ORG CC.OO.] en el [B-ORG Departamento] [I-ORG de] [E-ORG Justicia] ha denunciado hoy ms agresiones de alumnos a educadores en centros dependientes de esta [S-ORG consellería] ...

⋆ [A] La sección sindical de [S-ORG CC.OO.] en el [B-ORG Departamento] [I-ORG de] [E-ORG Justicia] ha denunciado hoy ms agresiones de alumnos a educadores en centros dependientes de esta consellería ...

#5 [SPANISH]: ... *and the Single Trade Union Confederation of Peasant Workers of Bolivia, agreed upon when the state of siege was ended last month.*

⋆ [B] ... y la [B-ORG Confederación] [I-ORG Sindical] [I-ORG Unica] [I-ORG de] [E-ORG Trabajadores] Campesinos de [S-ORG Bolivia] , pactadas cuando se dio fin al estado de sitio, el mes pasado .

⋆ [A] .. y la [B-ORG Confederación] [I-ORG Sindical] [I-ORG Unica] [I-ORG de] [I-ORG Trabajadores] [I-ORG Campesinos] [I-ORG de] [E-ORG Bolivia] , pactadas cuando se dio fin al estado de sitio, el mes pasado .

Table 4: Name Tagging results, each of which contains an English translation, result of the baseline model (B), and result of our model (A). The GREEN (RED) highlight indicates a correct (incorrect) tag.

[DUTCH] ... *Ingeborg Marx is her name, a formidable heavy weight to high above her head!*

⋆ [B] ... Zag ik zelfs onlangs niet dat een lief, mooi vrouwtje, Ingeborg Marx is haar naam, een formidabel zwaar gewicht tot hoog boven haar hoofd stak!

⋆ [CROSS-TASK] ... Zag ik zelfs onlangs niet dat een lief, mooi vrouwtje, [B-PER Ingeborg] [S-PER Marx] is haar naam, een formidabel zwaar gewicht tot hoog boven haar hoofd stak!

⋆ [CROSS-LINGUAL] ... Zag ik zelfs onlangs niet dat een lief, mooi vrouwtje, [B-PER Ingeborg] [E-PER Marx] is haar naam, een formidabel zwaar gewicht tot hoog boven haar hoofd stak!

Table 5: Comparing cross-task transfer and cross-lingual transfer on Dutch Name Tagging with 100 training sentences.

remove the character embeddings, highway networks, language-specific layer, and Dropout layer. As Table 6 shows, adding each component usually enhances the performance (F-score, %), while the impact also depends on the size of the target task data. For example, the language-specific layer slightly impairs the performance with only 10 training sentences. However, this is unsurprising as it introduces additional parameters that are only trained by the target task data.

Model	**0**	**10**	**100**	**200**	**All**
Basic	2.06	20.03	47.98	51.52	77.63
+C	1.69	24.22	48.53	56.26	83.38
+CL	9.62	25.97	49.54	56.29	83.37
+CLS	3.21	25.43	50.67	56.34	84.02
+CLSH	7.70	30.48	53.73	58.09	84.68
+CLSHD	12.12	35.82	57.33	63.27	86.00

Table 6: Performance comparison between models with different components (C: character embedding; L: shared LSTM; S: language-specific layer; H: highway networks; D: dropout).

3.6 Effect of the Amount of Auxiliary Task Data

For many low-resource languages, their related languages are also low-resource. To evaluate our model's sensitivity to the amount of auxiliary task data, we fix the size of main task data and downsample all auxiliary task data with sample rates from 1% to 50%. As Figure 7 shows, the performance goes up when we raise the sample rate from

1% to 20%. However, we do not observe significant improvement when we further increase the sample rate. By comparing scores in Figure 3 and Figure 7, we can see that using only 1% auxiliary data, our model already obtains 3.7%-9.7% absolute F-score gains. Due to space limitations, we only show curves for Dutch Name Tagging, while we observe similar results on other tasks. Therefore, we may conclude that our model does not heavily rely on the amount of auxiliary task data.

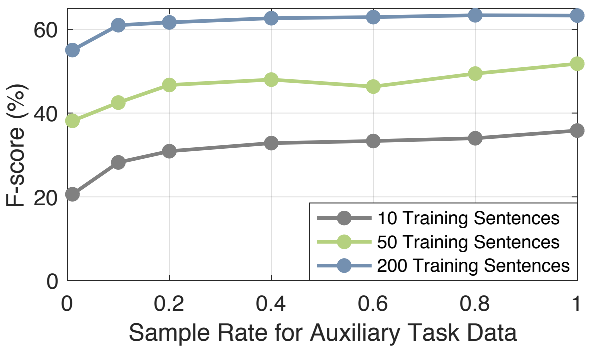


Figure 7: The effect of the amount of auxiliary task data on Dutch Name Tagging.

4 Related Work

Multi-task Learning has been applied in different NLP areas, such as machine translation (Luong et al., 2016; Dong et al., 2015; Domhan and Hieber, 2017), text classification (Liu et al., 2017), dependency parsing (Peng et al., 2017), textual entailment (Hashimoto et al., 2017), text summarization (Isonuma et al., 2017) and sequence labeling (Collobert and Weston, 2008; Søgaard and Goldberg, 2016; Rei, 2017; Peng and Dredze, 2017; Yang et al., 2017; von Däniken and Cieliebak, 2017; Aguilar et al., 2017; Liu et al., 2018)

Collobert and Weston (2008) is an early attempt that applies MTL to sequence labeling. The authors train a CNN model jointly on POS Tagging, Semantic Role Labeling, Name Tagging, chunking, and language modeling using parameter sharing. Instead of using other sequence labeling tasks, Rei (2017) and Liu et al. (2018) take language modeling as the secondary training objective to extract semantic and syntactic knowledge from large scale raw text without additional supervision. In (Yang et al., 2017), the authors propose three transfer models for cross-domain, cross-application, and cross-lingual transfer for sequence labeling, and also simulate a low-resource setting by downsampling the training data. By contrast, we combine cross-task transfer and cross-lingual transfer within a unified architecture to transfer different types of knowledge from multiple auxiliary tasks simultaneously. In addition, because our model is designed for low-resource settings, we share components among models in a different way (e.g., the LSTM layer is shared across all models). Differing from most MTL models, which perform supervisions for all tasks on the outermost layer, (Søgaard and Goldberg, 2016) proposes an MTL model which supervised tasks at different levels. It shows that supervising low-level tasks such as POS Tagging at lower layer obtains better performance.

5 Conclusions and Future Work

We design a multi-lingual multi-task architecture for low-resource settings. We evaluate the model on sequence labeling tasks with three language pairs. Experiments show that our model can effectively transfer different types of knowledge to improve the main model. It substantially outperforms the mono-lingual single-task baseline model, cross-lingual transfer model, and cross-task transfer model.

The next step of this research is to apply this architecture to other types of tasks, such as Event Extract and Semantic Role Labeling that involve structure prediction. We also plan to explore the possibility of integrating incremental learning into this architecture to adapt a trained model for new tasks rapidly.

Acknowledgments

This work was supported by the U.S. DARPA LORELEI Program No. HR0011-15-C-0115 and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

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

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Pius von Däniken and Mark Cieliebak. 2017. Transfer learning and sentence level features for named entity recognition on tweets. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *EMNLP*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *NAACL HLT*.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *EMNLP*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng

Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. In *ACL*.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *ICML*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *NAACL HLT*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. CoNLL 2017 shared task: multilingual parsing from raw text to universal dependencies. In *CoNLL*.

Two Methods for Domain Adaptation of Bilingual Tasks: Delightfully Simple and Broadly Applicable

Viktor Hangya[1], **Fabienne Braune**[1,2], **Alexander Fraser**[1], **Hinrich Schütze**[1]
[1]Center for Information and Language Processing
LMU Munich, Germany
[2]Volkswagen Data Lab Munich, Germany
{hangyav, fraser}@cis.uni-muenchen.de
fabienne.braune@volkswagen.de

Abstract

Bilingual tasks, such as bilingual lexicon induction and cross-lingual classification, are crucial for overcoming data sparsity in the target language. Resources required for such tasks are often out-of-domain, thus domain adaptation is an important problem here. We make two contributions. First, we test a delightfully simple method for *domain adaptation of bilingual word embeddings*. We evaluate these embeddings on two bilingual tasks involving different domains: cross-lingual twitter sentiment classification and medical bilingual lexicon induction. Second, we tailor a *broadly applicable semi-supervised classification* method from computer vision to these tasks. We show that this method also helps in low-resource setups. Using both methods together we achieve large improvements over our baselines, by using only additional unlabeled data.

1 Introduction

In this paper we study two bilingual tasks that strongly depend on bilingual word embeddings (BWEs). Previously, specialized domain adaptation approaches to such tasks were proposed. We instead show experimentally that a simple adaptation process involving only unlabeled text is highly effective. We then show that a semi-supervised classification method from computer vision can be applied successfully for further gains in cross-lingual classification.

Our BWE adaptation method is delightfully simple. We begin by adapting monolingual word embeddings to the target domain for source and target languages by simply building them using both general and target-domain unlabeled data. As a second step we use post-hoc mapping (Mikolov et al., 2013b), i.e., we use a seed lexicon to transform the word embeddings of the two languages into the same vector space. We show experimentally for the first time that the domain-adapted bilingual word embeddings we produce using this extremely simple technique are highly effective. We study two quite different tasks and domains, where resources are lacking, showing that our simple technique performs well for both of them: cross-lingual twitter sentiment classification and medical bilingual lexicon induction. In previous work, task-dependent approaches were used for this type of domain adaptation. Our approach is simple and task independent.

Second, we adapt the semi-supervised image classification system of Häusser et al. (2017) for NLP problems for the first time. This approach is broadly applicable to many NLP classification tasks where unlabeled data is available. We tailor it to both of our cross-lingual tasks. The system exploits unlabeled data during the training of classifiers by learning similar features for similar labeled and unlabeled training examples, thereby extracting information from unlabeled examples as well. As we show experimentally, the system further improves cross-lingual knowledge transfer for both of our tasks.

After combining both techniques, the results of sentiment analysis are competitive with systems that use annotated data in the target language, an impressive result considering that we require no target-language annotated data. The method also yields impressive improvements for bilingual lexicon induction compared with baselines trained on in-domain data. We show that this system requires the high-quality domain-adapted bilingual word embeddings we previously created to use unlabeled data well.

2 Previous Work

2.1 Bilingual Word Embeddings

Many approaches have been proposed for creating high quality BWEs using different bilingual signals. Following Mikolov et al. (2013b), many authors (Faruqui and Dyer, 2014; Xing et al., 2015; Lazaridou et al., 2015; Vulić and Korhonen, 2016) map monolingual word embeddings (MWEs) into the same bilingual space. Others leverage parallel texts (Hermann and Blunsom, 2014; Gouws et al., 2015) or create artificial cross-lingual corpora using seed lexicons or document alignments (Vulić and Moens, 2015; Duong et al., 2016) to train BWEs.

In contrast, our aim is not to improve the intrinsic quality of BWEs, but to adapt BWEs to specific domains to enhance their performance on bilingual tasks in these domains. Faruqui et al. (2015), Gouws and Søgaard (2015), Rothe et al. (2016) have previously studied domain adaptation of bilingual word embeddings, showing it to be highly effective for improving downstream tasks. However, importantly, their proposed methods are based on specialized domain lexicons (such as, e.g., sentiment lexicons) which contain task specific word relations. Our delightfully simple approach is, in contrast, effectively task independent (in that it only requires unlabeled in-domain text), which is an important strength.

2.2 Cross-Lingual Sentiment Analysis

Sentiment analysis is widely applied, and thus ideally we would have access to high quality supervised models in all human languages. Unfortunately, good quality labeled datasets are missing for many languages. Training models on resource rich languages and applying them to resource poor languages is therefore highly desirable. Cross-lingual sentiment classification (CLSC) tackles this problem (Mihalcea et al., 2007; Banea et al., 2010; Wan, 2009; Lu et al., 2011; Balamurali and Joshi, 2012; Gui et al., 2013). Recent CLSC approaches use BWEs as features of deep learning architectures which allows us to use a model for target-language sentiment classification, even when the model was trained only using source-language supervised training data. Following this approach we perform CLSC on Spanish tweets using English training data. Even though Spanish is not resource-poor we simulate this by using only English annotated data.

Xiao and Guo (2013) proposed a cross-lingual log-bilinear document model to learn distributed representations of words, which can capture both the semantic similarities of words across languages and the predictive information with respect to the classification task. Similarly, Tang and Wan (2014) jointly embedded texts in different languages into a joint semantic space representing sentiment. Zhou et al. (2014) employed aligned sentences in the BWE learning process, but in the sentiment classification process only representations in the source language are used for training, and representations in the target language are used for predicting labels. An important weakness of these three works was that aligned sentences were required.

Some work has trained sentiment-specific BWEs using annotated sentiment information in both languages (Zhou et al., 2015, 2016), which is desirable, but this is not applicable to our scenario. Our goal is to adapt BWEs to a specific domain without requiring additional task-specific engineering or knowledge sources beyond having access to plentiful target-language in-domain unlabeled text. Both of the approaches we study in this work fit this criterion, the delightfully simple method for adapting BWEs can improve the performance of any off-the-shelf classifier that is based on BWEs, while the broadly applicable semi-supervised approach of Häusser et al. (2017) can improve the performance of any off-the-shelf classifier.

2.3 Bilingual Lexicon Induction (BLI)

BLI is an important task that has been addressed by a large amount of previous work. The goal of BLI is to automatically extract word translation pairs using BWEs. While BLI is often used to provide an intrinsic evaluation of BWEs (Lazaridou et al., 2015; Vulić and Moens, 2015; Vulić and Korhonen, 2016) it is also useful for tasks such as machine translation (Madhyastha and España Bohnet, 2017). Most work on BLI using BWEs focuses on frequent words in high-resource domains such as parliament proceedings or news texts. Recently Heyman et al. (2017) tackled BLI of words in the medical domain. This task is useful for many applications such as terminology extraction or OOV mining for machine translation of medical texts. Heyman et al. (2017) show that when only a small amount of medical data is available, BLI using BWEs tends to perform poorly. Especially BWEs obtained using post-hoc mapping (Mikolov et al., 2013b; Lazaridou et al., 2015) fail on this task. Consequently, Heyman et al. (2017) build BWEs using aligned documents and then engineer a specialized classification-based approach to BLI. In contrast, our delightfully simple approach to create high-quality BWEs for the medical domain requires only monolingual data. We show that our adapted BWEs yield impressive improvements over non-adapted BWEs in this task with both cosine similarity and with the classifier of Heyman et al. (2017). In addition, we show that the broadly applicable method can push performance further using easily accessible unlabeled data.

3 Adaptation of BWEs

BWEs trained on *general domain* texts usually result in lower performance when used in a system for a *specific domain*. There are two reasons for this. (i) Vocabularies of specific domains contain words that are not used in the general case, e.g., names of medicines or diseases. (ii) The meaning of a word varies across domains; e.g., "apple" mostly refers to a fruit in general domains, but is an electronic device in many product reviews.

The delightfully simple method adapts general domain BWEs in a way that preserves the semantic knowledge from general domain data and leverages *monolingual* domain specific data to create domain-specific BWEs. Our domain-adaptation approach is applicable to any language-pair in which monolingual data is available. Unlike other methods, our approach is task independent: it only requires unlabeled in-domain target language text.

3.1 Approach

To create domain adapted BWEs, we first train MWEs (monolingual word embeddings) in both languages and then map those into the same space using post-hoc mapping (Mikolov et al., 2013b). We train MWEs for both languages by concatenating monolingual out-of-domain and in-domain data. The out-of-domain data allows us to create accurate distributed representations of common vocabulary while the in-domain data embeds domain specific words. We then map the two MWEs using a small seed lexicon to create the adapted BWEs. Because post-hoc mapping only requires a seed lexicon as bilingual signal it can easily be used with (cheap) monolingual data.

For **post-hoc mapping**, we use Mikolov et al. (2013b)'s approach. This model assumes a $W \in \mathbb{R}^{d_1 \times d_2}$ matrix which maps vectors from the source to the target MWEs where d_1 and d_2 are the embedding space dimensions. A seed lexicon of $(x_i, y_i) \in L \subseteq \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$ pairs is needed where x_i and y_i are source and target MWEs. W can be learned using ridge regression by minimizing the L_2-regularized mapping error between the source x_i and the target y_i vectors:

$$\min_W \sum_i ||Wx_i - y_i||_2^2 + \lambda ||W||_2^2 \qquad (1)$$

where λ is the regularization weight. Based on the source embedding x, we then compute a target embedding as Wx.

We create MWEs with word2vec skipgram (Mikolov et al., 2013a)[1] and estimate W with *scikit-learn* (Pedregosa et al., 2011). We use default parameters.

4 Cross-Lingual Sentiment Classification

In CLSC, an important application of BWEs, we train a supervised sentiment model on training data available in the source (a resource rich language) and apply it to the target (a resource poor language, for which there is typically no training data available). Because BWEs embed source and target words in the same space, annotations in the source (represented as BWEs) enable transfer learning. For CLSC of tweets, a drawback of BWEs trained on non-twitter data is that they do not produce embeddings for twitter-specific vocabulary, e.g., slang words like English *coool* and (Mexican) Spanish *chido*, resulting in lost information when a sentiment classifier uses them.

4.1 Training Data for Twitter Specific BWEs

As comparable non-twitter data we use OpenSubtitles (Lison and Tiedemann, 2016) which contains 49.2M English and Spanish subtitle sentences respectively (**Subtitle**). The reason behind choosing Subtitles is that although it is out-of-domain it contains slang words similar to tweets thus serving as a strong baseline in our setup. We experiment with two monolingual twitter data sets:

(i) **22M_tweets**: Downloaded[2] English (17.2M) and Spanish (4.8M) tweets using the public

[1]https://github.com/dav/word2vec
[2]We downloaded for a month starting on 2016-10-15.

Twitter Streaming API[3] with language filters *en* and *es*

(ii) a **BACKGROUND** corpus of 296K English and 150K Spanish (non-annotated) tweets released with the test data of the RepLab task (Amigó et al., 2013) described below

All twitter data was tokenized using Bird et al. (2009) and lowercased. User names, URLs, numbers, emoticons and punctuation were removed.

As lexicon for the mapping, we use the BNC word frequency list (Kilgarriff, 1997), a list of 6,318 frequent English lemmas and their Spanish translations, obtained from Google Translate. Note that we do not need a domain-specific lexicon in order to get good quality adapted BWEs.

4.2 Training Data for Sentiment Classifiers

For sentiment classification, we use data from the *RepLab 2013* shared task (Amigó et al., 2013). The data is annotated with positive, neutral and negative labels and contains English and Spanish tweets. We used the official English training set (26.6K tweets) and the Spanish test set (14.9K) in the resource-poor setup. We only use the 7.2K Spanish labeled training data for comparison reasons in §6.2, which we will discuss later.

The shared task was on target-level sentiment analysis, i.e., given a pair (document, target entity), the gold annotation is based on whether the sentiment expressed by the document is about the target. For example: *I cried on the back seat of my BMW!* where *BMW* is the target would be negative in the sentence-level scenario. However, it is neutral in the target-level case because the negative sentiment is not related to BMW. The reason for using this dataset is that it contains comparable English and Spanish tweets annotated for sentiment. There are other twitter datasets for English (Nakov et al., 2016) and Spanish (Garcıa-Cumbreras et al., 2016), but they were downloaded at different times and were annotated using different annotation methodologies, thus impeding a clean and consistent evaluation.

4.3 Sentiment Systems

For evaluating our adapted BWEs on the *RepLab* dataset we used a **target-aware** sentiment classifier introduced by Zhang et al. (2016). The network first embeds input words using pre-trained BWEs and feeds them to a bi-directional gated neural network. Pooling is applied on the hidden representations of the left and right context of the target mention respectively. Finally, gated neurons are used to model the interaction between the target mention and its surrounding context. During training we hold our pre-trained BWEs fixed and keep the default parameters of the model.

We also implement Kim (2014)'s *CNN-non-static* system, which does not use the target information in a given document (**target-ignorant**). The network first embeds input words using pre-trained BWEs and feeds them to a convolutional layer with multiple window sizes. Max pooling is applied on top of convolution followed by a fully connected network with one hidden layer. We used this system as well because it performed comparably to the target-aware system. The reason for this is that only 1% of the used data contains more than one target and out of these rare cases only 14% have differing sentiment labels in the same sentence, which are the difficult cases of target-level sentiment analysis. We used the default parameters as described in (Kim, 2014) with the exception of using 1000 feature maps and 30 epochs, based on our initial experiments. Word embeddings are fixed during the training just as for the target-aware classifier.

4.4 Results

As we previously explained we evaluate our adaptation method on the task of target-level sentiment classification using both target-aware and target-ignorant classifiers. For all experiments, our two baselines are off-the-shelf classifiers using non-adapted BWEs, i.e., BWEs trained only using Subtitles. Our goal is to show that our BWE adaptation method can improve the performance of such classifiers. We train our adapted BWEs on the concatenation of Subtitle and 22M_tweets or BACKGROUND respectively. In addition, we also report results with BWEs trained only on tweets.

To train the sentiment classifiers we use the English Replab training set and we evaluate on the Spanish test set. To show the performance that can be reached in a monolingual setup, we report results obtained by using annotated Spanish sentiment data instead of English (oracle). We train two oracle sentiment classifiers using (i) MWEs trained on only the Spanish part of Subtitle and (ii)

[3]dev.twitter.com/streaming/overview

			target-	
			aware	ignorant
oracle		MWE_Subtitle	62.17%	63.27%
oracle		BWE_Subtitle	62.46%	63.50%
domain adaptation		Baseline	55.14%	59.05%
domain adaptation	del. simple	BACKGROUND	56.79%	58.50%
domain adaptation	del. simple	22M_tweets	59.44%	61.14%
domain adaptation	del. simple	Subtitle+BACKGROUND	58.64%	59.34%
domain adaptation	del. simple	Subtitle+22M_tweets	60.99%	61.06%

Table 1: Accuracy of the BWE adaptation approach on the target-level sentiment classification task. The oracle systems used Spanish sentiment training data instead of English.

BWEs trained on Subtitle using posthoc mapping. The difference between the two is that the embeddings of (ii) are enriched with English words which can be beneficial for the classification of Spanish tweets because they often contain a few English words.

We do not compare with word embedding adaptation methods relying on specialized resources. The point of our work is to study task-independent methods and to the best of our knowledge ours is the first such attempt. Similarly, we do not compare against machine translation based sentiment classifiers (e.g., (Zhou et al., 2016)) because for their adaptation in-domain parallel data would be needed.

Table 1 gives results for both classifiers. It shows that the adaptation of Subtitle based BWEs with data from Twitter (22M_tweets and BACKGROUND) clearly outperforms the Baseline in all cases. The target-aware system performed poorly with the baseline BWEs and could benefit significantly from the adaptation approach. The target-ignorant performed better with the baseline BWEs but could also benefit from the adaptation. Comparing results with the Twitter-dataset-only based BWEs, the 22M_tweets performed better even though the BACKGROUND dataset is from the same topic as the RepLab train and test sets. Our conjecture is that the latter is too small to create good BWEs. In combination with Subtitles, 22M_tweets also yields better results than when combined with BACKGROUND. Although the best accuracy was reached using the 22M_tweets-only based BWEs, it is only slightly better then the adapted Subtitles+22M_tweets based BWEs. In §6 we show that both the semantic knowledge from Subtitles and the domain-specific information from tweets are needed to further improve results.

Comparing the two classifiers we can say that they performed similarly in terms of their best results. On the other hand, the target-ignorant system had better results on average. This might seem surprising at first because the system does not use the target as information. But considering the characteristics of RepLab, i.e., that the number of tweets that contains multiple targets is negligible, using the target offers no real advantage.

Although we did not focus on the impact of the seed lexicon size, we ran post-hoc mapping with different sizes during our preliminary experiments. With 1,000 and 100 word pairs in the lexicon the target-ignorant system suffered 0.5% and 4.0% drop in average of our setups respectively.

To summarize the result: using adapted BWEs for the Twitter CLSC task improves the performance of off-the-shelf classifiers.

5 Medical Bilingual Lexicon Induction

Another interesting downstream task for BWEs is bilingual lexicon induction. Given a list of words in a source language, the goal of BLI is to mine translations for each word in a chosen target language. The medical bilingual lexicon induction task proposed in (Heyman et al., 2017) aims to mine medical words using BWEs trained on a very small amount of English and Dutch monolingual medical data. Due to the lack of resources in this domain, good quality BWEs are hard to build using in-domain data only. We show that by enriching BWEs with general domain knowledge (in the form of general domain monolingual corpora) better results can be achieved on this medical domain task.

5.1 Experimental Setup

We evaluate our improved BWEs on the dataset provided by Heyman et al. (2017). The monolingual medical data consists of English and Dutch medical articles from Wikipedia. The English (resp. Dutch) articles contain 52,336 (resp. 21,374) sentences. A total of 7,368 manually annotated word translation pairs occurring in the English (source) and Dutch (target) monolingual corpora are provided as gold data. This set is split 64%/16%/20% into trn/dev/test. 20% of the English words have multiple translations. Given an English word, the task is to find the correct Dutch translation.

As monolingual general-domain data we use

	cosine similarity		classifier	
	F_1 (top)	F_1 (all)	F_1 (top)	F_1 (all)
Baseline	13.43	9.84	37.73	36.61
Baseline BNC lexicon	-	-	20.73	21.78
Adapted medical lexicon	14.18	14.15	40.71	38.09
Adapted BNC lexicon	16.29	16.71	22.10	21.50

Table 2: We report F_1 results for medical BLI with the cosine similarity and the classifier based systems. We present baseline and our proposed domain adaptation method using both general and medical lexicons.

the English and Dutch data from Europarl (v7) (Koehn, 2005), a corpus of 2 million sentence pairs. Although Europarl is a parallel corpus, we use it in a monolingual way and shuffle each side of the corpus before training. By using massive cheap data we create high-quality MWEs in each language which are still domain-specific (due to inclusion of medical data). To obtain an out-of-domain seed lexicon, we translated the English words in BNC to Dutch using Google Translate (just as we did before for the Twitter CLSC task). We then use the out-of-domain BNC and the in-domain medical seed lexicons in separate experiments to create BWEs with post-hoc mapping. Note, we did not concatenate the two lexicons because (i) they have a small common subset of source words which have different target words, thus having a negative effect on the mapping and (ii) we did not want to modify the medical seed lexicon because it was taken from previous work.

5.2 BLI Systems

To perform BLI we use two methods. Because BWEs represent words from different languages in a shared space, BLI can be performed via *cosine similarity* in this space. In other words, given a BWE representing two languages V_s and V_t, the translation of each word $s \in V_s$ can be induced by taking the word $t \in V_t$ whose representation $\vec{x_t}$ in the BWE is closest to the representation $\vec{x_s}$.

As the second approach we use a *classifier based system* proposed by Heyman et al. (2017). This neural network based system is comprised of two main modules. The first is a character-level LSTM which aims to learn orthographic similarity of word pairs. The other is the concatenation of the embeddings of the two words using embedding layers with the aim of learning the similarity among semantic representations of the words. Dense layers are applied on top of the two modules before the output soft-max layer. The classifier is trained using positive and negative word pair examples and a pre-trained word embedding model. Negative examples are randomly generated for each positive one in the training lexicon. We used default parameters as reported by Heyman et al. (2017) except for the t classification thresholds (used at prediction time). We fine-tuned these on dev. We note that the system works with pre-trained MWEs as well (and report these as official baseline results) but it requires BWEs for candidate generation at prediction time, thus we use BWEs for the system's input for all experiments. In preliminary work, we had found that MWE and BWE results are similar.

5.3 Results

Heyman et al. (2017)'s results are our *baseline*. Table 2 compares its performance with our adapted BWEs, with both cosine similarity and classification based systems. "top" F_1 scores are based on the most probable word as prediction only; "all" F_1 scores use all words as prediction whose probability is above the threshold. It can be seen that the cosine similarity based system using adapted BWEs clearly outperforms the non-adapted BWEs which were trained in a resource poor setup.[4] Moreover, the best performance was reached using the general seed lexicon for the mapping which is due to the fact that general domain words have better quality embeddings in the MWE models, which in turn gives a better quality mapping.

The classification based system performs significantly better comparing to cosine similarity by exploiting the seed lexicon better. Using adapted BWEs as input word embeddings for the system further improvements were achieved which shows the better quality of our BWEs. Simulating an even poorer setup by using a general lexicon, the

[4]The results for cosine similarity in (Heyman et al., 2017) are based on BWESG-based BWEs (Vulić and Moens, 2016) trained on a small document aligned parallel corpus without using a seed lexicon.

performance gain of the classifier is lower. This shows the significance of the medical seed lexicon for this system. On the other hand, adapted BWEs have better performance compared to non-adapted ones using the best translation while they have just slightly lower F_1 using multiple translations. This result shows that while with adapted BWEs the system predicts better "top" translations, it has a harder time when predicting "all" due to the increased vocabulary size.

To summarize: we have shown that adapted BWEs increase performance for this task and domain; and they do so independently of the task-specific system that is used.

6 Semi-Supervised Learning

In addition to the experiments that show our BWE-adaptation method's task and language independence, we investigate ways to further incorporate unlabeled data to overcome data sparsity.

Häusser et al. (2017) introduce a semi-supervised method for neural networks that makes associations from the vector representation of labeled samples to those of unlabeled ones and back. This lets the learning exploit unlabeled samples as well. While Häusser et al. (2017) use their model for image classification, we adapt it to CLSC of tweets and medical BLI. We show that our semi-supervised model requires adapted BWEs to be effective and yields significant improvements. This innovative method is general and can be applied to any classification when unlabeled text is available.

6.1 Model

Häusser et al. (2017)'s basic assumption is that the embeddings of labeled and unlabeled samples – i.e., the representations in the neural network on which the classification layer is applied – are similar within the same class. To achieve this, walking cycles are introduced: a cycle starts from a labeled sample, goes to an unlabeled one and ends at a labeled one. A cycle is correct if the start and end samples are in the same class. The probability of going from sample A to B is proportional to the cosine similarity of their embeddings. To maximize the number of correct cycles, two loss functions are employed: Walker loss and Visit loss.

Walker loss penalizes incorrect walks and encourages a uniform probability distribution of walks to the correct class. It is defined as:

$$\mathcal{L}_{walker} := H(T, P^{aba}) \tag{2}$$

where H is the cross-entropy function, P^{aba}_{ij} is the probability that a cycle starts from sample i and ends at j and T is the uniform target distribution:

$$T_{ij} := \begin{cases} 1/(\#c(i)) & \text{if } c(i) = c(j) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $c(i)$ is the class of sample i and $\#c(i)$ is the number of occurrences of $c(i)$ in the labeled set.

Visit loss encourages cycles to visit all unlabeled samples, rather than just those which are the most similar to labeled samples. It is defined as:

$$\mathcal{L}_{visit} := H(V, P^{visit})$$

$$P^{visit}_j := \langle P^{ab}_{ij} \rangle_i \tag{4}$$

$$V_j := \frac{1}{U}$$

where H is cross-entropy, P^{ab}_{ij} is the probability that a cycle starts from sample i and goes to j and U is the number of unlabeled samples.

The total loss during training is the sum of the walker, visit and classification (cross-entropy between predicted and gold labels) losses which is minimized using Adam (Kingma and Ba, 2015).

We adapt this model (including the two losses) to sentiment classification, focusing on the target-ignorant classifier, and the classifier based approach for BLI. We will call these systems **semisup**[5]. Due to the fact that we initialize the embedding layers for both classifiers with BWEs the models are able to make some correct cycles at the beginning of the training and improve them later on. We will describe the labeled and unlabeled datasets used in the subsequent sections below.

We use Häusser et al. (2017)'s implementation of the losses, with 1.0, 0.5 and 1.0 weights for the walker, visit and classification losses, respectively, for CLSC based on preliminary experiments. We fine-tuned the weights for BLI on dev for each experiment.

[5]We publicly release our implementation: https://github.com/hangyav/biadapt

		semisup
	Baseline	58.67% (-0.38%)
domain adaptation	BACKGROUND	57.41% (-1.09%)
	22M_tweets	60.19% (-0.95%)
	Subtitle+BACKGROUND	60.31% (0.97%)
	Subtitle+22M_tweets	63.23% (2.17%)

Table 3: Accuracy on CLSC of the adapted BWE approach with the semisup (target-ignorant with additional loss functions) system comparing to the target-ignorant in brackets.

6.2 Semi-Supervised CLSC

As in §4.4, we use pre-trained BWEs to initialize the classifier and use English sentiment training data as the labeled set. Furthermore, we use the Spanish sentiment training data as the unlabeled set, ignoring its annotation. This setup is very similar to real-word low-resource scenarios: unlabeled target-language tweets are easy to download while labeled English ones are available.

Table 3 gives results for adapted BWEs and shows that semisup helps only when word embeddings are adapted to the Twitter domain. As mentioned earlier, semisup compares labeled and unlabeled samples based on their vector representations. By using BWEs based on only Subtitles, we lose too many embeddings of similar English and Spanish tweets. On the other hand, if we use only tweet-based BWEs we lose good quality semantic knowledge which can be learned from more standard text domains. By combining the two domains we were able to capture both sides. For Subtitle+22M_tweets, we even get very close to the best oracle (BWE_Subtitle) in Table 1 getting only 0.27% less accuracy – an impressive result keeping in mind that we did not use labeled Spanish data.

The RepLab dataset contains tweets from 4 topics: automotive, banking, university, music. We manually analyzed similar tweets from the labeled and unlabeled sets. We found that when using semisup, English and Spanish tweets from the same topics are more similar in the embedding space than occurs without the additional losses. Topics differ in how they express sentiment – this may explain why semisup increases performance for RepLab.

Adding supervision. To show how well semisup can exploit the unlabeled data we used both English and Spanish sentiment training data together to train the sentiment classifiers. Table 4 shows that by using annotated data in both languages we get clearly better results than when using only one language. Tables 3 and 4 show that for Subtitle+22M_tweets based BWEs, the semisup approach achieved high improvement (2.17%) comparing to target-ignorant with English training data only, while it achieved lower improvement (0.97%) with the Subtitle+BACKGROUND based BWEs. On the other hand, adding labeled Spanish data caused just a slight increase comparing to semisup with Subtitle+22M_tweets based BWEs (0.59%), while in case of Subtitle+BACKGROUND we got significant additional improvement (2.61%). This means that with higher quality BWEs, unlabeled target-language data can be exploited better.

It can also be seen that the target-aware system outperformed the target-ignorant system using additional labeled target-language data. The reason could be that it is a more complex network and therefore needs more data to reach high performance.

The results in table 4 are impressive: our target-level system is strongly competitive with the official shared task results. We achieved high accuracy on the Spanish test set by using only English training data. Comparing our best system which used all training data to the official results (Amigó et al., 2013) we would rank 2nd even though our system is not fine-tuned for the RepLab dataset. Furthermore, we also outperformed the oracles when using annotated data from both languages which shows the additional advantage of using BWEs.

6.3 Semi-Supervised BLI

For BLI experiments with semisup we used word pairs from the medical seed lexicon as the labeled set (with negative word pairs generated as described in §5.2). As opposed to CLSC and the work of (Häusser et al., 2017), for this task we do not have an unlabeled set, and therefore we need to generate it. We developed two scenarios. For the first, **BNC**, we generate a general unlabeled set using English words from the **BNC** lexicon and generate 10 pairs out of each word by using the 5 most similar Dutch words based on the corresponding BWEs and 5 random Dutch words. For the second scenario, **medical**, we generate an in-domain unlabeled set by generating for each English word in the **medical** lexicon the 3 most similar Dutch

		lang	target-aware	target-ignorant
oracle	MWE_Subtitle	Es	62.17%	63.27%
	BWE_Subtitle	Es	62.46%	63.50%
domain adaptation	Subtitle+BACKGROUND	En	58.64%	59.34%
	Subtitle+BACKGROUND	En+Es	64.01%	62.92% (2.61%)
	Subtitle+22M_tweets	En	60.99%	61.06%
	Subtitle+22M_tweets	En+Es	64.24%	63.82% (0.59%)

Table 4: Accuracy on CLSC of both target-aware and target-ignorant systems using English or/and Spanish sentiment training data. Column *lang* shows the language of the used training data. Differences comparing to semisup are indicated in brackets.

	F_1 (top)	F_1 (all)
Baseline+BNC	35.04 (-0.66)	34.98 (-1.40)
Baseline+medical	36.20 (0.50)	36.55 (0.16)
Adapted+BNC	41.01 (0.30)	39.04 (0.95)
Adapted+medical	41.44 (0.73)	37.51 (-0.57)

Table 5: Results with the semi-supervised system for BLI. Differences comparing to previous results are indicated in brackets. Baseline results are compared to rerun experiments of Heyman et al. (2017) using BWEs instead of MWEs.

words based on BWEs and for each of these we use the 5 most similar English words (ignoring the words which are in the original medical lexicon) and 5 negative words. The idea behind these methods is to automatically generate an unlabeled set that hopefully has a similar positive and negative word pair distribution to the distribution in the labeled set.

Results in Table 5 show that adding semisup to the classifier further increases performance for BLI as well. For the baseline system, when using only in-domain text for creating BWEs, only the medical unlabeled set was effective, general domain word pairs could not be exploited due to the lack of general semantic knowledge in the BWE model. On the other hand, by using our domain adapted BWEs, which contain both general domain and in-domain semantical knowledge, we can exploit word pairs from both domains. Results for adapted BWEs increased in 3 out of 4 cases, where the only exception is when using multiple translations for a given source word (which may have been caused by the bigger vocabulary size).

These results show that adapted BWEs are needed to exploit unlabeled data well which leads to an impressive overall 3.71 increase compared with the best result in previous work (Heyman et al., 2017), by using only unlabeled data.

7 Conclusion

Bilingual word embeddings trained on general domain data yield poor results in out-of-domain tasks. We presented experiments on two different low-resource task/domain combinations. Our delightfully simple task independent method to adapt BWEs to a specific domain uses unlabeled monolingual data only. We showed that with the support of adapted BWEs the performance of off-the-shelf methods can be increased for both cross-lingual Twitter sentiment classification and medical bilingual lexicon induction. Furthermore, by adapting the broadly applicable semi-supervised approach of Häusser et al. (2017) (which until now has only been applied in computer vision) we were able to effectively exploit unlabeled data to further improve performance. We showed that, when also using high-quality adapted BWEs, the performance of the semi-supervised systems can be significantly increased by using unlabeled data at classifier training time. In addition, CLSC results are competitive with a system that uses target-language labeled data, even when we use no such target-language labeled data.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable input. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement № 640550).

References

Enrique Amigó, Jorge Carrillo de Albornoz, Irina Chugur, Adolfo Corujo, Julio Gonzalo, Tamara Martín, Edgar Meij, Maarten de Rijke, Damiano Spina, Enrique Amigo, Jorge Carrillo de Albornoz, Tamara Martin, and Maarten de Rijke. 2013. Overview of replab 2013: Evaluating online reputation monitoring systems. In *Proc. CLEF*.

A.R. Balamurali and Adity Joshi. 2012. Cross-lingual sentiment analysis for indian languages using linked wordnets. In *Proc. COLING*.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2010. Multilingual subjectivity: Are more languages better? In *Proc. COLING*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proc. EMNLP*.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. NAACL-HLT*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. EACL*.

Miguel Ángel Garcıa-Cumbreras, Julio Villena-Román, Eugenio Martınez-Cámara, Manuel Carlos Díaz-Galiano, María-Teresa Martín-Valdivia, and L. Alfonso Ureña-López. 2016. Overview of tass 2016. In *Proc. TASS*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proc. ICML*.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proc. NAACL-HLT*.

Lin Gui, Ruifeng Xu, Qin Lu, Jun Xu, Jian Xu, Bin Liu, and Wang Xiaolong. 2013. A mixed model for cross lingual opinion analysis. In *Proc. NLPCC*.

Philip Häusser, Alexander Mordvintsev, and Daniel Cremers. 2017. Learning by Association - A versatile semi-supervised training method for neural networks. In *Proc. CVPR*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proc. ACL*.

Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2017. Bilingual lexicon induction by learning to combine word-level and character-level representations. In *Proc. EACL*.

Adam Kilgarriff. 1997. Putting frequencies in the dictionary. *International Journal of Lexicography*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proc. ACL*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proc. LREC*.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K. Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proc. ACL*.

Pranava Swaroop Madhyastha and Cristina España Bohnet. 2017. Learning bilingual projections of embeddings for vocabulary expansion in machine translation. In *Proc. RepL4NLP*.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proc. ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. ICLR*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proc. SemEval*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense Word Embeddings by Orthogonal Transformation. In *Proc. NAACL-HLT*.

Xuewei Tang and Xiaojun Wan. 2014. Learning bilingual embedding model for cross-language sentiment classification. In *Proc. WI-IAT*.

Ivan Vulić and Anna Korhonen. 2016. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. In *Proc. ACL*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proc. ACL*.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proc. ACL*.

Min Xiao and Yuhong Guo. 2013. Semi-supervised representation learning for cross-lingual text classification. In *Proc. EMNLP*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proc. NAACL-HLT*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated Neural Networks for Targeted Sentiment Analysis. In *Proc. AAAI 2016*.

Guangyou Zhou, Tingting He, and Jun Zhao. 2014. Bridging the Language Gap: Learning Distributed Semantics for Cross-Lingual Sentiment Classification. In *Proc. NLPCC*.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proc. ACL*.

Xinjie Zhou, Xianjun Wan, and Jianguo Xiao. 2016. Cross-lingual sentiment classification with bilingual document representation learning. In *Proc. ACL*.

Knowledgeable Reader: Enhancing Cloze-Style Reading Comprehension with External Commonsense Knowledge

Todor Mihaylov and **Anette Frank**
Research Training Group AIPHES
Department of Computational Linguistics, Heidelberg University
Heidelberg, Germany
{mihaylov,frank}@cl.uni-heidelberg.de

Abstract

We introduce a neural reading comprehension model that integrates *external commonsense knowledge*, encoded as a key-value memory, in a cloze-style setting. Instead of relying on only document-to-question interaction or discrete features as in prior work, our model attends to relevant external knowledge and combines this knowledge with the context representation before inferring the answer. This allows the model to attract and imply knowledge from an external knowledge source that is not explicitly stated in the text, but that is relevant for inferring the answer. Our model improves results over a very strong baseline on a hard *Common Nouns* dataset, making it a strong competitor of much more complex models. By including knowledge *explicitly*, our model can also provide *evidence* about the background knowledge used in the RC process.

1 Introduction

Reading comprehension (RC) is a language understanding task similar to question answering, where a system is expected to read a given passage of text and answer questions about it. Cloze-style reading comprehension is a task setting where the question is formed by replacing a token in a sentence of the story with a placeholder (left part of Figure 1).

In contrast to many previous complex models (Weston et al., 2015; Dhingra et al., 2017; Cui et al., 2017; Munkhdalai and Yu, 2016; Sordoni et al., 2016) that perform *multi-turn reading* of a story and a question before inferring the correct answer, we aim to tackle the cloze-style RC task in a way that resembles how humans solve it: using, in addition, background knowledge. We develop

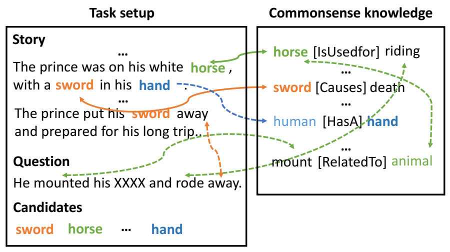


Figure 1: Cloze-style reading comprehension with external commonsense knowledge.

a neural model for RC that can successfully deal with tasks where most of the information to infer answers from is given in the document (story), but where additional information is needed to predict the answer, which can be retrieved from a knowledge base and added to the context representations explicitly.[1] An illustration is given in Figure 1.

Such knowledge may be *commonsense knowledge* or *factual background knowledge about entities and events* that is not explicitly expressed but can be found in a knowledge base such as ConceptNet (Speer et al., 2017), BabelNet (Navigli and Ponzetto, 2012), Freebase (Tanon et al., 2016) or domain-specific KBs collected with Information Extraction approaches (Fader et al., 2011; Mausam et al., 2012; Bhutani et al., 2016). Thus, we aim to define a neural model that encodes pre-selected knowledge in a memory, and that learns to include the available knowledge as an enrichment to the context representation.

The main difference of our model to prior state-of-the-art is that instead of relying only on document-to-question interaction or discrete features while performing multiple hops over the document, our model (i) *attends to relevant selected*

[1] *'Context representation'* refers to a vector representation computed from textual information only (i.e., document (story) or question).

external knowledge and (ii) *combines this knowledge with the context representation before inferring the answer*, in a single hop. This allows the model to explicitly imply knowledge that is not stated in the text, but is relevant for inferring the answer, and that can be found in an external knowledge source. Moreover, by including knowledge explicitly, our model *provides evidence and insight about the used knowledge in the RC.*

Our main contributions are: **(i)** We develop a method for integrating knowledge in a simple but effective reading comprehension model (*AS Reader*, Kadlec et al. (2016)) and improve its results significantly whereas other models employ features or multiple hops. **(ii)** We examine two sources of common knowledge: WordNet (Miller et al., 1990) and ConceptNet (Speer et al., 2017) and show that this type of knowledge is important for answering *common nouns* questions and also improves slightly the performance for *named entities*.

(iii) We show that knowledge facts can be added directly to the text-only representation, enriching the neural context encoding. **(iv)** We demonstrate the effectiveness of the injected knowledge by case studies and data statistics in a qualitative evaluation study.

2 Reading Comprehension with Background Knowledge Sources

In this work, we examine the impact of using external knowledge as supporting information for the task of *cloze style* reading comprehension.

We build a system with two modules. The first, *Knowledge Retrieval*, performs *fact retrieval* and selects a number of facts f_1, ..., f_p that might be relevant for connecting story, question and candidate answers. The second, main module, *the Knowledgeable Reader*, is a knowledge-enhanced neural module. It uses the input of the story context tokens $d_{1..m}$, the question tokens $q_{1..n}$, the set of answer candidates $a_{1..k}$ and a set of 'relevant' background knowledge facts $f_{1..p}$ in order to select the right answer. To include external knowledge for the RC task, we encode each fact $f_{1..p}$ and use attention to select the most relevant among them for each token in the story and question. We expect that enriching the text with additional knowledge about the mentioned concepts will improve the prediction of correct answers in a strong *single-pass* system. See Figure 1 for illustration.

2.1 Knowledge Retrieval

In our experiments we use knowledge from the Open Mind Common Sense (OMCS, Singh et al. (2002)) part of ConceptNet, a crowd-sourced resource of commonsense knowledge with a total of ∼630k facts. Each fact f_i is represented as a triple f_i=*(subject, relation, object)*, where *subject* and *object* can be multi-word expressions and *relation* is a relation type. An example is: *([bow]$_{subj}$, [IsUsedFor]$_{rel}$, [hunt, animals]$_{obj}$)*

We experiment with three set-ups: using (i) all facts from OMCS that pertain to ConceptNet, referred to as *CN5All*, (ii) using all facts from *CN5All* excluding some WordNet relations referred to as *CN5Sel(ected)* (see Section 3), and using (iii) facts from OMCS that have *source* set to *WordNet* (*CN5WN3*).

Retrieving relevant knowledge. For each instance (D, Q, $A_{1..10}$) we retrieve relevant commonsense background facts. We first retrieve facts that contain lemmas that can be looked up via tokens contained in any D(ocument), Q(uestion) or A(nswer candidates). We add a weight value for each node: 4, if it contains a lemma of a candidate token from A; 3, if it contains a lemma from the tokens of Q; and 2 if it contains a lemma from the tokens of D. The selected weights are chosen heuristically such that they model relative fact importance in different interactions as A+A > A+Q > A+D > D+Q > D+D. We weight the fact triples that contain these lemmas as nodes, by summing the weights of the subject and object arguments. Next, we sort the knowledge triples by this overall weight value. To limit the memory of our model, we run experiments with different sizes of the top number of facts (P) selected from all instance fact candidates, $P \in \{50, 100, 200\}$. As additional retrieval limitation, we force the number of facts per answer candidate to be the same, in order to avoid a frequency bias for an answer candidate that appears more often in the knowledge source. Thus, if we select the maximum 100 facts for each task instance and we have 10 answer candidates $a_{i=1..10}$, we retrieve the top 10 facts for each candidate a_i that has either a subject or an object lemma for a token in a_i. If the same fact contains lemmas of two candidates a_i and a_j ($j > i$), we add the fact once for a_i and do not add the same fact again for a_j. If several facts have the same weight, we take

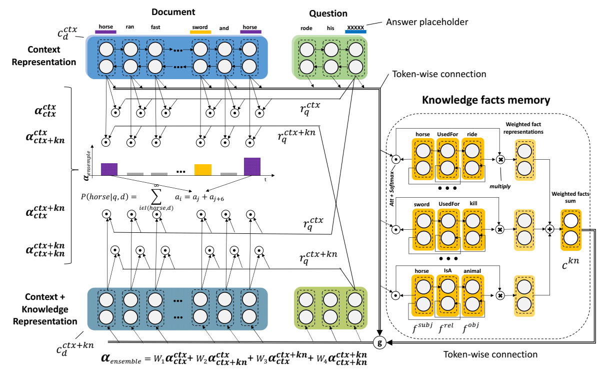


Figure 2: The Knowledgeable Reader combines plain *context* & *enhanced* (*context* + *knowledge*) repres. of *D* and *Q* and retrieved knowledge from the explicit memory with the *Key-Value* approach.

the first in the order of the list[2], i.e., the order of retrieval from the database. If one candidate has less than 10 facts, the overall fact candidates for the sample will be less than the maximum (100).

2.2 Neural Model: Extending the Attention Sum Reader with a Knowledge Memory

We implement our *Knowledgeable Reader (KnReader)* using as a basis the *Attention Sum Reader* as one of the strongest core models for single-hop RC. We extend it with a knowledge fact memory that is filled with pre-selected facts. Our aim is to examine how adding commonsense knowledge to a simple yet effective model can improve the RC process and to show some evidence of that by attending on the incorporated knowledge facts. The model architecture is shown in Figure 2.

Base Attention Model. The Attention-Sum Reader (Kadlec et al., 2016), our base model for RC reads the input of story tokens $d_{1..n}$, the question tokens $q_{1..m}$, and the set of candidates $a_{1..10}$ that occur in the story text. The model calculates the attention between the question representation r_q and the story token context encodings of the candidate tokens $a_{1..10}$ and sums the attention scores for the candidates that appear multiple times in the story. The model selects as answer the candidate that has the highest attention score.

Word Embeddings Layer. We represent input document and question tokens w by looking up their embedding representations $e_i = Emb(w_i)$, where Emb is an embedding lookup function. We apply dropout (Srivastava et al., 2014) with keep probability $p = 0.8$ to the output of the embeddings lookup layer.

Context Representations. To represent the document and question contexts, we first encode the tokens with a Bi-directional GRU (Gated Recurrent Unit) (Chung et al., 2014) to obtain context-encoded representations for document ($c^{ctx}_{d_1..n}$) and question ($c^{ctx}_{q_1..m}$) encoding:

$$c^{ctx}_{d_1..n} = BiGRU^{ctx}(e_{d_1..n}) \in \mathbb{R}^{n \times 2h} \quad (1)$$

$$c^{ctx}_{q_1..m} = BiGRU^{ctx}(e_{q_1..m}) \in \mathbb{R}^{m \times 2h} \quad (2)$$

, where d_i and q_i denote the ith token of a text sequence d (document) and q (question), respectively, n and m is the size of d and q and h the output hidden size (256) of a single GRU unit. $BiGRU$ is defined in (3), with e_i a word embedding vector

$$BiGRU^{ctx}(e_i, h_{i_{prev}}) = \begin{matrix}[\overrightarrow{GRU}(e_i, \overrightarrow{h_{i_{prev}}}), \\ \overleftarrow{GRU}(e_i, \overleftarrow{h_{i_{prev}}})]\end{matrix} \quad (3)$$

, where $h_{i_{prev}} = [\overrightarrow{h_{i_{prev}}}, \overleftarrow{h_{i_{prev}}}]$, and $\overrightarrow{h_{i_{prev}}}$ and $\overleftarrow{h_{i_{prev}}}$ are the previous hidden states of the forward and backward layers. Below we use $BiGRU^{ctx}(e_i)$ without the hidden state, for short.

Question Query Representation. For the question we construct a single vector representation r^{ctx}_q by retrieving the token representation at the placeholder (XXXX) index *pl* (cf. Figure 2):

$$r^{ctx}_q = c^{ctx}_{q_i..m}[pl] \in \mathbb{R}^{2h} \quad (4)$$

where $[pl]$ is an element pickup operation.

Our question vector representation is different from the original *AS Reader* that builds the question by concatenating the *last states* of a forward and backward layer $[\overrightarrow{GRU}(e_m), \overleftarrow{GRU}(e_1)]$. We changed the original representation as we observed some very long questions and in this way aim to prevent the context encoder from 'forgetting' where the placeholder is.

Answer Prediction: Q^{ctx} to D^{ctx} Attention. In order to predict the correct answer to the given question, we rank the given answer candidates $a_1..a_L$ according to the normalized attention sum score between the context (ctx) representation of the question placeholder r^{ctx}_q and the representation of the candidate tokens in the document:

$$P(a_i|q,d) = softmax(\sum \alpha_{i_j}) \quad (5)$$

$$\alpha_{i_j} = Att(r^{ctx}_q, c^{ctx}_{d_j}), i \in [1..L] \quad (6)$$

[2]We also experimented with re-ranking the facts with the same weight sums using tf-idf but we did not notice a difference in performance.

, where j is an index pointer from the list of indices that point to the candidate a_i token occurrences in the document context representation c_d. Att is a dot product.

Enriching Context Representations with Knowledge (Context+Knowledge). To enhance the representation of the context, we add knowledge, retrieved as a set of knowledge facts.

Knowledge Encoding. For each instance in the dataset, we retrieve a number of relevant facts (cf. Section 2.1). Each retrieved fact is represented as a triple $f = (w^{subj}_{1..L_{subj}}, w^{rel}_0, w^{obj}_{1..L_{obj}})$, where $w^{subj}_{1..L_{subj}}$ and $w^{obj}_{1..L_{obj}}$ are a multi-word expressions representing the $subject$ and $object$ with sequence lengths L_{subj} and L_{obj}, and w^{rel}_0 is a word token corresponding to a relation.[3] As a result of fact encoding, we obtain a separate knowledge memory for each instance in the data.

To encode the knowledge we use a $BiGRU$ to encode the triple argument tokens into the following context-encoded representations:

$$f^{subj}_{last} = BiGRU(Emb(w^{subj}_{1..L_{subj}}), 0) \quad (7)$$

$$f^{rel}_{last} = BiGRU(Emb(w^{rel}_0), f^{subj}_{last}) \quad (8)$$

$$f^{obj}_{last} = BiGRU(Emb(w^{obj}_{1..L_{subj}}), f^{rel}_{last}) \quad (9)$$

, where f^{subj}_{last}, f^{rel}_{last}, f^{obj}_{last} are the final hidden states of the context encoder $BiGRU$, that are also used as initial representations for the encoding of the next triple attribute in left-to-right order. See *Supplement* for comprehensive visualizations. The motivation behind this encoding is: (i) We encode the knowledge fact attributes in the same vector space as the plain tokens; (ii) we preserve the triple directionality; (iii) we use the relation type as a way of filtering the *subject* information to initialize the *object*.

Querying the Knowledge Memory. To enrich the context representation of the document and question tokens with the facts collected in the knowledge memory, we select a single *sum of weighted fact representations* for each token using Key-Value retrieval (Miller et al., 2016). In our model the *key* $M^{k(ey)}_i$ can be either f^{subj}_{last} or f^{obj}_{last} and the *value* $M^{v(alue)}_i$ is f^{obj}_{last}.

For each context-encoded token $c^{ctx}_{s_i}$ ($s = d, q$; i the token index) we attend over all knowledge memory keys M^k_i in the retrieved P knowledge facts. We use an attention function Att, scale the scalar attention value using $softmax$, multiply it with the value representation M^v_i and sum the result into a single vector value representation $c^{kn}_{s_i}$:

$$c^{kn}_{s_i} = \sum softmax(Att(c^{ctx}, M^k_{1..P}))^T M^v_{1..P} \quad (10)$$

Att is a dot product, but it can be replaced with another attention function. As a result of this operation, the context token representation $c^{ctx}_{s_i}$ and the corresponding retrieved knowledge $c^{kn}_{s_i}$ are in the same vector space $\in \mathbb{R}^{2h}$.

Combine Context and Knowledge $(ctx+kn)$**.** We combine the original context token representation $c^{ctx}_{s_i}$, with the acquired knowledge representation $c^{kn}_{s_i}$ to obtain $c^{ctx+kn}_{s_i}$:

$$c^{ctx+kn}_{s_i} = \gamma c^{ctx}_{s_i} + (1-\gamma) c^{kn}_{s_i} \quad (11)$$

, where $\gamma = 0.5$. We keep γ static but it can be replaced with a gating function.

Answer Prediction: $Q^{ctx(+kn)}$ **to** $D^{ctx(+kn)}$**.** To rank answer candidates $a_1..a_L$ we use attention sum similar to Eq.5 over an attention $\alpha^{ensemble}_{i_j}$ that combines attentions between context (ctx) and context+knowledge ($ctx+kn$) representations of the question ($r^{ctx(+kn)}_q$) and candidate token occurrences a_{i_j} in the document $c^{ctx(+kn)}_{d_j}$:

$$P(a_i|q,d) = softmax(\sum \alpha^{ensemble}_{i_j}) \quad (12)$$

$$\alpha^{ensemble}_{i_j} = \begin{array}{r} W_1 Att(r^{ctx}_q, c^{ctx}_{d_j}) \\ +W_2 Att(r^{ctx}_q, c^{ctx+kn}_{d_j}) \\ +W_3 Att(r^{ctx+kn}_q, c^{ctx}_{d_j}) \\ +W_4 Att(r^{ctx+kn}_q, c^{ctx+kn}_{d_j}) \end{array} \quad (13)$$

, where j is an index pointer from the list of indices that point to the candidate a_i token occurrences in the document context representation $c^{ctx(+kn)}_d$. $W_{1..4}$ are scalar weights initialized with 1.0 and optimized during training.[4] We propose the combination of ctx and $ctx + kn$ attentions because our task does not provide supervision whether the knowledge is needed or not.

[3]The 0 in w^{rel}_0 indicates that we encode the relation as a single *relation type* word. Ex. */r/IsUsedFor*.

[4]An example for learned $W_{1..4}$ is (2.13, 1.41, 1.49, 1.84) in setting (CBT CN, CN5Sel, Subj-Obj as k-v, 50 facts).

	CN	NE
Train	120,769 / 470	108,719 / 433
Dev	2,000 / 448	2,000 / 412
Test	2,500 / 461	2,500 / 424
Vocab	53,185	53,063

Table 1: Characteristics of Children Book Test datasets. CN: *Common Nouns*, NE: *Named Entities*. Cells for *Train, Dev, Test* show overall numbers of examples and average story size in tokens.

3 Data and Task Description

We experiment with knowledge-enhanced cloze-style reading comprehension using the *Common Nouns* and *Named Entities* partitions of the Children's Book Test (CBT) dataset (Hill et al., 2015).

In the CBT cloze-style task a system is asked to read a children story context of 20 sentences. The following 21st sentence involves a placeholder token that the system needs to predict, by choosing from a given set of 10 candidate words from the document. An example with suggested external knowledge facts is given in Figure 1. While in its *Common Nouns* setup, the task can be considered as a language modeling task, Hill et al. (2015) show that humans can answer the questions without the full context with an accuracy of only *64.4%* and a language model alone with *57.7%*. By contrast, the human performance when given the full context is at *81.6%*. Since the best neural model (Munkhdalai and Yu, 2016) achieves only *72.0%* on the task, we hypothesize that the task itself can benefit from external knowledge. The characteristics of the data are shown in Table 1.

Other popular cloze-style datasets such as CNN/Daily Mail (Hermann et al., 2015) or WhoDidWhat (Onishi et al., 2016) are mainly focused on finding *Named Entities* where the benefit of adding commonsense knowledge (as we show for the *NE* part of CBT) would be more limited.

Knowledge Source. As a source of commonsense knowledge we use the *Open Mind Common Sense* part of ConceptNet 5.0 that contains 630k fact triples. We refer to this entire source as *CN5All*. We conduct experiments with subparts of this data: *CN5WN3* which is the WordNet 3 part of *CN5All* (213k triples) and *CN5Sel*, which excludes the following WordNet relations: *RelatedTo, IsA, Synonym, SimilarTo, HasContext*.

4 Related Work

Cloze-Style Reading Comprehension. Following the original MCTest (Richardson et al., 2013) dataset multiple-choice version of cloze-style RC) recently several large-scale, automatically generated datasets for cloze-style reading comprehension gained a lot of attention, among others the 'CNN/Daily Mail' (Hermann et al., 2015; Onishi et al., 2016) and the Children's Book Test (CBTest) data set (Hill et al., 2015). Early work introduced simple but good *single turn models* (Hermann et al., 2015; Kadlec et al., 2016; Chen et al., 2016), that read the document once with the question representation 'in mind' and select an answer from a given set of candidates. More complex models (Weston et al., 2015; Dhingra et al., 2017; Cui et al., 2017; Munkhdalai and Yu, 2016; Sordoni et al., 2016) perform *multi-turn reading* of the story context and the question, before inferring the correct answer or use features (GA Reader, Dhingra et al. (2017). Performing multiple hops and *modeling a deeper relation between question and document* was further developed by several models (Seo et al., 2017; Xiong et al., 2016; Wang et al., 2016, 2017; Shen et al., 2016) on another generation of RC datasets, e.g. SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017) or TriviaQA (Joshi et al., 2017).

Integrating Background Knowledge in Neural Models. Integrating background knowledge in a neural model was proposed in the *neural-checklist model* by Kiddon et al. (2016) for text generation of recipes. They copy words from a list of ingredients instead of inferring the word from a global vocabulary. Ahn et al. (2016) proposed a language model that copies fact attributes from a topic knowledge memory. The model predicts a fact in the knowledge memory using a gating mechanism and given this fact, the next word to be selected is copied from the fact attributes. The knowledge facts are encoded using embeddings obtained using *TransE* (Bordes et al., 2013). Yang et al. (2017) extended a *seq2seq* model with attention to external facts for dialogue and recipe generation and a co-reference resolution-aware language model. A similar model was adopted by He et al. (2017) for answer generation in dialogue. Incorporating external knowledge in a neural model has proven beneficial for several other tasks: Yang and Mitchell (2017) incorporated knowledge directly into the *LSTM* cell state to improve event and entity extraction. They used knowledge embeddings trained on WordNet (Miller et al., 1990) and NELL (Mitchell et al., 2015) using the *BILINEAR* (Yang et al., 2014) model.

Work similar to ours is by Long et al. (2017), who have introduced a new task of Rare Entity Prediction. The task is to read a paragraph from WikiLinks (Singh et al., 2012) and to fill a blank field in place of a missing entity. Each missing entity is characterized with a short description derived from Freebase, and the system needs to choose one from a set of pre-selected candidates to fill the field. While the task is superficially similar to cloze-style reading comprehension, it differs considerably: first, when considering the text without the externally provided entity information, it is clearly ambiguous. In fact, the task is more similar to Entity Linking tasks in the Knowledge Base Population (KBP) tracks at TAC 2013-2017, which aim at detecting specific entities from Freebase. Our work, by contrast, examines the impact of injecting external knowledge in a reading comprehension, or NLU task, where the knowledge is drawn from a commonsense knowledge base, ConceptNet in our case. Another difference is that in their setup, the reference knowledge for the candidates is explicitly provided as a single, fixed set of knowledge facts (the entity description), encoded in a single representation. In our work, we are retrieving (typically) distinct sets of knowledge facts that might (or might not) be relevant for understanding the story and answering the question. Thus, in our setup, we crucially depend on the ability of the attention mechanism to retrieve relevant pieces of knowledge. Our aim is to examine to what extent commonsense knowledge can contribute to and improve the cloze-style RC task, that in principle is supposed to be solvable without explicitly given additional knowledge. We show that by integrating external commonsense knowledge we achieve clear improvements in reading comprehension performance over a strong baseline, and thus we can speculate that humans, when solving this RC task, are similarly using commonsense knowledge as implicitly understood background knowledge.

Recent unpublished work in Weissenborn et al. (2017) is driven by similar intentions. The authors exploit knowledge from ConceptNet to improve the performance of a reading comprehension model, experimenting on the recent SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017) datasets. While the source of the background knowledge is the same, the way of integrating this knowledge into the model and task is different. (i) We are using attention to select unordered fact triples using key-value retrieval and (ii) we integrate the knowledge that is considered relevant explicitly for each token in the context. The model of Weissenborn et al. (2017), by contrast, explicitly reads the acquired additional knowledge sequentially after reading the document and question, but transfers the background knowledge implicitly, by refining the word embeddings of the words in the document and the question with the words from the supporting knowledge that share the same lemma. In contrast to the implicit knowledge transfer of Weissenborn et al. (2017), our explicit attention over external knowledge facts can deliver insights about the used knowledge and how it interacts with specific context tokens (see Section 6).

5 Experiments and Results

We perform quantitative analysis through experiments. We study the impact of the used knowledge and different model components that employ the external knowledge. Some of the experiments below focus only on the *Common Nouns (CN)* dataset, as it has been shown to be more challenging than *Named Entities (NE)* in prior work.

5.1 Model Parameters

We experiment with different model parameters.

Number of facts. We explore different sizes of knowledge memories, in terms of number of acquired facts. If not stated otherwise, we use 50 facts per example.

Key-Value Selection Strategy. We use two strategies for defining key and value (Key/Value): *Subj/Obj* and *Obj/Obj*, where *Subj* and *Obj* are the subject and object attributes in the fact triples and they are selected as *Key* and *Value* for the KV memory *(see Section 2.2, Querying the Knowledge Memory)*. If not stated otherwise, we use the *Subj/Obj* strategy.

Answer Selection Components. If not stated otherwise, we use ensemble attention $\alpha_{ensemble}$ (combinations of *ctx* and *ctx+kn*) to rank the answers. We call this our *Full model* (see Sec. 2.2).

Source	Dev	Test
CN5All	71.40	66.72
CN5WN3 (WN3)	70.70	68.48
CN5Sel(ected)	**71.85**	67.64

Table 2: Results with different knowledge sources, for CBT-CN (Full model, 50 facts).

# facts	50	100	200	500
Dev	**71.85**	71.35	71.40	71.20
Test	67.64	67.44	**68.12**	67.24

Table 3: Results for CBT (CN) with different numbers of facts. (Full model, CN5Sel)

Hyper-parameters. For our experiments we use pre-trained Glove (Pennington et al., 2014) embeddings, *BiGRU* with hidden size 256, batch size of 64 and learning reate of 0.001 as they were shown (Kadlec et al., 2016) to perform good on the AS Reader.

5.2 Empirical Results

We perform experiments with the different model parameters described above. We report accuracy on the *Dev* and *Test* and use the results on *Dev* set for pruning the experiments.

Knowledge Sources. We experiment with different configuration of *ConceptNet* facts (see Section 3). Results on the *CBT CN* dataset are shown in Table 2. *CN5Sel* works best on the *Dev* set but *CN5WN3* works much better on *Test*. Further experiments use the *CN5Sel* setup.

Number of facts. We further experiment with different numbers of facts on the *Common Nouns* dataset (Table 3). The best result on the *Dev* set is for 50 facts so we use it for further experiments.

Component ablations. We ensemble the attentions from different combinations of the interaction between the question and document *context (ctx)* representations and *context+knowledge (ctx+kn)* representations in order to infer the right answer (see Section 2.2, Answer Ranking).

Table 4 shows that the combination of different interactions between *ctx* and *ctx+kn* representations leads to clear improvement over the *w/o knowledge* setup, in particular for the *Common Nouns* dataset. We also performed ablations for a model with 100 facts (see *Supplement*).

Key-Value Selection Strategy. Table 5 shows that for the *NE* dataset, the two strategies perform equally well on the *Dev* set, whereas the *Subj/Obj* strategy works slightly better on the *Test* set. For *Common Nouns*, *Subj/Obj* is better.

	NE		CN	
D_{repr} to Q_{repr} **interaction**	**Dev**	**Test**	**Dev**	**Test**
D_{ctx}, Q_{ctx} (w/o know)	75.50	70.30	68.20	64.80
D_{ctx+kn}, Q_{ctx+kn}	76.45	69.68	70.85	66.32
D_{ctx}, Q_{ctx+kn}	**77.10**	69.72	70.80	66.32
D_{ctx+kn}, Q_{ctx}	75.65	**70.88**	71.20	**67.96**
Full model	76.80	70.24	**71.85**	67.64
w/o D_{ctx}, Q_{ctx}	75.95	70.24	70.65	67.12
w/o D_{ctx+kn}, Q_{ctx+kn}	76.20	69.80	70.75	67.00
w/o D_{ctx}, Q_{ctx+kn}	76.55	70.52	71.75	66.32
w/o D_{ctx+kn}, Q_{ctx}	76.05	70.84	70.80	66.80

Table 4: Results for different combinations of interactions between document (D) and question (Q) *context (ctx)* and *context + knowledge (ctx+kn)* representations. (CN5Sel, 50 facts)

	NE		CN	
Key/Value	**Dev**	**Test**	**Dev**	**Test**
Subj/Obj	76.65	71.52	71.85	67.64
Obj/Obj	76.70	71.28	71.25	67.48

Table 5: Results for key-value knowledge retrieval and integration. (CN5Sel, 50 facts). *Subj/Obj* means: we attend over the fact subject (Key) and take the weighted fact object as value (Value).

	NE		CN	
Models	**dev**	**test**	**dev**	**test**
Human (ctx + q)	-	81.6	-	81.6
Single interaction				
LSTMs (ctx + q) (Hill et al., 2015)	51.2	41.8	62.6	56.0
AS Reader	73.8	68.6	68.8	63.4
AS Reader (our impl)	75.5	70.3	68.2	64.8
KnReader (ours)	77.4	71.4	71.8	67.6
Multiple interactions				
MemNNs (Weston et al., 2015)	70.4	66.6	64.2	63.0
EpiReader (Trischler et al., 2016)	74.9	69.0	71.5	67.4
GA Reader (Dhingra et al., 2017)	77.2	71.4	71.6	68.0
IAA Reader (Sordoni et al., 2016)	75.3	69.7	72.1	69.2
AoA Reader (Cui et al., 2017)	75.2	68.6	72.2	69.4
GA Reader (+feat)	77.8	72.0	74.4	70.7
NSE (Munkhdalai and Yu, 2016)	77.0	71.4	74.3	71.9

Table 6: Comparison of KnReader to existing end-to-end neural models on the benchmark datasets.

Comparison to Previous Work. Table 6 compares our model (*Knowledgeable Reader*) to previous work on the CBT datasets. We show the results of our model with the settings that performed best on the *Dev* sets of the two datasets *NE* and *CN*: for *NE*, (D_{ctx+kn}, Q_{ctx}) with 100 facts; for *CN* the *Full model* with *50 facts*, both with *CN5Sel*.

Note that our work focuses on the impact of external knowledge and employs a *single inter-*

action (single-hop) between the document context and the question so we primarily compare to and aim at improving over similar models. *KnReader* clearly outperforms prior single-hop models on both datasets. While we do not improve over the state of the art, our model stands well among other models that perform multiple hops. In the *Supplement* we also give comparison to ensemble models and some models that use re-ranking strategies.

6 Discussion and Analysis

6.1 Analysis of the empirical results.

Our experiments examined key parameters of the KnReader. As expected, injection of background knowledge yields only small improvements over the baseline model for *Named Entities*. However, on this dataset our single-hop model is competitive to most multi-hop neural architectures.

The integration of knowledge clearly helps for the *Common Nouns* task. The impact of knowledge sources (Table 2) is different on the *Dev* and *Test* sets which indicates that either the model or the data subsets are sensitive to different knowledge types and retrieved knowledge. Table 5 shows that attending over the *Subj* of the knowledge triple is slightly better than *Obj*. This shows that using a *Key-Value* memory is valuable. A reason for lower performance of *Obj/Obj* is that the model picks facts that are similar to the candidate tokens, not adding much new information. From the empirical results we see that training and evaluation with less facts is slightly better. We hypothesize that this is related to the lack of supervision on the retrieved and attended knowledge.

6.2 Interpreting Component Importance

Figure 3 shows the impact on prediction accuracy of individual components of the *Full model*, including the interaction between D and Q with ctx or $ctx + kn$ (w/o ctx-only). The values for each component are obtained from the attention weights, without retraining the model. The difference between blue (left) and orange (right) values indicates how much the module contributes to the model. Interestingly, the ranking of the contribution ($D_{ctx}, Q_{ctx+kn} > D_{ctx+kn}, Q_{ctx} > D_{ctx+kn}$, Q_{ctx+kn}) corresponds to the component importance ablation on the Dev set, lines 5-8, Table 4.

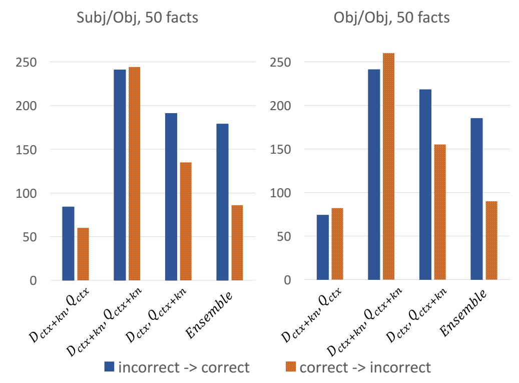


Figure 3: # of items with reversed prediction (±correct) for each combination of (*ctx+kn*, *ctx*) for Q and D. We report the number of *wrong* → *correct* (blue) and *correct* → *wrong* (orange) changes when switching from score w/o knowledge to score w/ knowledge. The best model type is *Ensemble*. (*Full model w/o* D_{ctx}, Q_{ctx}).

6.3 Qualitative Data Investigation

We will use the attention values of the interactions between $D_{ctx(+kn)}$ and $Q_{ctx(+kn)}$ and attentions to facts from each candidate token and the question placeholder to interpret how knowledge is employed to make a prediction for a single example.

Method: Interpreting Model Components. We manually inspect examples from the evaluation sets where *KnReader* improves prediction (blue (left) category, Fig. 3) or makes the prediction worse (orange (right) category, Fig. 3). Figure 4 shows the question with placeholder, followed by answer candidates and their associated attention weights as assigned by the model *w/o knowledge*. The matrix shows selected facts and their assigned weights for the question and the candidate tokens. Finally, we show the attention weights determined by the knowledge-enhanced D to Q interactions. The attention to the correct answer (*head*) is low when the model considers the text alone (*w/o knowledge*). When adding retrieved knowledge to the Q only (row $ctx, ctx + kn$) and to both Q and D (row $ctx + kn, ctx + kn$) the score improves, while when adding knowledge to D alone (row $ctx + kn, ctx$) the score remains ambiguous. The combined score *Ensemble* (see Eq. 13) then takes the final decision for the answer. In this example, the question can be answered without the story. The model tries to find knowledge that is related to *eyes*. The fact *eyes /r/PartOf head* is not contained in the retrieved knowledge but in-

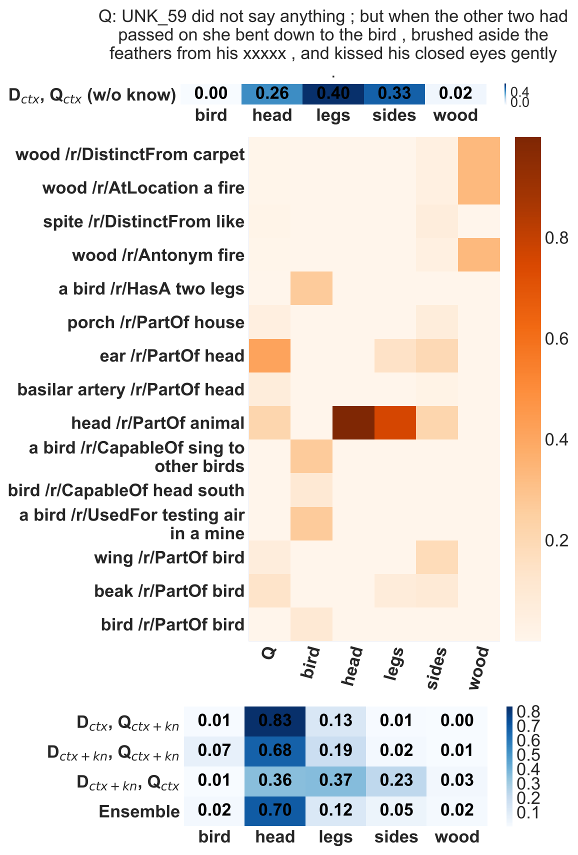


Figure 4: Interpreting the components of *KnReader*. Adding knowledge to Q and D increases the score for the correct answer. Results for top 5 candidates are shown. (*Full model, CN data, CN5Sel, Subj/Obj, 50 facts*)

stead the model selects the fact *ear /r/PartOf head* which receives the highest attention from Q. The weighted *Obj* representation (*head*) is added to the question with the highest weight, together with *animal* and *bird* from the next highly weighted facts This results in a high score for the Q_{ctx} to D_{ctx+kn} interaction with candidate *head*. See *Supplement for more details*.

Using the method described above, we analyze several example cases (presented in *Supplement*) that highlight different aspects of our model. Here we summarize our observations.

(i.) Answer prediction from Q or Q+D. In both human and machine RC, questions can be answered based on the question alone (Figure 4) or jointly with the story context (Case 2, *Suppl.*). We show that empirically, enriching the question with knowledge is crucial for the first type, while enrichment of Q and D is required for the second.

(ii.) Overcoming frequency bias.. We show that when appropriate knowledge is available and selected, the model is able to correct a frequency bias towards an incorrect answer (Cases 1 and 3).

(iii.) Providing appropriate knowledge. We observe a lack of knowledge regarding events (e.g. *take off* vs. *put on clothes*, Case 2; *climb up*, Case 5). Nevertheless relevant knowledge from CN5 can help predicting infrequent candidates (Case 2).

(iv.) Knowledge, Q and D encoding. The context encoding of facts allows the model to detect knowledge that is semantically related, but not surface near to phrases in Q and D (Case 2). The model finds facts to non-trivial paraphrases (e.g. *undressed–naked*, Case 2).

7 Conclusion and Future Work

We propose a neural cloze-style reading comprehension model that incorporates external commonsense knowledge, building on a single-turn neural model. Incorporating external knowledge improves its results with a relative error rate reduction of 9% on *Common Nouns*, thus the model is able to compete with more complex RC models. We show that the types of knowledge contained in ConceptNet are useful. We provide quantitative and qualitative evidence of the effectiveness of our model, that learns how to select relevant knowledge to improve RC. The attractiveness of our model lies in its *transparency and flexibility*: due to the attention mechanism, we can trace and analyze the facts considered in answering specific questions. This opens up for deeper investigation and future improvement of RC models in a targeted way, allowing us to investigate what knowledge sources are required for different data sets and domains. Since our model directly integrates background knowledge with the document and questioncontext representations, it can be adapted to very different task settings where we have a pair of two arguments (i.e. *entailment*, *question answering, etc.*) In future work, we will investigate even tighter integration of the attended knowledge and stronger reasoning methods.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We thank the reviewers for their helpful questions and comments.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. In *CoRR*, volume abs/1608.00318.

Nikita Bhutani, H V Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 55–64, Austin, Texas. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Junyoung Chung, Çalar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv e-prints*, abs/1412.3555. Presented at the Deep Learning workshop at NIPS2014.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602. Association for Computational Linguistics.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 199–208. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. volume abs/1511.02301.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics (ACL) 2017.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918. Association for Computational Linguistics.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics.

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 825–834, Copenhagen, Denmark. Association for Computational Linguistics.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller.

1990. Introduction to wordnet: An on-line lexical database*. In *International Journal of Lexicography*, volume 3, pages 235–244.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Tsendsuren Munkhdalai and Hong Yu. 2016. Reasoning with Memory Augmented Neural Networks for Language Comprehension. In *International Conference on Learning Representations (ICLR) 2017*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235, Austin, Texas. Association for Computational Linguistics.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bi-Directional Attention Flow for Machine Comprehension. In *Proceedings of International Conference of Learning Representations 2017*, pages 1–12.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*.

Parmjit Singh, T Lin, E.T. Mueller, G Lim, T Perkins, and W.L. Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *Lecture Notes in Computer Science*, volume 2519, pages 1223–1237.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015.

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. abs/1606.02245.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *AAAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. volume 15, pages 1929–1958.

Thomas Pellissier Tanon, Denny Vrande, San Francisco, Sebastian Schaffert, and Thomas Steiner. 2016. From Freebase to Wikidata : The Great Migration. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 128–137. Association for Computational Linguistics.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198. Association for Computational Linguistics.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *CoRR*, abs/1612.04211.

Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural NLU systems. *CoRR*, abs/1706.02596.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations (ICLR), 2015*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR), 2017*, volume abs/1611.01604.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *International Conference on Learning Representations (ICLR), 2015*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1851–1860. Association for Computational Linguistics.

Multi-Relational Question Answering from Narratives: Machine Reading and Reasoning in Simulated Worlds

Igor Labutov **Bishan Yang**
Machine Learning Dept.
Carnegie Mellon University
Pittsburgh, PA 15213
ilabutov@cs.cmu.edu
bishan@cs.cmu.edu

Anusha Prakash
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213
anushap@andrew.cmu.edu

Amos Azaria
Computer Science Dept.
Ariel University
Israel
amos.azaria@ariel.ac.il

Abstract

Question Answering (QA), as a research field, has primarily focused on either knowledge bases (KBs) or free text as a source of knowledge. These two sources have historically shaped the kinds of questions that are asked over these sources, and the methods developed to answer them. In this work, we look towards a practical use-case of *QA over user-instructed knowledge* that uniquely combines elements of both *structured QA* over knowledge bases, and *unstructured QA* over narrative, introducing the task of *multi-relational QA over personal narrative*. As a first step towards this goal, we make three key contributions: (i) we generate and release TEXTWORLDSQA, a set of five diverse datasets, where each dataset contains dynamic narrative that describes entities and relations in a simulated world, paired with variably compositional questions over that knowledge, (ii) we perform a thorough evaluation and analysis of several state-of-the-art QA models and their variants at this task, and (iii) we release a lightweight Python-based framework we call TEXTWORLDS for easily generating arbitrary additional worlds and narrative, with the goal of allowing the community to create and share a growing collection of diverse worlds as a test-bed for this task.

1 Introduction

Personal devices that interact with users via natural language conversation are becoming ubiquitous (e.g., Siri, Alexa), however, very little of that conversation today allows the user to teach, and then query, new knowledge. Most of the focus in these personal devices has been on Question Answering (QA) over general world-knowledge (e.g., *“who was the president in 1980”* or *“how many ounces are in a cup”*). These devices open a new and exciting possibility of enabling end-users to teach machines in natural language, e.g., by expressing the state of their personal world to its virtual assistant (e.g., via narrative about people and events in that user’s life) and enabling the user to ask questions over that personal knowledge (e.g., *“which engineers in the QC team were involved in the last meeting with the director?”*).

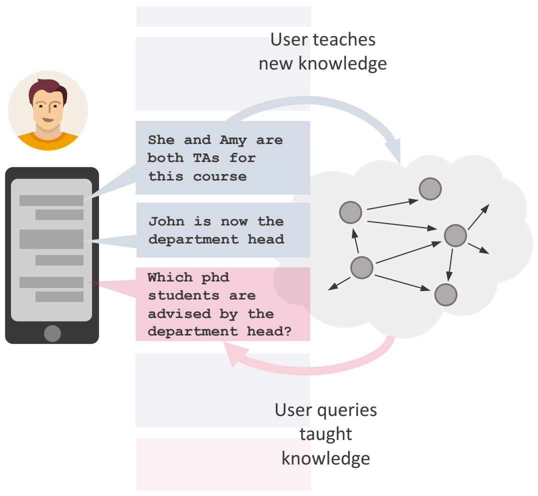

User teaches new knowledge
She and Amy are both TAs for this course
John is now the department head
Which phd students are advised by the department head?
User queries taught knowledge

Figure 1: Illustration of our task: relational question answering from dynamic knowledge expressed via personal narrative

This type of questions highlight a unique blend of two conventional streams of research in Question Answering (QA) – QA over *structured* sources such as knowledge bases (KBs), and QA over *unstructured* sources such as free text. This blend is a natural consequence of our problem setting: (i) users may choose to express rich relational knowledge about their world, in turn enabling them to pose complex **composi-**

Academic Department World

```
1. There is an associate professor named Andy
2. He returned from a sabbatical
3. This professor currently has funding
4. There is a masters level course called G301
5. That course is taught by him
6. That class is part of the mechanical
   engineering department
7. Roslyn is a student in this course
8. U203 is a undergraduate level course
9. Peggy and that student are TAs for this
   course
…

What students are advised by a professor with
funding?
  [Albertha, Roslyn, Peggy, Lucy, Racquel]

What assistant professors advise students who
passed their thesis proposal?
  [Andy]

Which courses have masters student TAs?
  [G301, U101 ]

Who are the professors working on unsupervised
machine learning?
 [Andy, Hanna]
```

Software Engineering World

```
1. There is a new important mobile project
2. That project is in the implementation stage
3. Hiram is a tester on mobile project
4. Mobile project has moved to the deployment
   stage
5. Andrew created a new issue for mobile
   project: fails with apache stack
6. Andrew is no longer assigned to that project
7. That developer resolved the changelog needs
   to be added issue
…

Are there any developers assigned to projects in
the evaluation stage?
 [Tawnya, Charlott, Hiram]

Who is the null pointer exception during parsing
issue assigned to?
 Hiram

Are there any issues that are resolved for
experimental projects?
 [saving data throws exception,
  wrong pos tag on consecutive words]
```

Figure 2: Illustrative snippets from two sample worlds. We aim to generate natural-sounding first-person narratives from five diverse worlds, covering a range of different events, entities and relations.

tional queries (e.g., *"all CS undergrads who took my class last semester"*), while at the same time (ii) personal knowledge generally evolves through time and has an open and growing set of relations, making natural language the only practical interface for creating and maintaining that knowledge by non-expert users. In short, the task that we address in this work is: **multi-relational question answering from dynamic knowledge expressed via narrative**.

Although we hypothesize that question-answering over personal knowledge of this sort is ubiquitous (e.g., between a professor and their administrative assistant, or even if just in the user's head), such interactions are rarely recorded, presenting a significant practical challenge to collecting a sufficiently large real-world dataset of this type. At the same time, we hypothesize that the technical challenges involved in developing models for relational question answering from narrative would not be fundamentally impacted if addressed via sufficiently rich, but controlled simulated narratives. Such simulations also offer the advantage of enabling us to directly experiment with stories and queries of different complexity, potentially offering additional insight into the fundamental challenges of this task.

While our problem setting blends the problems of relational question answering over knowledge bases and question answering over text, our hypothesis is that end-to-end QA models may learn to answer such multisentential relational queries, without relying on an intermediate knowledge base representation. In this work, we conduct an extensive evaluation of a set of state-of-the-art end-to-end QA models on our task and analyze their results.

2 Related Work

Question answering has been mainly studied in two different settings: KB-based and text-based. KB-based QA mostly focuses on parsing questions to logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2012; Berant et al., 2013; Kwiatkowski et al., 2013; Yih et al., 2015) in order to better retrieve answer candidates from a knowledge base. Text-based QA aims to directly answer questions from the input text. This includes works on early information retrieval-based methods (Banko et al., 2002; Ahn et al., 2004) and methods that build on extracted structured representations from both the question and the input text (Sachan et al., 2015; Sachan and Xing, 2016; Khot et al., 2017; Khashabi et al., 2018b). Although these structured presentations make reasoning more effective, they rely on sophisticated

NLP pipelines and suffer from error propagation. More recently, end-to-end neural architectures have been successfully applied to text-based QA, including Memory-augmented neural networks (Sukhbaatar et al., 2015; Miller et al., 2016; Kumar et al., 2016) and attention-based neural networks (Hermann et al., 2015; Chen et al., 2016; Kadlec et al., 2016; Dhingra et al., 2017; Xiong et al., 2017; Seo et al., 2017; Chen et al., 2017). In this work, we focus on QA over text (where the text is generated from a supporting KB) and evaluate several state-of-the-art memory-augmented and attention-based neural architectures on our QA task. In addition, we consider a sequence-to-sequence model baseline (Bahdanau et al., 2015), which has been widely used in dialog (Vinyals and Le, 2015; Ghazvininejad et al., 2017) and recently been applied to generating answer values from Wikidata (Hewlett et al., 2016).

There are numerous datasets available for evaluating the capabilities of QA systems. For example, MCTest (Richardson et al., 2013) contains comprehension questions for fictional stories. Allen AI Science Challenge (Clark, 2015) contains science questions that can be answered with knowledge from text books. RACE (Lai et al., 2017) is an English exam dataset for middle and high school Chinese students. MULTIRC (Khashabi et al., 2018a) is a dataset that focuses on evaluating multi-sentence reasoning skills. These datasets all require humans to carefully design multiple-choice questions and answers, so that certain aspects of the comprehension and reasoning capabilities are properly evaluated. As a result, it is difficult to collect them at scale. Furthermore, as the knowledge required for answering each question is not clearly specified in these datasets, it can be hard to identify the limitations of QA systems and propose improvements.

Weston et al. (2015) proposes to use synthetic QA tasks (the BABI dataset) to better understand the limitations of QA systems. BABI builds on a simulated physical world similar to interactive fiction (Montfort, 2005) with simple objects and relations and includes 20 different reasoning tasks. Various types of end-to-end neural networks (Sukhbaatar et al., 2015; Lee et al., 2015; Peng et al., 2015) have demonstrated promising accuracies on this dataset. However, the performance can hardly translate to real-world QA datasets, as BABI uses a small vocabulary (150 words) and short sentences with limited language variations (e.g., nesting sentences, coreference). A more sophisticated QA dataset with a supporting KB is WIKIMOVIES (Miller et al., 2016), which contains 100k questions about movies, each of them is answerable by using either a KB or a Wikipedia article. However, WIKIMOVIES is highly domain-specific, and similar to BABI, the questions are designed to be in simple forms with little compositionality and hence limit the difficulty level of the tasks.

Our dataset differs in the above datasets in that (i) it contains five different realistic domains permitting cross-domain evaluation to test the ability of models to generalize beyond a fixed set of KB relations, (ii) it exhibits rich referring expressions and linguistic variations (vocabulary much larger than the BABI dataset), (iii) questions in our dataset are designed to be deeply compositional and can cover multiple relations mentioned across multiple sentences.

Other large-scale QA datasets include Cloze-style datasets such as CNN/Daily Mail (Hermann et al., 2015), Children's Book Test (Hill et al., 2015), and Who Did What (Onishi et al., 2016); datasets with answers being spans in the document, such as SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), and TriviaQA (Joshi et al., 2017); and datasets with human generated answers, for instance, MS MARCO (Nguyen et al., 2016) and SearchQA (Dunn et al., 2017). One common drawback of these datasets is the difficulty in accessing a system's capability of integrating information across a document context. Kočiskỳ et al. (2017) recently emphasized this issue and proposed NarrativeQA, a dataset of fictional stories with questions that reflect the complexity of narratives: characters, events, and evolving relations. Our dataset contains similar narrative elements, but it is created with a supporting KB and hence it is easier to analyze and interpret results in a controlled setting.

3 TEXTWORLDS: Simulated Worlds for Multi-Relational QA from Narratives

In this work, we synthesize narratives in five diverse worlds, each containing a thousand narratives and where each narrative describes the evolution of a simulated user's world from a first-person perspective. In each narrative, the simulated user may introduce new knowledge, update existing knowledge or express a state change (e.g., *"Homework 3 is now due on Friday"* or *"Samantha passed her thesis defense"*). Each narrative is interleaved with questions about the current state of the world, and questions range in complexity depending on the amount of knowledge that needs to be integrated to answer them. This allows us to benchmark a range of QA models at their ability to answer questions that require different extents of relational reasoning to be answered.

The set of worlds that we simulate as part of this work are as follows:

1. MEETING WORLD: This world describes situations related to professional meetings, e.g., meetings being set/cancelled, people attending meetings, topics of meetings.

2. HOMEWORK WORLD: This world describes situations from the first-person perspective of a student, e.g., courses taken, assignments in different courses, deadlines of assignments.

3. SOFTWARE ENGINEERING WORLD: This world describes situations from the first-person perspective of a software development manager, e.g., task assignment to different project team members, stages of software development, bug tickets.

4. ACADEMIC DEPARTMENT WORLD: This world describes situations from the first-person perspective of a professor, e.g., teaching assignments, faculty going/returning from sabbaticals, students from different departments taking/dropping courses.

5. SHOPPING WORLD: This world describes situations about a person shopping for various occasions, e.g., adding items to a shopping list, purchasing items at different stores, noting where items are on sale.

3.1 Narrative

Each world is represented by a set of entities $\mathcal{E}$ and a set of unary, binary or ternary relations $\mathcal{R}$. Formally, a single step in one simulation of a world involves a combination of instantiating new entities and defining new (or mutating existing) relations between entities. Practically, we implement each world as a collection of classes and methods, with each step of the simulation creating or mutating class instances by sampling entities and methods on those entities. By design, these classes and methods are easy to extend, to either enrich existing worlds or create new ones. Each simulation step is then expressed as a natural language statement, which is added to the narrative. In the process of generating a natural language expression, we employ a rich mechanism for generating anaphora, such as *"meeting with John about the performance review"* and *"meeting that I last added"*, in addition to simple pronoun references. This allows us to generate more natural and flowing narratives. These references are generated and composed automatically by the underlying TEXTWORLDS framework, significantly reducing the effort needed to build new worlds. Furthermore, all generated stories also provide additional annotation that maps all entities to underlying gold-standard KB ids, allowing to perform experiments that provide models with different degrees of access to the "simulation oracle".

Statistics	Value
# of total stories	5,000
# of total questions	1,207,022
Avg. # of entity mentions (per story)	217.4
Avg. # of correct answers (per question)	8.7
Avg. # of statements in stories	100
Avg. # of tok. in stories	837.5
Avg. # of tok. in questions	8.9
Avg. # of tok. in answers	1.5
Vocabulary size (tok.)	1,994
Vocabulary size (entity)	10,793

Table 1: TEXTWORLDSQA dataset statistics

We generate 1,000 narratives within each world, where each narrative consists of 100 sentences, plus up to 300 questions interleaved randomly within the narrative. See Figure 1 for two example narratives. Each story in a given world samples its entities from a large general pool of entity names collected from the web (e.g., *people names*, *university names*). Although some entities do overlap between stories, each story in a given world contains a unique flow of events and entities involved in those events. See Table 1 for the data statistics.

3.2 Questions

Formally, questions are queries over the knowledge-base in the state defined up to the point when the question is asked in the narrative. In the narrative, the questions are expressed

Dataset	**Questions**			
	Single Entity/Relation	Multiple entities		
		Single relation	Two relations	Three relations
MEETING	57,590 (41.16%)	46,373 (33.14%)	30,391 (21.72%)	5,569 (3.98%)
HOMEWORK	45,523 (24.10%)	17,964 (9.51%)	93,669 (49.59%)	31,743 (16.80%)
SOFTWARE	47,565 (20.59%)	51,302 (22.20%)	66,026 (28.58%)	66,150 (28.63%)
ACADEMIC	46,965 (24.81%)	54,581 (28.83%)	57,814 (30.53%)	29,982 (15.83%)
SHOPPING	111,522 (26.25%)	119,890 (28.22%)	107,418 (25.29%)	85,982 (20.24%)
All	309,165 (26.33%)	290,110 (24.71%)	355,318 (30.27%)	219,426 (18.69%)

Table 2: Dataset statistics by question type.

in natural language, employing the same anaphora mechanism used in generating the narrative (e.g., *"who is attending the last meeting I added?"*).

We categorize generated questions into four types, reflecting the number and types of facts required to answer them; questions that require more facts to answer are typically more compositional in nature. We categorize each question in our dataset into one of the following four categories:

Single Entity/Single Relation Answers to these questions are a single entity, e.g. *"what is John's email address?"*, or expressed in lambda-calculus notation:

$$\lambda x.\texttt{EmailAddress}(\texttt{John}, x)$$

The answers to these questions are found in a single sentence in the narrative, although it is possible that the answer may change through the course of the narrative (e.g., *"John's new office is GHC122"*).

Multi-Entity/Single Relation Answers to these questions can be multiple entities but involve a single relation, e.g., *"Who is enrolled in the Math class?"*, or expressed in lambda calculus notation:

$$\lambda x.\texttt{TakingClass}(x, \texttt{Math})$$

Unlike the previous category, answers to these questions can be sets of entities.

Multi-Entity/Two Relations Answers to these questions can be multiple entities and involve two relations, e.g., *"Who is enrolled in courses that I am teaching?"*, or expressed in lambda calculus:

$$\lambda x.\exists y.\texttt{EnrolledInClass}(x, y) \\ \wedge \texttt{CourseTaughtByMe}(y)$$

Multi-Entity/Three Relations Answers to these questions can be multiple entities and involve three relations, e.g., *"Which undergraduates are enrolled in courses that I am teaching?"*, or expressed in lambda calculus notation:

$$\lambda x.\exists y.\texttt{EnrolledInClass}(x, y) \\ \wedge \texttt{CourseTaughtByMe}(y) \\ \wedge \texttt{Undergrad}(x)$$

In the data that we generate, answers to questions are always sets of spans in the narrative (the reason for this constraint is for easier evaluation of several existing machine-reading models; this assumption can easily be relaxed in the simulation). In all of our evaluations, we will partition our results by one of the four question categories listed above, which we hypothesize correlates with the difficulty of a question.

4 Methods

We develop several baselines for our QA task, including a logistic regression model and four different neural network models: Seq2Seq (Bahdanau et al., 2015), MemN2N (Sukhbaatar et al., 2015), BiDAF (Seo et al., 2017), and DrQA (Chen et al., 2017). These models generate answers in different ways, e.g., predicting a single entity, predicting spans of text, or generating answer sequences. Therefore, we implement two experimental settings: ENTITY and RAW. In the ENTITY setting, given a question and a story, we treat all the entity spans in the story as candidate answers, and the prediction task becomes a classification problem. In the RAW setting, a model needs to predict the answer spans. For logistic regression and MemN2N, we adopt the ENTITY setting as they are naturally classification models. This ideally provides an upper bound on the performance when considering answer candidate generation. For all the other models, we can apply the RAW setting.

4.1 Logistic Regression

The logistic regression baseline predicts the likelihood of an answer candidate being a true answer.

For each answer candidate e and a given question, we extract the following features: (1) The frequency of e in the story; (2) The number of words within e; (3) Unigrams and bigrams within e; (4) Each non-stop question word combined with each non-stop word within e; (5) The average minimum distance between each non-stop question word and e in the story; (6) The common words (excluding stop words) between the question and the text surrounding of e (within a window of 10 words); (7) Sum of the frequencies of the common words to the left of e, to the right e, and both. These features are designed to help the model pick the correct answer spans. They have shown to be effective for answer prediction in previous work (Chen et al., 2016; Rajpurkar et al., 2016).

We associate each answer candidate with a binary label indicating whether it is a true answer. We train a logistic regression classifier to produce a probability score for each answer candidate. During test, we search for an optimal threshold that maximizes the F1 performance on the validation data. During training, we optimize the cross-entropy loss using Adam (Kingma and Ba, 2014) with an initial learning rate of 0.01. We use a batch size of $10,000$ and train with 5 epochs. Training takes roughly 10 minutes for each domain on a Titan X GPU.

4.2 Seq2Seq

The seq2seq model is based on the sequence to sequence model presented in (Bahdanau et al., 2015), which includes an attention model. Bahdanau et al. (Bahdanau et al., 2015) have used this model to build a neural based machine translation performing at the state-of-the-art. We adopt this model to fit our own domain by including a preprocessing step in which all statements are concatenated with a dedicated token, while eliminating all previously asked questions, and the current question is added at the end of the list of statements. The answers are treated as a sequence of words. We use word embeddings (Zou et al., 2013), as it was shown to improve accuracy. We use 3 GRU (Cho et al., 2014) connected layers, each with a capacity of 256. Our batch size was set to 16. We use gradient descent with an initial learning rate of 0.5 and a decay factor of 0.99, iterating on the data for $50,000$ steps (5 epochs). The training process for each domain took approximately 48 hours on a Titan X GPU.

4.3 MemN2N

End-To-End Memory Network (MemN2N) is a neural architecture that encodes both long-term and short-term context into a memory and iteratively reads from the memory (i.e., multiple hops) relevant information to answer a question (Sukhbaatar et al., 2015). It has been shown to be effective for a variety of question answering tasks (Weston et al., 2015; Sukhbaatar et al., 2015; Hill et al., 2015).

In this work, we directly apply MemN2N to our task with a small modification. Originally, MemN2N was designed to produce a single answer for a question, so at the prediction layer, it uses softmax to select the best answer from the answer candidates. In order to account for multiple answers for a given question, we modify the prediction layer to apply the logistic function and optimize the cross entropy loss instead. For training, we use the parameter setting as in a publicly available MemN2N [1] except that we set the embedding size to 300 instead of 20. We train the model for 100 epochs and it takes about 2 hours for each domain on a Titan X GPU.

4.4 BiDAF-M

BiDAF (Bidirectional Attention Flow Networks) (Seo et al., 2017) is one of the top-performing models on the span-based question answering dataset SQuAD (Rajpurkar et al., 2016). We reimplement BiDAF with simplified parameterizations and change the prediction layer so that it can predict multiple answer spans.

Specifically, we encode the input story $\{x_1, ..., x_T\}$ and a given question $\{q_1, ..., q_J\}$ at the character level and the word level, where the character level uses CNNs and the word level uses pre-trained word vectors. The concatenation of the character and word embeddings are passed to a bidirectional LSTM to produce a contextual embedding for each word in the story context and in the question. Then, we apply the same bidirectional attention flow layer to model the interactions between the context and question embeddings, producing question-aware feature vectors for each word in the context, denoted as $\mathbf{G} \in \mathbb{R}^{d_g \times T}$. $\mathbf{G}$ is then fed into a bidirectional LSTM layer to obtain a feature matrix $\mathbf{M}_1 \in \mathbb{R}^{d_1 \times T}$ for predicting the start offset of the answer span, and $\mathbf{M}_1$ is then passed into

[1] https://github.com/domluna/memn2n

Within-World	MEETING	HOMEWORK	SOFTWARE	DEPARTMENT	SHOPPING	Avg. F1
Logistic Regression	50.1	55.7	60.9	55.9	61.1	56.7
Seq2Seq	22.5	32.6	16.7	39.1	31.5	28.5
MemN2N	55.4	46.6	69.5	67.3	46.3	57.0
BiDAF-M	81.8	76.9	68.4	68.2	68.7	72.8
DrQA-M	81.2	83.6	79.1	76.4	76.5	79.4
Cross-World	MEETING	HOMEWORK	SOFTWARE	DEPARTMENT	SHOPPING	Avg. F1
Logistic Regression	9.0	9.1	11.1	9.9	7.2	9.3
Seq2Seq	8.8	3.5	1.9	5.4	2.6	4.5
MemN2N	23.6	2.9	4.7	14.6	0.07	9.2
BiDAF-M	34.0	6.9	16.1	22.2	3.9	16.6
DrQA-M	46.5	12.2	23.1	28.5	9.3	23.9

Table 3: F_1 scores for different baselines evaluated on both *within-world* and *across-world* settings.

another bidirectional LSTM layer to obtain a feature matrix $\mathbf{M}_2 \in \mathbb{R}^{d_2 \times T}$ for predicting the end offset of the answer span. We then compute two probability scores for each word i in the narrative: $\mathbf{p}^{start} = \text{sigmoid}(\mathbf{w}_1^T[\mathbf{G};\mathbf{M}_1])$ and $\mathbf{p}^{end} = \text{sigmoid}(\mathbf{w}_2^T[\mathbf{G};\mathbf{M}_1;\mathbf{M}_2])$, where $\mathbf{w}_1$ and $\mathbf{w}_2$ are trainable weights. The training objective is simply the sum of cross-entropy losses for predicting the start and end indices.

We use 50 1D filters for CNN character embedding, each with a width of 5. The word embedding size is 300 and the hidden dimension for LSTMs is 128. For optimization, we use Adam (Kingma and Ba, 2014) with an initial learning rate of 0.001, and use a minibatch size of 32 for 15 epochs. The training process takes roughly 20 hours for each domain on a Titan X GPU.

4.5 DrQA-M

DrQA (Chen et al., 2017) is an open-domain QA system that has demonstrated strong performance on multiple QA datasets. We modify the Document Reader component of DrQA and implement it in a similar framework as BiDAF-M for fair comparisons. First, we employ the same character-level and word-level encoding layers to both the input story and a given question. We then use the concatenation of the character and word embeddings as the final embeddings for words in the story and in the question. We compute the aligned question embedding (Chen et al., 2017) as a feature vector for each word in the story and concatenate it with the story word embedding and pass it into a bidirectional LSTM to obtain the contextual embeddings $\mathbf{E} \in \mathbb{R}^{d \times T}$ for words in the story. Another bidirectional LSTM is used to obtain the contextual embeddings for the question, and self-attention is used to compress them into one single vector $\mathbf{q} \in \mathbb{R}^{d}$. The final prediction layer uses a bilinear term to compute scores for predicting the start offset: $\mathbf{p}^{start} = \text{sigmoid}(\mathbf{q}^T\mathbf{W}_1\mathbf{E})$ and another bilinear term for predicting the end offset: $\mathbf{p}^{end} = \text{sigmoid}(\mathbf{q}^T\mathbf{W}_2\mathbf{E})$, where $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weights. The training loss is the same as in BiDAF-M, and we use the same parameter setting. Training takes roughly 10 hours for each domain on a Titan X GPU.

5 Experiments

We use two evaluation settings for measuring performance at this task: *within-world* and *across-world*. In the *within-world* evaluation setting, we test on the same world that the model was trained on. We then compute the precision, recall and F_1 for each question and report the macro-average F1 score for questions in each world. In the *across-world* evaluation setting, the model is trained on four out of the five worlds, and tested on the remaining world. The *across-world* regime is obviously more challenging, as it requires the model to be able to learn to generalize to unseen relations and vocabulary. We consider the *across-world* evaluation setting to be the main evaluation criteria for any future models used on this dataset, as it mimics the practical requirement of any QA system used in personal assistants: it has to be able to answer questions on any new domain the user introduces to the system.

5.1 Results

We draw several important observations from our results. First, we observe that more compositional questions (i.e., those that integrate multiple relations) are more challenging for most models - as

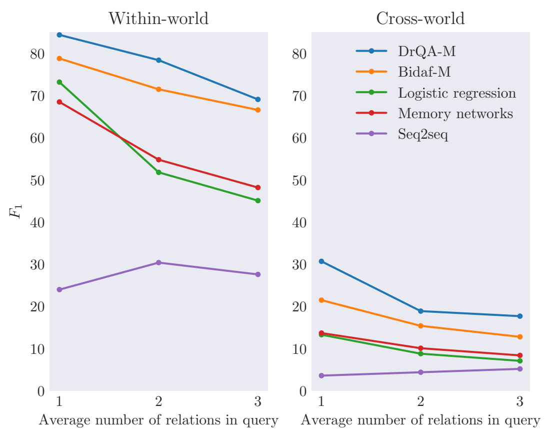


Figure 3: F_1 score breakdown based on the number of relations involved in the questions.

all models (except Seq2seq) decrease in performance with the number of relations composed in a question (Figure 5.1). This can be in part explained by the fact that more composition questions are typically longer, and also require the model to integrate more sources of information in the narrative in order to answer them. One surprising observation from our results is that the performance on questions that ask about a single relation and have only a single answer is lower than questions that ask about a single relation but that can have multiple answers (see detailed results in the Appendix). This is in part because questions that can have multiple answers typically have canonical entities as answers (e.g., person's name), and these entities generally repeat in the text, making it easier for the model to find the correct answer.

Table 3 reports the overall (macro-average) F1 scores for different baselines. We can see that BiDAF-M and DrQA-M perform surprisingly well in the *within-world* evaluation even though they do not use any entity span information. In particular, DrQA-M outperforms BiDAF-M which suggests that modeling question-context interactions using simple bilinear terms have advantages over using more complex bidirectional attention flows. The lower performance of MemN2N suggests that its effectiveness on the BABI dataset does not directly transfer to our dataset. Note that the original MemN2N architecture uses simple bag-of-words and position encoding for sentences. This may work well on dataset with a simple vocabulary, for example, MemN2N performs the best in the SOFTWARE world as the SOFTWARE world has a smaller vocabulary compared to other worlds. In general, we believe that better text representations for questions and narratives can lead to improved performance. Seq2Seq model also did not perform as well. This is due to the inherent difficulty of generation and encoding long sequences. We found that it performs better when training and testing on shorter stories (limited to 30 statements). Interestingly, the logistic regression baseline performs on a par with MemN2N, but there is still a large performance gap to BiDAF-M and DrQA-M, and the gap is greater for questions that compose multiple relations.

In the *across-world* setting, the performance of all methods dramatically decreases.[2] This suggests the limitations of these methods in generalizing to unseen relations and vocabulary. The span-based models BiDAF-M and DrQA-M have an advantage in this setting as they can learn to answer questions based on the alignment between the question and the narrative. However, the low performance still suggests their limitations in transferring question answering capabilities.

6 Conclusion

In this work, we have taken the first steps towards the task of multi-relational question answering expressed through personal narrative. Our hypothesis is that this task will become increasingly important as users begin to teach personal knowledge about their world to the personal assistants embedded in their devices. This task naturally synthesizes two main branches of question answering research: QA over KBs and QA over free text. One of our main contributions is a collection of diverse datasets that feature rich compositional questions over a dynamic knowledge graph expressed through simulated narrative. Another contribution of our work is a thorough set of experiments and analysis of different types of end-to-end architectures for QA at their ability to answer multi-relational questions of varying degrees of compositionality. Our long-term goal is that both the data and the simulation code we release will inspire and motivate the community to look towards the vision of letting end-users teach our personal assistants about the world around us.

[2]In order to allow generalization across different domains for the Seq2Seq model, we replace entities appearing in each story with an id that correlates to their appearance order. After the model outputs its prediction, the entity ids are converted back to the entity phrase.

The TEXTWORDSQA dataset and the code can be downloaded at https://igorlabutov.github.io/textworldsqa.github.io/

7 Acknowledgments

This paper was supported in part by Verizon InMind (Azaria and Hong, 2016). One of the GPUs used in this work was donated by Nvidia.

References

David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Müller, Maarten de Rijke, Stefan Schlobach, M Voorhees, and L Buckland. 2004. Using wikipedia at the trec qa track. In *TREC*. Citeseer.

Amos Azaria and Jason Hong. 2016. Recommender system with personality. In *RecSys*, pages 207–210.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Michele Banko, Eric Brill, Susan Dumais, and Jimmy Lin. 2002. Askmsr: Question answering using the worldwide web. In *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 7–9.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Peter Clark. 2015. Elementary school science and math tests as a driver for ai: take the aristo challenge! In *AAAI*, pages 4019–4021.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *ACL*.

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. In *AAAI*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *ACL*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *ACL*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018a. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Daniel Khashabi, Tushar Khot Ashish Sabharwal, and Dan Roth. 2018b. Question answering as global reasoning over semantic abstractions. In *AAAI*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2015. Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Nick Montfort. 2005. *Twisty Little Passages: an approach to interactive fiction*. Mit Press.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457*.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. 2015. Towards neural network-based reasoning. *arXiv preprint arXiv:1508.05508*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 239–249.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 486–492.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Dataset	**Questions**											
	Single Entity/Relation			Multiple Entities								
				Single Relation			Two Relations			Three Relations		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Logistic Regression												
MEETING	42.0	78.1	51.0	50.6	74.6	56.6	33.3	66.3	41.1	31.8	57.6	38.0
HOMEWORK	39.7	57.8	44.2	98.6	99.1	98.8	57.4	78.7	62.2	25.4	42.0	28.0
SOFTWARE	55.0	73.3	59.0	54.3	98.2	66.5	58.2	76.0	62.3	46.3	84.6	56.4
DEPARTMENT	42.6	65.9	48.0	59.0	82.5	65.1	38.8	52.7	41.2	42.5	64.6	46.9
SHOPPING	53.1	70.2	56.2	79.6	83.4	79.0	53.1	60.5	52.3	53.4	67.9	56.0
Average	46.5	69.1	51.7	68.4	87.6	73.2	48.2	66.8	51.8	39.9	63.3	45.1
Sequence-to-Sequence												
MEETING	27.9	18.3	22.1	48.1	12.1	19.3	42.1	15.0	22.1	33.7	19.7	24.8
HOMEWORK	16.3	9.0	11.6	71.9	9.3	16.4	75.3	35.9	48.6	32.9	15.6	21.1
SOFTWARE	42.5	21.5	28.5	44.8	8.5	14.2	50.0	6.3	11.2	45.5	7.4	12.7
DEPARTMENT	49.9	35.6	41.5	54.1	20.3	29.6	57.2	38.0	45.7	43.9	39.7	41.7
SHOPPING	25.8	16.0	19.8	71.3	28.2	40.5	33.3	19.3	24.4	46.9	31.4	37.6
Average	32.5	20.1	24.7	58.0	15.7	24.0	51.6	22.9	30.4	40.6	22.7	27.6
MemN2N												
MEETING	56.9	56.0	54.7	66.8	58.4	58.6	57.0	57.5	54.8	38.7	40.7	38.8
HOMEWORK	42.6	41.2	41.3	97.9	63.7	73.9	60.4	47.9	49.4	36.5	29.0	30.1
SOFTWARE	68.5	71.6	68.5	72.9	73.2	70.9	69.7	67.3	66.1	75.0	74.8	72.6
DEPARTMENT	56.3	74.3	61.3	78.5	87.0	80.2	59.4	76.6	63.2	57.8	74.2	61.6
SHOPPING	51.3	45.4	45.5	74.9	54.1	59.0	45.6	40.6	40.2	44.3	37.6	37.9
Average	55.1	57.7	54.3	78.2	67.3	68.5	58.4	58.0	54.8	50.4	51.3	48.2
BIDAF-M												
MEETING	87.6	92.4	88.2	78.6	86.1	79.2	68.9	89.6	74.6	73.9	94.4	80.0
HOMEWORK	79.9	97.4	84.5	86.8	81.0	82.4	76.4	90.0	78.9	47.0	78.5	55.5
SOFTWARE	48.0	89.4	57.4	68.5	93.6	75.8	62.4	86.1	67.5	62.7	90.9	71.3
DEPARTMENT	57.0	64.6	58.1	73.6	85.9	76.6	67.0	83.2	70.8	63.1	71.4	64.0
SHOPPING	60.5	87.1	66.9	76.7	90.9	79.8	57.1	89.0	65.8	53.2	88.5	62.0
Average	66.6	86.2	71.0	76.8	87.5	78.8	66.4	87.6	71.5	60.0	84.7	66.6
DrQA-M												
MEETING	77.1	94.2	81.0	80.6	95.8	85.1	68.6	95.7	76.8	64.1	97.9	74.3
HOMEWORK	88.8	97.9	91.4	85.2	80.2	81.4	85.0	94.7	87.9	51.6	85.8	60.2
SOFTWARE	72.7	96.0	78.9	78.6	93.3	82.7	79.4	89.4	80.9	66.3	93.2	74.5
DEPARTMENT	67.1	97.9	76.1	80.3	95.0	84.1	67.1	94.4	74.8	55.8	95.2	66.9
SHOPPING	71.5	93.9	77.7	86.4	94.8	88.7	62.8	91.1	71.4	62.4	90.7	69.7
Average	75.4	96.0	81.0	82.2	91.8	84.4	72.6	93.1	78.4	60.0	92.6	69.1

Table 4: Test performance at the task of question answering by question type using the *within-world* evaluation.

Dataset	Questions			
	Single Entity/Relation	Across Entities		
		Single Relation	Two Relations	Three Relations
Logistic Regression				
MEETING	8.8	10.9	7.2	5.6
HOMEWORK	7.5	20.2	8.5	6.7
SOFTWARE	8.2	12.0	12.9	10.6
DEPARTMENT	7.4	14.4	9.7	6.1
SHOPPING	8.2	9.0	5.9	6.6
Average	8.0	13.3	8.8	7.1
Sequence-to-Sequence				
MEETING	7.4	8.1	10.0	14.0
HOMEWORK	4.2	2.9	3.1	2.3
SOFTWARE	5.0	0.6	0.9	1.1
DEPARTMENT	5.5	4.0	5.6	5.6
SHOPPING	2.5	2.6	2.3	2.8
Average	4.9	3.6	4.4	5.2
MemN2N				
MEETING	9.0	34.2	33.0	27.4
HOMEWORK	3.3	12.4	1.0	2.5
SOFTWARE	13.4	0.8	3.2	2.9
DEPARTMENT	12.9	20.8	13.0	9.4
SHOPPING	0.1	0.07	0.05	0.03
Average	7.8	13.7	10.1	8.4
BIDAF-M				
MEETING	31.1	40.2	30.4	30.0
HOMEWORK	10.4	20.3	2.3	7.8
SOFTWARE	19.2	13.4	22.7	9.1
DEPARTMENT	23.3	30.5	19.0	13.5
SHOPPING	5.6	3.2	2.6	3.4
Average	17.9	21.5	15.4	12.8
DrQA-M				
MEETING	44.5	58.8	33.3	37.1
HOMEWORK	19.8	30.1	5.9	9.4
SOFTWARE	26.4	23.4	24.0	19.4
DEPARTMENT	31.0	38.8	24.4	15.7
SHOPPING	19.3	2.3	6.7	7.1
Average	28.2	30.7	18.9	17.7

Table 5: Test performance (F_1 score) at the task of question answering by question type using the *across-world* evaluation.

Simple and Effective Multi-Paragraph Reading Comprehension

Christopher Clark[*]
University of Washington
csquared@cs.washington.edu

Matt Gardner
Allen Institute for Artificial Intelligence
mattg@allenai.org

Abstract

We introduce a method of adapting neural paragraph-level question answering models to the case where entire documents are given as input. Most current question answering models cannot scale to document or multi-document input, and naively applying these models to each paragraph independently often results in them being distracted by irrelevant text. We show that it is possible to significantly improve performance by using a modified training scheme that teaches the model to ignore non-answer containing paragraphs. Our method involves sampling multiple paragraphs from each document, and using an objective function that requires the model to produce globally correct output. We additionally identify and improve upon a number of other design decisions that arise when working with document-level data. Experiments on TriviaQA and SQuAD shows our method advances the state of the art, including a 10 point gain on TriviaQA.

1 Introduction

Teaching machines to answer arbitrary user-generated questions is a long-term goal of natural language processing. For a wide range of questions, existing information retrieval methods are capable of locating documents that are likely to contain the answer. However, automatically extracting the answer from those texts remains an open challenge. The recent success of neural models at answering questions given a related paragraph (Wang et al., 2017c; Tan et al., 2017) suggests they have the potential to be a key part of a solution to this problem. Most neural models are unable to scale beyond short paragraphs, so typically this requires adapting a paragraph-level model to process document-level input.

There are two basic approaches to this task. Pipelined approaches select a single paragraph from the input documents, which is then passed to the paragraph model to extract an answer (Joshi et al., 2017; Wang et al., 2017a). Confidence based methods apply the model to multiple paragraphs and return the answer with the highest confidence (Chen et al., 2017a). Confidence methods have the advantage of being robust to errors in the (usually less sophisticated) paragraph selection step, however they require a model that can produce accurate confidence scores for each paragraph. As we shall show, naively trained models often struggle to meet this requirement.

In this paper we start by proposing an improved pipelined method which achieves state-of-the-art results. Then we introduce a method for training models to produce accurate per-paragraph confidence scores, and we show how combining this method with multiple paragraph selection further increases performance.

Our pipelined method focuses on addressing the challenges that come with training on document-level data. We use a linear classifier to select which paragraphs to train and test on. Since annotating entire documents is expensive, data of this sort is typically distantly supervised, meaning only the answer text, not the answer spans, are known. To handle the noise this creates, we use a summed objective function that marginalizes the model's output over all locations the answer text occurs. We apply this approach with a model design that integrates some recent ideas in reading comprehension models, including self-attention (Cheng et al., 2016) and bi-directional attention (Seo et al., 2016).

[*]Work completed while interning at the Allen Institute for Artificial Intelligence

Our confidence method extends this approach to better handle the multi-paragraph setting. Previous approaches trained the model on questions paired with paragraphs that are known *a priori* to contain the answer. This has several downsides: the model is not trained to produce low confidence scores for paragraphs that do not contain an answer, and the training objective does not require confidence scores to be comparable between paragraphs. We resolve these problems by sampling paragraphs from the context documents, including paragraphs that do not contain an answer, to train on. We then use a shared-normalization objective where paragraphs are processed independently, but the probability of an answer candidate is marginalized over all paragraphs sampled from the same document. This requires the model to produce globally correct output even though each paragraph is processed independently.

We evaluate our work on TriviaQA (Joshi et al., 2017) in the wiki, web, and unfiltered setting. Our model achieves a nearly 10 point lead over published prior work. We additionally perform an ablation study on our pipelined method, and we show the effectiveness of our multi-paragraph methods on a modified version of SQuAD (Rajpurkar et al., 2016) where only the correct document, not the correct paragraph, is known. Finally, we combine our model with a web search backend to build a demonstration end-to-end QA system[1], and show it performs well on questions from the TREC question answering task (Voorhees et al., 1999). We release our code[2] to facilitate future work.

2 Pipelined Method

In this section we propose a pipelined QA system, where a single paragraph is selected and passed to a paragraph-level question answering model.

2.1 Paragraph Selection

If there is a single source document, we select the paragraph with the smallest TF-IDF cosine distance with the question. Document frequencies are computed using the individual paragraphs within the document. If there are multiple input documents, we found it beneficial to use a linear classifier that uses the same TF-IDF score, whether the paragraph was the first in its document, how many tokens preceded it, and the number of question words it includes as features. The classifier is trained on the distantly supervised objective of selecting paragraphs that contain at least one answer span. On TriviaQA web, relative to truncating the document as done by prior work, this improves the chance of the selected text containing the correct answer from 83.1% to 85.1%.

2.2 Handling Noisy Labels

Question: Which British general was killed at Khartoum in 1885?
Answer: Gordon
Context: In February 1885 Gordon returned to the Sudan to evacuate Egyptian forces. Khartoum came under siege the next month and rebels broke into the city, killing Gordon and the other defenders. The British public reacted to his death by acclaiming 'Gordon of Khartoum', a saint. However, historians have suggested that Gordon...

Figure 1: Noisy supervision can cause many spans of text that contain the answer, but are not situated in a context that relates to the question (red), to distract the model from learning from more relevant spans (green).

In a distantly supervised setup we label all text spans that match the answer text as being correct. This can lead to training the model to select unwanted answer spans. Figure 1 contains an example. To handle this difficulty, we use a summed objective function similar to the one from Kadlec et al. (2016), that optimizes the negative log-likelihood of selecting any correct answer span. The models we consider here work by independently predicting the start and end token of the answer span, so we take this approach for both predictions. For example, the objective for predicting the answer start token becomes $-\log\left(\sum_{a\in A} p_a\right)$ where A is the set of tokens that start an answer and p_i is the answer-start probability predicted by the model for token i. This objective has the advantage of being agnostic to how the model distributes probability mass across the possible answer spans, allowing the model to focus on only the most relevant spans.

2.3 Model

We use a model with the following layers (shown in Figure 2):

Embedding: We embed words using pretrained word vectors. We concatenate these with character-derived word embeddings, which are

[1]https://documentqa.allenai.org
[2]https://github.com/allenai/document-qa

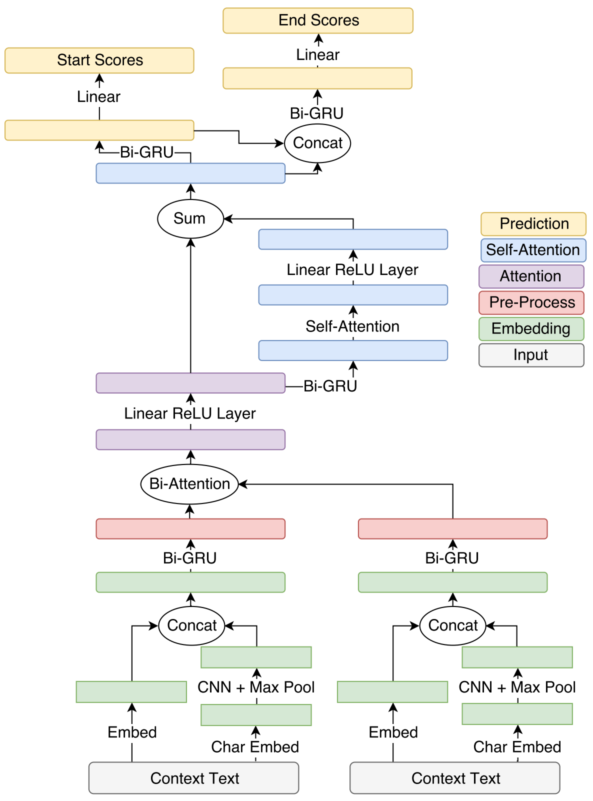


Figure 2: High level outline of our model.

produced by embedding characters using a learned embedding matrix and then applying a convolutional neural network and max-pooling.

Pre-Process: A shared bi-directional GRU (Cho et al., 2014) is used to process the question and passage embeddings.

Attention: The attention mechanism from the Bi-Directional Attention Flow (BiDAF) model (Seo et al., 2016) is used to build a query-aware context representation. Let $\mathbf{h_i}$ and $\mathbf{q_j}$ be the vector for context word i and question word j, and n_q and n_c be the lengths of the question and context respectively. We compute attention between context word i and question word j as:

$$a_{ij} = \mathbf{w_1} \cdot \mathbf{h_i} + \mathbf{w_2} \cdot \mathbf{q_j} + \mathbf{w_3} \cdot (\mathbf{h_i} \odot \mathbf{q_j})$$

where $\mathbf{w_1}$, $\mathbf{w_2}$, and $\mathbf{w_3}$ are learned vectors and $\odot$ is element-wise multiplication. We then compute an attended vector $\mathbf{c_i}$ for each context token as:

$$p_{ij} = \frac{e^{a_{ij}}}{\sum_{j=1}^{n_q} e^{a_{ij}}} \qquad \mathbf{c_i} = \sum_{j=1}^{n_q} \mathbf{q_j} p_{ij}$$

We also compute a query-to-context vector $\mathbf{q_c}$:

$$m_i = \max_{1 \leq j \leq n_q} a_{ij}$$

$$p_i = \frac{e^{m_i}}{\sum_{i=1}^{n_c} e^{m_i}} \qquad \mathbf{q_c} = \sum_{i=1}^{n_c} \mathbf{h_i} p_i$$

The final vector for each token is built by concatenating $\mathbf{h_i}$, $\mathbf{c_i}$, $\mathbf{h_i} \odot \mathbf{c_i}$, and $\mathbf{q_c} \odot \mathbf{c_i}$. In our model we subsequently pass the result through a linear layer with ReLU activations.

Self-Attention: Next we use a layer of residual self-attention. The input is passed through another bi-directional GRU. Then we apply the same attention mechanism, only now between the passage and itself. In this case we do not use query-to-context attention and we set $a_{ij} = -inf$ if $i = j$.

As before, we pass the concatenated output through a linear layer with ReLU activations. The result is then summed with the original input.

Prediction: In the last layer of our model a bi-directional GRU is applied, followed by a linear layer to compute answer start scores for each token. The hidden states are concatenated with the input and fed into a second bi-directional GRU and linear layer to predict answer end scores. The softmax function is applied to the start and end scores to produce answer start and end probabilities.

Dropout: We apply variational dropout (Gal and Ghahramani, 2016) to the input to all the GRUs and the input to the attention mechanisms at a rate of 0.2.

3 Confidence Method

We adapt this model to the multi-paragraph setting by using the un-normalized and un-exponentiated (i.e., before the softmax operator is applied) score given to each span as a measure of the model's confidence. For the boundary-based models we use here, a span's score is the sum of the start and end score given to its start and end token. At test time we run the model on each paragraph and select the answer span with the highest confidence. This is the approach taken by Chen et al. (2017a).

Our experiments in Section 5 show that these confidence scores can be very poor if the model is only trained on answer-containing paragraphs, as done by prior work. Table 1 contains some qualitative examples of the errors that occur.

We hypothesize that there are two key sources of error. First, for models trained with the softmax objective, the pre-softmax scores for all spans can be arbitrarily increased or decreased by a constant value without changing the resulting softmax probability distribution. As a result, nothing prevents models from producing scores that are arbitrarily all larger or all smaller for one paragraph

Question	Low Confidence Correct Extraction	High Confidence Incorrect Extraction
When is the Members Debate held?	**Immediately after Decision Time** a "Members Debate" is held, which lasts for 45 minutes...	...majority of the Scottish electorate voted for it in a referendum to be held on **1 March 1979** that represented at least...
How many tree species are in the rainforest?	...one 2001 study finding a quarter square kilometer (62 acres) of Ecuadorian rainforest supports more than **1,100** tree species	The affected region was approximately **1,160,000** square miles (3,000,000 km2) of rainforest, compared to 734,000 square miles
Who was Warsz?	....In actuality, Warsz was a 12th/13th century **nobleman** who owned a village located at the modern....	One of the most famous people born in Warsaw was **Maria Sklodowska - Curie**, who achieved international...
How much did the initial LM weight in kg?	The initial LM model weighed approximately 33,300 pounds (**15,000** kg), and...	The module was 11.42 feet (3.48 m) tall, and weighed approximately 12,250 pounds (**5,560** kg)

Table 1: Examples from SQuAD where a model was less confident in a correct extraction from one paragraph (left) than in an incorrect extraction from another (right). Even if the passage has no correct answer and does not contain any question words, the model assigns high confidence to phrases that match the category the question is asking about. Because the confidence scores are not well-calibrated, this confidence is often higher than the confidence assigned to correct answer spans in different paragraphs, even when those correct spans have better contextual evidence.

than another. Second, if the model only sees paragraphs that contain answers, it might become too confident in heuristics or patterns that are only effective when it is known *a priori* that an answer exists. For example, the model might become too reliant on selecting answers that match semantic type the question is asking about, causing it be easily distracted by other entities of that type when they appear in irrelevant text. This kind of error has also been observed when distractor sentences are added to the context (Jia and Liang, 2017)

We experiment with four approaches to training models to produce comparable confidence scores, shown in the following subsections. In all cases we will sample paragraphs that do not contain an answer as additional training points.

3.1 Shared-Normalization

In this approach a modified objective function is used where span start and end scores are normalized across all paragraphs sampled from the same context. This means that paragraphs from the same context use a shared normalization factor in the final softmax operations. We train on this objective by including multiple paragraphs from the same context in each mini-batch. The key idea is that this will force the model to produce scores that are comparable between paragraphs, even though it does not have access to information about what other paragraphs are being considered.

3.2 Merge

As an alternative to the previous method, we experiment with concatenating all paragraphs sampled from the same context together during training. A paragraph separator token with a learned embedding is added before each paragraph.

3.3 No-Answer Option

We also experiment with allowing the model to select a special "no-answer" option for each paragraph. First we re-write our objective as:

$$-\log\left(\frac{e^{s_a}}{\sum_{i=1}^{n} e^{s_i}}\right) - \log\left(\frac{e^{g_b}}{\sum_{j=1}^{n} e^{g_j}}\right) =$$

$$-\log\left(\frac{e^{s_a+g_b}}{\sum_{i=1}^{n}\sum_{j=1}^{n} e^{s_i+g_j}}\right)$$

where s_j and g_j are the scores for the start and end bounds produced by the model for token j, and a and b are the correct start and end tokens. We have the model compute another score, z, to represent the weight given to a "no-answer" possibility. Our revised objective function becomes:

$$-\log\left(\frac{(1-\delta)e^{z} + \delta e^{s_a+g_b}}{e^{z} + \sum_{i=1}^{n}\sum_{j=1}^{n} e^{s_i+g_j}}\right)$$

where δ is 1 if an answer exists and 0 otherwise. If there are multiple answer spans we use the same objective, except the numerator includes the summation over all answer start and end tokens.

We compute z by adding an extra layer at the end of our model. We build input vectors by taking the summed hidden states of the RNNs used to predict the start/end token scores weighed by the start/end probabilities, and using a learned attention vector on the output of the self-attention layer.

These vectors are fed into a two layer network with an 80 dimensional hidden layer and ReLU activations that produces z as its only output.

3.4 Sigmoid

As a final baseline, we consider training models with the sigmoid loss objective function. That is, we compute a start/end probability for each token by applying the sigmoid function to the start/end scores of each token. A cross entropy loss is used on each individual probability. The intuition is that, since the scores are being evaluated independently of one another, they are more likely to be comparable between different paragraphs.

4 Experimental Setup

4.1 Datasets

We evaluate our approach on four datasets: TriviaQA unfiltered (Joshi et al., 2017), a dataset of questions from trivia databases paired with documents found by completing a web search of the questions; TriviaQA wiki, the same dataset but only including Wikipedia articles; TriviaQA web, a dataset derived from TriviaQA unfiltered by treating each question-document pair where the document contains the question answer as an individual training point; and SQuAD (Rajpurkar et al., 2016), a collection of Wikipedia articles and crowdsourced questions.

4.2 Preprocessing

We note that for TriviaQA web we do not subsample as was done by Joshi et al. (2017), instead training on the all 530k training examples. We also observe that TriviaQA documents often contain many small paragraphs, so we restructure the documents by merging consecutive paragraphs together up to a target size. We use a maximum paragraph size of 400 unless stated otherwise. Paragraph separator tokens with learned embeddings are added between merged paragraphs to preserve formatting information. We are also careful to mark all spans of text that would be considered an exact match by the official evaluation script, which includes some minor text pre-processing, as answer spans, not just spans that are an exact string match with the answer text.

4.3 Sampling

Our confidence-based approaches are trained by sampling paragraphs from the context during training. For SQuAD and TriviaQA web we take the top four paragraphs as judged by our paragraph ranking function (see Section 2.1). We sample two different paragraphs from those four each epoch to train on. Since we observe that the higher-ranked paragraphs are more likely to contain the context needed to answer the question, we sample the highest ranked paragraph that contains an answer twice as often as the others. For the merge and shared-norm approaches, we additionally require that at least one of the paragraphs contains an answer span, and both of those paragraphs are included in the same mini-batch. For TriviaQA wiki we repeat the process but use the top 8 paragraphs, and for TriviaQA unfiltered we use the top 16, because much more context is given in these settings.

Model	EM	F1
baseline (Joshi et al., 2017)	41.08	47.40
BiDAF	50.21	56.86
BiDAF + TF-IDF	53.41	59.18
BiDAF + sum	56.22	61.48
BiDAF + TF-IDF + sum	57.20	62.44
our model + TF-IDF + sum	61.10	66.04

Table 2: Results on TriviaQA web using our pipelined method.

4.4 Implementation

We train the model with the Adadelta optimizer (Zeiler, 2012) with a batch size 60 for TriviaQA and 45 for SQuAD. At test time we select the most probable answer span of length less than or equal to 8 for TriviaQA and 17 for SQuAD. The GloVe 300 dimensional word vectors released by Pennington et al. (2014) are used for word embeddings. On SQuAD, we use a dimensionality of size 100 for the GRUs and of size 200 for the linear layers employed after each attention mechanism. We found for TriviaQA, likely because there is more data, using a larger dimensionality of 140 for each GRU and 280 for the linear layers is beneficial. During training, we maintain an exponential moving average of the weights with a decay rate of 0.999. We use the weight averages at test time. We do not update the word vectors during training.

5 Results

5.1 TriviaQA Web and TriviaQA Wiki

First, we do an ablation study on TriviaQA web to show the effects of our proposed methods for our pipeline model. We start with a baseline following the one used by Joshi et al. (2017). This

Model	Web		Web Verified		Wiki		Wiki Verified	
	EM	F1	EM	F1	EM	F1	EM	F1
Baseline (Joshi et al., 2017)	40.74	47.06	49.54	55.80	40.32	45.91	44.86	50.71
Smarnet (Chen et al., 2017b)	40.87	47.09	51.11	55.98	42.41	48.84	50.51	55.90
Mnemonic Reader (Hu et al., 2017)	46.65	52.89	56.96	61.48	46.94	52.85	54.45	59.46
(Weissenborn et al., 2017a)	50.56	56.73	63.20	67.97	48.64	55.13	53.42	59.92
Neural Cascade (Swayamdipta et al., 2017)	53.75	58.57	63.20	66.88	51.59	55.95	58.90	62.53
S-Norm (ours)	66.37	71.32	79.97	83.70	63.99	68.93	67.98	72.88

Table 3: Published TriviaQA results. Our approach advances the state of the art by about 10 points on these datasets[4]

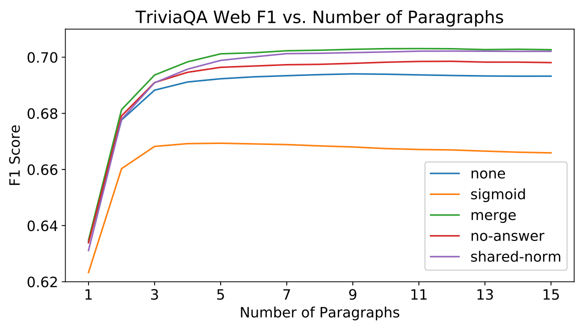


Figure 3: Results on TriviaQA web when applying our models to multiple paragraphs from each document. Most of our training methods improve the model's ability to utilize more text.

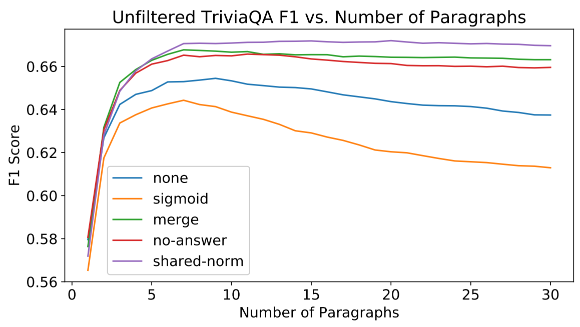


Figure 4: Results for our confidence methods on TriviaQA unfiltered. The shared-norm approach is the strongest, while the baseline model starts to lose performance as more paragraphs are used.

system uses BiDAF (Seo et al., 2016) as the paragraph model, and selects a random answer span from each paragraph each epoch to train on. The first 400 tokens of each document are used during training, and the first 800 during testing. When using the TF-IDF paragraph selection approach, we instead break the documents into paragraphs of size 400 when training and 800 when testing, and select the top-ranked paragraph to feed into the model. As shown in Table 2, our baseline outperforms the results reported by Joshi et al. (2017) significantly, likely because we are not subsampling the data. We find both TF-IDF ranking and the sum objective to be effective. Using our refined model increases the gain by another 4 points.

Next we show the results of our confidence-based approaches. For this comparison we split documents into paragraphs of at most 400 tokens, and rank them using TF-IDF cosine distance. Then we measure the performance of our proposed approaches as the model is used to independently process an increasing number of these paragraphs, and the highest confidence answer is selected as the final output. The results are shown in Figure 3.

On this dataset even the model trained without any of the proposed training methods ("none") improves as more paragraphs are used, showing it does a passable job at focusing on the correct paragraph. The no-answer option training approach lead to a significant improvement, and the shared-norm and merge approaches are even better.

We use the shared-norm approach for evaluation on the TriviaQA test sets. We found that increasing the paragraph size to 800 at test time, and to 600 during training, was slightly beneficial, allowing our model to reach 66.04 EM and 70.98 F1 on the dev set. As shown in Table 3, our model is firmly ahead of prior work on both the TriviaQA web and TriviaQA wiki test sets. Since our submission, a few additional entries have been added to the public leader for this dataset[5], although to the best of our knowledge these results have not yet been published.

5.2 TriviaQA Unfiltered

Next we apply our confidence methods to TriviaQA unfiltered. This dataset is of particular interest because the system is not told which document contains the answer, so it provides a plausible simulation of answering a question using a document

[4]Comparison made of 5/01/2018.
[5]https://competitions.codalab.org/competitions/17208

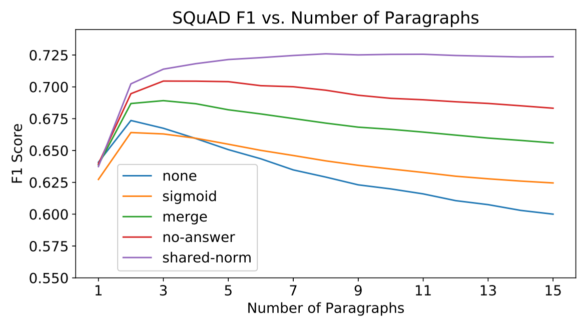


Figure 5: Results for our confidence methods on document-level SQuAD. The shared-norm model is the only model that does not lose performance when exposed to large numbers of paragraphs.

retrieval system. We show the same graph as before for this dataset in Figure 4. Our methods have an even larger impact on this dataset, probably because there are many more relevant and irrelevant paragraphs for each question, making paragraph selection more important.

Note the naively trained model starts to lose performance as more paragraphs are used, showing that errors are being caused by the model being overly confident in incorrect extractions. We achieve a score of 61.55 EM and 67.61 F1 on the dev set. This advances the only prior result reported for this dataset, 50.6 EM and 57.3 F1 from Wang et al. (2017b), by 10 points.

5.3 SQuAD

We additionally evaluate our model on SQuAD. SQuAD questions were not built to be answered independently of their context paragraph, which makes it unclear how effective of an evaluation tool they can be for document-level question answering. To assess this we manually label 500 random questions from the training set.

We categorize questions as:

1. Context-independent, meaning it can be understood independently of the paragraph.
2. Document-dependent, meaning it can be understood given the article's title. For example, "What individual is the school named after?" for the document "Harvard University".
3. Paragraph-dependent, meaning it can only be understood given its paragraph. For example, "What was the first step in the reforms?".

We find 67.4% of the questions to be context-independent, 22.6% to be document-dependent, and the remaining 10% to be paragraph-dependent. There are many document-dependent questions because questions are frequently about the subject of the document. Since a reasonably high fraction of the questions can be understood given the document they are from, and to isolate our analysis from the retrieval mechanism used, we choose to evaluate on the document-level. We build documents by concatenating all the paragraphs in SQuAD from the same article together into a single document.

Given the correct paragraph (i.e., in the standard SQuAD setting) our model reaches 72.14 EM and 81.05 F1 and can complete 26 epochs of training in less than five hours. Most of our variations to handle the multi-paragraph setting caused a minor (up to half a point) drop in performance, while the sigmoid version fell behind by a point and a half.

We graph the document-level performance in Figure 5. For SQuAD, we find it crucial to employ one of the suggested confidence training techniques. The base model starts to drop in performance once more than two paragraphs are used. However, the shared-norm approach is able to reach a peak performance of 72.37 F1 and 64.08 EM given 15 paragraphs. Given our estimate that 10% of the questions are ambiguous if the paragraph is unknown, our approach appears to have adapted to the document-level task very well.

Finally, we compare the shared-norm model with the document-level result reported by Chen et al. (2017a). We re-evaluate our model using the documents used by Chen et al. (2017a), which consist of the same Wikipedia articles SQuAD was built from, but downloaded at different dates. The advantage of this dataset is that it does not allow the model to know *a priori* which paragraphs were filtered out during the construction of SQuAD. The disadvantage is that some of the articles have been edited since the questions were written, so some questions may no longer be answerable. Our model achieves 59.14 EM and 67.34 F1 on this dataset, which significantly outperforms the 49.7 EM reported by Chen et al. (2017a).

5.4 Curated TREC

We perform one final experiment that tests our model as part of an end-to-end question answering system. For document retrieval, we re-implement the pipeline from Joshi et al. (2017). Given a question, we retrieve up to 10 web documents us-

[7]https://github.com/brmson/yodaqa/wiki/Benchmarks

Model	Accuracy
S-Norm (ours)	53.31
YodaQA with Bing (Baudiš, 2015),	37.18
YodaQA (Baudiš, 2015),	34.26
DrQA + DS (Chen et al., 2017a)	25.7

Table 4: Results on the Curated TREC corpus, YodaQA results extracted from its github page[7]

Category	proportion
Sentence reading errors	35.2
Paragraph reading errors	17.6
Document coreference errors	14.1
Part of answer extracted	7.1
Required background knowledge	5.8
Answer indirectly stated	20.2

Table 5: Error analysis on TriviaQA web.

ing a Bing web search of the question, and all Wikipedia articles about entities the entity linker TAGME (Ferragina and Scaiella, 2010) identifies in the question. We then use our linear paragraph ranker to select the 16 most relevant paragraphs from all these documents, which are passed to our model to locate the final answer span. We choose to use the shared-norm model trained on the TriviaQA unfiltered dataset since it is trained using multiple web documents as input. We use the same heuristics as Joshi et al. (2017) to filter out trivia or QA websites to ensure questions cannot be trivially answered using webpages that directly address the question. A demo of the system is publicly available[8].

We find accuracy on the TriviaQA unfiltered questions remains almost unchanged (within half a percent exact match score) when using our document retrieval method instead of the given documents, showing our pipeline does a good job of producing evidence documents that are similar to the ones in the training data.

We test the system on questions from the TREC QA tasks (Voorhees et al., 1999), in particular a curated set of questions from Baudiš (2015), the same dataset used in Chen et al. (2017a). We apply our system to the 694 test questions without retraining on the train questions.

We compare against DrQA (Chen et al., 2017a) and YodaQA (Baudiš, 2015). It is important to note that these systems use different document corpora (Wikipedia for DrQA, and Wikipedia, several knowledge bases, and optionally Bing web search for YodaQA) and different training data (SQuAD and the TREC training questions for DrQA, and TREC only for YodaQA), so we cannot make assertions about the relative performance of individual components. Nevertheless, it is instructive to show how the methods we experiment with in this work can advance an end-to-end QA system.

The results are listed in Table 4. Our method outperforms prior work, breaking the 50% accuracy mark. This is a strong proof-of-concept that neural paragraph reading combined with existing document retrieval methods can advance the state-of-the-art on general question answering. It also shows that, despite the noise, the data from TriviaQA is sufficient to train models that can be effective on out-of-domain QA tasks.

[8]https://documentqa.allenai.org/

5.5 Discussion

We found that models that have only been trained on answer-containing paragraphs can perform very poorly in the multi-paragraph setting. The results were particularly bad for SQuAD; we think this is partly because the paragraphs are shorter, so the model had less exposure to irrelevant text.

The shared-norm approach consistently outperformed the other methods, especially on SQuAD and TriviaQA unfiltered, where many paragraphs were needed to reach peak performance. Figures 3, 4, and 5 show this technique has a minimal effect on the performance when only one paragraph is used, suggesting the model's per-paragraph performance is preserved. Meanwhile, it can be seen the accuracy of the shared-norm model never drops as more paragraphs are added, showing it successfully resolves the problem of being distracted by irrelevant text.

The no-answer and merge approaches were moderately effective, we suspect because they at least expose the model to more irrelevant text. However, these methods do not address the fundamental issue of requiring confidence scores to be comparable between independent applications of the model to different paragraphs, which is why we think they lagged behind. The sigmoid objective function reduces the paragraph-level performance considerably, especially on the TriviaQA datasets. We suspect this is because it is vulnerable to label noise, as discussed in Section 2.2.

5.6 Error Analysis

We perform an error analysis by labeling 200 random TriviaQA web dev-set errors made by the shared-norm model. We found 40.5% of the errors were caused because the document did not contain sufficient evidence to answer the question, and 17% were caused by the correct answer not being contained in the answer key. The distribution of the remaining errors is shown in Table 5.

We found quite a few cases where a sentence contained the answer, but the model was unable to extract it due to complex syntactic structure or paraphrasing. Two kinds of multi-sentence reading errors were also common: cases that required connecting multiple statements made in a single paragraph, and long-range coreference cases where a sentence's subject was named in a previous paragraph. Finally, some questions required background knowledge, or required the model to extract answers that were only stated indirectly (e.g., examining a list to extract the nth element). Overall, these results suggest good avenues for improvement are to continue advancing the sentence and paragraph level reading comprehension abilities of the model, and adding a mechanism to handle document-level coreferences.

6 Related Work

Reading Comprehension Datasets. The state of the art in reading comprehension has been rapidly advanced by neural models, in no small part due to the introduction of many large datasets. The first large scale datasets for training neural reading comprehension models used a Cloze-style task, where systems must predict a held out word from a piece of text (Hermann et al., 2015; Hill et al., 2015). Additional datasets including SQuAD (Rajpurkar et al., 2016), WikiReading (Hewlett et al., 2016), MS Marco (Nguyen et al., 2016) and TriviaQA (Joshi et al., 2017) provided more realistic questions. Another dataset of trivia questions, Quasar-T (Dhingra et al., 2017), was introduced recently that uses ClueWeb09 (Callan et al., 2009) as its source for documents. In this work we choose to focus on SQuAD because it is well studied, and TriviaQA because it is more challenging and features documents and multi-document contexts (Quasar T is similar, but was released after we started work on this project).

Neural Reading Comprehension. Neural reading comprehension systems typically use some form of attention (Wang and Jiang, 2016), although alternative architectures exist (Chen et al., 2017a; Weissenborn et al., 2017b). Our model follows this approach, but includes some recent advances such as variational dropout (Gal and Ghahramani, 2016) and bi-directional attention (Seo et al., 2016). Self-attention has been used in several prior works (Cheng et al., 2016; Wang et al., 2017c; Pan et al., 2017). Our approach to allowing a reading comprehension model to produce a per-paragraph no-answer score is related to the approach used in the BiDAF-T (Min et al., 2017) model to produce per-sentence classification scores, although we use an attention-based method instead of max-pooling.

Open QA. Open question answering has been the subject of much research, especially spurred by the TREC question answering track (Voorhees et al., 1999). Knowledge bases can be used, such as in (Berant et al., 2013), although the resulting systems are limited by the quality of the knowledge base. Systems that try to answer questions using natural language resources such as YodaQA (Baudiš, 2015) typically use pipelined methods to retrieve related text, build answer candidates, and pick a final output.

Neural Open QA. Open question answering with neural models was considered by Chen et al. (2017a), where researchers trained a model on SQuAD and combined it with a retrieval engine for Wikipedia articles. Our work differs because we focus on explicitly addressing the problem of applying the model to multiple paragraphs. A pipelined approach to QA was recently proposed by Wang et al. (2017a), where a ranker model is used to select a paragraph for the reading comprehension model to process. More recent work has considered evidence aggregation techniques (Wang et al., 2017b; Swayamdipta et al., 2017). Our work shows paragraph-level models that produce well-calibrated confidence scores can effectively exploit large amounts of text without aggregation, although integrating aggregation techniques could further improve our results.

7 Conclusion

We have shown that, when using a paragraph-level QA model across multiple paragraphs, our training method of sampling non-answer-containing paragraphs while using a shared-norm objective function can be very beneficial. Combining this with our suggestions for paragraph selection, using the summed training objective, and our model design allows us to advance the state of the art on TriviaQA. As shown by our demo, this work can be directly applied to building deep-learning-powered open question answering systems.

References

Petr Baudiš. 2015. YodaQA: A Modular Question Answering System Pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 Data Set.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to Answer Open-Domain Questions. *arXiv preprint arXiv:1704.00051*.

Zheqian Chen, Rongqin Yang, Bin Cao, Zhou Zhao, Deng Cai, and Xiaofei He. 2017b. Smarnet: Teaching Machines to Read and Comprehend Like Human. *arXiv preprint arXiv:1710.02772*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. *arXiv preprint arXiv:1601.06733*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for Question Answering by Search and Reading. *arXiv preprint arXiv:1707.03904*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in neural information processing systems*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A Novel Large-scale Language Understanding Task over Wikipedia. *arXiv preprint arXiv:1608.03542*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *arXiv preprint arXiv:1511.02301*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic Reader: Machine Comprehension with Iterative Aligning and Multi-hop Answer Pointing.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. *arXiv preprint arXiv:1707.07328*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv preprint arXiv:1705.03551*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text Understanding with the Attention Sum Reader Network. *arXiv preprint arXiv:1603.01547*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. *arXiv preprint arXiv:1702.02171*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. MEMEN: Multi-layer Embedding with Memory Networks for Machine Comprehension. *arXiv preprint arXiv:1707.09098*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv preprint arXiv:1606.05250*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional Attention Flow for Machine Comprehension. *CoRR*, abs/1611.01603.

Swabha Swayamdipta, Ankur P. Parikh, and Tom Kwiatkowski. 2017. Multi-Mention Learning for Reading Comprehension with Neural Cascades.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-Net: From Answer Extraction to Answer Generation for Machine Reading Comprehension. *arXiv preprint arXiv:1706.04815*.

Ellen M Voorhees et al. 1999. The TREC-8 Question Answering Track Report. In *Trec*.

Shuohang Wang and Jing Jiang. 2016. Machine Comprehension Using Match-LSTM and Answer Pointer. *arXiv preprint arXiv:1608.07905*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R: Reinforced Reader-Ranker for Open-Domain Question Answering. *arXiv preprint arXiv:1709.00023*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017b. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017c. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2017a. Dynamic Integration of Background Knowledge in Neural NLU Systems. *arXiv preprint arXiv:1706.02596*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017b. FastQA: A Simple and Efficient Neural Architecture for Question Answering. *arXiv preprint arXiv:1703.04816*.

Matthew D Zeiler. 2012. ADADELTA: an Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

Semantically Equivalent Adversarial Rules for Debugging NLP Models

Marco Tulio Ribeiro
University of Washington
marcotcr@cs.uw.edu

Sameer Singh
University of California, Irvine
sameer@uci.edu

Carlos Guestrin
University of Washington
guestrin@cs.uw.edu

Abstract

Complex machine learning models for NLP are often brittle, making different predictions for input instances that are extremely similar semantically. To automatically detect this behavior for individual instances, we present *semantically equivalent adversaries* (SEAs) – semantic-preserving perturbations that induce changes in the model's predictions. We generalize these adversaries into *semantically equivalent adversarial rules* (SEARs) – simple, universal replacement rules that induce adversaries on many instances. We demonstrate the usefulness and flexibility of SEAs and SEARs by detecting bugs in black-box state-of-the-art models for three domains: machine comprehension, visual question-answering, and sentiment analysis. Via user studies, we demonstrate that we generate high-quality local adversaries for more instances than humans, and that SEARs induce four times as many mistakes as the bugs discovered by human experts. SEARs are also actionable: retraining models using data augmentation significantly reduces bugs, while maintaining accuracy.

1 Introduction

With increasing complexity of models for tasks like classification (Joulin et al., 2016), machine comprehension (Rajpurkar et al., 2016; Seo et al., 2017), and visual question answering (Zhu et al., 2016), models are becoming increasingly challenging to debug, and to determine whether they are ready for deployment. In particular, these complex models are prone to brittleness: different ways of phrasing the same sentence can often cause the model to

In the United States especially, several high-profile cases such as Debra LaFave, Pamela Rogers, and Mary Kay Letourneau have caused increased scrutiny on teacher misconduct.

(a) Input Paragraph

Q: What has been the result of this publicity?
A: increased scrutiny on teacher misconduct

(b) Original Question and Answer

Q: What haL been the result of this publicity?
A: teacher misconduct

(c) Adversarial Q & A (Ebrahimi et al., 2018)

Q: What's been the result of this publicity?
A: teacher misconduct

(d) **Semantically Equivalent Adversary**

Figure 1: Adversarial examples for question answering, where the model predicts the correct answer for the question and input paragraph (1a and 1b). It is possible to fool the model by adversarially changing a single character (1c), but at the cost of making the question nonsensical. A **Semantically Equivalent Adversary** (1d) results in an incorrect answer while preserving semantics.

output different predictions. While held-out accuracy is often useful, it is not sufficient: practitioners consistently overestimate their model's generalization (Patel et al., 2008) since test data is usually gathered in the same manner as training and validation. When deployed, these seemingly accurate models encounter sentences that are written very differently than the ones in the training data, thus making them prone to mistakes, and fragile with respect to distracting additions (Jia and Liang, 2017). These problems are exacerbated by the variability in language, and by cost and noise in annotations, making such bugs challenging to detect and fix.

A particularly challenging issue is *oversensitivity* (Jia and Liang, 2017): a class of bugs where models output different predictions for very similar inputs. These bugs are prevalent in image classifi-

Transformation Rules	#Flips
(`WP` *is*→`WP`*'s*)	70 (1%)
(*?*→*??*)	202(3%)

(a) Example Rules

Original: What is the oncorhynchus also called? **A:** chum salmon
Changed: What's the oncorhynchus also called? **A:** keta

(b) Example for (`WP` *is*→`WP`*'s*)

Original: How long is the Rhine? **A:** 1,230 km
Changed: How long is the Rhine?? **A:** more than 1,050,000

(c) Example for (*?*→*??*)

Figure 2: **Semantically Equivalent Adversarial Rules:** For the task of question answering, the proposed approach identifies transformation rules for questions in (a) that result in paraphrases of the queries, but lead to incorrect answers (#Flips is the number of times this happens in the validation data). We show examples of rephrased questions that result in incorrect answers for the two rules in (b) and (c).

cation (Szegedy et al., 2014), a domain where one can measure the magnitude of perturbations, and many small-magnitude changes are imperceptible to the human eye. For text, however, a single word addition can change semantics (e.g. adding "not"), or have no semantic impact for the task at hand.

Inspired by adversarial examples for images, we introduce *semantically equivalent adversaries* (**SEA**s) – text inputs that are perturbed in semantics-preserving ways, but induce changes in a black box model's predictions (example in Figure 1). Producing such adversarial examples systematically can significantly aid in debugging ML models, as it allows users to detect problems that happen in the real world, instead of oversensitivity only to malicious attacks such as intentionally scrambling, misspelling, or removing words (Bansal et al., 2014; Ebrahimi et al., 2018; Li et al., 2016).

While SEAs describe local brittleness (i.e. are specific to particular predictions), we are also interested in bugs that affect the model more globally. We represent these via simple replacement rules that induce SEAs on multiple predictions, such as in Figure 2, where a simple contraction of "is"after `Wh` pronouns (what, who, whom) (2b) makes 70 (1%) of the previously correct predictions of the model "flip" (i.e. become incorrect). Perhaps more surprisingly, adding a simple "?" induces mistakes in 3% of examples. We call such rules *semantically equivalent adversarial rules* (**SEAR**s).

In this paper, we present SEAs and SEARs, designed to unveil local and global oversensitivity bugs in NLP models. We first present an approach to generate semantically equivalent adversaries, based on paraphrase generation techniques (Lapata et al., 2017), that is model-agnostic (i.e. works for any black box model). Next, we generalize SEAs into semantically equivalent rules, and outline the properties for optimal rule sets: semantic equivalence, high adversary count, and non-redundancy. We frame the problem of finding such a set as a submodular optimization problem, leading to an accurate yet efficient algorithm.

Including the human into the loop, we demonstrate via user studies that SEARs help users uncover important bugs on a variety of state-of-the-art models for different tasks (sentiment classification, visual question answering). Our experiments indicate that SEAs and SEARs make humans significantly better at detecting impactful bugs – SEARs uncover bugs that cause 3 to 4 times more mistakes than human-generated rules, in much less time. Finally, we show that SEARs are actionable, enabling the human to close the loop by fixing the discovered bugs using a data augmentation procedure.

2 Semantically Equivalent Adversaries

Consider a black box model f that takes a sentence x and makes a prediction $f(x)$, which we want to debug. We identify adversaries by generating paraphrases of x, and getting predictions from f until the original prediction is changed.

Given an indicator function $\text{SemEq}(x, x')$ that is 1 if x is semantically equivalent to x' and 0 otherwise, we define a *semantically equivalent adversary* (SEA) as a semantically equivalent instance that changes the model prediction in Eq (1). Such adversaries are important in evaluating the robustness of f, as each is an undesirable bug.

$$\text{SEA}(x, x') = \mathbb{1}\left[\text{SemEq}(x, x') \wedge f(x) \neq f(x')\right] \quad (1)$$

While there are various ways of scoring semantic similarity between pairs of texts based on embeddings (Le and Mikolov, 2014; Wieting and Gimpel, 2017), they do not explicitly penalize unnatural sentences, and generating sentences requires surrounding context (Le and Mikolov, 2014) or training a separate model. We turn instead to paraphrasing based on neural machine translation (Lapata et al., 2017), where $P(x'|x)$ (the probability of a paraphrase x' given original sentence x) is proportional to translating x into multiple pivot languages and then taking the score of back-translating the translations into the original language. This approach scores semantics and "plausibility" simultaneously (as translation models have "built in" language models) and allows for easy paraphrase generation, by linearly combining the paths of each back-decoder when back-translating.

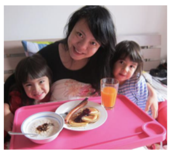
Unfortunately, given source sentences x and z, $P(x'|x)$ is not comparable to $P(z'|z)$, as each has a different normalization constant, and heavily depends on the shape of the distribution around x or z. If there are multiple perfect paraphrases near x, they will all share probability mass, while if there is a paraphrase much better than the rest near z, it will have a higher score than the ones near x, even if the paraphrase quality is the same. We thus define the semantic score $S(x, x')$ as a ratio between the probability of a paraphrase and the probability of the sentence itself:

$$S(x, x') = \min\left(1, \frac{P(x'|x)}{P(x|x)}\right) \tag{2}$$

We define $\text{SemEq}(x, x') = \mathbb{1}[S(x, x') \geq \tau]$, i.e. x' is semantically equivalent to x if the similarity score between x and x' is greater than some threshold τ (which we crowdsource in Section 5). In order to generate adversaries, we generate a set of paraphrases Π_x around x via beam search and get predictions on Π_x using the black box model until an adversary is found, or until $S(x, x') < \tau$. We may be interested in the best adversary for a particular instance, i.e. $\text{argmax}_{x' \in \Pi_x} S(x, x')\text{SEA}_x(x')$, or we may consider multiple SEAs for generalization purposes. We illustrate this process in Figure 3, where we generate SEAs for a VQA model by generating paraphrases around the question, and checking when the model prediction changes. The first two adversaries with highest $S(x, x')$ are semantically equivalent, the third maintains the semantics enough for it to be a useful adversary, and the fourth is ungrammatical and thus not useful.

3 Semantically Equivalent Adversarial Rules (SEARs)

While finding the best adversary for a particular instance is useful, humans may not have time or patience to examine too many SEAs, and may not be able to generalize well from them in order to understand and fix the most impactful bugs. In this section, we address the problem of generalizing local adversaries into Semantically Equivalent Adversarial Rules (SEARs), search and replace rules that produce semantic adversaries with little or no change in semantics, when applied to a corpus of sentences. Assuming that humans have limited time, and are thus willing to look at B rules, we propose a method for selecting such a set of rules given a reference dataset X.



What color is the tray?	**Pink**
What colour is the tray?	Green
Which color is the tray?	Green
What color is it?	Green
How color is tray?	Green

Figure 3: **Visual QA Adversaries:** Paraphrasing questions to find adversaries for the original question (top, in bold) asked of a given image. Adversaries are sorted by decreasing semantic similarity.

A rule takes the form $r = (a \rightarrow c)$, where the first instance of the antecedent a is replaced by the consequent c for every instance that includes a, as we previously illustrated in Figure 2a. The output after applying rule r on a sentence x is represented as the function call $r(x)$, e.g. if $r =$(*movie*$\rightarrow$*film*), r("Great movie!") = "Great film!".

Proposing a set of rules: In order to generalize a SEA x' into a candidate rule, we must represent the changes that took place from $x \rightarrow x'$. We will use x = "What color is it?" and x' = "Which color is it?" from Figure 4 as a running example.

One approach is exact matching: selecting the minimal contiguous sequence that turns x into x', (*What*$\rightarrow$*Which*) in the example. Such changes may not always be semantics preserving, so we also propose further rules by including the immediate context (previous and/or next word with respect to the sequence), e.g. (*What color*$\rightarrow$*Which color*). Adding such context, however, may make rules very specific, thus restricting their value. To allow for generalization, we also represent the antecedent of proposed rules by a product of their raw text with coarse and fine-grained Part-of-Speech tags, and allow these tags to happen in the consequent if they match the antecedent. In the running example, we would propose rules like (*What color*$\rightarrow$*Which color*), (*What* NOUN$\rightarrow$*Which* NOUN), (WP *color*$\rightarrow$*Which color*), etc.

We generate SEAs and propose rules for every $x \in X$, which gives us a set of candidate rules (second box in Figure 4, *for* loop in Algorithm 1).

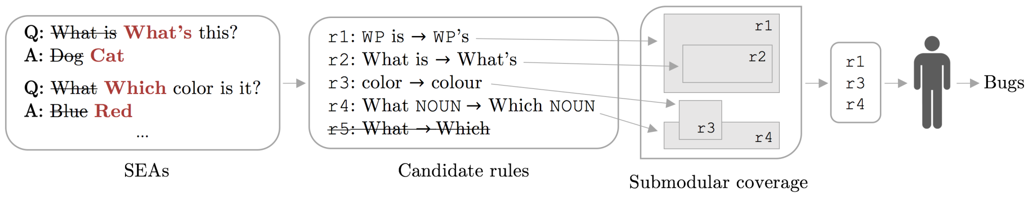


Figure 4: **SEAR process.** (1) SEAs are generalized into candidate rules, (2) rules that are not semantically equivalent are filtered out, e.g. `r5`: (*What→Which*), (3) rules are selected according to Eq (3), in order to maximize coverage and avoid redundancy (e.g. rejecting `r2`, valuing `r1` more highly than `r4`), and (4) a user vets selected rules and keeps the ones that they think are bugs.

Selecting a set of rules: Given a set of candidate rules, we want to select a set R such that $|R| \leq B$, and the following properties are met:

1. Semantic Equivalence: Application of the rules in the set should produce semantically equivalent instances. This is equivalent to considering rules that have a high probability of inducing semantically equivalent instances when applied, i.e. $E[\text{SemEq}(x, r(x))] \geq 1-\delta$. This is the *Filter* step in Algorithm 1. For example, consider the rule (*What→Which*) in Fig 4 which produces some semantically equivalent instances, but also produces many instances that are unnatural (e.g. "What is he doing?" → "Which is he doing?"), and is thus filtered out by this criterion.

2. High adversary count: The rules in the set should induce as many SEAs as possible in validation data. Furthermore, each of the induced SEAs should have as high of a semantic similarity score as possible, i.e. for each rule $r \in R$ we want to maximize $\sum_{x\in X} S(x, r(x))\text{SEA}(x, r(x))$. In Figure 4, `r1` induces more and more similar mistakes when compared to $r4$, and is thus superior to `r4`.

3. Non-redundancy: Different rules in the set may induce the same SEAs, or may induce different SEAs for the same instances. Ideally, rules in the set should cover as many instances in the validation as possible, rather than focus on a small set of fragile predictions. Furthermore, rules should not be repetitive to the user. In Figure 4 (mid), `r1` covers a superset of `r2`'s adversaries, making `r2` completely redundant and thus not included in R.

Properties 2 and 3 combined suggest a weighted coverage problem, where a rule r covers an instance x if $\text{SEA}(x, r(x))$, the weight of the connection being given by $S(x, r(x))$. We thus want to

Algorithm 1 Generating SEARs for a model

```
Require: Classifier $f$, Correct instances $X$
Require: Hyperparameters, $\delta$, $\tau$, Budget $B$
  $\mathcal{R} \leftarrow \{\}$ {Set of rules}
  for all $x \in X$ do
    $X' = \text{GenParaphrases}(X, \tau)$
    $\mathcal{A} \leftarrow \{x' \in X' \mid f(x) \neq f(x')\}$ {SEAs; §2}
    $\mathcal{R} \leftarrow \mathcal{R} \cup \text{Rules}(\mathcal{A})$
  end for
  $\mathcal{R} \leftarrow \text{Filter}(\mathcal{R}, \delta, \tau)$ {Remove low scoring SEARs}
  $\mathcal{R} \leftarrow \text{SubMod}(\mathcal{R}, B)$ {high count / score, diverse }
  return $\mathcal{R}$
```

find the set of semantically equivalent rules that:

$$\max_{R, |R|<B} \sum_{x\in X} \max_{r\in R} S(x, r(x))\text{SEA}(x, r(x)) \quad (3)$$

While Eq (3) is NP-hard, the objective is monotone submodular (Krause and Golovin, 2014), and thus a greedy algorithm that iteratively adds the rule with the highest marginal gain offers a constant-factor approximation guarantee of $1 - 1/e$ to the optimum. This is the *SubMod* procedure in Algorithm 1, represented pictorially in Figure 4, where the output is a set of rules given to a human, who judges if they are really bugs or not.

4 Illustrative Examples

Before evaluating the utility of SEAs and SEARs with user studies, we show examples in state-of-the-art models for different tasks. Note that we treat these models as black boxes, not using internals or gradients in any way when discovering these bugs.

Machine Comprehension: We take the AllenNLP (Gardner et al., 2017) implementation of BiDaF (Seo et al., 2017) for Machine Comprehension, and display some high coverage SEARs for it in Table 1 (also, Figures 1 and 2a). For each rule,

SEAR	Questions / SEAs	f(x)	Flips
What VBZ → What's	~~What is~~ What's the NASUWT?	~~Trade unions~~ Teachers in Wales	2%
	~~What is~~ What's a Hauptlied?	~~main hymn~~ Veni redemptor gentium	
What NOUN → Which NOUN	~~What resource~~ Which resource was mined in the Newcastle area?	~~coal~~ wool	1%
	~~What health~~ Which health problem did Tesla have in 1879?	~~nervous breakdown~~ relations	
What VERB → So what VERB	~~What was~~ So what was Ghandi's work called?	~~Satyagraha~~ Civil Disobedience	2%
	~~What is~~ So what is a new trend in teaching?	~~Co-teaching~~ educational institutions	
What VBD → And what VBD	~~What did~~ And what did Tesla develop in 1887?	~~an induction motor~~ laboratory	2%
	~~What was~~ And what was Kenneth Swezey's job?	~~journalist~~ sleep	

Table 1: SEARs for Machine Comprehension

SEAR	Questions / SEAs	f(x)	Flips
WP VBZ→ WP's	~~What has~~ What's been cut?	~~Cake~~ Pizza	3.3%
	~~Who is~~ Who's holding the baby	~~Woman~~ Man	
What NOUN→ Which NOUN	~~What~~ Which kind of floor is it?	~~Wood~~ Marble	3.9%
	~~What~~ Which color is the jet?	~~Gray~~ White	
color →colour	What ~~color~~ colour is the tray?	~~Pink~~ Green	2.2%
	What ~~color~~ colour is the jet?	~~Gray~~ Blue	
ADV is→ ADV's	~~Where is~~ Where's the jet?	~~Sky~~ Airport	2.1%
	~~How is~~ How's the desk?	~~Messy~~ Empty	

Table 2: SEARs for Visual QA

we display two example questions with the corresponding SEA, the prediction (with corresponding change) and the percentage of "flips" - instances previously predicted correctly on the validation data, but predicted incorrectly after the application of the rule. The rule (*What* VBZ→*What's*) generalizes the SEA on Figure 1, and shows that the model is fragile with respect to contractions (flips 2% of all correctly predicted instances on the validation data). The second rule uncovers a bug with respect to simple question rephrasing, while the third and fourth rules show that the model is not robust to a more conversational style of asking questions.

Visual QA: We show SEARs for a state-of-the-art visual question-answering model (Zhu et al., 2016) in Table 2. Even though the contexts are different (paragraphs for machine comprehension, images for VQA), it is interesting that both models display similar bugs. The fact that VQA is fragile to "Which" questions is because questions of this form are not in the training set, while (*color*→*colour*) probably stems from an American bias in data collection. Changes induced by these four rules flip more than 10% of the predictions in the validation data, which is of critical concern if the model is being evaluated for production.

SEAR	Reviews / SEAs	f(x)	Flips
movie → film	Yeah, the ~~movie~~ film pretty much sucked .	~~Neg~~ Pos	2%
	This is not ~~movie~~ film making .	~~Neg~~ Pos	
film → movie	Excellent ~~film~~ movie .	~~Pos~~ Neg	1%
	I'll give this ~~film~~ movie 10 out of 10 !	~~Pos~~ Neg	
is →was	Ray Charles ~~is~~ was legendary .	~~Pos~~ Neg	4%
	It ~~is~~ was a really good show to watch .	~~Pos~~ Neg	
this →that	Now ~~this~~ that is a movie I really dislike .	~~Neg~~ Pos	1%
	The camera really likes her in ~~this~~ that movie.	~~Pos~~ Neg	
DET NOUN is →it is	~~The movie is~~ It is terrible	~~Neg~~ Pos	1%
	~~The dialog is~~ It is atrocious	~~Neg~~ Pos	

Table 3: SEARs for Sentiment Analysis

Sentiment Analysis: Finally, in Table 3 we display SEARs for a fastText (Joulin et al., 2016) model for sentiment analysis trained on movie reviews. Surprisingly, many of its predictions change for perturbations that have no sentiment connotations, even in the presence of polarity-laden words.

5 User Studies

We compare automatically discovered SEAs and SEARs to user-generated adversaries and rules, and propose a way to fix the bugs induced by SEARs.

Our evaluation benchmark includes two tasks: visual question answering (VQA) and sentiment analysis on movie review sentences. We choose these tasks because a human can quickly look at a prediction and judge if it is correct or incorrect, can easily perturb instances, and judge if two instances in a pair are semantically equivalent or not. Since our focus is debugging, throughout the experiment we only considered SEAs and SEARs on examples that are originally predicted correctly (i.e. every adversary is also by construction a mistake). The user interfaces for all experiments in this section are included in the supplementary material.

5.1 Implementation Details

The paraphrasing model (Lapata et al., 2017) requires translation models to and from different languages. We train neural machine translation models using the default parameters of OpenNMT-py (Klein et al., 2017) for English↔Portuguese and English↔French models, on 2 million and 1 million parallel sentences (respectively) from EuroParl, news, and other sources (Tiedemann, 2012). We use the spacy library (`http://spacy.io`) for POS tagging. For SEAR generation, we set $\delta = 0.1$ (i.e. at least 90% equivalence). We generate a set of candidate adversaries as described in Section 2, and ask mechanical turkers to judge them

	Human vs SEA	Human vs HSEA
Neither	145 (48%)	127 (42%)
Only Human	47 (16%)	38 (13%)
Only SEA	**54 (18%)**	**72 (24%)**
Both	**54 (18%)**	**63 (21%)**

(a) Visual Question-Answering

	Human vs SEA	Human vs HSEA
Neither	177 (59%)	161 (54%)
Only Human	45 (15%)	40 (13%)
Only SEA	**47 (16%)**	**63 (21%)**
Both	**31 (10%)**	**36 (12%)**

(b) Sentiment Analysis

Table 4: **Finding Semantically Equivalent Adversaries:** we compare how often humans produce semantics-preserving adversaries, when compared to our automatically generated adversaries (SEA, left) and our adversaries filtered by humans (HSEA, right). There are four possible outcomes: neither produces a semantic equivalent adversary (i.e. they either do not produce an adversary or the adversary produced is not semantically equivalent), both do, or only one is able to do so.

for semantic equivalence. Using these evaluations, we identify $\tau = 0.0008$ as the value that minimizes the entropy in the induced splits, and use it for the remaining experiments. Source code and pretrained language models are available at `https://github.com/marcotcr/sears`.

For VQA, we use the multiple choice *telling* system and dataset of Zhu et al. (2016), using their implementation, with default parameters. The training data consists of questions that begin with "What", "Where", "When", "Who", "Why", and "How". The task is multiple choice, with four possible answers per instance. For sentiment analysis, we train a fastText (Joulin et al., 2016) model with unigrams and bigrams (embedding size of 50) on RottenTomato movie reviews (Pang and Lee, 2005), and evaluate it on IMDB sentence-sized reviews (Kotzias et al., 2015), simulating the common case where a model trained on a public dataset is applied to new data from a similar domain.

5.2 Can humans find good adversaries?

In this experiment, we compare our method for generating SEAs with user's ability to discover semantic-preserving adversaries. We take a random sample of 100 correctly-predicted instances for each task. In the first condition (**human**), we display each instance to 3 Amazon Mechanical Turk workers, and give them 10 attempts at creating semantically equivalent adversaries (with immediate feedback as to whether or not their attempts changed the prediction). Next, we ask them to choose the adversary that is semantically closest to the original instance, out of the candidates they generated. In the second condition (**SEA**), we generate adversaries for each of the instances, and pick the best adversary according to the semantic scorer. The third condition (**HSEA**) is a collaboration between our method and humans: we take the top 5 adversaries ranked by $S(x, x')$, and ask workers to pick the one closest to the original instance, rather than asking them to generate the adversaries.

To evaluate whether the proposed adversaries are semantically equivalent, we ask a separate set of workers to evaluate the similarity between each adversary and the original instance (with the image as context for VQA), on a scale of 1 (completely unrelated) to 5 (exactly the same meaning). Each adversary is evaluated by at least 10 workers, and considered equivalent if the median score ≥ 4. We thus obtain 300 comparisons between *human* and *SEA*, and 300 between *human* and *HSEA*.

The results in Table 4a and 4b are consistent across tasks: both models are susceptible to SEAs for a large fraction of predictions, and our fully automated method is able to produce SEAs as often as humans (left columns). On the other hand, asking humans to choose from generated SEAs (HSEA) yields much better results than asking humans to generate them (right columns), or using the highest scored SEA. The semantic scorer does make mistakes, so the top adversary is not always semantically equivalent, but a good quality SEA is often in the top 5, and is easily identified by users.

On both datasets, the automated method or humans were able to generate adversaries at the exclusion of the other roughly one third of the time, which indicates that they do not generate the same adversaries. Humans generate paraphrases differently than our method: the average character edit distance of our SEAs is 6.2 for VQA and 9.0 for Sentiment, while for humans it is 18.1 and 43.3, respectively. This is illustrated by examples in Table 5 - in Table 5a we see examples where very compact changes generate adversaries (humans were not able to find these changes though). The examples in Table 5b indicate that humans can generate adversaries that: (1) make use of the visual context in VQA, which our method does not, and (2) sig-

Dataset	Original	SEA
VQA	Where are the men? What kind of meat is on the boy's plate?	Where are the males? What sort of meat is on the boy's plate?
Sentiment	They are so easy to love, but even more easy to identify with. Today the graphics are crap.	They're so easy to love, but even more easy to identify with. Today, graphics are bullshit.

(a) Automatically generated adversaries, examples where humans failed to generate SEAs (**Only SEA**)

Dataset	Original	Human-generated SEA
VQA	How many suitcases? Where is the blue van?	How many suitcases are sitting on the shelf? What is the blue van's location?
Sentiment	(very serious spoilers) this movie was a huge disappointment. Also great directing and photography.	serious spoilers this movie did not deliver what I hoped Photography and directing were on point.

(b) Human generated adversaries, examples where our approach failed to generate SEAs (**Only Human**)

Table 5: Examples of generated adversaries

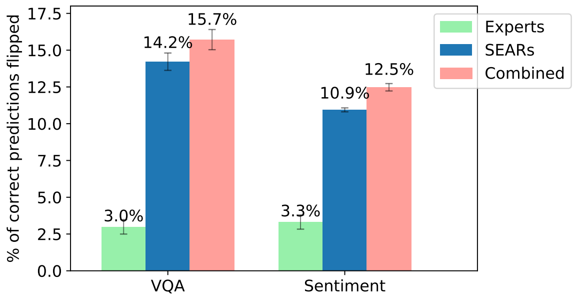


Figure 5: Mistakes induced by expert-generated rules (green), SEARs (blue), and a combination of both (pink), with standard error bars.

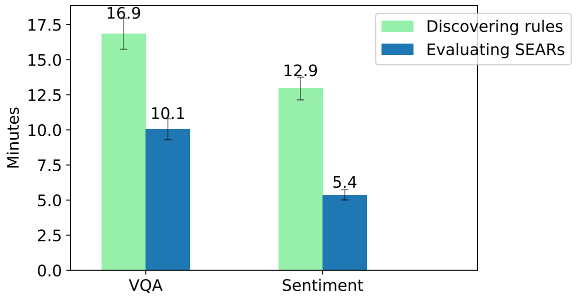


Figure 6: Time for users to create rules (green) and to evaluate SEARs (blue), with standard error bars

nificantly change the sentence structure, which the translation-based semantic scorer does not.

5.3 Can experts find high-impact bugs?

Here we investigate whether experts are able to detect high-impact global bugs, i.e. devise rules that flip many predictions, and compare them to generated SEARs. Instead of AMT workers, we have 26 expert subjects: students, graduates, or professors who have taken at least a graduate course in machine learning or NLP[1]. The experiment setup is as follows: for each task, subjects are given an interface where they see examples in the validation data, perturb those examples, and get predictions. The interface also allows them to create search and replace rules, with immediate feedback on how many mistakes are induced by their rules. They also see the list of examples where the rules apply, so they can verify semantic equivalence. Subjects are instructed to try to maximize the number of mistakes induced in the validation data (i.e. maximize "mistake coverage"), but only through semantically equivalent rules. They can try as many rules as they like, and are asked to select the best set of at most 10 rules at the end. This is quite a challenging task for humans (yet another reason to prefer algorithmic approaches), but we are not aware of any existing automated methods. Finally, we instruct subjects they could finish each task in about 15 minutes (some took longer, some ended earlier), in order to keep the total time reasonable.

After creating their rules for VQA and sentiment analysis, the subjects evaluate 20 SEARs (one rule at a time) for each task, and accept only semantically equivalent rules. When a subject rejects a rule, we recompute the remaining set according to Eq (3) in real time. If a subject accepts more than 10 rules, only the first 10 are considered, in order to ensure a fair comparison against the expert-generated rules.

We compare expert-generated rules with accepted SEARs (each subject's rules are compared to the SEARs they accepted) in terms of the percentage of the correct predictions that "flip" when the rules are applied. This is what we asked the subjects to maximize, and all the rules were ones deemed to be semantic equivalent by the subjects themselves. We also consider the union of expert-generated rules and accepted SEARs. The results in Figure 5 show that on both datasets, the filtered SEARs induce a much higher rate of mistakes than the rules the subjects themselves created, with a small increase when the union of both sets is taken. Furthermore, subjects spent less time evaluating

[1]We have an IRB/consent form, and personal information was only collected as needed to compensate subjects

	Error rate	
	Validation	Sensitivity
Visual QA		
Original Model	44.4.%	12.6%
SEAR Augmented	45.7 %	1.4%
Sentiment Analysis		
Original Model	22.1%	12.6%
SEAR Augmented	21.3%	3.4%

Table 6: **Fixing bugs using SEARs:** Effect of retraining models using SEARs, both on original validation and on sensitivity dataset. Retraining significantly reduces the number of bugs, with statistically insignificant changes to accuracy.

SEARs than trying to create their own rules (Figure 6). SEARs for sentiment analysis contain fewer POS tags, and are thus easier to evaluate for semantic equivalence than for VQA.

Discovering these bugs is hard for humans (even experts) without SEARs: not only do they need to imagine rules that maintain semantic equivalence, they must also discover the model's weak spots. Making good use of POS tags is also a challenge: only 50% of subjects attempt rules with POS tags for VQA, 36% for sentiment analysis.

Experts accepted 8.69 rules (on average) out of 20 for VQA as semantically equivalent, and 17.32 out of 20 for sentiment analysis. Similar to the previous experiment, errors made by the semantic scorer lead to rules that are not semantically equivalent (e.g. Table 7). With minimal human intervention, however, SEARs vastly outperform human experts in finding impactful bugs.

5.4 Fixing bugs using SEARs

Once such bugs are discovered, it is natural to want to fix them. The global and deterministic nature of SEARs make them actionable, as they represent bugs in a systematic manner. Once impactful bugs are identified, we use a simple data augmentation procedure: applying SEARs to the training data, and retraining the model on the original training augmented with the generated examples.

We take the rules that are accepted by ≥ 20 subjects as accepted bugs, a total of 4 rules (in Table 2) for VQA, and 16 rules for sentiment (including ones in Table 3). We then augment the training data by applying these rules to it, and retrain the models. To check if the bugs are still present, we create a sensitivity dataset by applying these SEARs to instances predicted correctly on the validation. A model not prone to the bugs described by these rules should not change any of its predictions, and should thus have error rate 0% on this sensitivity data. We also measure accuracy on the original validation data, to make sure that our bug-fixing procedure is not decreasing accuracy.

Table 6 shows that the incidence of these errors is greatly reduced after augmentation, with negligible changes to the validation accuracy (on both tasks, the changes are consistent with the effect of retraining with different seeds). These results show that SEARs are useful not only for discovering bugs, but are also actionable through a simple augmentation technique for any model.

6 Related Work

Previous work on debugging primarily focuses on *explaining* predictions in validation data in order to uncover bugs (Ribeiro et al., 2016, 2018; Kulesza et al., 2011), or find labeling errors (Zhang et al., 2018; Koh and Liang, 2017). Our work is complementary to these techniques, as they provide no mechanism to detect oversensitivity bugs. We are able to uncover these bugs even when they are not present in the data, since we generate sentences.

Adversarial examples for image recognition are typically indistinguishable to the human eye (Szegedy et al., 2014). These are more of a security concern than bugs per se, as images with adversarial noise are not "natural", and not expected to occur in the real world outside of targeted attacks. Adversaries are usually specific to predictions, and even universal adversarial perturbations (Moosavi-Dezfooli et al., 2017) are not natural, semantically meaningful to humans, or actionable. "Imperceptible" adversarial noise does not carry over from images to text, as adding or changing a single word in a sentence can drastically alter its meaning. Jia and Liang (2017) recognize that a true analog to detect oversensitivity would need semantic-preserving perturbations, but do not pursue an automated solution due to the difficulty of paraphrase generation. Their adversaries are whole sentence concatenations, generated by manually defined rules tailored to reading comprehension, and each adversary is specific to an individual instance. Zhao et al. (2018) generate natural text adversaries by projecting the input data to a latent space using a generative adversarial networks (GANs), and searching for adversaries close to the original instance in this latent space. Apart from the challenge of training GANs to generate high quality text, there is no guarantee that an example close in the latent space is semantically equivalent. Ebrahimi et al. (2018), along with proposing character-level changes that are not semantic-preserving, also propose a heuristic that replaces single words adversarially to preserve semantics. This approach not only depends on having white-box access to the model, but is also not able to generate many adversaries (only $\sim 1.6\%$ for sentiment analysis, compare to $\sim 33\%$ for SEAs in Table 4b). Developed concurrently with our work, Iyyer et al. (2018) proposes a neural paraphrase model based on back-translated data, which is able to produce paraphrases that have different sentence structures from the original. They use paraphrases to generate adversaries and try to post-process nonsensical outputs, but they do not explicitly reject non-semantics preserving ones, nor do they try to induce rules from individual adversaries. In any case, their adversaries are also useful for data augmentation, in experiments similar to ours.

In summary, previous work on text adversaries change semantics, only generate local (instance-specific) adversaries (Zhao et al., 2018; Iyyer et al., 2018), or are tailored for white-box models (Ebrahimi et al., 2018) or specific tasks (Jia and Liang, 2017). In contrast, SEAs expose oversensitivity for specific predictions of black-box models for a variety of tasks, while SEARs are intuitive and actionable global rules that induce a high number of high-quality adversaries. To our knowledge, we are also the first to evaluate human performance in adversarial generation (semantics-preserving or otherwise), and our extensive evaluation shows that SEAs and SEARs detect individual bugs and general patterns better than humans can.

7 Limitations and Future Work

Having demonstrated the usefulness of SEAs and SEARs in a variety of domains, we now describe their limitations and opportunities for future work.

Semantic scoring errors: Paraphrasing is still an active area of research, and thus our semantic scorer is sometimes incorrect in evaluating rules for semantic equivalence. We show examples of SEARs that are rejected by users in Table 7 – the semantic scorer does not sufficiently penalize preposition changes, and is biased towards common terms. The presence of such errors is why we still need humans in the loop to accept or reject SEARs.

SEAR	**Questions / SEAs**	**f(x)**
on →in	What is ~~on~~ in the background?	~~A building~~ Mountains
	What is ~~on~~? in	~~Lights~~ The television
VBP→is	Where ~~are~~ is the water bottles	~~Table~~ Vending Maching
	Where ~~are~~ is the people gathered	~~living room~~ kitchen
VERB on → VERB	What is ~~on~~ the background?	~~A building~~ Mountains
	What are the planes parked ~~on~~?	~~Concrete~~ landing strip

Table 7: SEARs for VQA that are rejected by users

Other paraphrase limitations: Paraphrase models based on neural machine translation are biased towards maintaining the sentence structure, and thus do not produce certain adversaries (e.g. Table 5b), which recent work on paraphrasing (Iyyer et al., 2018) or generation using GANs (Zhao et al., 2018) may address. More critically, existing models are inaccurate for long texts, restricting SEAs and SEARs to sentences.

Better bug fixing: Our data augmentation has the human users accept/reject rules based on whether or not they preserve semantics. Developing more effective ways of leveraging the expert's time to close the loop, and facilitating more interactive collaboration between humans and SEARs are exciting areas for future work.

8 Conclusion

We introduced SEAs and SEARs – adversarial examples and rules that preserve semantics, while causing models to make mistakes. We presented examples of such bugs discovered in state-of-the-art models for various tasks, and demonstrated via user studies that non-experts and experts alike are much better at detecting local and global bugs in NLP models by using our methods. We also *close the loop* by proposing a simple data augmentation solution that greatly reduced oversensitivity while maintaining accuracy. We demonstrated that SEAs and SEARs can be an invaluable tool for debugging NLP models, while indicating their current limitations and avenues for future work.

Acknowledgements

We are grateful to Dan Weld, Robert L. Logan IV, and to the anonymous reviewers for their feedback. This work was supported in part by ONR award #N00014-13-1-0023, in part by NSF award #IIS-1756023, and in part by funding from FICO. The views expressed are of the authors and do not reflect the policy or opinion of the funding agencies.

References

Aayush Bansal, Ali Farhadi, and Devi Parikh. 2014. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision (ECCV)*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Deijing Dou. 2018. HotFlip: White-Box Adversarial Examples for NLP. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matt A Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics (NAACL)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*.

Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Knowledge Discovery and Data Mining (KDD)*.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*.

Todd Kulesza, Simone Stumpf, Weng-Keen Wong, Margaret M. Burnett, Stephen Perona, Andrew Jensen Ko, and Ian Oberst. 2011. Why-oriented end-user debugging of naive bayes text classification. *TiiS* 1:2:1–2:31.

Mirella Lapata, Rico Sennrich, and Jonathan Mallinson. 2017. Paraphrasing revisited with neural machine translation. In *European Chapter of the ACL (EACL)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. *CoRR* abs/1612.08220.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. 2008. Investigating statistical machine learning as a tool for software development. In *Human Factors in Computing Systems (CHI)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Knowledge Discovery and Data Mining (KDD)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *International Conference on Language Resources and Evaluation (LREC)*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. 2018. Training set debugging using trusted items. In *AAAI Conference on Artificial Intelligence*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7W: Grounded Question Answering in Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Style Transfer Through Back-Translation

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, Alan W Black
Carnegie Mellon University, Pittsburgh, PA, USA
{sprabhum,ytsvetko,rsalakhu,awb}@cs.cmu.edu

Abstract

Style transfer is the task of rephrasing the text to contain specific stylistic properties without changing the intent or affect within the context. This paper introduces a new method for automatic style transfer. We first learn a latent representation of the input sentence which is grounded in a language translation model in order to better preserve the meaning of the sentence while reducing stylistic properties. Then adversarial generation techniques are used to make the output match the desired style. We evaluate this technique on three different style transformations: sentiment, gender and political slant. Compared to two state-of-the-art style transfer modeling techniques we show improvements both in automatic evaluation of style transfer and in manual evaluation of meaning preservation and fluency.

1 Introduction

Intelligent, situation-aware applications must produce naturalistic outputs, lexicalizing the same meaning differently, depending upon the environment. This is particularly relevant for language generation tasks such as machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), caption generation (Karpathy and Fei-Fei, 2015; Xu et al., 2015), and natural language generation (Wen et al., 2017; Kiddon et al., 2016). In conversational agents (Ritter et al., 2011; Sordoni et al., 2015; Vinyals and Le, 2015; Li et al., 2016), for example, modulating the politeness style, to sound natural depending upon a situation: at a party with friends "Shut up! the video is starting!", or in a professional setting "Please be quiet, the video will begin shortly.".

These goals have motivated a considerable amount of recent research efforts focused at "controlled" language generation—aiming at separating the semantic content of *what* is said from the stylistic dimensions of *how* it is said. These include approaches relying on heuristic substitutions, deletions, and insertions to modulate demographic properties of a writer (Reddy and Knight, 2016), integrating stylistic and demographic speaker traits in statistical machine translation (Rabinovich et al., 2016; Niu et al., 2017), and deep generative models controlling for a particular stylistic aspect, e.g., politeness (Sennrich et al., 2016), sentiment, or tense (Hu et al., 2017; Shen et al., 2017). The latter approaches to style transfer, while more powerful and flexible than heuristic methods, have yet to show that in addition to transferring style they effectively preserve meaning of input sentences.

This paper introduces a novel approach to transferring style of a sentence while better preserving its meaning. We hypothesize—relying on the study of Rabinovich et al. (2016) who showed that author characteristics are significantly obfuscated by both manual and automatic machine translation—that grounding in back-translation is a plausible approach to rephrase a sentence while reducing its stylistic properties. We thus first use back-translation to rephrase the sentence and reduce the effect of the original style; then, we generate from the latent representation, using separate style-specific generators controlling for style (§2).

We focus on transferring author attributes: (1) gender and (2) political slant, and (3) on sentiment modification. The second task is novel: given a sentence by an author with a particular political leaning, rephrase the sentence to preserve its meaning but to confound classifiers of political slant (§3). The task of sentiment modification enables us to compare our approach with state-of-

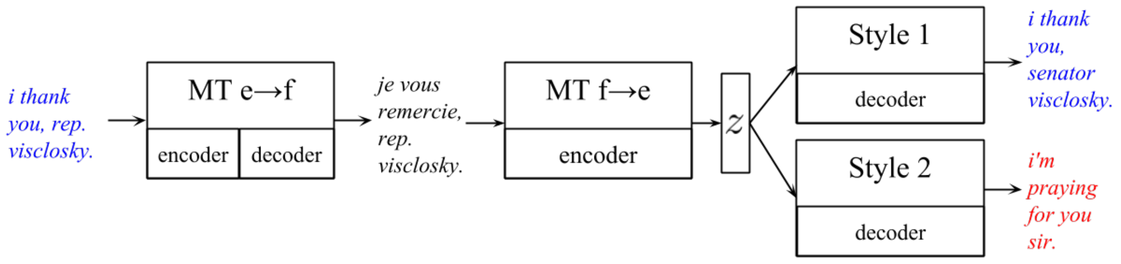

i thank you, rep. visclosky.
MT e→f
encoder
decoder
je vous remercie, rep. visclosky.
MT f→e
encoder
z
Style 1
decoder
i thank you, senator visclosky.
Style 2
decoder
i'm praying for you sir.

Figure 1: Style transfer pipeline: to rephrase a sentence and reduce its stylistic characteristics, the sentence is back-translated. Then, separate style-specific generators are used for style transfer.

the-art models (Hu et al., 2017; Shen et al., 2017).

Style transfer is evaluated using style classifiers trained on held-out data. Our back-translation style transfer model outperforms the state-of-the-art baselines (Shen et al., 2017; Hu et al., 2017) on the tasks of political slant and sentiment modification; 12% absolute improvement was attained for political slant transfer, and up to 7% absolute improvement in modification of sentiment (§5). Meaning preservation was evaluated manually, using A/B testing (§4). Our approach performs better than the baseline on the task of transferring gender and political slant. Finally, we evaluate the fluency of the generated sentences using human evaluation and our model outperforms the baseline in all experiments for fluency.

The main contribution of this work is a new approach to style transfer that outperforms state-of-the-art baselines in both the quality of input–output correspondence (meaning preservation and fluency), and the accuracy of style transfer. The secondary contribution is a new task that we propose to evaluate style transfer: transferring political slant.

2 Methodology

Given two datasets $\boldsymbol{X}_1 = \{\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_1^{(n)}\}$ and $\boldsymbol{X}_2 = \{\boldsymbol{x}_2^{(1)}, \ldots, \boldsymbol{x}_2^{(n)}\}$ which represent two different styles $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, respectively, our task is to generate sentences of the desired style while preserving the meaning of the input sentence. Specifically, we generate samples of dataset $\boldsymbol{X}_1$ such that they belong to style $\boldsymbol{s}_2$ and samples of $\boldsymbol{X}_2$ such that they belong to style $\boldsymbol{s}_1$. We denote the output of dataset $\boldsymbol{X}_1$ transfered to style $\boldsymbol{s}_2$ as $\hat{\boldsymbol{X}}_1 = \{\hat{\boldsymbol{x}}_2^{(1)}, \ldots, \hat{\boldsymbol{x}}_2^{(n)}\}$ and the output of dataset $\boldsymbol{X}_2$ transferred to style $\boldsymbol{s}_1$ as $\hat{\boldsymbol{X}}_2 = \{\hat{\boldsymbol{x}}_1^{(1)}, \ldots, \hat{\boldsymbol{x}}_1^{(n)}\}$.

Hu et al. (2017) and Shen et al. (2017) introduced state-of-the-art style transfer models that use variational auto-encoders (Kingma and Welling, 2014, VAEs) and cross-aligned auto-encoders, respectively, to model a latent content variable $\boldsymbol{z}$. The latent content variable $\boldsymbol{z}$ is a code which is not observed. The generative model conditions on this code during the generation process. Our aim is to design a latent code $\boldsymbol{z}$ which (1) represents the meaning of the input sentence grounded in back-translation and (2) weakens the style attributes of author's traits. To model the former, we use neural machine translation. Prior work has shown that the process of translating a sentence from a source language to a target language retains the meaning of the sentence but does not preserve the stylistic features related to the author's traits (Rabinovich et al., 2016). We hypothesize that a latent code $\boldsymbol{z}$ obtained through back-translation will normalize the sentence and devoid it from style attributes specific to author's traits.

Figure 1 shows the overview of the proposed method. In our framework, we first train a machine translation model from source language e to a target language f. We also train a back-translation model from f to e. Let us assume our styles $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ correspond to DEMOCRATIC and REPUBLICAN style, respectively. In Figure 1, the input sentence *i thank you, rep. visclosky.* is labeled as DEMOCRATIC. We translate the sentence using the $e \rightarrow f$ machine translation model and generate the parallel sentence in the target language f: *je vous remercie, rep. visclosky.* Using the fixed encoder of the $f \rightarrow e$ machine translation model, we encode this sentence in language f. The hidden representation created by this encoder of the back-translation model is used as $\boldsymbol{z}$. We condition our generative models on this $\boldsymbol{z}$. We then train two separate decoders for each style $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ to generate samples in these respective styles in source language e. Hence the sentence could be translated to the REPUBLICAN style using the decoder for $\boldsymbol{s}_2$. For example, the sentence *i'm praying for you sir.* is the REPUBLICAN ver-

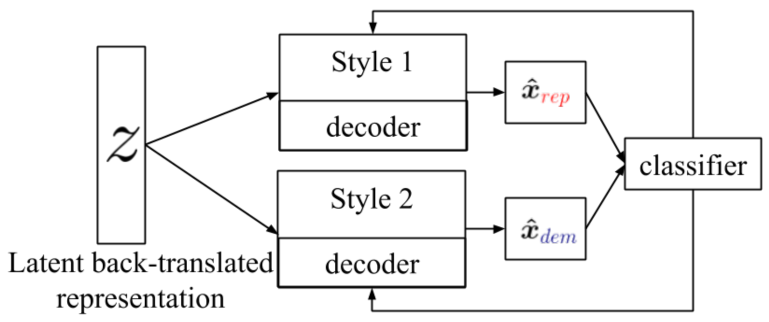


Figure 2: The latent representation from back-translation and the style classifier feedback are used to guide the style-specific generators.

sion of the input sentence and *i thank you, senator visclosky.* is the more DEMOCRATIC version of it.

Note that in this setting, the machine translation and the encoder of the back-translation model remain fixed. They are not dependent on the data we use across different tasks. This facilitates reusability and spares the need of learning separate models to generate $\boldsymbol{z}$ for a new style data.

2.1 Meaning-Grounded Representation

In this section we describe how we learn the latent content variable $\boldsymbol{z}$ using back-translation. The $e \rightarrow f$ machine translation and $f \rightarrow e$ back-translation models are trained using a sequence-to-sequence framework (Sutskever et al., 2014; Bahdanau et al., 2015) with style-agnostic corpus. The style-specific sentence *i thank you, rep. visclosky.* in source language e is translated to the target language f to get *je vous remercie, rep. visclosky.* The individual tokens of this sentence are then encoded using the encoder of the $f \rightarrow e$ back-translation model. The learned hidden representation is $\boldsymbol{z}$.

Formally, let $\boldsymbol{\theta}_E$ represent the parameters of the encoder of $f \rightarrow e$ translation system. Then $\boldsymbol{z}$ is given by:

$$\boldsymbol{z} = Encoder(\boldsymbol{x}_f; \boldsymbol{\theta}_E) \tag{1}$$

where, $\boldsymbol{x}_f$ is the sentence $\boldsymbol{x}$ in language f. Specifically, $\boldsymbol{x}_f$ is the output of $e \rightarrow f$ translation system when $\boldsymbol{x}_e$ is given as input. Since $\boldsymbol{z}$ is derived from a non-style specific process, this Encoder is not style specific.

2.2 Style-Specific Generation

Figure 2 shows the architecture of the generative model for generating different styles. Using the encoder embedding $\boldsymbol{z}$, we train multiple decoders for each style. The sentence generated by a decoder is passed through the classifier. The loss of the classifier for the generated sentence is used as feedback to guide the decoder for the generation process. The target attribute of the classifier is determined by the decoder from which the output is generated. For example, in the case of DEMOCRATIC decoder, the target attribute is DEMOCRATIC and for the REPUBLICAN decoder the target is REPUBLICAN.

2.2.1 Style Classifiers

We train a convolutional neural network (CNN) classifier to accurately predict the given style. We also use it to evaluate the error in the generated samples for the desired style. We train the classifier in a supervised manner. The classifier accepts either discrete or continuous tokens as inputs. This is done such that the generator output can be used as input to the classifier. We need labeled examples to train the classifier such that each instance in the dataset $\boldsymbol{X}$ should have a label in the set $\boldsymbol{s} = \{\boldsymbol{s}_1, \boldsymbol{s}_2\}$. Let $\boldsymbol{\theta}_C$ denote the parameters of the classifier. The objective to train the classifier is given by:

$$\mathcal{L}_{class}(\boldsymbol{\theta}_C) = \mathbb{E}_{\boldsymbol{X}}[\log q_C(\boldsymbol{s}|\boldsymbol{x})]. \tag{2}$$

To improve the accuracy of the classifier, we augment classifier's inputs with style-specific lexicons. We concatenate binary style indicators to each input word embedding in the classifier. The indicators are set to 1 if the input word is present in a style-specific lexicon; otherwise they are set to 0. Style lexicons are extracted using the log-odds ratio informative Dirichlet prior (Monroe et al., 2008), a method that identifies words that are statistically overrepresented in each of the categories.

2.2.2 Generator Learning

We use a bidirectional LSTM to build our decoders which generate the sequence of tokens $\hat{\boldsymbol{x}} = \{x_1, \cdots x_T\}$. The sequence $\hat{\boldsymbol{x}}$ is conditioned on the latent code $\boldsymbol{z}$ (in our case, on the machine translation model). In this work we use a corpus translated to French by the machine translation system as the input to the encoder of the back-translation model. The same encoder is used to encode sentences of both styles. The representation created by this encoder is given by Eq 1. Samples are generated as follows:

$$\begin{aligned} \hat{\boldsymbol{x}} \sim \boldsymbol{z} &= p(\hat{\boldsymbol{x}}|\boldsymbol{z}) && (3) \\ &= \prod_t p(\hat{x}_t|\hat{\boldsymbol{x}}^{<t}, \boldsymbol{z}) && (4) \end{aligned}$$

where, $\hat{\boldsymbol{x}}^{<t}$ are the tokens generated before $\hat{x}_t$.

Tokens are discrete and non-differentiable. This makes it difficult to use a classifier, as the generation process samples discrete tokens from the multinomial distribution parametrized using softmax function at each time step t. This non-differentiability, in turn, breaks down gradient propagation from the discriminators to the generator. Instead, following Hu et al. (2017) we use a continuous approximation based on softmax, along with the temperature parameter which anneals the softmax to the discrete case as training proceeds. To create a continuous representation of the output of the generative model which will be given as an input to the classifier, we use:

$$\hat{x}_t \sim \text{softmax}(\boldsymbol{o}_t/\tau),$$

where, $\boldsymbol{o}_t$ is the output of the generator and τ is the temperature which decreases as the training proceeds. Let $\boldsymbol{\theta}_G$ denote the parameters of the generators. Then the reconstruction loss is calculated using the cross entropy function, given by:

$$\mathcal{L}_{recon}(\boldsymbol{\theta}_G; \boldsymbol{x}) = \mathbb{E}_{q_E(\boldsymbol{z}|\boldsymbol{x})}[\log p_{gen}(\boldsymbol{x}|\boldsymbol{z})] \quad (5)$$

Here, the back-translation encoder E creates the latent code $\boldsymbol{z}$ by:

$$\boldsymbol{z} = E(\boldsymbol{x}) = q_E(\boldsymbol{z}|\boldsymbol{x}) \quad (6)$$

The generative loss $\mathcal{L}_{gen}$ is then given by:

$$\min_{\theta_{gen}} \mathcal{L}_{gen} = \mathcal{L}_{recon} + \lambda_c \mathcal{L}_{class} \quad (7)$$

where $\mathcal{L}_{recon}$ is given by Eq. (5), $\mathcal{L}_{class}$ is given by Eq (2) and λ_c is a balancing parameter.

We also use global attention of (Luong et al., 2015) to aid our generators. At each time step t of the generation process, we infer a variable length alignment vector $\boldsymbol{a}_t$:

$$\boldsymbol{a}_t = \frac{\exp(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s))}{\sum_{s'} \exp(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'}))} \quad (8)$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \textbf{dot}(\boldsymbol{h}_t^T, \bar{\boldsymbol{h}}_s), \quad (9)$$

where $\boldsymbol{h}_t$ is the current target state and $\bar{\boldsymbol{h}}_s$ are all source states. While generating sentences, we use the attention vector to replace unknown characters (UNK) using the copy mechanism in (See et al., 2017).

3 Style Transfer Tasks

Much work in computational social science has shown that people's personal and demographic characteristics—either publicly observable (e.g., age, gender) or private (e.g., religion, political affiliation)—are revealed in their linguistic choices (Nguyen et al., 2016). There are practical scenarios, however, when these attributes need to be modulated or obfuscated. For example, some users may wish to preserve their anonymity online, for personal security concerns (Jardine, 2016), or to reduce stereotype threat (Spencer et al., 1999). Modulating authors' attributes while preserving meaning of sentences can also help generate demographically-balanced training data for a variety of downstream applications.

Moreover, prior work has shown that the quality of language identification and POS tagging degrades significantly on African American Vernacular English (Blodgett et al., 2016; Jørgensen et al., 2015); YouTube's automatic captions have higher error rates for women and speakers from Scotland (Rudinger et al., 2017). Synthesizing balanced training data—using style transfer techniques—is a plausible way to alleviate bias present in existing NLP technologies.

We thus focus on two tasks that have practical and social-good applications, and also accurate style classifiers. To position our method with respect to prior work, we employ a third task of sentiment transfer, which was used in two state-of-the-art approaches to style transfer (Hu et al., 2017; Shen et al., 2017). We describe the three tasks and associated dataset statistics below. The methodology that we advocate is general and can be applied to other styles, for transferring various social categories, types of bias, and in multi-class settings.

Gender. In sociolinguistics, gender is known to be one of the most important social categories driving language choice (Eckert and McConnell-Ginet, 2003; Lakoff and Bucholtz, 2004; Coates, 2015). Reddy and Knight (2016) proposed a heuristic-based method to obfuscate gender of a writer. This method uses statistical association measures to identify gender-salient words and substitute them with synonyms typically of the opposite gender. This simple approach produces highly fluent, meaning-preserving sentences, but does not allow for more general rephrasing of sentence beyond single-word substitutions. In our work, we adopt this task of transferring the author's gender and adapt it to our experimental settings.

We used Reddy and Knight's (2016) dataset of reviews from Yelp annotated for two genders corresponding to markers of sex.[1] We split the reviews to sentences, preserving the original gender labels. To keep only sentences that are strongly indicative of a gender, we then filtered out gender-neutral sentences (e.g., *thank you*) and sentences whose likelihood to be written by authors of one gender is lower than 0.7.[2]

Political slant. Our second dataset is comprised of top-level comments on Facebook posts from all 412 current members of the United States Senate and House who have public Facebook pages (Voigt et al., 2018).[3] Only top-level comments that directly respond to the post are included. Every comment to a Congressperson is labeled with the Congressperson's party affiliation: democratic or republican. Topic and sentiment in these comments reveal commenter's political slant. For example, *defund them all, especially when it comes to the illegal immigrants .* and *thank u james, praying for all the work u do .* are republican, whereas *on behalf of the hard-working nh public school teachers- thank you !* and *we need more strong voices like yours fighting for gun control .* represent examples of democratic sentences. Our task is to preserve intent of the commenter (e.g., to thank their representative), but to modify their observable political affiliation, as in the example in Figure 1. We preprocessed and filtered the comments similarly to the gender-annotated corpus above.

Style	*class*	*train*	*dev*	*test*
gender	2.57M	2.67M	4.5K	535K
political	80K	540K	4K	56K
sentiment	2M	444K	63.5K	127K

Table 1: Sentence count in style-specific corpora.

Sentiment. To compare our work with the state-of-the-art approaches of style transfer for non-parallel corpus we perform sentiment transfer, replicating the models and experimental setups of Hu et al. (2017) and Shen et al. (2017). Given a positive Yelp review, a style transfer model will generate a similar review but with an opposite sentiment. We used Shen et al.'s (2017) corpus of reviews from Yelp. They have followed the standard practice of labeling the reviews with rating of higher than three as positive and less than three as negative. They have also split the reviews to sentences and assumed that the sentence has the same sentiment as the review.

Dataset statistics. We summarize below corpora statistics for the three tasks: transferring gender, political slant, and sentiment. The dataset for sentiment modification task was used as described in (Shen et al., 2017). We split Yelp and Facebook corpora into four disjoint parts each: (1) a training corpus for training a style classifier (*class*); (2) a training corpus (*train*) used for training the style-specific generative model described in §2.2; (3) development and (4) test sets. We have removed from training corpora *class* and *train* all sentences that overlap with development and test corpora. Corpora sizes are shown in Table 1.

Table 2 shows the approximate vocabulary sizes used for each dataset. The vocabulary is the same for both the styles in each experiment.

Style	gender	political	sentiment
Vocabulary	20K	20K	10K

Table 2: Vocabulary sizes of the datasets.

Table 3 summarizes sentence statistics. All the

[1]We note that gender may be considered along a spectrum (Eckert and McConnell-Ginet, 2003), but use gender as a binary variable due to the absence of corpora with continuous-valued gender annotations.

[2]We did not experiment with other threshold values.

[3]The posts and comments are all public; however, to protect the identity of Facebook users in this dataset Voigt et al. (2018) have removed all identifying user information as well as Facebook-internal information such as User IDs and Post IDs, replacing these with randomized ID numbers.

sentences have maximum length of 50 tokens.

Style	Avg. Length	%data
male	18.08	50.00
female	18.21	50.00
republican	16.18	50.00
democratic	16.01	50.00
negative	9.66	39.81
positive	8.45	60.19

Table 3: Average sentence length and class distribution of style corpora.

4 Experimental Setup

In what follows, we describe our experimental settings, including baselines used, hyperparameter settings, datasets, and evaluation setups.

Baseline. We compare our model against the "cross-aligned" auto-encoder (Shen et al., 2017), which uses style-specific decoders to align the style of generated sentences to the actual distribution of the style. We used the off-the-shelf sentiment model released by Shen et al. (2017) for the sentiment experiments. We also separately train this model for the gender and political slant using hyper-parameters detailed below.[4]

Translation data. We trained an English–French neural machine translation system and a French–English back-translation system. We used data from Workshop in Statistical Machine Translation 2015 (WMT15) (Bojar et al., 2015) to train our translation models. We used the French–English data from the Europarl v7 corpus, the news commentary v10 corpus and the common crawl corpus from WMT15. Data were tokenized using the Moses tokenizer (Koehn et al., 2007). Approximately 5.4M English–French parallel sentences were used for training. A vocabulary size of 100K was used to train the translation systems.

Hyperparameter settings. In all the experiments, the generator and the encoders are a two-layer bidirectional LSTM with an input size of 300 and the hidden dimension of 500. The generator samples a sentence of maximum length 50. All the generators use global attention vectors of size 500. The CNN classifier is trained with 100 filters of size 5, with max-pooling. The input to CNN is of size 302: the 300-dimensional word embedding plus two bits for membership of the word in our style lexicons, as described in §2.2.1. Balancing parameter λ_c is set to 15. For sentiment task, we have used settings provided in (Shen et al., 2017).

5 Results

We evaluate our approach along three dimensions. (1) Style transfer accuracy, measuring the proportion of our models' outputs that generate sentences of the desired style. The style transfer accuracy is performed using classifiers trained on held-out train data that were not used in training the style transfer models. (2) Preservation of meaning. (3) Fluency, measuring the readability and the naturalness of the generated sentences. We conducted human evaluations for the latter two.

In what follows, we first present the quality of our neural machine translation systems, then we present the evaluation setups, and then present the results of our experiments.

Translation quality. The BLEU scores achieved for English–French MT system is 32.52 and for French–English MT system is 31.11; these are strong translation systems. We deliberately chose a European language close to English for which massive amounts of parallel data are available and translation quality is high, to concentrate on the style generation, rather than improving a translation system. [5]

5.1 Style Transfer Accuracy

We measure the accuracy of style transfer for the generated sentences using a pre-trained style classifier (§2.2.1). The classifier is trained on data that is not used for training our style transfer generative models (as described in §3). The classifier has an accuracy of 82% for the gender-annotated corpus, 92% accuracy for the political slant dataset and 93.23% accuracy for the sentiment dataset.

[4]In addition, we compared our model with the current state-of-the-art approach introduced by Hu et al. (2017); Shen et al. (2017) use this method as baseline, obtaining comparable results. We reproduced the results reported in (Hu et al., 2017) using their tasks and data. However, the same model trained on our political slant datasets (described in §3), obtained an almost random accuracy of 50.98% in style transfer. We thus omit these results.

[5]Alternatively, we could use a pivot language that is typologically more distant from English, e.g., Chinese. In this case we hypothesize that stylistic traits would be even less preserved in translation, but the quality of back-translated sentences would be worse. We have not yet investigated how the accuracy of the translation model, nor the language of translation affects our models.

We transfer the style of test sentences and then test the classification accuracy of the generated sentences for the opposite label. For example, if we want to transfer the style of male Yelp reviews to female, then we use the fixed common encoder of the back-translation model to encode the test male sentences and then we use the female generative model to generate the female-styled reviews. We then test these generated sentences for the *female* label using the gender classifier.

Experiment	CAE	BST
Gender	**60.40**	57.04
Political slant	75.82	**88.01**
Sentiment	80.43	**87.22**

Table 4: Accuracy of the style transfer in generated sentences.

In Table 4, we detail the accuracy of each classifier on generated style-transfered sentences.[6] We denote the Shen et al.'s (2017) **C**ross-aligned **A**uto-**E**ncoder model as CAE and our model as **B**ack-translation for **S**tyle **T**ransfer (BST).

On two out of three tasks our model substantially outperforms the baseline, by up to 12% in political slant transfer, and by up to 7% in sentiment modification.

5.2 Preservation of Meaning

Although we attempted to use automatics measures to evaluate how well meaning is preserved in our transformations; measures such as BLEU (Papineni et al., 2002) and Meteor (Denkowski and Lavie, 2011), or even cosine similarity between distributed representations of sentences do not capture this distance well.

Meaning preservation in style transfer is not trivial to define as literal meaning is likely to change when style transfer occurs. For example "My girlfriend loved the desserts" vs "My partner liked the desserts". Thus we must relax the condition of literal meaning to *intent* or *affect* of the utterance within the context of the discourse. Thus if the intent is to criticize a restaurant's service in a review, changing "salad" to "chicken" could still have the same effect but if the intent is to order food that substitution would not be acceptable. Ideally we wish to evaluate transfer within some downstream task and ensure that the task has the same outcome even after style transfer. This is a hard evaluation and hence we resort to a simpler evaluation of the "meaning" of the sentence.

[6]In each experiment, we report aggregated results across directions of style transfer; same results broke-down to style categories are listed in the Supplementary Material.

Experiment	CAE	No Pref.	BST
Gender	15.23	41.36	**43.41**
Political slant	14.55	**45.90**	39.55
Sentiment	35.91	**40.91**	23.18

Table 5: Human preference for meaning preservation in percentages.

We set up a manual pairwise comparison following Bennett (2005). The test presents the original sentence and then, in random order, its corresponding sentences produced by the baseline and our models. For the gender style transfer we asked "Which transferred sentence maintains the same sentiment of the source sentence in the same semantic context (i.e. you can ignore if food items are changed)". For the task of changing the political slant, we asked "Which transferred sentence maintains the same semantic intent of the source sentence while changing the political position". For the task of sentiment transfer we have followed the annotation instruction in (Shen et al., 2017) and asked "Which transferred sentence is semantically equivalent to the source sentence with an opposite sentiment"

We then count the preferences of the eleven participants, measuring the relative acceptance of the generated sentences.[7] A third option "=" was given to participants to mark no preference for either of the generated sentence. The "no preference" option includes choices both are equally bad and both are equally good. We conducted three tests one for each type of experiment - gender, political slant and sentiment. We also divided our annotation set into short (#tokens $\leq$ 15) and long ($15 <$ #tokens ≤ 30) sentences for the gender and the political slant experiment. In each set we had 20 random samples for each type of style transfer. In total we had 100 sentences to be annotated. Note that we did not ask about appropriateness of the style transfer in this test, or fluency of outputs, only about meaning preservation.

The results of human evaluation are presented in Table 5. Although a no-preference option was chosen often—showing that state-of-the-art systems are still not on par with hu-

[7]None of the human judges are authors of this paper

man expectations—the BST models outperform the baselines in the gender and the political slant transfer tasks.

Crucially, the BST models significantly outperform the CAE models when transferring style in longer and harder sentences. Annotators preferred the CAE model only for 12.5% of the long sentences, compared to 47.27% preference for the BST model.

5.3 Fluency

Finally, we evaluate the fluency of the generated sentences. Fluency was rated from 1 (unreadable) to 4 (perfect) as is described in (Shen et al., 2017). We randomly selected 60 sentences each generated by the baseline and the BST model.

The results shown in Table 6 are averaged scores for each model.

Experiment	CAE	BST
Gender	2.42	**2.81**
Political slant	2.79	**2.87**
Sentiment	3.09	**3.18**
Overall	2.70	**2.91**
Overall Short	3.05	**3.11**
Overall Long	2.18	**2.62**

Table 6: Fluency of the generated sentences.

BST outperforms the baseline overall. It is interesting to note that BST generates significantly more fluent longer sentences than the baseline model. Since the average length of sentences was higher for the gender experiment, BST notably outperformed the baseline in this task, relatively to the sentiment task where the sentences are shorter. Examples of the original and style-transfered sentences generated by the baseline and our model are shown in the Supplementary Material.

5.4 Discussion

The loss function of the generators given in Eq. 5 includes two competing terms, one to improve meaning preservation and the other to improve the style transfer accuracy. In the task of sentiment modification, the BST model preserved meaning worse than the baseline, on the expense of being better at style transfer. We note, however, that the sentiment modification task is not particularly well-suited for evaluating style transfer: it is particularly hard (if not impossible) to disentangle the sentiment of a sentence from its propositional content, and to modify sentiment while preserving meaning or intent. On the other hand, the style-transfer accuracy for gender is lower for BST model but the preservation of meaning is much better for the BST model, compared to CAE model and to "No preference" option. This means that the BST model does better job at closely representing the input sentence while taking a mild hit in the style transfer accuracy.

6 Related Work

Style transfer with non-parallel text corpus has become an active research area due to the recent advances in text generation tasks. Hu et al. (2017) use variational auto-encoders with a discriminator to generate sentences with controllable attributes. The method learns a disentangled latent representation and generates a sentence from it using a code. This paper mainly focuses on sentiment and tense for style transfer attributes. It evaluates the transfer strength of the generated sentences but does not evaluate the extent of preservation of meaning in the generated sentences. In our work, we show a qualitative evaluation of meaning preservation.

Shen et al. (2017) first present a theoretical analysis of style transfer in text using non-parallel corpus. The paper then proposes a novel cross-alignment auto-encoders with discriminators architecture to generate sentences. It mainly focuses on sentiment and word decipherment for style transfer experiments.

Fu et al. (2018) explore two models for style transfer. The first approach uses multiple decoders for each type of style. In the second approach, style embeddings are used to augment the encoded representations, so that only one decoder needs to be learned to generate outputs in different styles. Style transfer is evaluated on scientific paper titles and newspaper tiles, and sentiment in reviews. This method is different from ours in that we use machine translation to create a strong latent state from which multiple decoders can be trained for each style. We also propose a different human evaluation scheme.

Li et al. (2018) first extract words or phrases associated with the original style of the sentence, delete them from the original sentence and then replace them with new phrases associated with the target style. They then use a neural model to fluently combine these into a final output. Junbo et al. (2017) learn a representation which is style-agnostic, using adversarial training of the auto-encoder.

Our work is also closely-related to a problem of paraphrase generation (Madnani and Dorr, 2010; Dong et al., 2017), including methods relying on (phrase-based) back-translation (Ganitkevitch et al., 2011; Ganitkevitch and Callison-Burch, 2014). More recently, Mallinson et al. (2017) and Wieting et al. (2017) showed how neural back-translation can be used to generate paraphrases. An additional related line of research is machine translation with non-parallel data. Lample et al. (2018) and Artetxe et al. (2018) have proposed sophisticated methods for unsupervised machine translation. These methods could in principle be used for style transfer as well.

7 Conclusion

We propose a novel approach to the task of style transfer with non-parallel text.[8] We learn a latent content representation using machine translation techniques; this aids grounding the meaning of the sentences, as well as weakening the style attributes. We apply this technique to three different style transfer tasks. In transfer of political slant and sentiment we outperform an off-the-shelf state-of-the-art baseline using a cross-aligned autoencoder. The political slant task is a novel task that we introduce. Our model also outperforms the baseline in all the experiments of fluency, and in the experiments for meaning preservation in generated sentences of gender and political slant. Yet, we acknowledge that the generated sentences do not always adequately preserve meaning.

This technique is suitable not just for style transfer, but for enforcing style, and removing style too. In future work we intend to apply this technique to *debiasing* sentences and *anonymization* of author traits such as gender and age.

In the future work, we will also explore whether an enhanced back-translation by pivoting through several languages will learn better grounded latent meaning representations. In particular, it would be interesting to back-translate through multiple target languages with a single source language (Johnson et al., 2016).

Measuring the separation of style from content is hard, even for humans. It depends on the task and the context of the utterance within its discourse. Ultimately we must evaluate our style transfer within some down-stream task where our style transfer has its intended use but we achieve the same task completion criteria.

[8] All the code and data used in the experiments will be released to facilitate reproducibility at https://github.com/shrimai/Style-Transfer-Through-Back-Translation

Acknowledgments

This work was funded by a fellowship from Robert Bosch, and in part by the National Science Foundation through award IIS-1526745. We would like to thank Sravana Reddy for sharing the Yelp corpus used in gender transfer experiments, Zhiting Hu for providing an implementation of a VAE-based baseline, and the 11 CMU graduate students who helped with annotation and manual evaluations. We are also grateful to the anonymous reviewers for their constructive feedback, and to Dan Jurafsky, David Jurgens, Vinod Prabhakaran, and Rob Voigt for valuable discussions at earlier stages of this work.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proc ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Christina L Bennett. 2005. Large scale evaluation of corpus-based synthesizers: Results and lessons from the blizzard challenge 2005. In *Ninth European Conference on Speech Communication and Technology*.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. In *Proc. EMNLP*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proc. WMT*, pages 1–46.

Jennifer Coates. 2015. *Women, men and language: A sociolinguistic account of gender differences in language*. Routledge.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proc. WMT*, pages 85–91.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886. Association for Computational Linguistics.

Penelope Eckert and Sally McConnell-Ginet. 2003. *Language and gender*. Cambridge University Press.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style Transfer in Text: Exploration and Evaluation. In *Proc. AAAI*.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proc. LREC*, pages 4276–4283.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proc. EMNLP*, pages 1168–1179.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proc. ICML*, pages 1587–1596.

Eric Jardine. 2016. Tor, what is it good for? political repression and the use of online anonymity-granting technologies. *New Media & Society*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of studying and processing dialects in social media. In *Proc. of the Workshop on Noisy User-generated Text*, pages 9–18.

Junbo, Zhao, Y. Kim, K. Zhang, A. M. Rush, and Y. LeCun. 2017. Adversarially Regularized Autoencoders for Generating Discrete Structures. *ArXiv e-prints*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proc. CVPR*, pages 3128–3137.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proc. EMNLP*, pages 329–339.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proc. ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL (demonstration sessions)*, pages 177–180.

Robin Tolmach Lakoff and Mary Bucholtz. 2004. *Language and woman's place: Text and commentaries*, volume 3. Oxford University Press, USA.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proc. ICLR*.

J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer. *ArXiv e-prints*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proc. ACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proce. EACL*, volume 1, pages 881–893.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*.

Dong Nguyen, A. Seza Doğruöz, Carolyn P. Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *Computational Linguistics*, 42(3):537–593.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proc. EMNLP*, pages 2804–2809.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

Ella Rabinovich, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. 2016. Personalized machine translation: Preserving original author traits. In *Proc. EACL*.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proc. of Workshop on Natural Language Processing and Computational Social Science*, pages 17–26.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proc. EMNLP*, pages 583–593.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proc. of the First Workshop on Ethics in Natural Language Processing*, page 74.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proc. NAACL*, pages 35–40.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proc. NIPS*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. NAACL*.

Steven J. Spencer, Claude M. Steele, and Diane M. Quinn. 1999. Stereotype Threat and Women's Math Performance. *Journal of Experimental Social Psychology*, 35:4–28.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proc. ICML Deep Learning Workshop*.

Rob Voigt, David Jurgens, Vinodkumar Prabhakaran, Dan Jurafsky, and Yulia Tsvetkov. 2018. RtGender: A corpus for studying differential responses to gender. In *Proc. LREC*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proc. EACL*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proc. EMNLP*, pages 274–285.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. ICML*, pages 2048–2057.

Generating Fine-Grained Open Vocabulary Entity Type Descriptions

Rajarshi Bhowmik and **Gerard de Melo**
Department of Computer Science
Rutgers University – New Brunswick
Piscataway, NJ, USA
`{rajarshi.bhowmik, gerard.demelo}@cs.rutgers.edu`

Abstract

While large-scale knowledge graphs provide vast amounts of structured facts about entities, a short textual description can often be useful to succinctly characterize an entity and its type. Unfortunately, many knowledge graph entities lack such textual descriptions. In this paper, we introduce a dynamic memory-based network that generates a short open vocabulary description of an entity by jointly leveraging induced fact embeddings as well as the dynamic context of the generated sequence of words. We demonstrate the ability of our architecture to discern relevant information for more accurate generation of type description by pitting the system against several strong baselines.

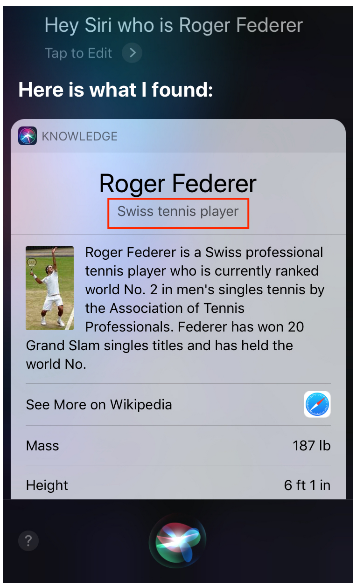

Hey Siri who is Roger Federer
Tap to Edit
Here is what I found:
KNOWLEDGE
Roger Federer
Swiss tennis player
Roger Federer is a Swiss professional tennis player who is currently ranked world No. 2 in men's singles tennis by the Association of Tennis Professionals. Federer has won 20 Grand Slam singles titles and has held the world No.
See More on Wikipedia
Mass 187 lb
Height 6 ft 1 in

Figure 1: A motivating example question that demonstrates the importance of short textual descriptions.

1 Introduction

Broad-coverage knowledge graphs such as Freebase, Wikidata, and NELL are increasingly being used in many NLP and AI tasks. For instance, DBpedia and YAGO were vital for IBM's Watson! Jeopardy system (Welty et al., 2012). Google's Knowledge Graph is tightly integrated into its search engine, yielding improved responses for entity queries as well as for question answering. In a similar effort, Apple Inc. is building an in-house knowledge graph to power Siri and its next generation of intelligent products and services.

Despite being rich sources of factual knowledge, cross-domain knowledge graphs often lack a succinct textual description for many of the existing entities. Fig. 1 depicts an example of a concise entity description presented to a user. Descriptions of this sort can be beneficial both to humans and in downstream AI and natural language processing tasks, including question answering (e.g., *Who is Roger Federer?*), named entity disambiguation (e.g., *Philadelphia* as a city vs. the film or even the brand of cream cheese), and information retrieval, to name but a few.

Additionally, descriptions of this sort can also be useful to determine the ontological type of an entity – another challenging task that often needs to be addressed in cross-domain knowledge graphs. Many knowledge graphs already provide ontological type information, and there has been substantial previous research on how to predict such types automatically for entities in knowledge graphs (Neelakantan and Chang, 2015; Miao et al., 2016; Kejriwal and Szekely, 2017), in semi-structured resources such as Wikipedia (Ponzetto and Strube, 2007; de Melo and Weikum, 2010), or even in unstructured text (Snow et al., 2006; Bansal et al., 2014; Tandon et al., 2015). However, most such work has targeted a fixed inventory of types from a given target ontology, many of which are more abstract in nature (e.g., *human* or *artifact*). In this work, we consider the task of generating more detailed open vocabulary descriptions (e.g., *Swiss tennis player*) that can readily be presented to end users, generated from facts in the knowledge graph.

Apart from type descriptions, certain knowledge graphs, such as Freebase and DBpedia, also provide a paragraph-length textual abstract for every entity. In the latter case, these are sourced from Wikipedia. There has also been research on generating such abstracts automatically (Biran and McKeown, 2017). While abstracts of this sort provide considerably more detail than ontological types, they are not sufficiently concise to be grasped at a single glance, and thus the onus is put on the reader to comprehend and summarize them.

Typically, a short description of an entity will hence need to be synthesized just by drawing on certain most relevant facts about it. While in many circumstances, humans tend to categorize entities at a level of abstraction commonly referred to as basic level categories (Rosch et al., 1976), in an information seeking setting, however, such as in Fig. 1, humans naturally expect more detail from their interlocutor. For example, *occupation* and *nationality* are often the two most relevant properties used in describing a person in Wikidata, while terms such as *person* or *human being* are likely to be perceived as overly unspecific. However, choosing such most relevant and distinctive attributes from the set of available facts about the entity is non-trivial, especially given the diversity of different kinds of entities in broad-coverage knowledge graphs. Moreover, the generated text should be coherent, succinct, and non-redundant.

To address this problem, we propose a dynamic memory-based generative network that can generate short textual descriptions from the available factual information about the entities. To the best of our knowledge, we are the first to present neural methods to tackle this problem. Previous work has suggested generating short descriptions using predefined templates (cf. Section 4). However, this approach severely restricts the expressivity of the model and hence such templates are typically only applied to very narrow classes of entities. In contrast, our goal is to design a broad-coverage open domain description generation architecture.

In our experiments, we induce a new benchmark dataset for this task by relying on Wikidata, which has recently emerged as the most popular crowdsourced knowledge base, following Google's designation of Wikidata as the successor to Freebase (Tanon et al., 2016). With a broad base of 19,000 casual Web users as contributors, Wikidata is a crucial source of machine-readable knowledge in many applications. Unlike DBpedia and Freebase, Wikidata usually contains a very concise description for many of its entities. However, because Wikidata is based on user contributions, many new entries are created that still lack such descriptions. This can be a problem for downstream tools and applications using Wikidata for background knowledge. Hence, even for Wikidata, there is a need for tools to generate fine-grained type descriptions. Fortunately, we can rely on the entities for which users have already contributed short descriptions to induce a new benchmark dataset for the task of automatically inducing type descriptions from structured data.

2 A Dynamic Memory-based Generative Network Architecture

Our proposed dynamic memory-based generative network consists of three key components: an input module, a dynamic memory module, and an output module. A schematic diagram of these are given in Fig. 2.

2.1 Input Module

The input to the input module is a set of N facts $F = \{f_1, f_2, \ldots, f_N\}$ pertaining to an entity. Each of these input facts are essentially (s, p, o) triples, for subjects s, predicates p, and objects o. Upon being encoded into a distributed vector representation, we refer to them as *fact embeddings*.

Although many different encoding schemes can be adopted to obtain such fact embeddings, we opt for a positional encoding as described by Sukhbaatar et al. (2015), motivated in part by the considerations given by Xiong et al. (2016). For completeness, we describe the positional encoding scheme here.

We encode each fact f_i as a vector $\mathbf{f_i} = \sum_{j=1}^{J} \mathbf{l_j} \circ \mathbf{w_j^i}$, where $\circ$ is an element-wise multiplication, and $\mathbf{l_j}$ is a column vector with the structure $l_{kj} = (1 - \frac{j}{J}) - (k/d)(1 - 2\frac{j}{J})$, with J being the number of words in the factual phrase, $\mathbf{w_j^i}$ as the embedding of the j-th word, and d as the dimensionality of the embedding. Details about how these factual phrases are formed for our data are

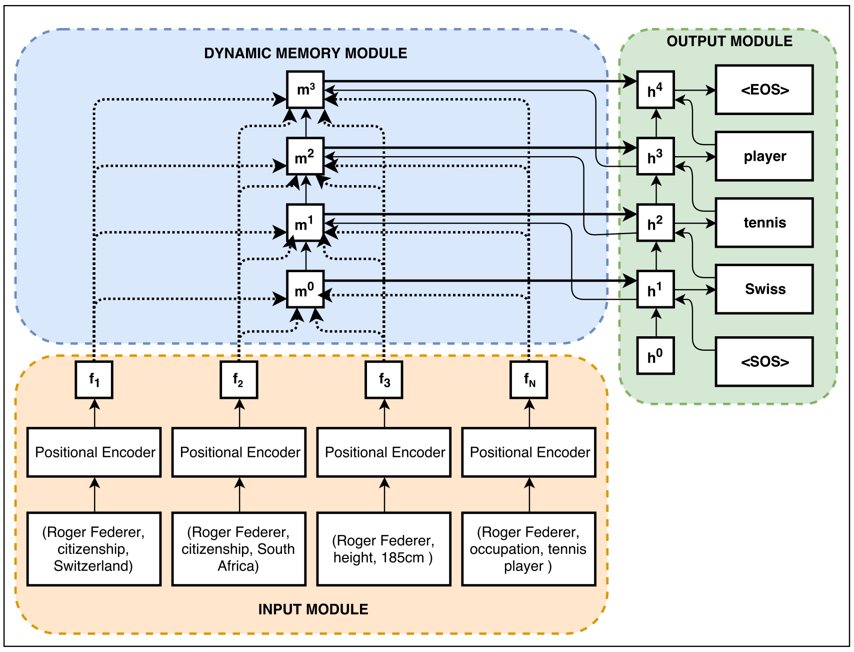

DYNAMIC MEMORY MODULE
OUTPUT MODULE
INPUT MODULE
m³
m²
m¹
m⁰
h⁴
h³
h²
h¹
h⁰
<EOS>
player
tennis
Swiss
<SOS>
f₁
f₂
f₃
f_N
Positional Encoder
(Roger Federer, citizenship, Switzerland)
(Roger Federer, citizenship, South Africa)
(Roger Federer, height, 185cm)
(Roger Federer, occupation, tennis player)

Figure 2: Model architecture.

given in Section 3.3.

Thus, the output of this module is a concatenation of N fact embeddings $\mathbf{F} = [\mathbf{f_1}; \mathbf{f_2}; \ldots; \mathbf{f_N}]$.

2.2 Dynamic Memory Module

The dynamic memory module is responsible for memorizing specific facts about an entity that will be useful for generating the next word in the output description sequence. Intuitively, such a memory should be able to update itself dynamically by accounting not only for the factual embeddings but also for the current context of the generated sequence of words.

To begin with, the memory is initialized as $\mathbf{m^{(0)}} = \max(\mathbf{0}, \mathbf{W_m F} + \mathbf{b_m})$. At each time step t, the memory module attempts to gather pertinent contextual information by attending to and summing over the fact embeddings in a weighted manner. These attention weights are scalar values informed by two factors: (1) how much information from a particular fact is used by the previous memory state $\mathbf{m}^{(t-1)}$, and (2) how much information of a particular fact is invoked in the current context of the output sequence $\mathbf{h}^{(t-1)}$. Formally,

$$\mathbf{x_i}^{(t)} = [|\mathbf{f_i} - \mathbf{h}^{(t-1)}|; |\mathbf{f_i} - \mathbf{m}^{(t-1)}|], \quad (1)$$

$$\mathbf{z_i}^{(t)} = \mathbf{W_2} \tanh(\mathbf{W_1 x_i}^{(t)} + \mathbf{b_1}) + \mathbf{b_2}, \quad (2)$$

$$a_i^{(t)} = \frac{\exp(\mathbf{z_i}^{(t)})}{\sum_{k=1}^{N} \exp(\mathbf{z_k}^{(t)})}, \quad (3)$$

where $|.|$ is the element-wise absolute difference and $[;]$ denotes the concatenation of vectors.

Having obtained the attention weights, we apply a soft attention mechanism to extract the current context vector at time t as

$$\mathbf{c}^{(t)} = \sum_{i=1}^{N} a_i^{(t)} \mathbf{f_i}. \quad (4)$$

This newly obtained context information is then used along with the previous memory state to update the memory state as follows:

$$\mathbf{C}^{(t)} = [\mathbf{m}^{(t-1)}; \mathbf{c}^{(t)}; \mathbf{h}^{(t-1)}] \quad (5)$$

$$\mathbf{m}^{(t)} = \max(\mathbf{0}, \mathbf{W_m C}^{(t)} + \mathbf{b_m}) \quad (6)$$

Such updated memory states serve as the input to the decoder sequence of the output module at each time step.

2.3 Output Module

The output module governs the process of repeatedly decoding the current memory state so as to emit the next word in an ordered sequence of output words. We rely on GRUs for this.

At each time step, the decoder GRU is presented as input a glimpse of the current memory state $\mathbf{m}^{(t)}$ as well as the previous context of the output sequence, i.e., the previous hidden state of the decoder $\mathbf{h}^{(t-1)}$. At each step, the resulting output of the GRU is concatenated with the context vector $\mathbf{c_i}^{(t)}$ and is passed through a fully connected layer and finally through a softmax layer. During training, we deploy *teacher forcing* at each step by providing the vector embedding of the previous correct word in the sequence as an additional input. During testing, when such a signal is not available, we use the embedding of the predicted word in the previous step as an additional input to the current step. Formally,

$$\mathbf{h}^{(t)} = \mathrm{GRU}([\mathbf{m}^{(t)}; \mathbf{w}^{(t-1)}], \mathbf{h}^{(t-1)}), \quad (7)$$

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{W_d}[\mathbf{h}^{(t)}; \mathbf{c}^{(t)}] + \mathbf{b_d}), \quad (8)$$

$$\hat{\mathbf{y}}^{(t)} = \mathrm{Softmax}(\mathbf{W_o}\tilde{\mathbf{h}}^{(t)} + \mathbf{b_o}), \quad (9)$$

where $[;]$ is the concatenation operator, $\mathbf{w}^{(t-1)}$ is vector embedding of the previous word in the sequence, and $\hat{\mathbf{y}}^{(t)}$ is the probability distribution for the predicted word over the vocabulary at the current step.

2.4 Loss Function and Training

Training this model amounts to picking suitable values for the model parameters θ, which include the matrices $\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{W_m}$, $\mathbf{W_d}$, $\mathbf{W_o}$ and the corresponding bias terms $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_m}$, $\mathbf{b_d}$, and $\mathbf{b_o}$ as well as the various transition and output matrices of the GRU.

To this end, if each of the training instances has a description with a maximum of M words, we can rely on the categorical cross-entropy over the entire output sequence as the loss function:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{M}\sum_{j=1}^{|\mathcal{V}|} y_j^{(t)} \log(\hat{y}_j^{(t)}). \quad (10)$$

where $y_j^{(t)} \in \{0, 1\}$ and $|\mathcal{V}|$ is the vocabulary size.

We train our model end-to-end using Adam as the optimization technique.

3 Evaluation

In this section, we describe the process of creating our benchmark dataset as well as the baseline methods and the experimental results.

3.1 Benchmark Dataset Creation

For the evaluation of our method, we introduce a novel benchmark dataset that we have extracted from Wikidata and transformed to a suitable format. We rely on the official RDF exports of Wikidata, which are generated regularly (Erxleben et al., 2014), specifically, the RDF dump dated 2016-08-01, which consists of 19,768,780 entities with 2,570 distinct properties. A pair of a property and its corresponding value represents a fact about an entity. In Wikidata parlance, such facts are called *statements*. We sample a dataset of 10K entities from Wikidata, and henceforth refer to the resulting dataset as WikiFacts10K. Our sampling method ensures that each entity in WikiFacts10K has an English description and at least 5 associated statements. We then transform each extracted statement into a phrasal form by concatenating the words of the property name and its value. For example, the (subject, predicate, object) triple (*Roger Federer*, *occupation*, *tennis player*) is transformed to *'occupation tennis player'*. We refer to these phrases as the *factual phrases*, which are embedded as described earlier. We randomly divide this dataset into training, validation, and test sets with a 8:1:1 ratio. We have made our code and data available[1] for reproducibility and to facilitate further research in this area.

3.2 Baselines

We compare our model against an array of baselines of varying complexity. We experiment with some variants of our model as well as several other state-of-the-art models that, although not specifically designed for this setting, can straightforwardly be applied to the task of generating descriptions from factual data.

1. **Facts-to-sequence Encoder-Decoder Model.** This model is a variant of the standard sequence-to-sequence encoder-decoder architecture described by Sutskever et al. (2014). However, instead of an input sequence, it here operates on a set of fact embeddings $\{\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_N}\}$, which are emitted by the positional encoder described in Section 2.1. We initialize the hidden state of the decoder with a linear transformation of the fact embeddings as $\mathbf{h}^{(0)} = \mathbf{WF} + \mathbf{b}$, where $\mathbf{F} = [\mathbf{f_1}; \mathbf{f_2}; \ldots; \mathbf{f_N}]$ is the concatenation of N fact embeddings.

   As an alternative, we also experimented with a sequence encoder that takes a separate fact embedding as input at each step and initializes the decoder hidden state with the final hidden state of the encoder. However, this approach did not yield us better results.

[1] https://github.com/kingsaint/Open-vocabulary-entity-type-description

Table 1: Automatic evaluation results of different models. For a detailed explanation of the baseline models, please refer to Section 3.2. The best performing model for each column is highlighted in boldface.

Model	B-1	B-2	B-3	B-4	ROUGE-L	METEOR	CIDEr
Facts-to-seq	0.404	0.324	0.274	0.242	0.433	0.214	1.627
Facts-to-seq w. Attention	0.491	0.414	0.366	0.335	0.512	0.257	2.207
Static Memory	0.374	0.298	0.255	0.223	0.383	0.185	1.328
DMN+	0.281	0.234	0.236	0.234	0.275	0.139	0.912
Our Model	**0.611**	**0.535**	**0.485**	**0.461**	**0.641**	**0.353**	**3.295**

2. **Facts-to-sequence Model with Attention Decoder.** The encoder of this model is identical to the one described above. The difference is in the decoder module that uses an attention mechanism.

   At each time step t, the decoder GRU receives a context vector $\mathbf{c}^{(t)}$ as input, which is an attention weighted sum of the fact embeddings. The attention weights and the context vectors are computed as follows:

$$\mathbf{x}^{(t)} = [\mathbf{w}^{(t-1)}; \mathbf{h}^{(t-1)}] \quad (11)$$

$$\mathbf{z}^{(t)} = \mathbf{W}\mathbf{x}^{(t)} + \mathbf{b} \quad (12)$$

$$\mathbf{a}^{(t)} = \mathrm{softmax}(\mathbf{z}^{(t)}) \quad (13)$$

$$\mathbf{c}^{(t)} = \max(\mathbf{0}, \sum_{i=1}^{N} a_i^{(t)} \mathbf{f}_\mathbf{i}) \quad (14)$$

   After obtaining the context vector, it is fed to the GRU as input:

$$\mathbf{h}^{(t)} = \mathrm{GRU}([\mathbf{w}^{(t-1)}; \mathbf{c}^{(t)}], \mathbf{h}^{(t-1)}) \quad (15)$$

3. **Static Memory Model.** This is a variant of our model in which we do not upgrade the memory dynamically at each time step. Rather, we use the initial memory state as the input to all of the decoder GRU steps.

4. **Dynamic Memory Network (DMN+).** We consider the approach proposed by Xiong et al. (2016), which supersedes Kumar et al. (2016). However, some minor modifications are needed to adapt it to our task. Unlike the bAbI dataset, our task does not involve any question. The presence of a question is imperative in DMN+, as it helps to determine the initial state of the episodic memory module. Thus, we prepend an interrogative phrase such as *"Who is"* or *"What is"* to every entity name. The question module of the DMN+ is hence presented with a question such as *"Who is Roger Federer?"* or *"What is Star Wars?"*. Another difference is in the output module. In DMN+, the final memory state is passed through a softmax layer to generate the answer. Since most answers in the bAbI dataset are unigrams, such an approach suffices. However, as our task is to generate a sequence of words as descriptions, we use a GRU-based decoder sequence model, which at each time step receives the final memory state $\mathbf{m}^{(T)}$ as input to the GRU. We restrict the number of memory update episodes to 3, which is also the preferred number of episodes in the original paper.

3.3 Experimental Setup

For each entity in the WikiFacts10K dataset, there is a corresponding set of facts expressed as factual phrases as defined earlier. Each factual phrase in turn is encoded as a vector by means of the positional encoding scheme described in Section 2.1. Although other variants could be considered, such as LSTMs and GRUs, we apply this standard fact encoding mechanism for our model as well as all our baselines for the sake of uniformity and fair comparison. Another factor that makes the use of a sequence encoder such as LSTMs or GRUs less suitable is that the set of input facts is essentially unordered without any temporal correlation between facts.

We fixed the dimensionality of the fact embeddings and all hidden states to be 100. The vocabulary size is 29K. Our models and all other baselines are trained for a maximum of 25 epochs with an early stopping criterion and a fixed learning rate of 0.001.

To evaluate the quality of the generated descriptions, we rely on the standard BLEU (B-1, B-2, B-3, B-4), ROUGE-L, METEOR and CIDEr metrics, as implemented by Sharma et al. (2017). Of course, we would be remiss not to point out that these metrics are imperfect. In general, they tend to be conservative in that they only reward generated descriptions that overlap substantially with the ground truth descriptions given in Wikidata. In reality, it may of course be the case that alternative descriptions are equally appropriate. In fact, inspecting the generated descriptions, we found that our method often indeed generates correct alternative descriptions. For instance, Darius Kaiser is described as a *cyclist*, but one could also describe him as a *German bicycle racer*. Despite their shortcomings, the aforementioned metrics have generally been found suitable for comparing supervised systems, in that systems with significantly higher scores tend to fare better at learning to reproduce ground truth captions.

3.4 Results

The results of the experiments are reported in Table 1. Across all metrics, we observe that our model obtains significantly better scores than the alternatives.

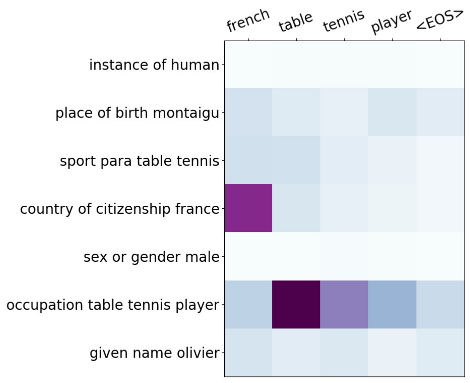
A facts-to-seq model exploiting our positional fact encoding performs adequately. With an additional attention mechanism (Facts-to-seq w. Attention), the results are even better. This is on account of the attention mechanism's ability to reconsider the attention distribution at each time step using the current context of the output sequence. The results suggest that this enables the model to more flexibly focus on the most pertinent parts of the input. In this regard, such a model thus resembles our approach. However, there are important differences between this baseline and our model. Our model not only uses the current context of the output sequence, but also memorizes how information of a particular fact has been used thus far, via the dynamic memory module. We conjecture that the dynamic memory module thereby facilitates generating longer description sequences more accurately by better tracking which parts have been attended to, as is empirically corroborated by the comparably higher BLEU scores for longer n-grams.

The analysis of the Static Memory approach amounts to an ablation study, as it only differs from our full model in lacking memory updates. The divergence of scores between the two variants suggests that the dynamic memory indeed is vital for more dynamically attending to the facts by taking into account the current context of the output sequence at each step. Our model needs to dynamically achieve different objectives at different time points. For instance, it may start off looking at several properties to infer a type of the appropriate granularity for the entity (e.g., *village*), while in the following steps it considers a salient property and emits the corresponding named entity for it as well as a suitable preposition (e.g., *in China*).

Finally, the poor results of the DMN+ approach show that a naïve application of a state-of-the-art dynamic memory architecture does not suffice to obtain strong results on this task. Indeed, the DMN+ is even outperformed by our Facts-to-seq baseline. This appears to stem from the inability of the model to properly memorize all pertinent facts in its encoder.

Analysis. In Figure 3, we visualize the attention distribution over facts. We observe how the model shifts its focus to different sorts of properties while generating successive words.

Table 2 provides a representative sample of the generated descriptions and their ground truth counterparts. A manual inspection reveals five distinct patterns. The first case is that of exact matches with the reference descriptions. The second involves examples on which there is a high overlap of words between the ground truth and generated descriptions, but the latter as a whole is incorrect because of semantic drift or other challenges. In some cases, the model may have never seen a word or named entity during training (e.g., *Hypocrisy*), or their frequency is very limited in the training set. While it has been shown that GRUs with an attention mechanism are capable of learning to copy random strings from the input (Gu et al., 2016), we conjecture that a dedicated copy mechanism may help to mitigate this problem, which we will explore in future research. In other cases, the model conflates semantically related concepts, as is evident from examples such as a *film* being described as a *filmmaker* and a *polo player* as a *water polo player*. Next, the third group involves generated descriptions that are more specific than the ground truth, but correct, while, in the fourth group, the generated outputs generalize the descriptions to a certain extent. For example, *American musician and pianist* is generalized as *American musician*, since *musician* is a hypernym of *pianist*. Finally, the last group consists of cases in which our model generated descriptions that are factually accurate and may be deemed appropriate despite diverging from the



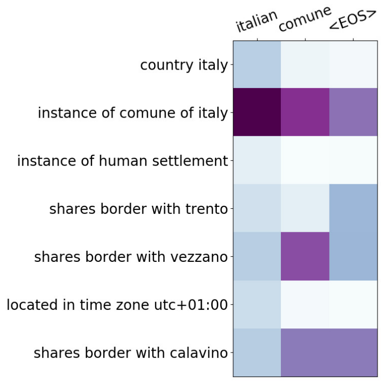


Figure 3: An example of attention distribution over the facts while emitting words. The *country of citizenship* property gets the most attention while generating the first word *French* of the left description. For generating the next three words, the fact *occupation* attracts the most attention. Similarly, *instance of* attracts the most attention when generating the sequence *Italian comune*.

Table 2: A representative sample of the generated descriptions and its comparison with the ground truth descriptions.

	Item	**Ground Truth Description**	**Generated Description**
Matches	Q20538915	painting by Claude Monet	painting by Claude Monet
	Q10592904	genus of fungi	genus of fungi
	Q669081	municipality in Austria	municipality in Austria
	Q23588047	microbial protein found in Mycobacterium abscessus	microbial protein found in Mycobacterium abscessus
Semantic drift	Q1777131	album by Hypocrisy	album by Mandy Moore
	Q16164685	polo player	water polo player
	Q849834	class of 46 electric locomotives	class of 20 british 0-6-0t locomotives
	Q1434610	1928 film	filmmaker
More specific	Q1865706	footballer	Finnish footballer
	Q19261036	number	natural number
	Q7807066	cricketer	English cricketer
	Q10311160	Brazilian lawyer	Brazilian lawyer and politician
More general	Q149658	main-belt asteroid	asteroid
	Q448330	American musician and pianist	American musician
	Q4801958	2011 Hindi film	Indian film
	Q7815530	South Carolina politician	American politician
Alternative	Q7364988	Dean of York	British academic
	Q1165984	cyclist	German bicycle racer
	Q6179770	recipient of the knight's cross	German general
	Q17660616	singer-songwriter	Canadian musician

reference descriptions to an extent that almost no overlapping words are shared with them. Note that such outputs are heavily penalized by the metrics considered in our evaluation.

4 Related Work

Type Prediction. There has been extensive work on predicting the ontological types of entities in large knowledge graphs (Neelakantan and Chang, 2015; Miao et al., 2016; Kejriwal and Szekely, 2017; Shimaoka et al., 2017), in semi-structured resources such as Wikipedia (Ponzetto and Strube, 2007; de Melo and Weikum, 2010), as well as in text (Del Corro et al., 2015; Yaghoobzadeh and Schütze, 2015; Ren et al., 2016). However, the major shortcoming of these sorts of methods, including those aiming at more fine-grained typing, is that they assume that the set of candidate types is given as input, and the main remaining challenge is to pick the correct one(s). In contrast, our work yields descriptions that often indicate the type of entity, but typically are more natural-sounding and descriptive (e.g. *French Impressionist artist*) than the oftentimes abstract ontological types (such as *human* or *artifact*) chosen by type prediction methods.

A separate, long-running series of work has obtained open vocabulary type predictions for named entities and concepts mentioned in text (Hearst, 1992; Snow et al., 2006), possibly also inducing taxonomies from them (Poon and Domingos, 2010; Velardi et al., 2013; Bansal et al., 2014). However, these methods typically just need to select existing spans of text from the input as the output description.

Text Generation from Structured Data. Research on methods to generate descriptions for entities has remained scant. Lebret et al. (2016) take Wikipedia infobox data as input and train a custom form of neural language model that, conditioned on occurrences of words in the input table, generates biographical sentences as output. However, their system is limited to a single kind of description (biographical sentences) that tend to share a common structure. Wang et al. (2016) focus on the problem of temporal ordering of extracted facts. Biran and McKeown (2017) introduced a template-based description generation framework for creating hybrid concept-to-text and text-to-text generation systems that produce descriptions of RDF entities. Their framework can be tuned for new domains, but does not yield a broad-coverage multi-domain model. Voskarides et al. (2017) first create sentence templates for specific entity relationships, and then, given a new relationship instance, generate a description by selecting the best template and filling the template slots with the appropriate entities from the knowledge graph. Kutlak et al. (2013) generates referring expressions by converting property-value pairs to text using a hand-crafted mapping scheme. Wiseman et al. (2017) considered the related task of mapping tables with numeric basketball statistics to natural language. They investigated an extensive array of current state-of-the-art neural pointer methods but found that template-based models outperform all neural models on this task by a significant margin. However, their method requires specific templates for each domain (for example, basketball games in their case). Applying template-based methods to cross-domain knowledge bases is highly challenging, as this would require too many different templates for different types of entities. Our dataset contains items of from a large number of diverse domains such as humans, books, films, paintings, music albums, genes, proteins, cities, scientific articles, etc., to name but a few.

Chen and Mooney (2008) studied the task of taking representations of observations from a sports simulation (Robocup simulator) as input, e.g. *pass(arg1=purple6, arg2=purple3)*, and generating game commentary. Liang et al. (2009) learned alignments between formal descriptions such as *rainChance(time=26-30,mode=Def)* and natural language weather reports. Mei et al. (2016) used LSTMs for these sorts of generation tasks, via a custom coarse-to-fine architecture that first determines which input parts to focus on.

Much of the aforementioned work essentially involves aligning small snippets in the input to the relevant parts in the training output and then learning to expand such input snippets into full sentences. In contrast, in our task, alignments between parts of the input and the output do not suffice. Instead, describing an entity often also involves considering all available evidence about that entity to infer information about it that is often not immediately given. Rather than verbalizing facts, our method needs a complex attention mechanism to predict an object's general type and consider the information that is most likely to appear salient to humans from across the entire input.

The WebNLG Challenge (Gardent et al., 2017) is another task for generating text from structured data. However, this task requires a textual verbalization of every triple. On the contrary, the task we consider in this work is quite complementary in that a verbalization of all facts one-by-one is not the sought result. Rather, our task requires synthesizing a short description by carefully selecting the most relevant and distinctive facts from the set of all available facts about the entity. Due to these differences, the WebNLG dataset was not suitable for the research question considered by our paper.

Neural Text Summarization. Generating entity descriptions is related to the task of text summarization. Most traditional work in this area was extractive in nature, i.e. it selects the most salient sentences from a given input text and concatenates them to form a shorter summary or presents them differently to the user (Yang et al., 2017). Abstractive summarization goes beyond this in generating new text not necessarily encountered in the input, as is typically necessary in our setting. The surge of sequence-to-sequence modeling of text via LSTMs naturally extends to the task of abstractive summarization by training a model to accept a longer sequence as input and learning to generate a shorter compressed sequence as a summary.

Rush et al. (2015) employed this idea to generate a short headline from the first sentence of a text. Subsequent work investigated the use of architectures such as pointer-generator networks to better cope with long input texts (See et al., 2017). Recently, Liu et al. (2018) presented a model that generates an entire Wikipedia article via a neural decoder component that performs abstractive summarization of multiple source documents. Our work differs from such previous work in that we do not consider a text sequence as input. Rather, our input are a series of entity relationships or properties, as reflected by our facts-to-sequence baselines in the experiments. Note that our task is in certain respects also more difficult than text summarization. While regular neural summarizers are often able to identify salient spans of text that can be copied to the output, our input is of a substantially different form than the desired output.

Additionally, our goal is to make our method applicable to any entity with factual information that may not have a corresponding Wikipedia-like article available. Indeed, Wikidata currently has 46 million items, whereas the English Wikipedia has only 5.6 million articles. Hence, for the vast majority of items in Wikidata, no corresponding Wikipedia article is available. In such cases, a summarization baseline will not be effective.

Episodic Memory Architectures. A number of neural models have been put forth that possess the ability to interact with a memory component. Recent advances in neural architectures that combine memory components with an attention mechanism exhibit the ability to extract and reason over factual information. A well-known example is the End-To-End Memory Network model by Sukhbaatar et al. (2015), which may make multiple passes over the memory input to facilitate multi-hop reasoning. These have been particularly successful on the bAbI test suite of artificial comprehension tests (Weston et al., 2015), due to their ability to extract and reason over the input.

At the core of the Dynamic Memory Networks (DMN) architecture (Kumar et al., 2016) is an episodic memory module, which is updated at each episode with new information that is required to answer a predefined question. Our approach shares several commonalities with DMNs, as it is also endowed with a dynamic memory of this sort. However, there are also a number of significant differences. First of all, DMN and its improved version DMN+ (Xiong et al., 2016) assume sequential correlations between the sentences and rely on them for reasoning purposes. To this end, DMN+ needs an additional layer of GRUs, which is used to capture sequential correlations among sentences. Our model does not need any such layer, as facts in a knowledge graph do not necessarily possess any sequential interconnections. Additionally, DMNs assume a predefined number of memory episodes, with the final memory state being passed to the answer module. Unlike DMNs, our model uses the dynamic context of the output sequence to update the memory state. The number of memory updates in our model flexibly depends on the length of the generated sequence. DMNs also have an additional question module as input, which guides the memory updates and also the output, while our model does not leverage any such guiding factor. Finally, in DMNs, the output is typically a unigram, whereas our model emits a sequence of words.

5 Conclusion

Short textual descriptions of entities facilitate instantaneous grasping of key information about entities and their types. Generating them from facts in a knowledge graph requires not only mapping the structured fact information to natural language, but also identifying the type of entity and then discerning the most crucial pieces of information for that particular type from the long list of input facts and compressing them down to a highly succinct form. This is very challenging in light of the very heterogeneous kinds of entities in our data.

To this end, we have introduced a novel dynamic memory-based neural architecture that updates its memory at each step to continually reassess the relevance of potential input signals. We have shown that our approach outperforms several competitive baselines. In future work, we hope to explore the potential of this architecture on further kinds of data, including multimodal data (Long et al., 2018), from which one can extract structured signals. Our code and data is freely available.[2]

Acknowledgments

This research is funded in part by ARO grant no. W911NF-17-C-0098 as part of the DARPA SocialSim program.

[2]https://github.com/kingsaint/Open-vocabulary-entity-type-description

References

Mohit Bansal, David Burkett, Gerard de Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1041–1051. http://www.aclweb.org/anthology/P14-1098.

Or Biran and Kathleen McKeown. 2017. Domain-adaptable hybrid generation of RDF entity descriptions. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*. pages 306–315. https://aclanthology.info/papers/I17-1031/i17-1031.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '08, pages 128–135. https://doi.org/10.1145/1390156.1390173.

Gerard de Melo and Gerhard Weikum. 2010. MENTA: Inducing multilingual taxonomies from Wikipedia. In Jimmy Huang, Nick Koudas, Gareth Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010)*. ACM, New York, NY, USA, pages 1099–1108.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. FINET: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 868–878. http://aclweb.org/anthology/D15-1103.

Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. 2014. Introducing Wikidata to the Linked Data Web. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandečić, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *Proceedings of the 13th International Semantic Web Conference (ISWC'14)*. Springer, volume 8796 of *LNCS*, pages 50–65.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*. Association for Computational Linguistics, Santiago de Compostela, Spain, pages 124–133. http://www.aclweb.org/anthology/W17-3518.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1631–1640. http://www.aclweb.org/anthology/P16-1154.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

Mayank Kejriwal and Pedro Szekely. 2017. Supervised typing of big graphs using semantic embeddings. *CoRR* abs/1703.07805. http://arxiv.org/abs/1703.07805.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, New York, New York, USA, volume 48 of *Proceedings of Machine Learning Research*, pages 1378–1387. http://proceedings.mlr.press/v48/kumar16.html.

Roman Kutlak, Kees van Deemter, and Christopher Stuart Mellish. 2013. Generation of referring expressions in large domains.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Generating text from structured data with application to the biography domain. *CoRR* abs/1603.07771. http://arxiv.org/abs/1603.07771.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 91–99. http://dl.acm.org/citation.cfm?id=1687878.1687893.

Peter Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. *CoRR* abs/1801.10198. http://arxiv.org/abs/1801.10198.

Xiang Long, Chuang Gan, and Gerard de Melo. 2018. Video captioning with multi-faceted attention. *Transactions of the Association for Computational Linguistics (TACL)* 6:173–184. https://transacl.org/ojs/index.php/tacl/article/view/1289.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of NAACL*.

Qingliang Miao, Ruiyu Fang, Shuangyong Song, Zhongguang Zheng, Lu Fang, Yao Meng, and Jun Sun. 2016. Automatic identifying entity type in Linked Data. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation, PACLIC 30, Seoul, Korea, October 28 - October 30, 2016*. http://aclweb.org/anthology/Y/Y16/Y16-3009.pdf.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 515–525. http://www.aclweb.org/anthology/N15-1054.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*. AAAI Press, AAAI'07, pages 1440–1445. http://dl.acm.org/citation.cfm?id=1619797.1619876.

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 296–305. http://dl.acm.org/citation.cfm?id=1858681.1858712.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *EMNLP*.

Eleanor Rosch, Carolyn B. Mervis, Wayne D. Gray, David M. Johnson, and Penny Boyes-Braem. 1976. Basic objects in natural categories. *Cognitive Psychology* .

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1073–1083. https://doi.org/10.18653/v1/P17-1099.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR* abs/1706.09799. http://arxiv.org/abs/1706.09799.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1271–1280. http://www.aclweb.org/anthology/E17-1119.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 801–808. https://doi.org/10.3115/1220175.1220276.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2440–2448. http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Niket Tandon, Gerard de Melo, Abir De, and Gerhard Weikum. 2015. Knowlywood: Mining activity knowledge from Hollywood narratives. In *Proceedings of CIKM 2015*.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From Freebase to Wikidata: The great migration. In *World Wide Web Conference*.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics* 39(3):665–707. https://doi.org/10.1162/COLI_a_00146.

Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 2017. Generating descriptions of entity relationships. In *ECIR 2017: 39th European Conference on Information Retrieval*. Springer, LNCS.

Yafang Wang, Zhaochun Ren, Martin Theobald, Maximilian Dylla, and Gerard de Melo. 2016. Summary generation for temporal extractions. In *Proceedings of 27th International Conference on Database and Expert Systems Applications (DEXA 2016)*.

Chris Welty, J. William Murdock, Aditya Kalyanpur, and James Fan. 2012. A comparison of hard filters and soft evidence for answer typing in Watson. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2012*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 243–256.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR* abs/1502.05698. http://arxiv.org/abs/1502.05698.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2253–2263. https://www.aclweb.org/anthology/D17-1239.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, ICML'16, pages 2397–2406. http://dl.acm.org/citation.cfm?id=3045390.3045643.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 715–725. http://aclweb.org/anthology/D15-1083.

Qian Yang, Yong Cheng, Sen Wang, and Gerard de Melo. 2017. HiText: Text reading with dynamic salience marking. In *Proceedings of WWW 2017 (Digital Learning Track)*. ACM.

Hierarchical Neural Story Generation

Angela Fan **Mike Lewis** **Yann Dauphin**

Facebook AI Research, Menlo Park
{angelafan, mikelewis, ynd}@fb.com

Abstract

We explore *story generation*: creative systems that can build coherent and fluent passages of text about a topic. We collect a large dataset of 300K human-written stories paired with writing prompts from an online forum. Our dataset enables hierarchical story generation, where the model first generates a premise, and then transforms it into a passage of text. We gain further improvements with a novel form of model fusion that improves the relevance of the story to the prompt, and adding a new gated multi-scale self-attention mechanism to model long-range context. Experiments show large improvements over strong baselines on both automated and human evaluations. Human judges prefer stories generated by our approach to those from a strong non-hierarchical model by a factor of two to one.

1 Introduction

Story-telling is on the frontier of current text generation technology: stories must remain thematically consistent across the complete document, requiring modeling very long range dependencies; stories require creativity; and stories need a high level plot, necessitating planning ahead rather than word-by-word generation (Wiseman et al., 2017).

We tackle the challenges of story-telling with a hierarchical model, which first generates a sentence called the *prompt* describing the topic for the story, and then conditions on this prompt when generating the story. Conditioning on the prompt or premise makes it easier to generate consistent stories because they provide grounding for the overall plot. It also reduces the tendency of standard sequence models to drift off topic.

Prompt: The Mage, the Warrior, and the Priest

Story: A light breeze swept the ground, and carried with it still the distant scents of dust and time-worn stone. The Warrior led the way, heaving her mass of armour and muscle over the uneven terrain. She soon crested the last of the low embankments, which still bore the unmistakable fingerprints of haste and fear. She lifted herself up onto the top the rise, and looked out at the scene before her. [...]

Figure 1: Example prompt and beginning of a story from our dataset. We train a hierarchical model that first generates a prompt, and then conditions on the prompt when generating a story.

We find that standard sequence-to-sequence (seq2seq) models (Sutskever et al., 2014) applied to hierarchical story generation are prone to degenerating into language models that pay little attention to the writing prompt (a problem that has been noted in other domains, such as dialogue response generation (Li et al., 2015a)). This failure is due to the complex and underspecified dependencies between the prompt and the story, which are much harder to model than the closer dependencies required for language modeling (for example, consider the subtle relationship between the first sentence and prompt in Figure 1).

To improve the relevance of the generated story to its prompt, we introduce a fusion mechanism (Sriram et al., 2017) where our model is trained on top of an pre-trained seq2seq model. To improve over the pre-trained model, the second model must focus on the link between the prompt and the story. For the first time, we show that fusion mechanisms can help seq2seq models build dependencies between their input and output.

Another major challenge in story generation is the inefficiency of modeling long documents with standard recurrent architectures—stories contain 734 words on average in our dataset. We improve efficiency using a convolutional architecture, al-

# Train Stories	272,600
# Test Stories	15,138
# Validation Stories	15,620
# Prompt Words	7.7M
# Story Words	200M
Average Length of Prompts	28.4
Average Length of Stories	734.5

Table 1: Statistics of WritingPrompts dataset

lowing whole stories to be encoded in parallel. Existing convolutional architectures only encode a bounded amount of context (Dauphin et al., 2017), so we introduce a novel gated self-attention mechanism that allows the model to condition on its previous outputs at different time-scales.

To train our models, we gathered a large dataset of 303,358 human generated stories paired with writing prompts from an online forum. Evaluating free form text is challenging, so we also introduce new evaluation metrics which isolate different aspects of story generation.

Experiments show that our fusion and self-attention mechanisms improve over existing techniques on both automated and human evaluation measures. Our new dataset and neural architectures allow for models which can creatively generate longer, more consistent and more fluent passages of text. Human judges prefer our hierarchical model's stories twice as often as those of a non-hierarchical baseline.

2 Writing Prompts Dataset

We collect a hierarchical story generation dataset[1] from Reddit's WritingPrompts forum.[2] WritingPrompts is a community where online users inspire each other to write by submitting story premises, or prompts, and other users freely respond. Each prompt can have multiple story responses. The prompts have a large diversity of topic, length, and detail. The stories must be at least 30 words, avoid general profanity and inappropriate content, and should be inspired by the prompt (but do not necessarily have to fulfill every requirement). Figure 1 shows an example.

We scraped three years of prompts and their associated stories using the official Reddit API. We clean the dataset by removing automated bot posts, deleted posts, special announcements, comments from moderators, and stories shorter than 30 words. We use NLTK for tokenization. The dataset models full text to generate immediately human-readable stories. We reserve 5% of the prompts for a validation set and 5% for a test set, and present additional statistics about the dataset in Table 1.

For our experiments, we limit the length of the stories to 1000 words maximum and limit the vocabulary size for the prompts and the stories to words appearing more than 10 times each. We model an unknown word token and an end of document token. This leads to a vocabulary size of 19,025 for the prompts and 104,960 for the stories. As the dataset is scraped from an online forum, the number of rare words and misspellings is quite large, so modeling the full vocabulary is challenging and computationally intensive.

3 Approach

The challenges of WritingPrompts are primarily in modeling long-range dependencies and conditioning on an abstract, high-level prompt. Recurrent and convolutional networks have successfully modeled sentences (Jozefowicz et al., 2016; Dauphin et al., 2017), but accurately modeling several paragraphs is an open problem. While seq2seq networks have strong performance on a variety of problems, we find that they are unable to build stories that accurately reflect the prompts. We will evaluate strategies to address these challenges in the following sections.

3.1 Hierarchical Story Generation

High-level structure is integral to good stories, but language models generate on a strictly-word-by-word basis and so cannot explicitly make high-level plans. We introduce the ability to plan by decomposing the generation process into two levels. First, we generate the premise or prompt of the story using the convolutional language model from Dauphin et al. (2017). The prompt gives a sketch of the structure of the story. Second, we use a seq2seq model to generate a story that follows the premise. Conditioning on the prompt makes it easier for the story to remain consistent and also have structure at a level beyond single phrases.

[1] www.github.com/pytorch/fairseq
[2] www.reddit.com/r/WritingPrompts/

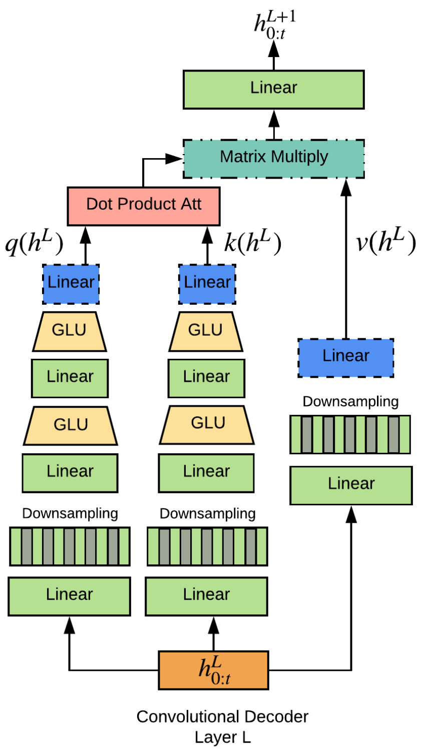
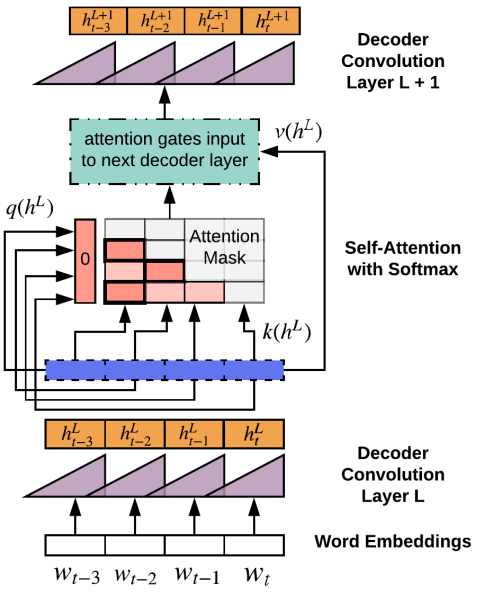


Figure 2: Self-Attention Mechanism of a single head, with GLU gating and downsampling. Multiple heads are concatenated, with each head using a separate downsampling function.



h^{L+1}_{t-3} h^{L+1}_{t-2} h^{L+1}_{t-1} h^{L+1}_{t}
Decoder Convolution Layer L + 1
attention gates input to next decoder layer
$v(h^L)$
$q(h^L)$
0
Attention Mask
Self-Attention with Softmax
$k(h^L)$
h^L_{t-3} h^L_{t-2} h^L_{t-1} h^L_t
Decoder Convolution Layer L
Word Embeddings
w_{t-3} w_{t-2} w_{t-1} w_t

Figure 3: Multihead self-attention mechanism. The decoder layer depicted attends with itself to gate the input of the subsequent decoder layer.

3.2 Efficient Learning with Convolutional Sequence-to-Sequence Model

The length of stories in our dataset is a challenge for RNNs, which process tokens sequentially. To transform prompts into stories, we instead build on the convolutional seq2seq model of Gehring et al. (2017), which uses deep convolutional networks as the encoder and decoder. Convolutional models are ideally suited to modeling long sequences, because they allow parallelism of computation within the sequence. In the Conv seq2seq model, the encoder and decoder are connected with attention modules (Bahdanau et al., 2015) that perform a weighted sum of encoder outputs, using attention at each layer of the decoder.

3.3 Modeling Unbounded Context with Gated Multi-Scale Self-attention

CNNs can only model a bounded context window, preventing the modeling of long-range dependencies within the output story. To enable modeling of unbounded context, we supplement the decoder with a self-attention mechanism (Sukhbaatar et al., 2015; Vaswani et al., 2017), which allows the model to refer to any previously generated words. The self-attention mechanism improves the model's ability to extract long-range context with limited computational impact due to parallelism.

Gated Attention: Similar to Vaswani et al. (2017), we use multi-head attention to allow each head to attend to information at different positions. However, the queries, keys and values are not given by linear projections but by more expressive gated deep neural nets with Gated Linear Unit (Dauphin et al., 2017) activations. We show that gating lends the self-attention mechanism crucial capacity to make fine-grained selections.

Multi-Scale Attention: Further, we propose to have each head operating at a different time scale, depicted in Figure 2. Thus the input to each head is *downsampled* a different amount—the first head sees the full input, the second every other input timestep, the third every third input timestep, etc. The different scales encourage the heads to attend to different information. The downsampling operation limits the number of tokens in the attention maps, making them sharper.

The output of a single attention head is given by

$$h^{L+1}_{0:t} = \text{Linear}\Big(v(h^L_{0:t-1}) \odot \text{softmax}(q(h^L_{0:t})k(h^L_{0:t})^\top)\Big) \quad (1)$$

where $h^L_{0:t}$ contains the hidden states up to time t

at layer L, and q, k, v are gated downsampling networks as shown in Figure 2. Unlike Vaswani et al. (2017), we allow the model to optionally attend to a 0 vector at each timestep, if it chooses to ignore the information of past timesteps (see Figure 3). This mechanism allows the model to recover the non-self-attention architecture and avoid attending to the past if it provides only noise. Additionally, we do not allow the self-attention mechanism to attend to the current timestep, only the past.

3.4 Improving Relevance to Input Prompt with Model Fusion

Unlike tasks such as translation, where the semantics of the target are fully specified by the source, the generation of stories from prompts is far more open-ended. We find that seq2seq models ignore the prompt and focus solely on modeling the stories, because the local dependencies required for language modeling are easier to model than the subtle dependencies between prompt and story.

We propose a fusion-based approach to encourage conditioning on the prompt. We train a seq2seq model that has access to the hidden states of a pretrained seq2seq model. Doing so can be seen as a type of boosting or residual learning that allows the second model to focus on what the first model failed to learn—such as conditioning on the prompt. To our knowledge, this paper is the first to show that fusion reduces the problem of seq2seq models degenerating into language models that capture primarily syntactic and grammatical information.

The cold fusion mechanism of Sriram et al. (2017) pretrains a language model and subsequently trains a seq2seq model with a gating mechanism that learns to leverage the final hidden layer of the language model during seq2seq training. We modify this approach by combining two seq2seq models as follows (see Figure 4):

$$g_t = \sigma(W[h_t^{\text{Training}}; h_t^{\text{Pretrained}}] + b)$$
$$h_t = g_t \circ [h_t^{\text{Training}}; h_t^{\text{Pretrained}}]$$

where the hidden state of the pretrained seq2seq model and training seq2seq model (represented by h_t) are concatenated to learn gates g_t. The gates are computed using a linear projection with the weight matrix W. The gated hidden layers are combined by concatenation and followed by more fully connected layers with GLU activations (see

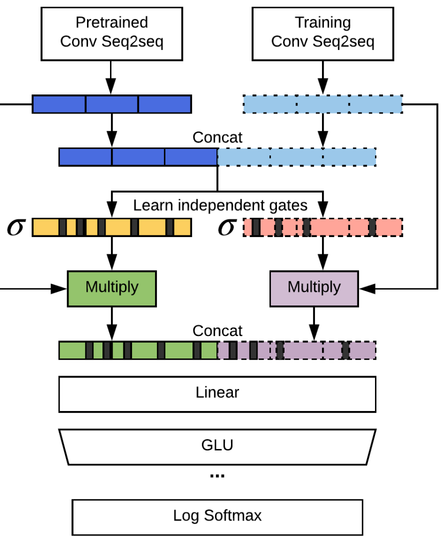


Figure 4: Diagram of our fusion model, which learns a second seq2seq model to improve a pretrained model. The separate hidden states are combined after gating through concatenation.

Appendix). We use layer normalization (Ba et al., 2016) after each fully connected layer.

4 Related Work

4.1 Story Generation

Sequence-to-sequence neural networks (Sutskever et al., 2014) have achieved state of the art performance on a variety of text generation tasks, such as machine translation (Sutskever et al., 2014) and summarization (Rush et al., 2015). Recent work has applied these models to more open-ended generation tasks, including writing Wikipedia articles (Liu et al., 2018) and poetry (Zhang and Lapata, 2014).

Previous work on story generation has explored seq2seq RNN architectures (Roemmele, 2016), but has focused largely on using various content to inspire the stories. For instance, Kiros et al. (2015) uses photos to inspire short paragraphs trained on romance novels, and Jain et al. (2017) chain a series of independent descriptions together into a short story. Martin et al. (2017) decompose story generation into two steps, first converting text into event representations, then modeling stories as sequences of events before translating back to natural language. Similarly, Harrison et al. (2017) generate summaries of movies as sequences of events using an RNN, then sample event representations using MCMC. They find this technique can generate text of the desired genre, but the movie plots are not interpretable (as the model outputs events, not raw text). However, we are not aware of previous work that has used hierarchical generation from a textual premise to improve the coherence and structure of stories.

4.2 Hierarchical Text Generation

Previous work has proposed decomposing the challenge of generating long sequences of text into a hierarchical generation task. For instance, Li et al. (2015b) use an LSTM to hierarchically learn word, then sentence, then paragraph embeddings, then transform the paragraph embeddings into text. Yarats and Lewis (2017) generate a discrete latent variable based on the context, then generates text conditioned upon it.

4.3 Fusion Models

Previous work has investigated the integration of language models with seq2seq models. The two models can be leveraged together without architectural modifications: Ramachandran et al. (2016) use language models to initialize the encoder and decoder side of the seq2seq model independently, and Chorowski and Jaitly (2016) combine the predictions of the language model and seq2seq model solely at inference time. Recent work has also proposed deeper integration. Gulcehre et al. (2015) combined a trained language model with a trained seq2seq model to learn a gating function that joins them. Sriram et al. (2017) propose training the seq2seq model given the fixed language model then learning a gate to filter the information from the language model.

5 Experimental Setup

5.1 Baselines

We evaluate a number of baselines:

(1) Language Models: Non-hierarchical models for story generation, which do not condition on the prompt. We use both the gated convolutional language (GCNN) model of Dauphin et al. (2017) and our additional self-attention mechanism.

(2) seq2seq: using LSTMs and convolutional seq2seq architectures, and Conv seq2seq with decoder self-attention.

(3) Ensemble: an ensemble of two Conv seq2seq with self-attention models.

(4) KNN: we also compare with a KNN model to find the closest prompt in the training set for each prompt in the test set. A TF-IDF vector for each prompt was created using FASTTEXT (Bojanowski et al., 2016) and FAISS (Johnson et al., 2017) was used for KNN search. The retrieved story from the training set is limited to 150 words to match the length of generated stories.

Model	**Valid Perplexity**	**Test Perplexity**
Conv seq2seq	45.27	45.54
+ self-attention	42.01	42.32
+ multihead	40.12	40.39
+ multiscale	38.76	38.91
+ gating	**37.37**	**37.94**

Table 2: Effect of new attention mechanism. Gated multi-scale attention significantly improves the perplexity on the WRITINGPROMPTS dataset.

5.2 Fusion Training

To train the fusion model, we first pretrain a Conv seq2seq with self-attention model on the WRITINGPROMPTS dataset. This pretrained model is fixed and provided to the second Conv seq2seq with self-attention model during training time. The two models are integrated with the fusion mechanism described in Section 3.4.

5.3 Training

We implement models with the fairseq-py library in PyTorch. Similar to Gehring et al. (2017), we train using the Nesterov accelerated gradient method (Sutskever et al., 2013) using gradient clipping (Pascanu et al., 2013). We perform hyperparameter optimization on each of our models by cross-validating with random search on a validation set. We provide model architectures in the appendix.

5.4 Generation

We generate stories from our models using a *top-k random sampling* scheme. At each timestep, the model generates the probability of each word in the vocabulary being the likely next word. We randomly sample from the $k = 10$ most likely candidates from this distribution. Then, subsequent timesteps generate words based on the previously selected words. We find this sampling strategy substantially more effective than beam search, which tends to produce common phrases and repetitive text from the training set (Vijayakumar et al., 2016; Shao et al., 2017). Sentences pro-

Model	# Parameters (mil)	Valid Perplexity	Test Perplexity
GCNN LM	123.4	54.50	54.79
GCNN + self-attention LM	126.4	51.84	51.18
LSTM seq2seq	110.3	46.83	46.79
Conv seq2seq	113.0	45.27	45.54
Conv seq2seq + self-attention	134.7	37.37	37.94
Ensemble: Conv seq2seq + self-attention	270.3	36.63	36.93
Fusion: Conv seq2seq + self-attention	255.4	**36.08**	**36.56**

Table 3: Perplexity on WRITINGPROMPTS. We dramatically improve over standard seq2seq models.

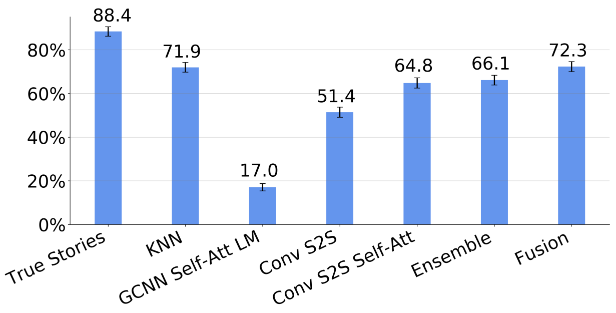


Figure 5: Human accuracy at pairing stories with the prompts used to generate them. People find that our fusion model significantly improves the link between the prompt and generated stories.

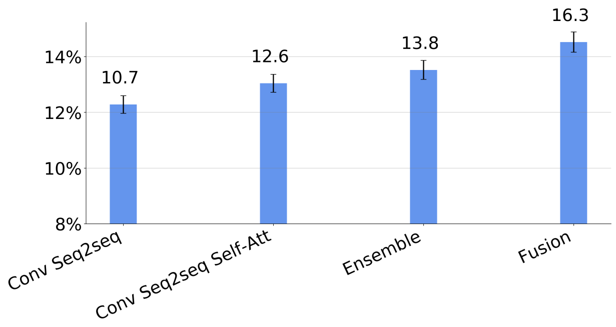


Figure 6: Accuracy of prompt ranking. The fusion model most accurately pairs prompt and stories.

duced by beam search tend to be short and generic. Completely random sampling can introduce very unlikely words, which can damage generation as the model has not seen such mistakes at training time. The restriction of sampling from the 10 most likely candidates reduces the risk of these low-probability samples.

For each model, we tune a temperature parameter for the softmax at generation time. To ease human evaluation, we generate stories of 150 words and do not generate unknown word tokens.

For prompt generation, we use a self-attentive GCNN language model trained with the same prompt-side vocabulary as the sequence-to-sequence story generation models. The language model to generate prompts has a validation perplexity of 63.06. Prompt generation is conducted using the top-k random sampling from the 10 most likely candidates, and the prompt is completed when the language model generates the end of prompt token.

5.5 Evaluation

We propose a number of evaluation metrics to quantify the performance of our models. Many commonly used metrics, such as BLEU for ma-

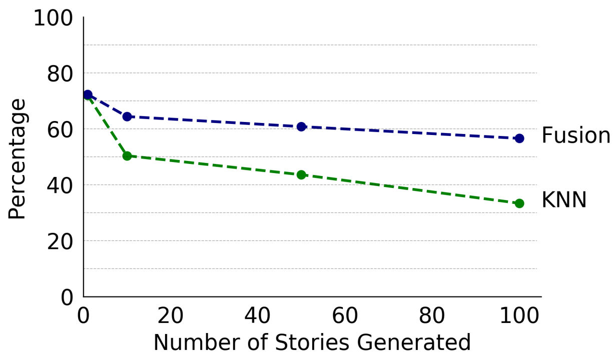


Figure 7: Accuracy on the prompt/story pairing task vs. number of generated stories. Our generative fusion model can produce many stories without degraded performance, while the KNN can only produce a limited number relevant stories.

Model	Human Preference
Language model	32.68%
Hierarchical Model	**67.32%**

Table 4: Effect of Hierarchical Generation. Human judges prefer stories that were generated hierarchically by first creating a premise and creating a full story based on it with a seq2seq model.

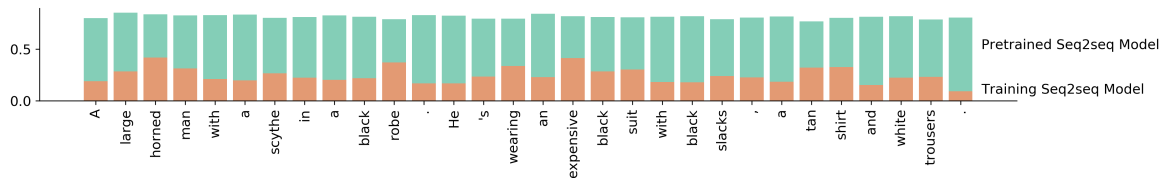


Figure 8: Average weighting of each model in our Fusion model for the beginning of the generated story for the prompt *Gates of Hell*. The fused model (orange) is primarily used for words which are closely related to the prompt, whereas generic words are generated by the pre-trained model (green).

chine translation or ROUGE for summarization, compute an n-gram overlap between the generated text and the human text—however, in our open-ended generation setting, these are not useful. We do not aim to generate a specific story; we want to generate viable and novel stories. We focus on measuring both the fluency of our models and their ability to adhere to the prompt.

For automatic evaluation, we measure *model perplexity* on the test set and *prompt ranking accuracy*. Perplexity is commonly used to evaluate the quality of language models, and it reflects how fluently the model can produce the correct next word given the preceding words. We use prompt ranking to assess how strongly a model's output depends on its input. Stories are decoded under 10 different prompts—9 randomly sampled prompts and 1 true corresponding prompt—and the likelihood of the story given the various prompts is recorded. We measure the percentage of cases where the true prompt is the most likely to generate the story. In our evaluation, we examined 1000 stories from the test set for each model.

For human evaluation, we use Amazon Mechanical Turk to conduct a *triple pairing task*. We use each model to generate stories based on held-out prompts from the test set. Then, groups of three stories are presented to the human judges. The stories and their corresponding prompts are shuffled, and human evaluators are asked to select the correct pairing for all three prompts. 105 stories per model are grouped into questions, and each question is evaluated by 15 judges.

Lastly, we conduct human evaluation to evaluate the importance of *hierarchical generation* for story writing. We use Amazon Mechanical Turk to compare the stories from hierarchical generation from a prompt with generation without a prompt. 400 pairs of stories were evaluated by 5 judges each in a blind test.

6 Results

We analyze the effect of our modeling improvements on the WRITINGPROMPTS dataset.

Effect of Hierarchical Generation: We explore leveraging our dataset to perform hierarchical story generation by first using a self-attentive GCNN language model to generate a prompt, and then using a fusion model to write a story given the generated prompt. We evaluate the effect of hierarchical generation using a human study in Table 4. 400 stories were generated from a self-attentive GCNN language model, and another 400 were generated from our hierarchical fusion model given generated prompts from a language model. In a blind comparison where raters were asked to choose the story they preferred reading, human raters preferred the hierarchical model 67% of the time.

Effect of new attention mechanism: Table 2 shows the effect of the proposed additions to the self-attention mechanism proposed by Vaswani et al. (2017). Table 3 shows that deep multi-scale self-attention and fusion each significantly improve the perplexity compared to the baselines. In combination these additions to the Conv seq2seq baseline reduce the perplexity by 9 points.

Effect of model fusion: Results in Table 3 show that adding our fusion mechanism substantially improves the likelihood of human-generated stories, and even outperforms an ensemble despite having fewer parameters. We observe in Figure 5 that fusion has a much more significant impact on the topicality of the stories. In comparison, ensembling has no effect on people's ability to associate stories with a prompt, but adding model fusion leads improves the pairing accuracy of the human judges by 7%. These results suggest that by training a second model on top of the first, we have encouraged that model to learn the challenging additional dependencies to relate to the source sequence. To our knowledge, these are the first results to show that fusion has such capabilities.

Comparison with Nearest Neighbours: Nearest Neighbour Search (KNN) provides a strong baseline for text generation. Figure 5 shows that the fusion model can match the performance of nearest neighbour search in terms of the connection between the story and prompt. The real value in our generative approach is that it can produce an unlimited number of stories, whereas KNN can never generalize from its training data. To quantify this improvement, Figure 7 plots the relevance of the *k*th best story to a given prompt; the performance of KNN degrades much more rapidly.

7 Discussion

7.1 Generation Quality

Our proposed fusion model is capable of generating unique text without copying directly from the training set. When analyzing 500 150-word generated stories from test-set prompts, the average longest common subsequence is 8.9. In contrast, the baseline Conv seq2seq model copies 10.2 words on average and the KNN baseline copies all 150 words from a story in the training set.

Figure 8 shows the values of the fusion gates for an example story, averaged at each timestep. The pretrained seq2seq model acts similarly to a language model producing common words and punctuation. The second seq2seq model learns to focus on rare words, such as *horned* and *robe*.

However, the fusion model has limitations. Using random sampling to generate can produce errors. For example, *can't* is tokenized to *ca n't*, and the model occasionally produces the first token but misses the second. A similar error is after one line of dialogue, the model may move to another line of dialogue without generating a newline token. A further obstacle is repetition. The model focuses frequently on what it has recently produced, which leads to the generation of similar text multiple times.

In the generation of prompts using the GCNN language model, we find that prompts are fairly generic compared to human prompts. Language models often struggle to model rare words accurately, as the probability distribution over the next word is dominated by more common words. This tends to produce similar prompts, particularly at the start — we see many prompts that start with *the man*. In contrast, many of the human prompts are very unique (e.g. prompting stories in fantasy worlds such as Harry Potter and Game of Thrones) and the language model rarely produces the specific vocabulary required by these settings.

7.2 Use of Attention

We analyze the encoder-decoder attention in the fusion model and find that unlike attention maps in machine translation, where each decoder timestep tends to attend to a different word on the encoder-side, the attention map for each decoder timestep looks similar and focuses mainly on salient words in the prompt. We further look at the usage of the self-attention layers within the decoder. While they could be leveraged to look at words generated very far in the past, at many timesteps the self-attention focuses on the recent past.

8 Conclusion

We have collected the first dataset for creative text generation based on short writing prompts. This new dataset pushes the boundaries of text generation by requiring longer range dependencies and conditioning on an abstract premise. Building on this dataset, we show through automatic and human evaluation that novel hierarchical models, self-attention mechanisms and model fusion significantly improves the fluency, topicality, and overall quality of the generated stories.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representation (ICLR)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks.

Generation with Fusion of Conv seq2seq + Self-Attention
Example Prompt 1: Aliens start abducting humans
It has been two weeks, and the last of my kind has gone. It is only a matter of time until there will be nothing left. I'm not sure what the hell is going on... I can't think. I can hear a distant scream. I think of a strange, alien sound. I try to speak, but am interrupted by something, something that feels like a drum, I ca not tell. I mean I'm just a little bit older than an average human. But I can, and I can feel the vibrations . I hear the sound of a ship approaching. The ground quakes at the force of the impact, and a loud explosion shatters the silence.
Example Prompt 2: The scientists have discovered something terrible .
The scientist stood there, a little dazed as he stared. "What is it?" He asked. "This...this...Thing...This is a virus. A chemical that can destroy entire planet and it is a very small, complex, chemical that could destroy any planet." The scientist replied. His lab assistant looked down at the tablet. "I've just discovered it. I can't believe it. It looks like it's made of some sort of chemical that's very dangerous." " Well, there's a virus on the ground. It's very effective..." " I can't believe what it is." he said, pointing to the scientist . " We don't know what this thing is. We haven't seen anything like it . We can't even see anything like this. " Dr. Jones stared at the scientist for a moment. "What do you mean what does it do ?" "It...It 's a monster."
Example Prompt 3: The silence before the storm comes
"I'm sorry," she said "I can't do this anymore." She sat in the grass, the grass blowing behind her as she sat staring at the sky. Her eyes were still closed but she was still staring at it. Her hand was still shaking, her eyes closed as she watched the rain roll in and out of view. She could see the world through her eyelids now. Her hands were still shaking like she hadn't eaten in days. Her fingers were still clenched tight and her fingers were now shaking. She wasn't even sure if she was going to cry or if she just wanted to cry. It wasn't until she was close enough to hear the words. "I ...I just...I need to...I ..." She was crying now and then, the rain was picking up. Her lips were shaking now and she could see that the storm was getting closer and closer. Her heart stopped when she heard the thunder. She knew it was coming. The storm was

Generation with Language Model + Self-Attention Baseline
Example 1:
I've always been the curious type. Growing up in the ghetto, I've been around for more than a year now. I still haven't heard it on my own. I'm not sure if I'm a good person. But I'm not the only one. I've been to the hospital, but it's not my first time. I've been to my psychiatrist's journals before. "Hey, are you ... are you okay?" I asked. "Yes" I said. "What's wrong?" she replied, "I was just ... Doing something." She said, "I've never been a bad person."
Example 2:
The man was an accountant. He had to be. He had to be the next president. I looked back over the top and saw that his wife was crying in the kitchen. I looked at the clock. It seemed to be coming slower, but I knew if I did it would not be long before I was in my own home. I wasn't sure. I had a hard time finding the right words to say. I was about to leave when he suddenly became angry and began talking to me. "Hello, sir, I'm John. What is your name?" "My name is Manuel and I'm a journalist." I said

Table 5: Example stories generated by the proposed hierarchical fusion approach compared to stories generated by a language model. Stories generated by the fusion model relate to the desired prompt and show increased coherence between sentences and ability to stay on one topic compared to the language modeling baseline.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Brent Harrison, Christopher Purdy, and Mark O Riedl. 2017. Toward automated story generation with markov chain monte carlo methods and deep neural networks.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *arXiv preprint arXiv:1707.05501*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015a. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015b. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Lara J Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2017. Event representations for automated story generation with deep neural nets. *arXiv preprint arXiv:1706.01331*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2016. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*.

Melissa Roemmele. 2016. Writing stories with help from recurrent neural networks. In *AAAI*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating long and diverse responses with neural conversation models. *arXiv preprint arXiv:1701.03185*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems (NIPS)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Denis Yarats and Mike Lewis. 2017. Hierarchical text generation and planning for strategic dialogue. *arXiv preprint arXiv:1712.05846*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

No Metrics Are Perfect: Adversarial Reward Learning for Visual Storytelling

Xin Wang[*], **Wenhu Chen**[*], **Yuan-Fang Wang** , **William Yang Wang**
University of California, Santa Barbara
{xwang,wenhuchen,yfwang,william}@cs.ucsb.edu

Abstract

Though impressive results have been achieved in visual captioning, the task of generating abstract stories from photo streams is still a little-tapped problem. Different from captions, stories have more expressive language styles and contain many imaginary concepts that do not appear in the images. Thus it poses challenges to behavioral cloning algorithms. Furthermore, due to the limitations of automatic metrics on evaluating story quality, reinforcement learning methods with hand-crafted rewards also face difficulties in gaining an overall performance boost. Therefore, we propose an Adversarial REward Learning (AREL) framework to learn an implicit reward function from human demonstrations, and then optimize policy search with the learned reward function. Though automatic evaluation indicates slight performance boost over state-of-the-art (SOTA) methods in cloning expert behaviors, human evaluation shows that our approach achieves significant improvement in generating more human-like stories than SOTA systems. Code will be made available here[1].

1 Introduction

Recently, increasing attention has been focused on visual captioning (Chen et al., 2015; Xu et al., 2016; Wang et al., 2018c), which aims at describing the content of an image or a video. Though it has achieved impressive results, its capability of performing human-like understanding is still restrictive. To further investigate machine's capabilities in understanding more complicated visual scenarios and composing more structured expressions, visual storytelling (Huang et al., 2016) has been proposed. Visual captioning is aimed at depicting the concrete content of the images, and its expression style is rather simple. In contrast, visual storytelling goes one step further: it summarizes the idea of a photo stream and tells a story about it. Figure 1 shows an example of visual captioning and visual storytelling. We have observed that stories contain rich **emotions** (*excited, happy, not want*) and **imagination** (*siblings, parents, school, car*). It, therefore, requires the capability to associate with concepts that do not explicitly appear in the images. Moreover, stories are more **subjective**, so there barely exists standard

Captions:
(a) A small boy and a girl are sitting together.
(b) Two kids sitting on a porch with their backpacks on.
(c) Two young kids with backpacks sitting on the porch.
(d) Two young children that are very close to one another.
(e) A boy and a girl smiling at the camera together.

Story #1: The brother and sister were ready for the first day of school. They were excited to go to their first day and meet new friends. They told their mom how happy they were. They said they were going to make a lot of new friends . Then they got up and got ready to get in the car .

Story #2: The brother did not want to talk to his sister. The siblings made up. They started to talk and smile. Their parents showed up. They were happy to see them.

Figure 1: An example of visual storytelling and visual captioning. Both captions and stories are shown here: each image is captioned with one sentence, and we also demonstrate two diversified stories that match the same image sequence.

[*] Equal contribution
[1] https://github.com/littlekobe/AREL

templates for storytelling. As shown in Figure 1, the same photo stream can be paired with diverse stories, different from each other. This heavily increases the evaluation difficulty.

So far, prior work for visual storytelling (Huang et al., 2016; Yu et al., 2017b) is mainly inspired by the success of visual captioning. Nevertheless, because these methods are trained by maximizing the likelihood of the observed data pairs, they are restricted to generate simple and plain description with limited expressive patterns. In order to cope with the challenges and produce more human-like descriptions, Rennie et al. (2016) have proposed a reinforcement learning framework. However, in the scenario of visual storytelling, the common reinforced captioning methods are facing great challenges since the hand-crafted rewards based on string matches are either too biased or too sparse to drive the policy search. For instance, we used the METEOR (Banerjee and Lavie, 2005) score as the reward to reinforce our policy and found that though the METEOR score is significantly improved, the other scores are severely harmed. Here we showcase an adversarial example with an average METEOR score as high as 40.2:

> *We had a great time to have a lot of the. They were to be a of the. They were to be in the. The and it were to be the. The, and it were to be the.*

Apparently, the machine is gaming the metrics. Conversely, when using some other metrics (*e.g.* BLEU, CIDEr) to evaluate the stories, we observe an opposite behavior: many relevant and coherent stories are receiving a very low score (nearly zero).

In order to resolve the strong bias brought by the hand-coded evaluation metrics in RL training and produce more human-like stories, we propose an Adversarial REward Learning (AREL) framework for visual storytelling. We draw our inspiration from recent progress in inverse reinforcement learning (Ho and Ermon, 2016; Finn et al., 2016; Fu et al., 2017) and propose the AREL algorithm to learn a more intelligent reward function. Specifically, we first incorporate a Boltzmann distribution to associate reward learning with distribution approximation, then design the adversarial process with two models – a **policy model** and a **reward model**. The policy model performs the primitive actions and produces the story sequence, while the reward model is responsible for learning the implicit reward function from human demonstrations. The learned reward function would be employed to optimize the policy in return.

For evaluation, we conduct both automatic metrics and human evaluation but observe a poor correlation between them. Particularly, our method gains slight performance boost over the baseline systems on automatic metrics; human evaluation, however, indicates significant performance boost. Thus we further discuss the limitations of the metrics and validate the superiority of our AREL method in performing more intelligent understanding of the visual scenes and generating more human-like stories.

Our main contributions are four-fold:

- We propose an adversarial reward learning framework and apply it to boost visual story generation.
- We evaluate our approach on the Visual Storytelling (VIST) dataset and achieve the state-of-the-art results on automatic metrics.
- We empirically demonstrate that automatic metrics are not perfect for either training or evaluation.
- We design and perform a comprehensive human evaluation via Amazon Mechanical Turk, which demonstrates the superiority of the generated stories of our method on relevance, expressiveness, and concreteness.

2 Related Work

Visual Storytelling Visual storytelling is the task of generating a narrative story from a photo stream, which requires a deeper understanding of the event flow in the stream. Park and Kim (2015) has done some pioneering research on storytelling. Chen et al. (2017) proposed a multimodal approach for storyline generation to produce a stream of entities instead of human-like descriptions. Recently, a more sophisticated dataset for visual storytelling (VIST) has been released to explore a more human-like understanding of grounded stories (Huang et al., 2016). Yu et al. (2017b) proposes a multi-task learning algorithm for both album summarization and paragraph generation, achieving the best results on the VIST dataset. But these methods are still based on behavioral cloning and lack the ability to generate more structured stories.

Reinforcement Learning in Sequence Generation Recently, reinforcement learning (RL) has gained its popularity in many sequence generation tasks such as machine translation (Bahdanau et al., 2016), visual captioning (Ren et al., 2017; Wang et al., 2018b), summarization (Paulus et al., 2017; Chen et al., 2018), etc. The common wisdom of using RL is to view generating a word as an action and aim at maximizing the expected return by optimizing its policy. As pointed in (Ranzato et al., 2015), traditional maximum likelihood algorithm is prone to exposure bias and label bias, while the RL agent exposes the generative model to its own distribution and thus can perform better. But these works usually utilize hand-crafted metric scores as the reward to optimize the model, which fails to learn more implicit semantics due to the limitations of automatic metrics.

Rethinking Automatic Metrics Automatic metrics, including BLEU (Papineni et al., 2002), CIDEr (Vedantam et al., 2015), METEOR (Banerjee and Lavie, 2005), and ROUGE (Lin, 2004), have been widely applied to the sequence generation tasks. Using automatic metrics can ensure rapid prototyping and testing new models with fewer expensive human evaluation. However, they have been criticized to be biased and correlate poorly with human judgments, especially in many generative tasks like response generation (Lowe et al., 2017; Liu et al., 2016), dialogue system (Bruni and Fernández, 2017) and machine translation (Callison-Burch et al., 2006). The naive overlap-counting methods are not able to reflect many semantic properties in natural language, such as coherence, expressiveness, etc.

Generative Adversarial Network Generative adversarial network (GAN) (Goodfellow et al., 2014) is a very popular approach for estimating intractable probabilities, which sidestep the difficulty by alternately training two models to play a min-max two-player game:

$$\min_D \max_G \mathop{E}_{x \sim p_{data}} [\log D(x)] + \mathop{E}_{z \sim p_z} [\log D(G(z))] ,$$

where G is the generator and D is the discriminator, and z is the latent variable. Recently, GAN has quickly been adopted to tackle discrete problems (Yu et al., 2017a; Dai et al., 2017; Wang et al., 2018a). The basic idea is to use Monte Carlo policy gradient estimation (Williams, 1992) to update the parameters of the generator.

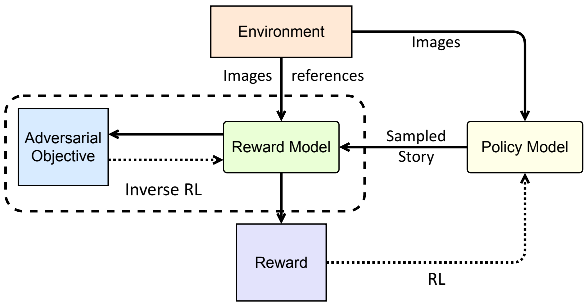


Figure 2: AREL framework for visual storytelling.

Inverse Reinforcement Learning Reinforcement learning is known to be hindered by the need for an extensive feature and reward engineering, especially under the unknown dynamics. Therefore, inverse reinforcement learning (IRL) has been proposed to infer expert's reward function. Previous IRL approaches include maximum margin approaches (Abbeel and Ng, 2004; Ratliff et al., 2006) and probabilistic approaches (Ziebart, 2010; Ziebart et al., 2008). Recently, adversarial inverse reinforcement learning methods provide an efficient and scalable promise for automatic reward acquisition (Ho and Ermon, 2016; Finn et al., 2016; Fu et al., 2017; Henderson et al., 2017). These approaches utilize the connection between IRL and energy-based model and associate every data with a scalar energy value by using Boltzmann distribution $p_\theta(x) \propto \exp(-E_\theta(x))$. Inspired by these methods, we propose a practical AREL approach for visual storytelling to uncover a robust reward function from human demonstrations and thus help produce human-like stories.

3 Our Approach

3.1 Problem Statement

Here we consider the task of visual storytelling, whose objective is to output a word sequence $W = (w_1, w_1, \cdots, w_T)$, $w_t \in \mathbb{V}$ given an input image stream of 5 ordered images $I = (I_1, I_2, \cdots, I_5)$, where $\mathbb{V}$ is the vocabulary of all output token. We formulate the generation as a markov decision process and design a reinforcement learning framework to tackle it. As described in Figure 2, our AREL framework is mainly composed of two modules: a **policy model** $\pi_\beta(W)$ and a **reward model** $R_\theta(W)$. The policy model takes an image sequence I as the input and performs sequential actions (choosing words w from the vocabulary $\mathbb{V}$) to form a narrative story W. The reward model

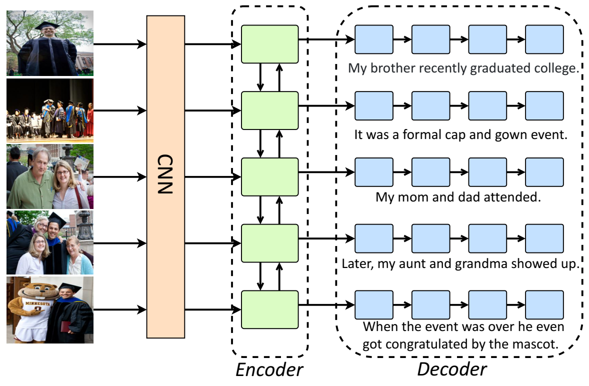

CNN
My brother recently graduated college.
It was a formal cap and gown event.
My mom and dad attended.
Later, my aunt and grandma showed up.
When the event was over he even got congratulated by the mascot.
Encoder
Decoder

Figure 3: Overview of the policy model. The visual encoder is a bidirectional GRU, which encodes the high-level visual features extracted from the input images. Its outputs are then fed into the RNN decoders to generate sentences in parallel. Finally, we concatenate all the generated sentences as a full story. Note that the five decoders share the same weights.

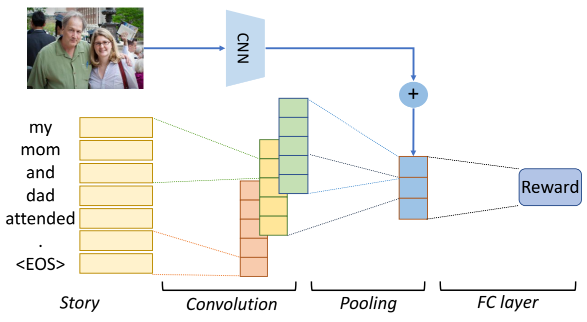


Figure 4: Overview of the reward model. Our reward model is a CNN-based architecture, which utilizes convolution kernels with size 2, 3 and 4 to extract bigram, trigram and 4-gram representations from the input sequence embeddings. Once the sentence representation is learned, it will be concatenated with the visual representation of the input image, and then be fed into the final FC layer to obtain the reward.

is optimized by the adversarial objective (see Section 3.3) and aims at deriving a human-like reward from both human-annotated stories and sampled predictions.

3.2 Model

Policy Model As is shown in Figure 3, the policy model is a CNN-RNN architecture. We fist feed the photo stream $I = (I_1, \cdots, I_5)$ into a pretrained CNN and extract their high-level image features. We then employ a visual encoder to further encode the image features as context vectors $h_i = [\overleftarrow{h_i}; \overrightarrow{h_i}]$. The visual encoder is a bidirectional gated recurrent units (GRU).

In the decoding stage, we feed each context vector h_i into a GRU-RNN decoder to generate a sub-story W_i. Formally, the generation process can be written as:

$$s_t^i = \text{GRU}(s_{t-1}^i, [w_{t-1}^i, h_i]) \,, \quad (1)$$

$$\pi_\beta(w_t^i | w_{1:t-1}^i) = softmax(W_s s_t^i + b_s) \,, \quad (2)$$

where s_t^i denotes the t-th hidden state of i-th decoder. We concatenate the previous token w_{t-1}^i and the context vector h_i as the input. W_s and b_s are the projection matrix and bias, which output a probability distribution over the whole vocabulary $\mathbb{V}$. Eventually, the final story W is the concatenation of the sub-stories W_i. β denotes all the parameters of the encoder, the decoder, and the output layer.

Reward Model The reward model $R_\theta(W)$ is a CNN-based architecture (see Figure 4). Instead of giving an overall score for the whole story, we apply the reward model to different story parts (sub-stories) W_i and compute partial rewards, where $i = 1, \cdots, 5$. We observe that the partial rewards are more fine-grained and can provide better guidance for the policy model.

We first query the word embeddings of the sub-story (one sentence in most cases). Next, multiple convolutional layers with different kernel sizes are used to extract the n-grams features, which are then projected into the sentence-level representation space by pooling layers (the design here is inspired by Kim (2014)). In addition to the textual features, evaluating the quality of a story should also consider the image features for relevance. Therefore, we then combine the sentence representation with the visual feature of the input image through concatenation and feed them into the final fully connected decision layer. In the end, the reward model outputs an estimated reward value $R_\theta(W)$. The process can be written in formula:

$$R_\theta(W) = W_r(f_{conv}(W) + W_i I_{CNN}) + b_r, \quad (3)$$

where W_r, b_r denotes the weights in the output layer, and f_{conv} denotes the operations in CNN. I_{CNN} is the high-level visual feature extracted from the image, and W_i projects it into the sentence representation space. θ includes all the parameters above.

3.3 Learning

Reward Boltzmann Distribution In order to associate story distribution with reward function, we apply EBM to define a Reward Boltzmann distribution:

$$p_\theta(W) = \frac{\exp(R_\theta(W))}{Z_\theta}\,, \tag{4}$$

Where W is the word sequence of the story and $p_\theta(W)$ is the approximate data distribution, and $Z_\theta = \sum_W \exp(R_\theta(W))$ denotes the partition function. According to the energy-based model (LeCun et al., 2006), the optimal reward function $R^*(W)$ is achieved when the Reward-Boltzmann distribution equals to the "real" data distribution $p_\theta(W) = p^*(W)$.

Adversarial Reward Learning We first introduce an empirical distribution $p_e(W) = \frac{\mathbb{1}(W \in D)}{|D|}$ to represent the empirical distribution of the training data, where D denotes the dataset with $|D|$ stories and $\mathbb{1}$ denotes an indicator function. We use this empirical distribution as the "good" examples, which provides the evidence for the reward function to learn from.

In order to approximate the Reward Boltzmann distribution towards the "real" data distribution $p^*(W)$, we design a min-max two-player game, where the Reward Boltzmann distribution p_θ aims at maximizing the its similarity with empirical distribution p_e while minimizing that with the "faked" data generated from policy model π_β. On the contrary, the policy distribution π_β tries to maximize its similarity with the Boltzmann distribution p_θ. Formally, the adversarial objective function is defined as

$$\max_\beta \min_\theta KL(p_e(W)||p_\theta(W)) - KL(\pi_\beta(W)||p_\theta(W))\,. \tag{5}$$

We further decompose it into two parts. First, because the objective J_β of the story generation policy is to minimize its similarity with the Boltzmann distribution p_θ, the optimal policy that minimizes KL-divergence is thus $\pi(W) \sim \exp(R_\theta(W))$, meaning if R_θ is optimal, the optimal $\pi_\beta = \pi^*$. In formula,

$$\begin{aligned} J_\beta =& -KL(\pi_\beta(W)||p_\theta(W)) \\ =& \mathop{E}_{W\sim\pi_\beta(W)}[R_\theta(W)] + H(\pi_\beta(W))\,, \end{aligned} \tag{6}$$

Algorithm 1 The AREL Algorithm.

```
1: for episode ← 1 to N do
2:     collect story $W$ by executing policy $\pi_\theta$
3:     if Train-Reward then
4:         $\theta \leftarrow \theta - \eta \times \frac{\partial J_\theta}{\partial \theta}$ (see Equation 9)
5:     else if Train-Policy then
6:         collect story $\tilde{W}$ from empirical $p_e$
7:         $\beta \leftarrow \beta - \eta \times \frac{\partial J_\beta}{\partial \beta}$ (see Equation 9)
8:     end if
9: end for
```

where H denotes the entropy of the policy model. On the other hand, the objective J_θ of the reward function is to distinguish between human-annotated stories and machine-generated stories. Hence it is trying to minimize the KL-divergence with the empirical distribution p_e and maximize the KL-divergence with the approximated policy distribution π_β:

$$\begin{aligned} J_\theta =& KL(p_e(W)||p_\theta(W)) - KL(\pi_\beta(W)||p_\theta(W)) \\ =& \sum_W [p_e(W)R_\theta(W) - \pi_\beta(W)R_\theta(W)] \\ & - H(p_e) + H(\pi_\beta)\,, \end{aligned} \tag{7}$$

Since $H(\pi_\beta)$ and $H(p_e)$ are irrelevant to θ, we denote them as constant C. Therefore, the objective J_θ can be further derived as

$$J_\theta = \mathop{E}_{W\sim p_e(W)}[R_\theta(W)] - \mathop{E}_{W\sim\pi_\beta(W)}[R_\theta(W)] + C\,. \tag{8}$$

Here we propose to use stochastic gradient descent to optimize these two models alternately. Formally, the gradients can be written as

$$\begin{aligned} \frac{\partial J_\theta}{\partial \theta} &= \mathop{E}_{W\sim p_e(W)} \frac{\partial R_\theta(W)}{\partial \theta} - \mathop{E}_{W\sim\pi_\beta(W)} \frac{\partial R_\theta(W)}{\partial \theta}\,, \\ \frac{\partial J_\beta}{\partial \beta} &= \mathop{E}_{W\sim\pi_\beta(W)} (R_\theta(W) + \log \pi_\theta(W) - b) \frac{\partial \log \pi_\beta(W)}{\partial \beta}\,, \end{aligned} \tag{9}$$

where b is the estimated baseline to reduce the variance.

Training & Testing As described in Algorithm 1, we introduce an alternating algorithm to train these two models using stochastic gradient descent. During testing, the policy model is used with beam search to produce the story.

4 Experiments and Analysis

4.1 Experimental Setup

VIST Dataset The VIST dataset (Huang et al., 2016) is the first dataset for sequential vision-to-language tasks including visual storytelling, which consists of 10,117 Flickr albums with 210,819 unique photos. In this paper, we mainly evaluate our AREL method on this dataset. After filtering the broken images[2], there are 40,098 training, 4,988 validation, and 5,050 testing samples. Each sample contains one story that describes 5 selected images from a photo album (mostly one sentence per image). And the same album is paired with 5 different stories as references. In our experiments, we used the same split settings as in (Huang et al., 2016; Yu et al., 2017b) for a fair comparison.

Evaluation Metrics In order to comprehensively evaluate our method on storytelling dataset, we adopted both the automatic metrics and human evaluation as our criterion. Four diverse automatic metrics were used in our experiments: BLEU, METEOR, ROUGE-L, and CIDEr. We utilized the open source evaluation code[3] used in (Yu et al., 2017b). For human evaluation, we employed the Amazon Mechanical Turk to perform two kinds of user studies (see Section 4.3 for more details).

Training Details We employ pretrained ResNet-152 model (He et al., 2016) to extract image features from the photo stream. We built a vocabulary of size 9,837 to include words appearing more than three times in the training set. More training details can be found at Appendix B.

4.2 Automatic Evaluation

In this section, we compare our AREL method with the state-of-the-art methods as well as standard reinforcement learning algorithms on automatic evaluation metrics. Then we further discuss the limitations of the hand-crafted metrics on evaluating human-like stories.

Comparison with SOTA on Automatic Metrics In Table 1, we compare our method with Huang et al. (2016) and Yu et al. (2017b), which report achieving best-known results on the VIST dataset. We first implement a strong baseline model (*XE-ss*), which share the same architecture with our policy model but is trained with cross-entropy loss and scheduled sampling. Besides, we adopt the traditional generative adversarial training for comparison (*GAN*). As shown in Table 1, our XE-ss model already outperforms the best-known results on the VIST dataset, and the GAN model can bring a performance boost. We then use the XE-ss model to initialize our policy model and further train it with *AREL*. Evidently, our AREL model performs the best and achieves the new state-of-the-art results across all metrics.

Method	B-1	B-2	B-3	B-4	M	R	C
Huang et al.	-	-	-	-	31.4	-	-
Yu et al.	-	-	21.0	-	34.1	29.5	7.5
XE-ss	62.3	38.2	22.5	13.7	34.8	**29.7**	8.7
GAN	62.8	38.8	23.0	14.0	35.0	29.5	9.0
AREL-s-50	63.8	38.9	22.9	13.8	34.9	29.4	9.5
AREL-t-50	63.4	39.0	23.1	**14.1**	**35.2**	29.6	9.5
AREL-s-100	**63.9**	**39.1**	23.0	13.9	35.0	**29.7**	**9.6**
AREL-t-100	63.8	**39.1**	**23.2**	**14.1**	35.0	29.5	9.4

Table 1: Automatic evaluation on the VIST dataset. We report BLEU (B), METEOR (M), ROUGH-L (R), and CIDEr (C) scores of the SOTA systems and the models we implemented, including XE-ss, GAN and AREL. AREL-s-N denotes AREL models with sigmoid as output activation and alternate frequency as N, while AREL-t-N denoting AREL models with tahn as the output activation (N = 50 or 100).

But, compared with the XE-ss model, the performance gain is minor, especially on METEOR and ROUGE-L scores. However, in Sec. 4.3, the extensive human evaluation has indicated that our AREL framework brings a significant improvement on generating human-like stories over the XE-ss model. The inconsistency of automatic evaluation and human evaluation lead to a suspect that these hand-crafted metrics lack the ability to fully evaluate stories' quality due to the complicated characteristics of the stories. Therefore, we conduct experiments to analyze and discuss the defects of the automatic metrics in section 4.2.

Limitations of Automatic Metrics As we claimed in the introduction, string-match-based automatic metrics are not perfect and fail to evaluate some semantic characteristics of the stories, like the expressiveness and coherence of the stories. In order to confirm our conjecture, we utilize automatic metrics as rewards to reinforce the visual storytelling model by adopting policy gradient with baseline to train the policy model. The quantitative results are demonstrated in Table 1.

Apparently, METEOR-RL and ROUGE-RL are severely ill-posed: they obtain the highest scores on their own metrics but damage the other met-

[2]There are only 3 (out of 21,075) broken images in the test set, which basically has no influence on the final results. Moreover, Yu et al. (2017b) also removed the 3 pictures, so it is a fair comparison.

[3]`https://github.com/lichengunc/vist_eval`

Method	B-1	B-2	B-3	B-4	M	R	C
XE-ss	62.3	38.2	22.5	13.7	34.8	**29.7**	8.7
BLEU-RL	62.1	38.0	22.6	13.9	34.6	29.0	8.9
METEOR-RL	**68.1**	35.0	15.4	6.8	**40.2**	30.0	1.2
ROUGE-RL	58.1	18.5	1.6	0	27.0	**33.8**	0
CIDEr-RL	61.9	37.8	22.5	13.8	34.9	29.7	8.1
AREL (avg)	**63.7**	**39.0**	**23.1**	**14.0**	**35.0**	29.6	**9.5**

Table 2: Comparison with different RL models with different metric scores as the rewards. We report the average scores of the AREL models as AREL (avg). Although METEOR-RL and ROUGE-RL models achieve very high scores on their own metrics, the underlined scores are severely damaged. Actually, they are gaming their own metrics with nonsense sentences.

Method	Win	Lose	Unsure
XE-ss	22.4%	71.7%	5.9%
BLEU-RL	23.4%	67.9%	8.7%
CIDEr-RL	13.8%	80.3%	5.9%
GAN	34.3%	60.5%	5.2%
AREL	**38.4%**	**54.2%**	**7.4%**

Table 3: Turing test results.

rics severely. We observe that these models are actually overfitting to a given metric while losing the overall coherence and semantical correctness. Same as METEOR score, there is also an adversarial example for ROUGE-L[4], which is nonsense but achieves an average ROUGE-L score of 33.8.

Besides, as can be seen in Table 1, after reinforced training, BLEU-RL and CIDEr-RL do not bring a consistent improvement over the XE-ss model. We plot the histogram distributions of both BLEU-3 and CIDEr scores on the test set in Figure 5. An interesting fact is that there are a large number of samples with nearly zero score on both metrics. However, we observed those "zero-score" samples are not pointless results; instead, lots of them make sense and deserve a better score than zero. Here is a "zero-score" example on BLEU-3:

I had a great time at the restaurant today. The food was delicious. I had a lot of food. The food was delicious. T had a great time.

The corresponding reference is

The table of food was a pleasure to see! Our food is both nutritious and beautiful! Our chicken was especially tasty! We love greens as they taste great and are healthy! The fruit was a colorful display that tantalized our palette..

Although the prediction is not as good as the reference, it is actually coherent and relevant to the theme "food and eating", which showcases the defeats of using BLEU and CIDEr scores as a reward for RL training.

Moreover, we compare the human evaluation scores with these two metric scores in Figure 5. Noticeably, both BLEU-3 and CIDEr have a poor correlation with the human evaluation scores. Their distributions are more biased and thus cannot fully reflect the quality of the generated stories. In terms of BLEU, it is extremely hard for machines to produce the exact 3-gram or 4-gram matching, so the scores are too low to provide useful guidance. CIDEr measures the similarity of a sentence to the majority of the references. However, the references to the same image sequence are photostream different from each other, so the score is very low and not suitable for this task. In contrast, our AREL framework can lean a more robust reward function from human-annotated stories, which is able to provide better guidance to the policy and thus improves its performances over different metrics.

Comparison with GAN We here compare our method with traditional GAN (Goodfellow et al., 2014), the update rule for generator can be generally classified into two categories. We demonstrate their corresponding objectives and ours as follows:

$$\begin{aligned} GAN1: \quad & J_\beta = \underset{W\sim p_\beta}{E}\left[-\log R_\theta(W)\right], \\ GAN2: \quad & J_\beta = \underset{W\sim p_\beta}{E}\left[\log(1-R_\theta(W))\right], \\ ours: \quad & J_\beta = \underset{W\sim p_\beta}{E}\left[-R_\theta(W)\right]. \end{aligned}$$

As discussed in Arjovsky et al. (2017), $GAN1$ is prone to the unstable gradient issue and $GAN2$ is prone to the vanishing gradient issue. Analytically, our method does not suffer from these two common issues and thus is able converge to optimum solutions more easily. From Table 1, we can observe slight gains of using AREL over GAN

[4]An adversarial example for ROUGE-L: *we the was a . and to the . we the was a . and to the . we the was a . and to the . we the was a . and to the . we the was a . and to the .*

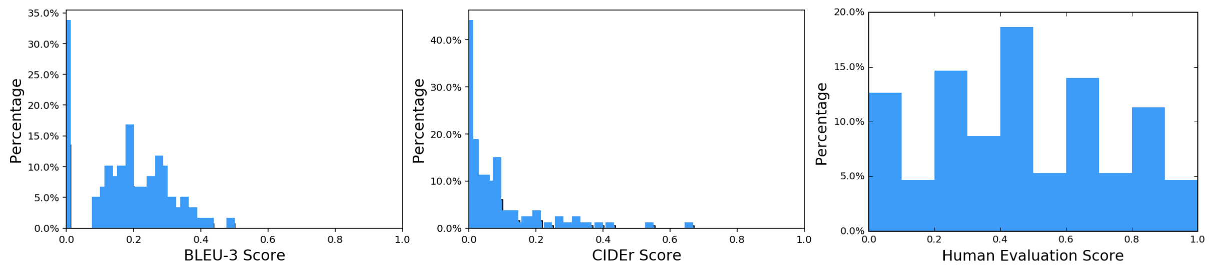


Figure 5: Metric score distributions. We plot the histogram distributions of BLEU-3 and CIDEr scores on the test set, as well as the human evaluation score distribution on the test samples. For a fair comparison, we use the Turing test results to calculate the human evaluation scores (see Section 4.3). Basically, 0.2 score is given if the generated story wins the Turing test, 0.1 for tie, and 0 if losing. Each sample has 5 scores from 5 judges, and we use the sum as the human evaluation score, so it is in the range [0, 1].

	AREL *vs* XE-ss			AREL *vs* BLEU-RL			AREL *vs* CIDEr-RL			AREL *vs* GAN		
Choice (%)	AREL	XE-ss	Tie	AREL	BLEU-RL	Tie	AREL	CIDEr-RL	Tie	AREL	GAN	Tie
Relevance	**61.7**	25.1	13.2	**55.8**	27.9	16.3	**56.1**	28.2	15.7	**52.9**	35.8	11.3
Expressiveness	**66.1**	18.8	15.1	**59.1**	26.4	14.5	**59.1**	26.6	14.3	**48.5**	32.2	19.3
Concreteness	**63.9**	20.3	15.8	**60.1**	26.3	13.6	**59.5**	24.6	15.9	**49.8**	35.8	14.4

Table 4: Pairwise human comparisons. The results indicate the consistent superiority of our AREL model in generating more human-like stories than the SOTA methods.

with automatic metrics, therefore we further deploy human evaluation for a better comparison.

4.3 Human Evaluation

Automatic metrics cannot fully evaluate the capability of our AREL method. Therefore, we perform two different kinds of human evaluation studies on Amazon Mechanical Turk: Turing test and pairwise human evaluation. For both tasks, we use 150 stories (750 images) sampled from the test set, each assigned to 5 workers to eliminate human variance. We batch six items as one assignment and insert an additional assignment as a sanity check. Besides, the order of the options within each item is shuffled to make a fair comparison.

Turing Test We first conduct five independent Turing tests for XE-ss, BLEU-RL, CIDEr-RL, GAN, and AREL models, during which the worker is given one human-annotated sample and one machine-generated sample, and needs to decide which is human-annotated. As shown in Table 3, our AREL model significantly outperforms all the other baseline models in the Turing test: it has much more chances to fool AMT worker (the ratio is AREL:XE-ss:BLEU-RL:CIDEr-RL:GAN = 45.8%:28.3%:32.1%:19.7%:39.5%), which confirms the superiority of our AREL framework in generating human-like stories. Unlike automatic metric evaluation, the Turing test has indicated a much larger margin between AREL and other competing algorithms. Thus, we empirically confirm that metrics are not perfect in evaluating many implicit semantic properties of natural language. Besides, the Turing test of our AREL model reveals that nearly half of the workers are fooled by our machine generation, indicating a preliminary success toward generating human-like stories.

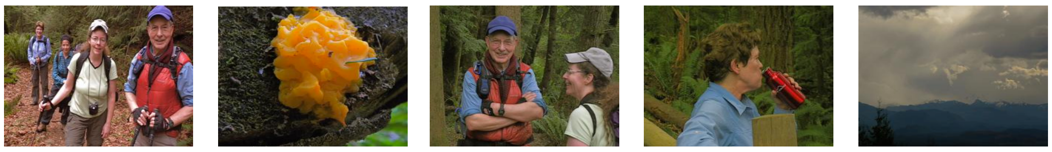
Pairwise Comparison In order to have a clear comparison with competing algorithms with respect to different semantic features of the stories, we further perform four pairwise comparison tests: AREL *vs* XE-ss/BLEU-RL/CIDEr-RL/GAN. For each photo stream, the worker is presented with two generated stories and asked to make decisions from the three aspects: relevance[5], expressiveness[6] and concreteness[7]. This head-to-head compete is designed to help us understand in what aspect our model outperforms the competing algorithms, which is displayed in Table 4.

Consistently on all the three comparisons, a large majority of the AREL stories trumps the competing systems with respect to their relevance,

[5]Relevance: the story accurately describes what is happening in the image sequence and covers the main objects.

[6]Expressiveness: coherence, grammatically and semantically correct, no repetition, expressive language style.

[7]Concreteness: the story should narrate concretely what is in the image rather than giving very general descriptions.



XE-ss	We took a trip to the mountains.	There were many different kinds of different kinds.	We had a great time.	He was a great time.	It was a beautiful day.
AREL	The family decided to take a trip to the countryside.	There were so many different kinds of things to see.	The family decided to go on a hike.	I had a great time.	At the end of the day, we were able to take a picture of the beautiful scenery.
Human-created Story	We went on a hike yesterday.	There were a lot of strange plants there.	I had a great time.	We drank a lot of water while we were hiking.	The view was spectacular.

Figure 6: Qualitative comparison example with XE-ss. The direct comparison votes (AREL:XE-ss:Tie) were 5:0:0 on Relevance, 4:0:1 on Expressiveness, and 5:0:0 on Concreteness.

expressiveness, and concreteness. Therefore, it empirically confirms that our generated stories are more relevant to the image sequences, more coherent and concrete than the other algorithms, which however is not explicitly reflected by the automatic metric evaluation.

4.4 Qualitative Analysis

Figure 6 gives a qualitative comparison example between AREL and XE-ss models. Looking at the individual sentences, it is obvious that our results are more grammatically and semantically correct. Then connecting the sentences together, we observe that the AREL story is more coherent and describes the photo stream more accurately. Thus, our AREL model significantly surpasses the XE-ss model on all the three aspects of the qualitative example. Besides, it won the Turing test (3 out 5 AMT workers think the AREL story is created by a human). In the appendix, we also show a negative case that fails the Turing test.

5 Conclusion

In this paper, we not only introduce a novel adversarial reward learning algorithm to generate more human-like stories given image sequences, but also empirically analyze the limitations of the automatic metrics for story evaluation. We believe there are still lots of improvement space in the narrative paragraph generation tasks, like how to better simulate human imagination to create more vivid and diversified stories.

Acknowledgment

We thank Adobe Research for supporting our language and vision research. We would also like to thank Licheng Yu for clarifying the details of his paper and the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings* *of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Elia Bruni and Raquel Fernández. 2017. Adversarial evaluation for open-domain dialogue generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–288.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Wenhu Chen, Guanlin Li, Shuo Ren, Shujie Liu, Zhirui Zhang, Mu Li, and Ming Zhou. 2018. Generative bridging network in neural sequence prediction. In *NAACL*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Zhiqian Chen, Xuchao Zhang, Arnold P. Boedihardjo, Jing Dai, and Chang-Tien Lu. 2017. Multimodal storytelling via generative adversarial imitation learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3967–3973.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *The IEEE International Conference on Computer Vision (ICCV)*.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*.

Justin Fu, Katie Luo, and Sergey Levine. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. 2017. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. *arXiv preprint arXiv:1709.06683*.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573.

Ting-Hao K. Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Aishwarya Agrawal, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ryan Lowe, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. *arXiv preprint arXiv:1708.07149*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Cesc C Park and Gunhee Kim. 2015. Expressing an image stream with a sequence of natural sentences. In *Advances in Neural Information Processing Systems*, pages 73–81.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2006. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM.

Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. 2017. Deep reinforcement learning-based image captioning with embedding reward. In *Proceeding of IEEE conference on Computer Vision and Pattern Recognition (CVPR)*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Jing Wang, Jianlong Fu, Jinhui Tang, Zechao Li, and Tao Mei. 2018a. Show, reward and tell: Automatic generation of narrative paragraph from photo stream by adversarial training. *AAAI*.

Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. 2018b. Video captioning via hierarchical reinforcement learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xin Wang, Yuan-Fang Wang, and William Yang Wang. 2018c. Watch, listen, and describe: Globally and locally aligned cross-modal attentions for video captioning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017a. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Licheng Yu, Mohit Bansal, and Tamara Berg. 2017b. Hierarchically-attentive rnn for album summarization and storytelling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 966–971, Copenhagen, Denmark. Association for Computational Linguistics.

Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA.

Bridging Languages through Images with Deep Partial Canonical Correlation Analysis

Guy Rotman[1], **Ivan Vulić**[2] and **Roi Reichart**[1]
[1] Faculty of Industrial Engineering and Management, Technion, IIT
[2] Language Technology Lab, University of Cambridge
grotman@campus.technion.ac.il
iv250@cam.ac.uk roiri@technion.ac.il

Abstract

We present a deep neural network that leverages images to improve bilingual text embeddings. Relying on bilingual image tags and descriptions, our approach conditions text embedding induction on the shared visual information for both languages, producing highly correlated bilingual embeddings. In particular, we propose a novel model based on Partial Canonical Correlation Analysis (PCCA). While the original PCCA finds linear projections of two views in order to maximize their canonical correlation conditioned on a shared third variable, we introduce a non-linear Deep PCCA (DPCCA) model, and develop a new stochastic iterative algorithm for its optimization. We evaluate PCCA and DPCCA on multilingual word similarity and cross-lingual image description retrieval. Our models outperform a large variety of previous methods, despite not having access to any visual signal during test time inference.[1]

1 Introduction

Research in multi-modal semantics deals with the *grounding problem* (Harnad, 1990), motivated by evidence that many semantic concepts, irrespective of the actual language, are grounded in the perceptual system (Barsalou and Wiemer-Hastings, 2005). In particular, recent studies have shown that performance on NLP tasks can be improved by joint modeling of text and vision, with multi-modal and perceptually enhanced representation learning outperforming purely textual representations (Feng and Lapata, 2010; Kiela and Bottou, 2014; Lazaridou et al., 2015).

These findings are not surprising, and can be explained by the fact that humans understand language not only by its words, but also by their visual/perceptual context. The ability to connect vision and language has also enabled new tasks which require both visual *and* language understanding, such as visual question answering (Antol et al., 2015; Fukui et al., 2016; Xu and Saenko, 2016), image-to-text retrieval and text-to-image retrieval (Kiros et al., 2014; Mao et al., 2014), image caption generation (Farhadi et al., 2010; Mao et al., 2015; Vinyals et al., 2015; Xu et al., 2015), and visual sense disambiguation (Gella et al., 2016).

While the main focus is still on monolingual settings, the fact that visual data can serve as a natural bridge between languages has sparked additional interest towards *multilingual multi-modal modeling*. Such models induce bilingual multi-modal spaces based on multi-view learning (Calixto et al., 2017; Gella et al., 2017; Rajendran et al., 2016).

In this work, we propose a novel effective approach for learning bilingual *text* embeddings *conditioned on shared visual information*. This additional perceptual modality bridges the gap between languages and reveals latent connections between concepts in the multilingual setup. The shared visual information in our work takes the form of images with word-level tags or sentence-level descriptions assigned in more than one language.

We propose a *deep* neural architecture termed Deep Partial Canonical Correlation Analysis (DPCCA) based on the Partial CCA (PCCA) method (Rao, 1969). To the best of our knowledge, PCCA has not been used in multilingual settings before. In short, PCCA is a variant of CCA which learns maximally correlated linear projections of two views (e.g., two language-specific "text-based views") conditioned on a shared third view (e.g.,

[1] Our code and data are available at: https://github.com/rotmanguy/DPCCA.

the "visual view"). We discuss the PCCA and DPCCA methods in §3 and show how they can be applied without having access to the shared images at test time inference.

PCCA inherits one disadvantageous property from CCA: both methods compute estimates for covariance matrices based on all training data. This would prevent feasible training of their deep non-linear variants, since deep neural nets (DNNs) are predominantly optimized via stochastic optimization algorithms. To resolve this major hindrance, we propose an effective optimization algorithm for DPCCA, inspired by the work of Wang et al. (2015b) on Deep CCA (DCCA) optimization.

We evaluate our DPCCA architecture on two semantic tasks: **1)** multilingual word similarity and **2)** cross-lingual image description retrieval. For the former, we construct and provide to the community a new Word-Image-Word (WIW) dataset containing bilingual lexicons for three languages with shared images for 5K+ concepts. WIW is used as training data for word similarity experiments, while evaluation is conducted on the standard multilingual SimLex-999 dataset (Hill et al., 2015; Leviant and Reichart, 2015).

The results reveal stable improvements over a large space of non-deep and deep CCA-style baselines in both tasks. Most importantly, **1)** PCCA is overall better than other methods which do not use the additional perceptual view; **2)** DPCCA outperforms PCCA, indicating the importance of non-linear transformations modeled through DNNs; **3)** DPCCA outscores DCCA, again verifying the importance of conditioning multilingual text embedding induction on the shared visual view; and **4)** DPCCA outperforms two recent multi-modal bilingual models which also leverage visual information (Gella et al., 2017; Rajendran et al., 2016).

2 Related Work

This work is related to two research threads: **1)** multi-modal models that combine vision and language, with a focus on multilingual settings; **2)** correlational multi-view models based on CCA which learn a shared vector space for multiple views.

Multi-Modal Modeling in Multilingual Settings Research in cognitive science suggests that human meaning representations are grounded in our perceptual system and sensori-motor experience (Harnad, 1990; Lakoff and Johnson, 1999; Louwerse, 2011). Visual context serves as a useful cross-lingual grounding signal (Bruni et al., 2014; Glavaš et al., 2017) due to its language invariance, even enabling the induction of word-level bilingual semantic spaces solely through tagged images obtained from the Web (Bergsma and Van Durme, 2011; Kiela et al., 2015). Vulić et al. (2016) combine text embeddings with visual features via simple techniques of concatenation and averaging to obtain bilingual multi-modal representations, with noted improvements over text-only embeddings on word similarity and bilingual lexicon extraction. However, similar to the monolingual model of Kiela and Bottou (2014), their models lack the training phase, and require the visual signal at test time.

Recent work from Gella et al. (2017) exploits visual content as a bridge between multiple languages by optimizing a contrastive loss function. Furthermore, Rajendran et al. (2016) extend the work of Chandar et al. (2016) and propose to use a pivot representation in multimodal multilingual setups, with English representations serving as the pivot. While these works learn shared multimodal multilingual vector spaces, we demonstrate improved performance with our models (see §7).

Finally, although not directly comparable, recent work in neural machine translation has constructed models that can translate image descriptions by additionally relying on visual features of the image provided (Calixto and Liu, 2017; Elliott et al., 2015; Hitschler et al., 2016; Huang et al., 2016; Nakayama and Nishida, 2017, *inter alia*).

Correlational Models CCA-based techniques support multiple views on related data: e.g., when coupled with a bilingual dictionary, input monolingual word embeddings for two different languages can be seen as two views of the same latent semantic signal. Recently, CCA-based models for bilingual text embedding induction were proposed. These models rely on the basic CCA model (Chandar et al., 2016; Faruqui and Dyer, 2014), its deep variant (Lu et al., 2015), and a CCA extension which supports more than two views (Funaki and Nakayama, 2015; Rastogi et al., 2015). In this work, we propose to use (D)PCCA, which organically supports our setup: it conditions the two (textual) views on a shared (visual) view.

CCA-based methods (including PCCA) require the estimation of covariance matrices over *all* training data (Kessy et al., 2017). This hinders the use of DNNs with these models, as DNNs are typically trained via stochastic optimization over mini-batches on very large training sets. To address this limitation, various optimization methods for Deep CCA were proposed. Andrew et al. (2013) use L-BFGS (Byrd et al., 1995) over all training samples, while Arora and Livescu (2013) and Yan and Mikolajczyk (2015) train with large batches. However, these methods suffer from high memory complexity with unstable numerical computations.

Wang et al. (2015b) have recently proposed a stochastic approach for CCA and DCCA which copes well with small and large batch sizes while preserving high model performance. They use orthogonal iterations to estimate a moving average of the covariance matrices, which improves memory consumption. Therefore, we base our novel optimization algorithm for DPCCA on this approach.

3 Methodology: Deep Partial CCA

Given two image descriptions $\boldsymbol{x}$ and $\boldsymbol{y}$ in two languages and an image $\boldsymbol{z}$ that they refer to, the task is to learn a shared bilingual space such that similar descriptions obtain similar representations in the induced space. The image $\boldsymbol{z}$ serves as a shared third view on the textual data during training. The representation model is then utilized in cross-lingual and monolingual tasks. In this paper we focus on the more realistic scenario where no relevant visual content is available at test time. For this goal we propose a novel Deep Partial CCA (DPCCA) framework.

In what follows, we first review the CCA model and its deep variant: DCCA. We then introduce our DPCCA architecture, and describe our new stochastic optimization algorithm for DPCCA.

3.1 CCA and Deep CCA

DCCA (Andrew et al., 2013) extends CCA by learning non-linear (instead of linear) transformations of features contained in the input matrices $\boldsymbol{X} \in \mathbb{R}^{D_x \times N}$ and $\boldsymbol{Y} \in \mathbb{R}^{D_y \times N}$, where D_x and D_y are input vector dimensionalities, and N is the number of input items. Since CCA is a special case of the non-linear DCCA (see below), we here briefly outline the more general DCCA model.

The DCCA architecture is illustrated in Figure 1a. Non-linear transformations are achieved through two DNNs $\boldsymbol{f} : \mathbb{R}^{D_x \times N} \rightarrow \mathbb{R}^{D'_x \times N}$ and $\boldsymbol{g} : \mathbb{R}^{D_y \times N} \rightarrow \mathbb{R}^{D'_y \times N}$ for $\boldsymbol{X}$ and $\boldsymbol{Y}$. D'_x and D'_y are the output dimensionalities. A final linear layer is added to resemble the linear CCA projection.

The goal is to project the features of $\boldsymbol{X}$ and $\boldsymbol{Y}$ into a shared L-dimensional ($1 \leq L \leq min(D'_x, D'_y)$) space such that the canonical correlation of the final outputs $\boldsymbol{F}(\boldsymbol{X}) = \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{f}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y}) = \boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{g}(\boldsymbol{Y})$ is maximized. $\boldsymbol{W} \in \mathbb{R}^{D'_x \times L}$ and $\boldsymbol{V} \in \mathbb{R}^{D'_y \times L}$ are projection matrices: they project the final outputs of the DNNs to the shared space. $\boldsymbol{W_f}$ and $\boldsymbol{V_g}$ (the parameters of $\boldsymbol{f}$ and $\boldsymbol{g}$) and the projection matrices are the model parameters: $\boldsymbol{W_F} = \{\boldsymbol{W_f}, \boldsymbol{W}\}$; $\boldsymbol{V_G} = \{\boldsymbol{V_g}, \boldsymbol{V}\}$.[2] Formally, the DCCA objective can be written as:

$$\max_{\boldsymbol{W_F},\boldsymbol{V_G}} Tr(\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FG}}) \quad \text{so that } \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} = \hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} = \boldsymbol{I}. \tag{1}$$

$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FG}} \equiv \frac{1}{N-1}\boldsymbol{F}(\boldsymbol{X})\boldsymbol{G}(\boldsymbol{Y})^{\boldsymbol{T}}$ is the estimation of the cross-covariance matrix of the outputs, and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} \equiv \frac{1}{N-1}\boldsymbol{F}(\boldsymbol{X})\boldsymbol{F}(\boldsymbol{X})^{\boldsymbol{T}}$, $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} \equiv \frac{1}{N-1}\boldsymbol{G}(\boldsymbol{Y})\boldsymbol{G}(\boldsymbol{Y})^{\boldsymbol{T}}$ are the estimations of the auto-covariance matrices of the outputs.[3] Further, following Wang et al. (2015b), the optimal solution of Eq. (1) is equivalent to the optimal solution of the following:

$$\min_{\boldsymbol{W_F},\boldsymbol{V_G}} \frac{1}{N-1}\|\boldsymbol{F}(\boldsymbol{X}) - \boldsymbol{G}(\boldsymbol{Y})\|_F^2 \quad s.t.\ \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} = \hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} = \boldsymbol{I}. \tag{2}$$

The main disadvantage of DCCA is its inability to support more than two views, and to learn conditioned on an additional shared view, which is why we introduce Deep Partial CCA.

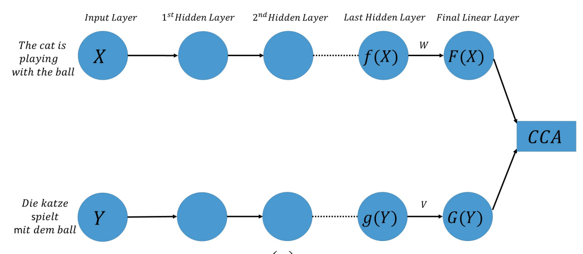
3.2 New Model: Deep Partial CCA

Figure 1b illustrates the architecture of DPCCA. The training data now consists of triplets $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i})_{1=1}^{N}$ from three views, forming the columns of $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$, where $\boldsymbol{x_i} \in \mathbb{R}^{D_x}, \boldsymbol{y_i} \in \mathbb{R}^{D_y}, \boldsymbol{z_i} \in \mathbb{R}^{D_z}$ for $i = 1, \ldots, N$. The objective is to maximize the canonical correlation of the first two views $\boldsymbol{X}$ and $\boldsymbol{Y}$ conditioned on the shared third variable $\boldsymbol{Z}$. Following Rao (1969)'s work on Partial CCA, we first consider two multivariate linear multiple regression models:

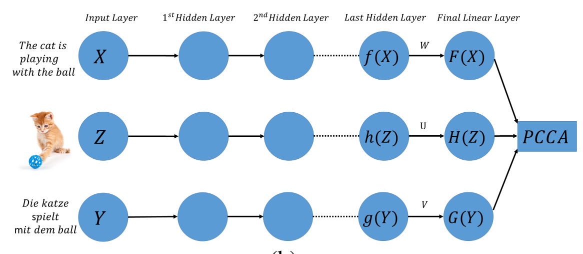
$$\boldsymbol{F}(\boldsymbol{X}) = \boldsymbol{AZ} + \boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z}), \tag{3}$$
$$\boldsymbol{G}(\boldsymbol{Y}) = \boldsymbol{BZ} + \boldsymbol{G}(\boldsymbol{Y}|\boldsymbol{Z}). \tag{4}$$

[2]For notational simplicity, we assume $\boldsymbol{f}(\boldsymbol{X})$ and $\boldsymbol{g}(\boldsymbol{Y})$ to have zero-means, otherwise it is possible to centralize them at the final layer of each network to the same effect.

[3]The CCA model can be seen as a special (linear) case of the more general DCCA model. The basic CCA objective can be recovered from the DCCA objective by simply setting $D'_x = D_x$, $D'_y = D_y$ and $\boldsymbol{f}(\boldsymbol{X}) = id_X$, $\boldsymbol{g}(\boldsymbol{Y}) = id_Y$; id is the identity mapping.



(a)



(b)

Figure 1: DCCA and DPCCA architectures. **(a)**: DCCA. $\boldsymbol{X}$ and $\boldsymbol{Y}$ (English and German image descriptions) are fed through two identical deep feed-forward neural networks followed by a final linear layer. The final nodes of the networks $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y})$ are then maximally correlated via the CCA objective. **(b)**: DPCCA. In addition, a third (shared) variable $\boldsymbol{Z}$ (an image) is either optimized via an identical architecture of the two main views (DPCCA Variant B, illustrated here) or kept fixed (DPCCA Variant A). The final nodes of the networks $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y})$ are maximally correlated conditioned on the final node in the middle network $\boldsymbol{H}(\boldsymbol{Z})$ (or directly on the input node $\boldsymbol{Z}$ in DPCCA Variant A).

$\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{L \times D_z}$ are matrices of coefficients, and $\boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z}), \boldsymbol{G}(\boldsymbol{Y}|\boldsymbol{Z}) \in \mathbb{R}^{L \times N}$ are normal random error matrices: residuals. We then minimize the mean-squared error regression criterion:

$$\min_{\boldsymbol{A}} \frac{1}{N-1} \|\boldsymbol{F}(\boldsymbol{X}) - \boldsymbol{A}\boldsymbol{Z}\|_F^2, \tag{5}$$

$$\min_{\boldsymbol{B}} \frac{1}{N-1} \|\boldsymbol{G}(\boldsymbol{Y}) - \boldsymbol{B}\boldsymbol{Z}\|_F^2. \tag{6}$$

After obtaining the optimal solutions for the coefficients, $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$, the residuals are as follows:

$$\begin{aligned}\boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z}) &= \boldsymbol{F}(\boldsymbol{X}) - \hat{\boldsymbol{A}}\boldsymbol{Z} \\ &= \boldsymbol{F}(\boldsymbol{X}) - \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FZ}} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{ZZ}}^{-1} \boldsymbol{Z}.\end{aligned} \tag{7}$$

$\boldsymbol{G}(\boldsymbol{Y}|\boldsymbol{Z})$ is computed in the analogous manner, now relying on $\boldsymbol{G}(\boldsymbol{Y})$ and $\hat{\boldsymbol{B}}\boldsymbol{Z}$. $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{S'Z}} \equiv \frac{1}{N-1}\boldsymbol{S}\boldsymbol{Z}^{\boldsymbol{T}}$ refers to the covariance matrix estimator of $\boldsymbol{S'}$ and $\boldsymbol{Z}$, where $(\boldsymbol{S'}, \boldsymbol{S}) \in \{(\boldsymbol{F}, \boldsymbol{F}(\boldsymbol{X})), (\boldsymbol{G}, \boldsymbol{G}(\boldsymbol{Y})), (\boldsymbol{Z}, \boldsymbol{Z})\}$.[4]

The canonical correlation between the residual matrices $\boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z})$ and $\boldsymbol{G}(\boldsymbol{Y}|\boldsymbol{Z})$ is referred to as the *partial canonical correlation*. The Deep PCCA objective can be obtained by replacing $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y})$ with their residuals in Eq. (2):

$$\begin{aligned}&\min_{\boldsymbol{W_F}, \boldsymbol{V_G}} \frac{1}{N-1} \|\boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z}) - \boldsymbol{G}(\boldsymbol{Y}|\boldsymbol{Z})\|_F^2 \\ &s.t.\ \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}|\boldsymbol{Z}} = \hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}|\boldsymbol{Z}} = \boldsymbol{I}.\end{aligned} \tag{8}$$

The computation of the conditional covariance matrix $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}|\boldsymbol{Z}}$ can be formulated as follows:

$$\begin{aligned}\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}|\boldsymbol{Z}} &\equiv \frac{1}{N-1} \boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z}) \boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z})^{\boldsymbol{T}} \\ &= \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} - \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FZ}} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{ZZ}}^{-1} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FZ}}^{\boldsymbol{T}}.\end{aligned} \tag{9}$$

[4] A small value $\epsilon > 0$ is added to the main diagonal of the covariance estimators for numerical stability.

The other conditional covariance matrix $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}|\boldsymbol{Z}}$ is again computed in the analogous manner, replacing $\boldsymbol{F}$ with $\boldsymbol{G}$ and $\boldsymbol{X}$ with $\boldsymbol{Y}$.[5]

While the (D)PCCA objective is computed over the residuals, after the network is trained (using multilingual texts and corresponding images) we can compute the representations of $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y})$ at test time without having access to images (see the network structure in Figure 1b). This heuristic enables the use of DPCCA in a real-life scenario in which images are unavailable at test time, and its encouraging results are demonstrated in §7.

Model Variants We consider two DPCCA variants : **1)** in DPCCA Variant A, the shared view $\boldsymbol{Z}$ is kept fixed; **2)** DPCCA Variant B also optimizes over $\boldsymbol{Z}$, as illustrated in Figure 1b. Variant A may be seen as a special case of Variant B.[6]

Variant B learns a non-linear function of the shared variable, $\boldsymbol{H}(\boldsymbol{Z}) = \boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{h}(\boldsymbol{Z})$, during training, where $\boldsymbol{h} : \mathbb{R}^{D_z \times N} \to \mathbb{R}^{D_{z'} \times N}$ is a DNN having the same architecture as $\boldsymbol{f}$ and $\boldsymbol{g}$. $\boldsymbol{U} \in \mathbb{R}^{D_{z'} \times L}$ is the final linear layer of $\boldsymbol{H}$, such that overall, the additional parameters of the model are $\boldsymbol{U_H} = \{\boldsymbol{U_h}, \boldsymbol{U}\}$. Instead of assuming a linear connection between $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y})$ to $\boldsymbol{Z}$, as in Variant A, we now assume that the linear connection takes place with $\boldsymbol{H}(\boldsymbol{Z})$. This assumption

[5] The original PCCA objective can be recovered by setting $D'_x = D_x$, $D'_y = D_y$ and $\boldsymbol{f}(\boldsymbol{X}) = id_X, \boldsymbol{g}(\boldsymbol{Y}) = id_Y$.

[6] For Variant A, in order for $\boldsymbol{Z}$ to be on the same range of values as in $\boldsymbol{F}$ and $\boldsymbol{G}$, we pass it through the activation function of the network, $\boldsymbol{Z} = \sigma(\boldsymbol{Z})$. Due to space constraints we discuss DPCCA Variant A in the supplementary material only.

changes Eq. (3) and Eq. (4) to:[7]

$$F(X) = A' \cdot H(Z) + F(X|H(Z)), \quad (10)$$
$$G(Y) = B' \cdot H(Z) + G(Y|H(Z)). \quad (11)$$

4 DPCCA: Optimization Algorithm

Training deep variants of CCA-style multi-view models is non-trivial due to estimation on the entire training set related to whitening constraints (i.e., the orthogonality of covariance matrices). To overcome this issue, Wang et al. (2015b) proposed a stochastic optimization algorithm for DCCA via non-linear orthogonal iterations (DCCA_NOI). Relying on the solution for DCCA (§4.1), we develop a new optimization algorithm for DPCCA in §4.2.

4.1 Optimization of DCCA

The DCCA optimization from Wang et al. (2015b), fully provided in Algorithm 1, relies on three key steps. First, the estimation of the covariance matrices in the form of $\hat{\Sigma}_{FF_t}$ at time t is calculated by a moving average over the minibatches:

$$\hat{\Sigma}_{FF_t} \leftarrow \rho\hat{\Sigma}_{FF_{t-1}} + (1-\rho)\left(\frac{|b_t|}{N-1}\right)^{-1} F(X_{b_t})F(X_{b_t})^T. \quad (12)$$

b_t is the minibatch at time t, X_{b_t} is the current input matrix at time t, and $\rho \in [0, 1]$ controls the ratio between the overall covariance estimation and the covariance estimation of the current minibatch.[8] This step eliminates the need of estimating the covariances over all training data, as well as the inherent bias when the estimate relies only on the current minibatch.

Second, the DCCA_NOI algorithm forces the whitening constraints to hold by performing an explicit matrix transformation in the form of:

$$\widetilde{F(X_{b_t})} = \hat{\Sigma}_{FF_t}^{-\frac{1}{2}} F(X_{b_t}). \quad (13)$$

According to Horn et al. (1988), if $\rho = 0$:

$$\left(\frac{|b_t|}{N-1}\right)^{-1} \widetilde{F(X_{b_t})}\widetilde{F(X_{b_t})}^T = I. \quad (14)$$

Finally, in order to optimize the DCCA objective (see Eq. (2)), the weights of the two DNNs are decoupled: i.e., the objective is disassembled into two separate mean-squared error objectives. Instead of trying to bring $F(X_{b_t})$ and $G(Y_{b_t})$ closer in one gradient descent step, two steps are performed: one of the views is fixed, and a gradient step over the other is performed, and so on, iteratively. The final objective functions at each time step are:

$$\min_{W_F} \frac{1}{|b_t|} \|F(X_{b_t}) - \widetilde{G(Y_{b_t})}\|_F^2, \quad (15)$$

$$\min_{V_G} \frac{1}{|b_t|} \|G(Y_{b_t}) - \widetilde{F(X_{b_t})}\|_F^2. \quad (16)$$

Wang et al. (2015b) show that the projection matrices W and V converge to the exact solutions of CCA as t$\rightarrow \infty$ when considering linear CCA.

Algorithm 1 The non-linear orthogonal iterations (NOI) algorithm for DCCA (DCCA_NOI)

```
Input: Data matrices $X \in \mathbb{R}^{D_x \times N}$, $Y \in \mathbb{R}^{D_y \times N}$, time constant $\rho$, learning rate $\eta$.

initialization: Initialize weights ($W_F$, $V_G$).
Randomly choose a minibatch ($X_{b_0}$, $Y_{b_0}$).
Initialize covariances:
$\hat{\Sigma}_{FF} \leftarrow \frac{N-1}{|b_0|} F(X_{b_0})F(X_{b_0})^T$
$\hat{\Sigma}_{GG} \leftarrow \frac{N-1}{|b_0|} G(Y_{b_0})G(Y_{b_0})^T$

  for $t = 1, 2, \ldots, n$ do
  Randomly choose a minibatch ($X_{b_t}$, $Y_{b_t}$).

  Update covariances:
  $\hat{\Sigma}_{FF} \leftarrow \rho\hat{\Sigma}_{FF} + (1-\rho)\frac{N-1}{|b_t|} F(X_{b_t})F(X_{b_t})^T$
  $\hat{\Sigma}_{GG} \leftarrow \rho\hat{\Sigma}_{GG} + (1-\rho)\frac{N-1}{|b_t|} G(Y_{b_t})G(Y_{b_t})^T$
  Fix $\widetilde{G(Y_{b_t})} = \hat{\Sigma}_{GG}^{-\frac{1}{2}} G(Y_{b_t})$, and compute $\nabla W_F$ with respect to:
  $\min_{W_F} \frac{1}{|b_t|} \|F(X_{b_t}) - \widetilde{G(Y_{b_t})}\|_F^2$
  Update parameters:
  $W_F \leftarrow W_F - \eta\nabla W_F$
  Fix $\widetilde{F(X_{b_t})} = \hat{\Sigma}_{FF}^{-\frac{1}{2}} F(X_{b_t})$, and compute $\nabla V_G$ with respect to:
  $\min_{V_G} \frac{1}{|b_t|} \|G(Y_{b_t}) - \widetilde{F(X_{b_t})}\|_F^2$
  Update parameters:
  $V_G \leftarrow V_G - \eta\nabla V_G$
  end for
Output: ($W_F$, $V_G$)
```

4.2 Optimization of DPCCA

Our DPCCA optimization is based on the DCCA_NOI algorithm with several adjustments. Besides the requirement to obtain the sample covariances $\hat{\Sigma}_{FF}$ and $\hat{\Sigma}_{GG}$, when calculating the conditional variables $F(X|Z)$, $G(Y|Z)$, $\hat{\Sigma}_{FF|Z}$ and $\hat{\Sigma}_{GG|Z}$, we additionally have to obtain the stochastic estimators $\hat{\Sigma}_{FZ}$, $\hat{\Sigma}_{GZ}$ and $\hat{\Sigma}_{ZZ}$. To this end, we use the moving average estimation from Eq. (12). Next, we define the whitening transformation on the residuals:

[7]Note that the matrices of coefficients A', $B' \in \mathbb{R}^{L \times L}$.

[8]Setting ρ to a high value indicates slow updates of the estimator; setting it low mostly erases the overall estimation and relies more on the current minibatch estimation.

$$\boldsymbol{F}(\widetilde{\boldsymbol{X}_{\boldsymbol{b_t}}|\boldsymbol{Z}}_{\boldsymbol{b_t}}) = \hat{\boldsymbol{\Sigma}}^{-\frac{1}{2}}_{\boldsymbol{FF_t|Z}}\boldsymbol{F}(\boldsymbol{X}_{\boldsymbol{b_t}}|\boldsymbol{Z}_{\boldsymbol{b_t}}), \tag{17}$$

$$\boldsymbol{G}(\widetilde{\boldsymbol{Y}_{\boldsymbol{b_t}}|\boldsymbol{Z}}_{\boldsymbol{b_t}}) = \boldsymbol{\Sigma}^{-\frac{1}{2}}_{\boldsymbol{GG_t|Z}}\boldsymbol{G}(\boldsymbol{Y}_{\boldsymbol{b_t}}|\boldsymbol{Z}_{\boldsymbol{b_t}}). \tag{18}$$

As before, the whitening constraints hold when ρ = 0. From here, we derive our two final objective functions over the residuals at time t:

$$\min_{\boldsymbol{W_F}} \frac{1}{|b_t|}\|\boldsymbol{F}(\boldsymbol{X}_{\boldsymbol{b_t}}|\boldsymbol{Z}_{\boldsymbol{b_t}}) - \boldsymbol{G}(\widetilde{\boldsymbol{Y}_{\boldsymbol{b_t}}|\boldsymbol{Z}}_{\boldsymbol{b_t}})\|^2_F, \tag{19}$$

$$\min_{\boldsymbol{V_G}} \frac{1}{|b_t|}\|\boldsymbol{G}(\boldsymbol{Y}_{\boldsymbol{b_t}}|\boldsymbol{Z}_{\boldsymbol{b_t}}) - \boldsymbol{F}(\widetilde{\boldsymbol{X}_{\boldsymbol{b_t}}|\boldsymbol{Z}}_{\boldsymbol{b_t}})\|^2_F. \tag{20}$$

Equivalently to Eq. (15)-(16) that replace Eq. (2), Eq. (19)-(20) replace Eq. (8) by performing stochastic, decoupled and unconstrained steps. As our algorithm performs CCA over the residuals, we gain the same guarantees as Wang et al. (2015b), now for the projection matrices of the residuals.

Algorithm 2 shows the full optimization procedure for the more complex DPCCA Variant B. The full algorithm for Variant A is provided in the supplementary material. The main difference is that with Variant B we replace $\boldsymbol{Z}$ with $\boldsymbol{H}(\boldsymbol{Z})$ in all equations where it appears, and we optimize over $\boldsymbol{U_H}$ along with $\boldsymbol{W_F}$ and $\boldsymbol{V_G}$ in Eq. (19) and Eq. (20), respectively.

Algorithm 2 The non-linear orthogonal iterations (NOI) algorithm for DPCCA Variant B

```
Input: Data matrices $\boldsymbol{X} \in \mathbb{R}^{D_x \times N}$, $\boldsymbol{Y} \in \mathbb{R}^{D_y \times N}$, $\boldsymbol{Z} \in \mathbb{R}^{D_z \times N}$, time constant $\rho$, learning rate $\eta$.

initialization: Initialize weights $(\boldsymbol{W_F}, \boldsymbol{V_G}, \boldsymbol{U_H})$.
Randomly choose a minibatch $(\boldsymbol{X_{b_0}}, \boldsymbol{Y_{b_0}}, \boldsymbol{Z_{b_0}})$.
Initialize covariances:
$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} \leftarrow \frac{N-1}{|b_0|}\boldsymbol{F}(\boldsymbol{X_{b_0}})\boldsymbol{F}(\boldsymbol{X_{b_0}})^{\boldsymbol{T}}$
$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} \leftarrow \frac{N-1}{|b_0|}\boldsymbol{G}(\boldsymbol{Y_{b_0}})\boldsymbol{G}(\boldsymbol{Y_{b_0}})^{\boldsymbol{T}}$
$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{HH}} \leftarrow \frac{N-1}{|b_0|}\boldsymbol{H}(\boldsymbol{Z_{b_0}})\boldsymbol{H}(\boldsymbol{Z_{b_0}})^{\boldsymbol{T}}$
$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FH}} \leftarrow \frac{N-1}{|b_0|}\boldsymbol{F}(\boldsymbol{X_{b_0}})\boldsymbol{H}(\boldsymbol{Z_{b_0}})^{\boldsymbol{T}}$
$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GH}} \leftarrow \frac{N-1}{|b_0|}\boldsymbol{G}(\boldsymbol{Y_{b_0}})\boldsymbol{H}(\boldsymbol{Z_{b_0}})^{\boldsymbol{T}}$

  for $t = 1, 2, \ldots, n$ do
  Randomly choose a minibatch $(\boldsymbol{X_{b_t}}, \boldsymbol{Y_{b_t}}, \boldsymbol{Z_{b_t}})$.
  Update covariances:
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} \leftarrow \rho\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} + (1-\rho)\frac{N-1}{|b_t|}\boldsymbol{F}(\boldsymbol{X_{b_t}})\boldsymbol{F}(\boldsymbol{X_{b_t}})^{\boldsymbol{T}}$
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} \leftarrow \rho\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} + (1-\rho)\frac{N-1}{|b_t|}\boldsymbol{G}(\boldsymbol{Y_{b_t}})\boldsymbol{G}(\boldsymbol{Y_{b_t}})^{\boldsymbol{T}}$
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{HH}} \leftarrow \rho\hat{\boldsymbol{\Sigma}}_{\boldsymbol{HH}} + (1-\rho)\frac{N-1}{|b_t|}\boldsymbol{H}(\boldsymbol{Z_{b_t}})\boldsymbol{H}(\boldsymbol{Z_{b_t}})^{\boldsymbol{T}}$
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FH}} \leftarrow \rho\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FH}} + (1-\rho)\frac{N-1}{|b_t|}\boldsymbol{F}(\boldsymbol{X_{b_t}})\boldsymbol{H}(\boldsymbol{Z_{b_t}})^{\boldsymbol{T}}$
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GH}} \leftarrow \rho\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GH}} + (1-\rho)\frac{N-1}{|b_t|}\boldsymbol{G}(\boldsymbol{Y_{b_t}})\boldsymbol{H}(\boldsymbol{Z_{b_t}})^{\boldsymbol{T}}$

  Update conditional variables:
  $\boldsymbol{F|H} \leftarrow \boldsymbol{F}(\boldsymbol{X_{b_t}}) - \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FH}}\hat{\boldsymbol{\Sigma}}^{-1}_{\boldsymbol{HH}}\boldsymbol{H}(\boldsymbol{Z_{b_t}})$
  $\boldsymbol{G|H} \leftarrow \boldsymbol{G}(\boldsymbol{Y_{b_t}}) - \hat{\boldsymbol{\Sigma}}_{\boldsymbol{GH}}\hat{\boldsymbol{\Sigma}}^{-1}_{\boldsymbol{HH}}\boldsymbol{H}(\boldsymbol{Z_{b_t}})$
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF|H}} \leftarrow \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FF}} - \hat{\boldsymbol{\Sigma}}_{\boldsymbol{FH}}\hat{\boldsymbol{\Sigma}}^{-1}_{\boldsymbol{HH}}\hat{\boldsymbol{\Sigma}}^{\boldsymbol{T}}_{\boldsymbol{FH}}$
  $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG|H}} \leftarrow \hat{\boldsymbol{\Sigma}}_{\boldsymbol{GG}} - \hat{\boldsymbol{\Sigma}}_{\boldsymbol{GH}}\hat{\boldsymbol{\Sigma}}^{-1}_{\boldsymbol{HH}}\hat{\boldsymbol{\Sigma}}^{\boldsymbol{T}}_{\boldsymbol{GH}}$

  Fix $\widetilde{\boldsymbol{G|H}} = \hat{\boldsymbol{\Sigma}}^{-\frac{1}{2}}_{\boldsymbol{GG|H}}\boldsymbol{G|H}$, and compute $\nabla\boldsymbol{W_F}$, $\nabla\boldsymbol{U_H}$ with respect to:
  $\min_{\boldsymbol{W_F},\boldsymbol{U_H}} \frac{1}{|b_t|}\|\boldsymbol{F|H} - \widetilde{\boldsymbol{G|H}}\|^2_F$
  Update parameters:
  $\boldsymbol{W_F} \leftarrow \boldsymbol{W_F} - \eta\nabla\boldsymbol{W_F}, \boldsymbol{U_H} \leftarrow \boldsymbol{U_H} - \eta\nabla\boldsymbol{U_H}$
  Fix $\widetilde{\boldsymbol{F|H}} = \hat{\boldsymbol{\Sigma}}^{-\frac{1}{2}}_{\boldsymbol{FF|H}}\boldsymbol{F|H}$, and compute $\nabla\boldsymbol{V_G}$, $\nabla\boldsymbol{U_H}$ with respect to:
  $\min_{\boldsymbol{V_G},\boldsymbol{U_H}} \frac{1}{|b_t|}\|\boldsymbol{G|H} - \widetilde{\boldsymbol{F|H}}\|^2_F$
  Update parameters:
  $\boldsymbol{V_G} \leftarrow \boldsymbol{V_G} - \eta\nabla\boldsymbol{V_G}, \boldsymbol{U_H} \leftarrow \boldsymbol{U_H} - \eta\nabla\boldsymbol{U_H}$
  end for
Output: $(\boldsymbol{W_F}, \boldsymbol{V_G}, \boldsymbol{U_H})$
```

5 Tasks and Data

Cross-lingual Image Description Retrieval The cross-lingual image description retrieval task is formulated as follows: taking an image description as a *query* in the source language, the system has to retrieve a set of *relevant descriptions* in the target language which describe the same image. Our evaluation assumes a *single-best* scenario, where only a single target description is relevant for each query. In addition, in our setup, images are not available during inference: retrieval is performed based solely on text queries. This enables a fair comparison between our model and many baseline models that cannot represent images and text in a shared space. Moreover, it allows us to test our model in the realistic setup where images are not available at test time. To avoid the use of images at retrieval time with DPCCA, we perform the retrieval on $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{Y})$, rather than on $\boldsymbol{F}(\boldsymbol{X}|\boldsymbol{Z})$ and $\boldsymbol{G}(\boldsymbol{Y}|\boldsymbol{Z})$ (see §3.2).

We use the Multi30K dataset (Elliott et al., 2016), originated from Flickr30K (Young et al., 2014) that is comprised of Flicker images described with 1-5 English descriptions per image. Multi30K adds German descriptions to a total of 30,014 images: most were written independently of the English descriptions, while some are direct translations. Each image is associated with one English and one German description. We rely on the original Multi30K splits with 29,000, 1,014, and 1,000 triplets for training, validation, and test, respectively.

Multilingual Word Similarity The word similarity task tests the correlation between automatic and human generated word similarity scores. We evaluate with the Multilingual SimLex-999 dataset (Leviant and Reichart, 2015): the 999 English (EN)

	EN-DE	EN-IT	EN-RU
Nouns	4606	4735	4106
Adjectives	405	416	348
Verbs	392	400	227
Adverbs	167	161	142
Prepositions	12	12	9
Total	**5598**	**5740**	**4838**

Table 1: WIW statistics: the number of WIW entries across POS classes in each language pair. The numbers of words per POS class are not summed to the total number of words as other (less frequent) POS tags are also represented.

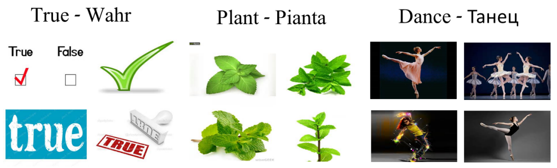


Figure 2: WIW examples from each of the three bilingual lexicons. Note that the designated words can be either abstract (*true*), express an action (*dance*) or be more concrete (*plant*).

word pairs from SimLex-999 (Hill et al., 2015) were translated to German (DE), Italian (IT), and Russian (RU), and similarity scores were crowdsourced from native speakers.

We introduce a new dataset termed *Word-Image-Word* (WIW), which we use to train word-level models for the multilingual word similarity task. WIW contains three bilingual lexicons (EN-DE, EN-IT, EN-RU) with images shared between words in a lexicon entry. Each WIW entry is a triplet: an English word, its translation in DE/IT/RU, and a set of images relevant to the pair.

English words were taken from the January 2017 Wikipedia dump. After removing stop words and punctuation, we extract the 6,000 most frequent words from the cleaned corpus not present in SimLex. DE/IT/RU words were obtained semi-automatically from the EN words using Google Translate. The images are crawled from the Bing search engine using MMFeat[9] (Kiela, 2016) by querying the EN words only. Following the suggestions from the study of Kiela et al. (2016), we save the top 20 images as relevant images.[10]

Table 1 provides a summary of the WIW dataset. The dataset contains both concrete and abstract words, and words of different POS tags.[11] This property has an influence on the image collection: similar to Kiela et al. (2014), we have noticed that images of more concrete concepts are less dispersed (see also examples from Figure 2).

6 Experimental Setup

Data Preprocessing and Embeddings For the sentence-level task, all descriptions were lowercased and tokenized. Each sentence is represented with one vector: the average of its word embeddings. For English, we rely on 500-dimensional English skip-gram word embeddings (Mikolov et al., 2013) trained on the January 2017 Wikipedia dump with bag-of-words contexts (window size of 5). For German we use the deWaC 1.7B corpus (Baroni et al., 2009) to obtain 500-dimensional German embeddings using the same word embedding model. For word similarity, to be directly comparable to previous work, we rely on 300-dim word vectors in EN, DE, IT, and RU from Mrkšić et al. (2017).

Visual features are extracted from the penultimate layer (FC7) of the VGG-19 network (Simonyan and Zisserman, 2015), and compressed to the dimensionality of the textual inputs by a Principal Component Analysis (PCA) step. For the word similarity task, we average the visual vectors across all images of each word pair as done in, e.g., (Vulić et al., 2016), before the PCA step.

Baseline Models We consider a wide variety of multi-view CCA-based baselines. First, we compare against the original (linear) **CCA** model (Hotelling, 1936), and its deep non-linear extension **DCCA** (Andrew et al., 2013). For DCCA: 1) we rely on its improved optimization algorithm from Wang et al. (2015a) which uses a stochastic approach with large minibatches; 2) we compare against the **DCCA_NOI** variant (Wang et al., 2015b) described by Algorithm 1, and another recent DCCA variant with the optimization algorithm based on a stochastic decorrelational loss (Chang et al., 2017) (**DCCA_SDL**); and 3) we also test the DCCA Autoencoder model (**DCCAE**) (Wang et al., 2015a), which offers a trade-off between maximizing the canonical correlation of two sets of variables and finding informative features for their reconstruction.

Another baseline is Generalized CCA (**GCCA**) (Funaki and Nakayama, 2015; Horst, 1961; Rastogi et al., 2015): a linear model which extends CCA to

[9]https://github.com/douwekiela/mmfeat.
[10]Offensive words and images are manually cleaned.
[11]POS tag information is taken from the NLTK toolkit for the English words.

three or more views. Unlike PCCA, GCCA does not condition two variables on the third shared one, but rather seeks to maximize the canonical correlations of all pairs of views. We also compare to Nonparametric CCA (**NCCA**) (Michaeli et al., 2016), and to a probabilistic variant of PCCA (**PPCCA**, Mukuta and Harada (2014)).

Finally, we compare with the two recent models which operate in the setup most similar to ours: 1) Bridge Correlational Networks (**BCN**) (Rajendran et al., 2016); and 2) Image Pivoting (**IMG_PIVOT**) from Gella et al. (2017). For both models, we report results only with the strongest variant based on the findings from the original papers, also verified by additional experimentation in our work.[12]

Hyperparameter Tuning The hyperparameters of the different models are tuned with a grid search over the following values: {2,3,4,5} for number of layers, {*tanh, sigmoid, ReLU*} as the activation functions (we use the same activation function in all the layers of the same network), {64,128,256} for minibatch size, {0.001,0.0001} for learning rate, and {128,256} for L (the size of the output vectors). The dimensions of all mid-layers are set to the input size. We use the Adam optimizer (Kingma and Ba, 2015), with the number of epochs set to 300.

For all participating models, we report test performance of the best hyperparameter on the validation set. For word similarity, following a standard practice (Levy et al., 2015; Vulić et al., 2017) we tune all models on one half of the SimLex data and evaluate on the other half, and vice versa. The reported score is the average of the two halves. Similarity scores for all tasks were computed using the *cosine similarity* measure.

7 Results and Discussion

Cross-lingual Image Description Retrieval We report two standard evaluation metrics: 1) *Recall at 1* (R@1) scores, and 2) the sentence-level *BLEU+1* metric (Lin and Och, 2004), a variant of BLEU which smooths terms for higher-order n-grams, making it more suitable for evaluating short sentences. The scores for the retrieval task with all models are summarized in Table 2.

[12] More details about preprocessing and baselines (including all *links to their code*), are in the the supplementary material. We use original readily available implementations of all baselines whenever this is possible, and our in-house implementations for baselines for which no code is provided by the original authors.

	R@1		**BLEU+1**	
Model	EN→DE	DE→EN	EN→DE	DE→EN
DPCCA (Variant A)	0.795	0.779	0.836	0.827
DPCCA (Variant B)	0.809	**0.794**	0.848	**0.839**
DPCCA(B)+DCCA_NOI (concat)	**0.826**	0.791	**0.863**	0.837
DCCA_NOI (Wang et al., 2015b)	0.812	0.788	0.849	0.830
DCCA_SDL (Chang et al., 2017)	0.507	0.487	0.552	0.533
DCCA (Wang et al., 2015a)	0.619	0.621	0.664	0.673
DCCAE (Wang et al., 2015a)	0.564	0.542	0.607	0.598
IMG_PIVOT (Gella et al., 2017)	0.772	0.763	0.789	0.781
BCN (Rajendran et al., 2016)	0.579	0.570	0.628	0.629
PCCA (Rao, 1969)	<u>0.785</u>	<u>0.737</u>	<u>0.825</u>	<u>0.787</u>
CCA (Hotelling, 1936)	0.764	0.704	0.803	0.754
GCCA (Funaki and Nakayama, 2015)	0.699	0.690	0.742	0.743
NCCA (Michaeli et al., 2016)	0.157	0.165	0.205	0.213
PPCCA (Mukuta and Harada, 2014)	0.035	0.050	0.063	0.086

Table 2: Results on cross-lingual image description retrieval. NN-based models are above the dashed line. Best overall results are in bold. Best results with non-deep models are underlined.

The results clearly demonstrate the superiority of DPCCA (with a slight advantage to the more complex Variant B) and of the concatenation of their representation with that of the DCCA_NOI (strongest) baseline. Furthermore, the non-deep, linear PCCA achieves strong results: it outscores all non-deep models, as well as all deep models except from DCCA_NOI, IMG_PIVOT in one case, and its deep version: DPCCA. This emphasizes our contribution in proposing PCCA for multilingual processing with images as a cross-lingual bridge.

The results suggest that: 1) the inclusion of visual information in the training process helps the retrieval task even without such information during inference. DPCCA outscores all DCCA variants (either alone or through a concatenation with the DCCA_NOI representation), and PCCA outscores the original two-view CCA model; and 2) deep, non-linear architectures are useful: our DPCCA outperforms the linear PCCA model.

We also note clear improvements over the two recent models which also rely on visual information: IMG_PIVOT and BCN. The gain over IMG_PIVOT is observed despite the fact that IMG_PIVOT is a more complex multi-modal model which relies on RNNs, and is tailored to sentence-level tasks. Finally, the scores from Table 2 suggest that improved performance can be achieved by an ensemble model, that is, a simple concatenation of DPCCA (B) and DCCA_NOI.

Multilingual Word Similarity The results, presented as standard Spearman's rank correlation scores, are summarized in Table 3: we present fine-grained results over different POS classes for EN and DE, and compare them to the results from

Model	English-German					
	EN-Adj	EN-Verbs	EN-Nouns	DE-Adj	DE-Verbs	DE-Nouns
DPCCA (Variant A)	**0.640**	0.311	0.369	0.430	**0.321**	**0.404**
DPCCA (Variant B)	0.626	**0.316**	**0.382**	**0.462**	0.319	0.399
DCCA_NOI (Wang et al., 2015b)	0.611	0.308	0.361	0.441	0.297	0.398
DCCA (Wang et al., 2015a)	0.618	0.261	0.327	0.404	0.290	0.362
PCCA (Rao, 1969)	0.614	0.296	0.340	0.305	0.143	0.340
CCA (Hotelling, 1936)	0.557	0.297	0.321	0.284	0.157	0.346
GCCA (Funaki and Nakayama, 2015)	0.636	0.280	0.378	0.446	0.277	0.398
INIT_EMB	0.582	0.160	0.306	0.407	0.164	0.285

Table 3: Results on EN and DE SimLex-999 (POS-based evaluation). All scores are Spearman's rank correlations. INIT_EMB refers to initial pre-trained monolingual word embeddings (see §6).

Model	EN-DE WIW		EN-IT WIW		EN-RU WIW	
	EN	DE	EN	IT	EN	RU
DPCCA (A)	0.398	**0.400**	0.412	**0.429**	0.404	**0.407**
DPCCA (B)	**0.405**	**0.400**	0.413	0.427	**0.413**	0.402
PCCA	0.374	0.301	0.370	0.386	0.374	0.374
DCCA_NOI	0.390	0.398	0.413	0.422	0.407	0.398
GCCA	0.395	0.386	**0.414**	0.407	0.412	0.396
INIT_EMB	0.321	0.278	0.321	0.361	0.321	0.385

Table 4: Results (Spearman rank correlation) of our models and the strongest baselines on Multilingual SimLex-999 (all data).

a selection of strongest baselines. Further, Table 4 presents results on all SimLex word pairs. The POS class result patterns for EN-IT and EN-RU are very similar to the patterns in Table 3 and are provided in the supplementary material. First, the results over the initial monolingual embeddings before training (INIT_EMB) clearly indicate that multilingual information is beneficial for the word similarity task. We observe improvements with all models (the only exception being extremely low-scoring PPCCA and NCCA, not shown). Moreover, by additionally grounding concepts from two languages in the visual modality it is possible to further boost word similarity scores. This result is in line with prior work in monolingual settings (Chrupała et al., 2015; Kiela and Bottou, 2014; Lazaridou et al., 2015), which have shown to profit from multi-modal features.

The results on the POS classes represented in SimLex-999 (nouns, verbs, adjectives, Table 3) form our main finding: conditioning the multilingual representations on a shared image leads to improvements in verb and adjective representations. While for nouns one of the DPCCA variants is the best performing model for both languages, the gaps from the best performing baselines are much smaller. This is interesting since, e.g., verbs are more abstract than nouns (Hartmann and Søgaard, 2017; Hill et al., 2014). Considering the fact that SimLex-999 consists of 666 noun pairs, 222 verb pairs and 111 adjective pairs, this is the reason that the gains of DPCCA over the strongest baselines across the entire evaluation set are more modest (Table 4). We note again that the same patterns presented in Table 3 for EN-DE – more prominent verb and adjective gains and a smaller gain on nouns – also hold for EN-IT and EN-RU (see the supplementary material).

8 Conclusion and Future Work

We addressed the problem of utilizing images as a bridge between languages to learn improved bilingual text representations. Our main contribution is two-fold. First, we proposed to use the Partial CCA (PCCA) method. In addition, we proposed a stochastic optimization algorithm for the deep version of PCCA that overcomes the challenges posed by the covariance estimation required by the method. Our experiments reveal the effectiveness of these methods for both sentence-level and word-level tasks. Crucially, our proposed solution does not require access to images at inference/test time, in line with the realistic scenario where images that describe sentential queries are not readily available.

In future work we plan to improve our methods by exploiting the internal structure of images and sentences as well as by effectively integrating signals from more than two languages.

Acknowledgments

IV is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). GR and RR are supported by the Infomedia Magnet Grant and by an AOL grant on ”connected experience technologies”.

References

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *Proceedings of ICML*, pages 1247–1255.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, Lawrence C. Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of ICCV*, pages 2425–2433.

Raman Arora and Karen Livescu. 2013. Multi-view CCA-based acoustic features for phonetic recognition across speakers and domains. In *Proceedings of ICASSP*, pages 7135–7139.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Lawrence W. Barsalou and Katja Wiemer-Hastings. 2005. Situating abstract concepts. In D. Pecher and R. Zwaan, editors, *Grounding cognition: The role of perception and action in memory, language, and thought*, pages 129–163.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of IJCAI*, pages 1764–1769.

Elia Bruni, Nam Khanh Tram, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Iacer Calixto and Qun Liu. 2017. Incorporating global visual features into attention-based neural machine translation. In *Proceedings of EMNLP*, pages 992–1003.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Multilingual multi-modal embeddings for natural language processing. *arXiv preprint arXiv:1702.01101*.

Sarath Chandar, Mitesh M Khapra, Hugo Larochelle, and Balaraman Ravindran. 2016. Correlational neural networks. *Neural Computation*, 28:257–285.

Xiaobin Chang, Tao Xiang, and Timothy M. Hospedales. 2017. Deep multi-view learning with stochastic decorrelation loss. *CoRR*, abs/1707.09669.

Grzegorz Chrupała, Ákos Kádár, and Afra Alishahi. 2015. Learning language through pictures. In *Proceedings of ACL*, pages 112–118.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multilingual image description with neural sequence models. *arXiv preprint arXiv:1510.04709*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Proceedings of ECCV*, pages 15–29.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, pages 462–471.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Proceedings of NAACL-HLT*, pages 91–99.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of EMNLP*, pages 457–468.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of EMNLP*, pages 585–590.

Spandana Gella, Mirella Lapata, and Frank Keller. 2016. Unsupervised visual sense disambiguation for verbs using multimodal embeddings. In *Proceedings of NAACL-HLT*, pages 182–192.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations. In *Proceedings of EMNLP*, pages 2839–2845.

Goran Glavaš, Ivan Vulić, and Simone Paolo Ponzetto. 2017. If sentences could see: Investigating visual information for semantic textual similarity. In *Proceedings of IWCS*.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1–3).

Mareike Hartmann and Anders Søgaard. 2017. Limitations of cross-lingual learning from image search. *CoRR*, abs/1709.05914.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for concrete and abstract concept meaning. *Transactions of the ACL*, 2:285–296.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal pivots for image caption translation. In *Proceedings of ACL*, pages 2399–2409.

Berthold K.P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. 1988. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of Optical Society of America*, 5(7):1127–1135.

Paul Horst. 1961. Generalized canonical correlations and their applications to experimental data. *Journal of Clinical Psychology*, 17(4):331–347.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of WMT*, pages 639–645.

Agnan Kessy, Alex Lewin, and Korbinian Strimmer. 2017. Optimal whitening and decorrelation. *The American Statistician*.

Douwe Kiela. 2016. MMFeat: A toolkit for extracting multi-modal features. In *Proceedings of ACL System Demonstrations*, pages 55–60.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP*, pages 36–45.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of ACL*, pages 835–841.

Douwe Kiela, Anita Lilla Verő, and Stephen Clark. 2016. Comparing data sources and architectures for deep visual representation learning in semantics. In *Proceedings of EMNLP*, pages 447–456.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred ConvNet features. In *Proceedings of EMNLP*, pages 148–158.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR (Conference Track)*.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of ICML*, pages 595–603.

George Lakoff and Mark Johnson. 1999. *Philosophy in the flesh: The embodied mind and its challenge to Western thought*.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of NAACL-HLT*, pages 153–163.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR, abs/1508.00106*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: A method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of COLING*, pages 501–507.

Max M. Louwerse. 2011. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science*, 59(1):617–645.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of NAACL-HLT*, pages 250–256.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-RNN). In *Proceedings of ICLR (Conference Track)*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L Yuille. 2014. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*.

Tomer Michaeli, Weiran Wang, and Karen Livescu. 2016. Nonparametric canonical correlation analysis. In *Proceedings of ICML*, pages 1967–1976.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR (Conference Track)*.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5(1):309–324.

Yusuke Mukuta and Harada. 2014. Probabilistic partial canonical correlation analysis. In *Proceedings of ICML*, pages 1449–1457.

Hideki Nakayama and Noriki Nishida. 2017. Zero-resource machine translation by multimodal encoder–decoder network with multimedia pivot. *Machine Translation*, 31(1-2):49–64.

Janarthanan Rajendran, Mitesh M. Khapra, Sarath Chandar, and Balaraman Ravindran. 2016. Bridge correlational neural networks for multilingual multimodal representation learning. In *Proceedings of NAACL-HLT*, pages 171–181.

B. Raja Rao. 1969. Partial canonical correlations. *Trabajos de estadistica y de investigación operativa*, 20(2-3):211–219.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of NAACL-HLT*, pages 556–566.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of ICLR (Workshop Track)*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of CVPR*, pages 3156–3164.

Ivan Vulić, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-modal representations for improved bilingual lexicon learning. In *Proceedings of ACL*, pages 188–194. ACL.

Ivan Vulić, Roy Schwartz, Ari Rappoport, Roi Reichart, and Anna Korhonen. 2017. Automatic selection of context configurations for improved class-specific word representations. In *Proceedings of CoNLL*, pages 112–122.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. 2015a. On deep multi-view representation learning. In *Proceedings of ICML*, pages 1083–1092.

Weiran Wang, Raman Arora, Karen Livescu, and Nathan Srebro. 2015b. Stochastic optimization for deep CCA via nonlinear orthogonal iterations. In *Proceedings of Communication, Control, and Computing*, pages 688–695.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *Proceedings of ECCV*, pages 451–466.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*, pages 2048–2057.

Fei Yan and Krystian Mikolajczyk. 2015. Deep correlation for matching images and text. In *Proceedings of CVPR*, pages 3441–3450.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the ACL*, 2:67–78.

Illustrative Language Understanding: Large-Scale Visual Grounding with Image Search

Jamie Ryan Kiros* **William Chan*** **Geoffrey E. Hinton**

Google Brain Toronto
{kiros, williamchan, geoffhinton}@google.com

Abstract

We introduce Picturebook, a large-scale lookup operation to ground language via 'snapshots' of our physical world accessed through image search. For each word in a vocabulary, we extract the top-k images from Google image search and feed the images through a convolutional network to extract a word embedding. We introduce a multimodal gating function to fuse our Picturebook embeddings with other word representations. We also introduce Inverse Picturebook, a mechanism to map a Picturebook embedding back into words. We experiment and report results across a wide range of tasks: word similarity, natural language inference, semantic relatedness, sentiment/topic classification, image-sentence ranking and machine translation. We also show that gate activations corresponding to Picturebook embeddings are highly correlated to human judgments of concreteness ratings.

1 Introduction

Constructing grounded representations of natural language is a promising step towards achieving human-like language learning. In recent years, a large amount of research has focused on integrating vision and language to obtain visually grounded word and sentence representations. One source of grounding, which has been utilized in existing work, is image search engines. Search engines allow us to obtain correspondences between language and images that are far less restricted than existing multimodal datasets which typically have restricted vocabularies. While true natural language understanding may require fully embodied cognition, search engines allow us to get a form of quasi-grounding from high-coverage 'snapshots' of our physical world provided by the interaction of millions of users.

One place to incorporate grounding is in the lookup table that maps tokens to vectors. The dominant approach to learning distributed word representations is through indexing a learned matrix. While immensely successful, this lookup operation is typically learned through co-occurrence objectives or a task-dependent reward signal. A very different way to obtain word embeddings is to aggregate features obtained by using the word as a query for an image search engine. This involves retrieving the top-k images from a search engine, running those through a convolutional network and aggregating the results. These word embeddings are grounded via the retrieved images. While several authors have considered this approach, it has been largely limited to a few thousand queries and only a small number of tasks.

In this paper we introduce Picturebook embeddings produced by image search using words as queries. Picturebook embeddings are obtained through a convolutional network trained with a semantic ranking objective on a proprietary image dataset with over 100+ million images (Wang et al., 2014). Using Google image search, a Picturebook embedding for a word is obtained by concatenating the k-feature vectors of our convolutional network on the top-k retrieved search results. The main contributions of our work are as follows:

- We obtain Picturebook embeddings for the 2.2 million words that occur in the Glove vocabulary (Pennington et al., 2014) [1], allowing each word to have a Glove embedding and a parallel grounded word representation. This collection of word representations that we visually

*Both authors contributed equally to this work.

[1]Common Crawl, 840B tokens

ground via image search is 2-3 orders of magnitude larger than prior work.

- We introduce a multimodal gating mechanism to selectively choose between Glove and Picturebook embeddings in a task-dependent way. We apply our approach to over a dozen datasets and several different tasks: word similarity, sentence relatedness, natural language inference, topic/sentiment classification, image sentence ranking and Machine Translation (MT).
- We introduce Inverse Picturebook to perform the inverse lookup operation. Given a Picturebook embedding, we find the closest words which would generate the embedding. This is useful for generative modelling tasks.
- We perform an extensive analysis of our gating mechanism, showing that the gate activations for Picturebook embeddings are highly correlated with human judgments of concreteness. We also show that Picturebook gate activations are negatively correlated with image dispersion (Kiela et al., 2014), indicating that our model selectively chooses between word embeddings based on their abstraction level.
- We highlight the importance of the convolutional network used to extract embeddings. In particular, networks trained with semantic labels result in better embeddings than those trained with visual labels, even when evaluating similarity on concrete words.

2 Related Work

The use of image search for obtaining word representations is not new. Table 1 illustrates existing methods that utilize image search and the tasks considered in their work. There has also been other work using other image sources such as ImageNet (Kiela and Bottou, 2014; Collell and Moens, 2016) over the WordNet synset vocabulary, and using Flickr photos and captions (Joulin et al., 2016). Our approach differs from the above methods in three main ways: a) we obtain search-grounded representations for over 2 million words as opposed to a few thousand, b) we apply our representations to a higher diversity of tasks than previously considered, and c) we introduce a multimodal gating mechanism that allows for a more flexible integration of features than mere concatenation.

Our work also relates to existing multimodal models combining different representations of the data (Hill and Korhonen, 2014). Various work has also fused text-based representations with image-based representations (Bruni et al., 2014; Lazaridou et al., 2015; Chrupala et al., 2015; Mao et al., 2016; Silberer et al., 2017; Kiela et al., 2017; Collell et al., 2017; Zablocki et al., 2018) and representations derived from a knowledge-graph (Thoma et al., 2017). More recently, gating-based approaches have been developed for fusing traditional word embeddings with visual representations. Arevalo et al. (2017) introduce a gating mechanism inspired by the LSTM while Kiela et al. (2018) describe an asymmetric gate that allows one modality to 'attend' to the other. The work that most closely matches ours is that of Wang et al. (2018) who also consider fusing Glove embeddings with visual features. However, their analysis is restricted to word similarity tasks and they require text-to-image regression to obtain visual embeddings for unseen words, due to the use of ImageNet. The use of image search allows us to obtain visual embeddings for a virtually unlimited vocabulary without needing a mapping function.

Method	tasks
(Bergsma and Durme, 2011)	bilingual lexicons
(Bergsma and Goebel, 2011)	lexical preference
(Kiela et al., 2014)	word similarity
(Kiela et al., 2015a)	lexical entailment detection
(Kiela et al., 2015b)	bilingual lexicons
(Shutova et al., 2016)	metaphor identification
(Bulat et al., 2015)	predicting property norms
(Kiela, 2016)	toolbox
(Vulic et al., 2016)	bilingual lexicons
(Kiela et al., 2016)	word similarity
(Anderson et al., 2017)	decoding brain activity
(Glavas et al., 2017)	semantic text similarity
(Bhaskar et al., 2017)	abstract vs concrete nouns
(Hartmann and Sogaard, 2017)	bilingual lexicons
(Bulat et al., 2017)	decoding brain activity

Table 1: Existing methods that use image search for grounding and their corresponding tasks.

3 Picturebook Embeddings

Our Picturebook embeddings ground language using the 'snapshots' returned by an image search engine. Given a word (or phrase), we image search for the top-k images and extract the images. We then pass each image through a CNN trained with a semantic ranking objective to extract its embedding. Our Picturebook embeddings reflect the search rankings by concatenating the individual embeddings in the order of the search results. We can perform all of these operations offline to construct a matrix E_p representing the Picturebook embeddings over a vocabulary.

3.1 Inducing Picturebook Embeddings

The convolutional network used to obtain Picturebook embeddings is based off of Wang et al. (2014). Let p_i, p_i^+, p_i^- denote a triplet of query, positive and negative images, respectively. We define the following hinge loss for a given triplet as follows:

$$l(p_i, p_i^+, p_i^-) = \max\{0, g + D(f(p_i), f(p_i^+)) - D(f(p_i), f(p_i^-))\} \tag{1}$$

where $f(p_i)$ represents the embedding of image p_i, $D(\cdot,\cdot)$ is the Euclidean distance and g is a margin (gap) hyperparameter. Suppose we have available pairwise relevance scores $r_{i,j} = r(p_i, p_j)$ indicating the similarity of images p_i and p_j. The objective function that is optimized is given by:

$$\begin{aligned}&\min \sum_i \xi_i + \lambda \|W\|_2^2 \\ &s.t. : l(p_i, p_i^+, p_i^-) \le \xi_i \\ &\quad \forall p_i, p_i^+, p_i^- \text{ such that } r(p_i, p_i^+) > r(p_i, p_i^-)\end{aligned} \tag{2}$$

where ξ_i are slack variables and W is a vector of the network's model parameters. The model is trained end-to-end using a proprietary dataset with 100+ million images. We refer the reader to Wang et al. (2014) for additional details of training, including the specifics of the architecture used.

After the model is trained, we can use the convolutional network as a feature extractor for images by computing an embedding vector $f(p)$ for an image p. Suppose we would like to obtain a Picturebook embedding for a given word w. We first perform an image search with query w to obtain a ranked list of images $p_1^w, \dots, p_k^w$. The Picturebook embedding for a word w is then represented as:

$$e_p(w) = [f(p_1^w); f(p_2^w); \dots; f(p_k^w)] \tag{3}$$

namely, the concatenation of the feature vectors in ranked order. In our model, each embedding results in a 64-dimensional vector with the final Picturebook embedding being $64 * k$ dimensions. Most of our experiments use $k = 10$ images resulting in a word embedding size of 640. To obtain the full collection of embeddings, we run the full Glove vocabulary (2.2M words) through image search to obtain a corresponding Picturebook embedding to each word in the Glove vocabulary.

3.2 Visual vs Semantic Similarity

The training procedure is heavily influenced by the choice of similarity function $r_{i,j}$. We consider two types of image similarity: visual and semantic. As an example, an image of a blue car would have high visual similarity to other blue cars but would have higher semantic similarity to cars of the same make, independent of color. In our experiments we consider two types of Picturebook embedding: one trained through optimizing for visual similarity and another for semantic similarity. As we will show in our experiments, the semantic Picturebook embeddings result in representations that are more useful for natural language processing tasks than the visual embeddings.

3.3 Multimodal Fusion Gating

Picturebook embeddings on their own are likely to be useful for representing concrete words but it is not clear whether they will be of benefit for abstract words. Consequently, we would like to fuse our Picturebook embeddings with other sources of information, for example Glove embeddings (Pennington et al., 2014) or randomly initialized embeddings that will be trained. Let $e_g = e_g(w)$ be our other embedding (i.e., Glove) for a word w and $e_p = e_p(w)$ be our Picturebook embedding. We fuse our embeddings using a multimodal gating mechanism:

$$g = \sigma(e_g, e_p) \tag{4}$$

$$e = g \odot \phi(e_g) + (1 - g) \odot \psi(e_p) \tag{5}$$

where σ is a 1 hidden layer DNN with ReLU activations and $\mathrm{sigmoid}$ outputs, ϕ and ψ are 1 hidden layer DNNs with ReLU activations and $\tanh$ outputs. The gating DNN σ allows the model to learn how visual a word is as a function of its input e_p and e_g. Similar gating mechanisms can be found in LSTMs (Hochreiter and Schmidhuber, 1997) and other multimodal models (Arevalo et al., 2017; Wang et al., 2018; Kiela et al., 2018). On some experiments we found it beneficial to include a skip connection from the hidden layer of σ. We chose this form of fusion over other approaches, such as CCA variants and metric learning methods, to allow for easier interpretability and analysis. We leave comparison of alternative fusion strategies for future work.

3.4 Contextual Gating

The gating described above is non-contextual, in the sense that each embedding computes a gate value independent of the context the words occur in. In some cases it may be beneficial to use contextual gates that are aware of the sentence that words appear in to decide how to weight Glove and Picturebook embeddings. For contextual gates, we use the same approach as above except we replace the controller $\sigma(e_g, e_p)$ with inputs that have been fed through a bidirectional-LSTM, e.g. $\sigma(\text{BiLSTM}(e_g),\text{BiLSTM}(e_p))$. We experiment with contextual gating for all experiments that use a bidirectional-LSTM encoder.

3.5 Inverse Picturebook

Picturebook embeddings can be seen as a form of implicit image search: given a word (or phrase), image search the word query and concatenate the embeddings of the images produced by a CNN. Up until now, we have only discussed scenarios where we have a word and we want to perform this implicit search operation. In generative modelling problems (i.e., MT), we want to perform the opposite operation. Given a Picturebook embedding, we want to find the closest word or phrase aligned to the representation. For example, given the word 'bicycle' in English and its Picturebook embedding, we want to find the closest French word that would generate this representation (i.e., 'vélo'). We want to perform this inverse image search operation given its Picturebook embedding.

We introduce a differentiable mechanism which allows us to align words across source and target languages in the Picturebook embedding domain. Let h be our internal representation of our model (i.e., seq2seq decoder state), and e_i be the i-th word embedding from our Picturebook embedding matrix E_p:

$$p(y_i|h) = \frac{\exp(\langle h, e_i \rangle)}{\sum_j \exp(\langle h, e_j \rangle)} \tag{6}$$

Given a representation h, Equation 6 simply finds the most similar word in the embedding space. This can be easily implemented by setting the output softmax matrix as the transpose of the Picturebook embedding matrix E_p. In practice, we find adding additional parameters helps with learning:

$$p(y_i|h) = \frac{\exp(\langle h, e_i + e'_i \rangle + b_i)}{\sum_j \exp(\langle h, e_j + e'_j \rangle + b_j)} \tag{7}$$

where e'_i is a trainable weight vector per word and b_i is a trainable bias per word. A similar technique to tie the softmax matrix as the transpose of the embedding matrix can be found in language modelling (Press and Wolf, 2017; Inan et al., 2017).

4 Experiments

To evaluate the effectiveness of our embeddings, we perform both quantitative and qualitative evaluation across a wide range of natural language processing tasks. Hyperparameter details of each experiment are included in the appendix. Since the use of Picturebook embeddings adds extra parameters to our models, we include a baseline for each experiment (either based on Glove or learned embeddings) that we extensively tune. In most experiments, we end up with baselines that are stronger than what has previously been reported.

4.1 Nearest neighbours

In order to get a sense of the representations our model learns, we first compute nearest neighbour results of several words, shown in Table 2. These results can be interpreted as follows: the words that appear as neighbours are those which have semantically similar images to that of the query. Often this captures visual similarity as well. Some words capture multimodality, such as 'deep' referring both to deep sea as well as to AI. Searching for cities returns cities which have visually similar characteristics. Words like 'sun' also return the corresponding word in different languages, such as 'Sol' in Spanish and 'Soleil' in French. Finally, it's worth highlighting that the most frequent association of a word may not be what is represented in image search results. For example, the word 'is' returns words related to terrorists and ISIS and 'it' returns words related to scary and clowns due to the 2017 film of the same name. We also report nearest neighbour examples across languages in Appendix A.1.

4.2 Word similarity

Our first quantitative experiment aims to determine how well Picturebook embeddings capture word similarity. We use the SimLex-999 dataset (Hill et al., 2015) and report results across 9 categories: all (the whole evaluation), adjectives, nouns, verbs, concreteness quartiles and the hardest 333 pairs. For the concreteness quartiles, the first quartile corresponds to the most abstract words, while the last corresponds to the most concrete words. The hardest pairs are those for which similarity is difficult to distinguish from relatedness. This is an interesting category since image-based word embeddings are perhaps less likely to confuse similarity with relatedness than distributional-based methods. For Glove, scores

language	deep	network	Melbourne	association	sun	life	not
interdisciplinary	deepest	internet	Austin	inclusion	prominence	praising	Nosign
languages	deep-sea	cyberspace	Raleigh	committees	Sol	rejoicing	prohibited
literacy	manta	networks	Cincinnati	social	Soleil	freedom	Forbidden
sociology	depths	blueprints	Yokohama	groupe	Sole	glorifying	no
multilingual	Jarvis	connectivity	Cleveland	members	Venere	worshipping	no-fly
inclusion	cyber	interconnections	Tampa	participation	Marte	healed	forbid
communications	AI	blueprint	Pittsburgh	personnel	eclipses	praise	10
linguistics	hackers	AI	Boston	involvement	Venus	healing	prohibiting
values	restarting	interconnected	Rochester	staffing	eclipse	trust	forbidden
user-generated	diver	tech	Frankfurt	meetings	fireballs	happiness	Stop

Table 2: Nearest neighbours of words. Results are retrieved over the 100K most frequent words.

Model	all	adjs	nouns	verbs	conc-q1	conc-q2	conc-q3	conc-q4	hard
Glove	40.8	**62.2**	42.8	19.6	**43.3**	41.6	42.3	40.2	27.2
Picturebook	37.3	11.7	48.2	17.3	14.4	27.5	46.2	**60.7**	28.8
Glove + Picturebook	**45.5**	46.2	**52.1**	**22.8**	36.7	**41.7**	**50.4**	57.3	**32.5**
Picturebook (Visual)	31.3	11.1	38.8	20.4	13.9	26.1	38.7	47.7	23.9
Picturebook (Semantic)	37.3	11.7	48.2	17.3	14.4	27.5	46.2	60.7	28.8
Picturebook (1)	24.5	2.6	33.5	12.1	4.7	17.8	32.8	47.8	13.6
Picturebook (2)	28.4	6.5	38.9	9.0	5.0	21.3	34.3	55.1	15.7
Picturebook (3)	30.3	11.9	41.9	3.1	2.6	24.3	37.5	58.3	18.4
Picturebook (5)	34.4	6.8	44.5	18.0	9.0	27.9	42.8	58.3	25.9
Picturebook (10)	37.3	11.7	48.2	17.3	14.4	27.5	46.2	60.7	28.8

Table 3: SimLex-999 results (Spearman's ρ). Best results overall are bolded. Best results per section are underlined. Bracketed numbers signify the number of images used. Some rows are copied across sections for ease of reading.

are computed via cosine similarity. For computing a score between 2 word pairs with Picturebook, we set $s(w^{(1)}, w^{(2)}) = -min_{i,j}\ d(e_i^{(1)}, e_j^{(2)})$. [2] That is, the score is minus the smallest cosine distance between all pairs of images of the two words. Note that this reduces to negative cosine distance when using only 1 image per word. We also report results combining Glove and Picturebook by summing their two independent similarity scores. By default, we use 10 images for each embedding using the semantic convolutional network.

Table 3 displays our results, from which several observations can be made. First, we observe that combining Glove and Picturebook leads to improved similarity across most categories. For adjectives and the most abstract category, Glove performs significantly better, while for the most concrete category Picturebook is significantly better. This result confirms that Glove and Picturebook capture very different properties of words. Next we observe that the performance of Picturebook gets progressively better across each concreteness quartile rating, with a 20 point improvement over Glove for the most concrete category. For the hardest subset of words, Picturebook performs slightly better than Glove while Glove performs better across all pairs. We also compare to a convolutional network trained with visual similarity. We observe a performance difference between our visual and semantic embeddings: on all categories except verbs, the semantic embeddings outperform visual ones, even on the most concrete categories. This indicates the importance of the type of similarity used for training the model. Finally we note that adding more images nearly consistently improves similarity scores across categories. Kiela et al. (2016) showed that after 10-20 images, performance tends to saturate. All subsequent experiments use 10 images with semantic Picturebook.

4.3 Sentential Inference and Relatedness

We next consider experiments on 3 pairwise prediction datasets: SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2017) and SICK (Marelli et al., 2014). The first two are natural language inference tasks and the third is a sentence semantic relatedness task. We explore the use of two types of sentential encoders: Bag-of-Words (BoW) and BiLSTM-Max (Conneau et al., 2017a).

[2] We found scoring all pairs of images to outperform scoring only the corresponding equally ranked image.

Model	SNLI		MultiNLI		SICK Relatedness		
	dev	test	dev-mat	dev-mis	test-p	test-s	test-mse
Glove (bow)	85.2	84.2	70.5	69.9	86.8	79.8	25.2
Picturebook (bow)	84.0	83.8	67.9	67.1	85.8	79.3	27.0
Glove + Picturebook (bow)	86.2	85.2	71.3	70.9	**87.2**	**80.9**	**24.4**
BiLSTM-Max (Conneau et al., 2017a)	85.0	84.5					
Glove	86.8	86.3	74.1	**74.5**			
Picturebook	85.2	85.1	70.7	70.3			
Glove + Picturebook	86.7	86.1	73.7	73.7			
Glove + Picturebook + Contextual Gating	**86.9**	**86.5**	**74.2**	74.4			

Table 4: Classification accuracies are reported for SNLI and MulitNLI. For SICK we report Pearson, Spearman and MSE. Higher is better for all metrics except MSE. Best results overall per column are bolded. Best results per section are underlined.

Three sets of features are used: Glove only, Picturebook only and Glove + Picturebook. For the latter, we use multimodal gating for all encoders and contextual gating in the BiLSTM-Max model. For SICK, we follow previous work and report average results across 5 runs (Tai et al., 2015). Due to the small size of the dataset, we only experiment with BoW on SICK. The full details of hyperparameters are discussed in Appendix B.

Table 4 displays our results. For BoW models, adding Picturebook embeddings to Glove results in significant gains across all three tasks. For BiLSTM-Max, our contextual gating sets a new state-of-the-art on SNLI sentence encoding methods (methods without interaction layers), outperforming the recently proposed methods of Im and Cho (2017); Shen et al. (2018). It is worth noting the effect that different encoders have when using our embeddings. While non-contextual gating is sufficient to improve bag-of-words methods, with BiLSTM-Max it slightly hurts performance over the Glove baseline. Adding contextual gating was necessary to improve over the Glove baseline on SNLI. Finally we note the strength of our own Glove baseline over the reported results of Conneau et al. (2017a), from which we improve on their accuracy from 85.0 to 86.8 on the development set. [3]

4.4 Sentiment and Topic Classification

Our next set of experiments aims to determine how well Picturebook embeddings do on tasks that are primarily non-visual, such as topic and sentiment classification. We experiment with 7 datasets provided by Zhang et al. (2015) and compare bag-of-words models against n-gram baselines provided by the authors as well as fastText (Joulin et al., 2017). Hyperparameter details are reported in Appendix B.

Our experimental results are provided in Table 5. Perhaps unsurprisingly, adding Picturebook to Glove matches or only slightly improves on 5 out of 7 tasks and obtains a lower result on AG News and Yahoo. Our results show that Picturebook embeddings, while minimally aiding in performance, can perform reasonably well on their own - outperforming the n-gram baselines of (Zhang et al., 2015) on 5 out of 7 tasks and the unigram fastText baseline on all 7 tasks. This result shows that our embeddings are able to work as a general text embedding, though they typically lag behind Glove. We note that the best performing methods on these tasks are based on convolutional neural networks (Conneau et al., 2017b).

4.5 Image-Sentence Ranking

We next consider experiments that map images and sentences into a common vector space for retrieval. Here, we utilize VSE++ (Faghri et al., 2017) as our base model and evaluate on the COCO dataset (Lin et al., 2014). VSE++ improves over the original CNN-LSTM embedding method of Kiros et al. (2015a) by using hard negatives instead of summing over contrastive examples. We re-implement their model with 2 modifications: 1) we replace the unidirectional LSTM encoder with a BiLSTM-Max sentence encoder and 2) we use Inception-V3 (Szegedy et al., 2016) as our CNN instead of ResNet 152 (He et al., 2016). As in previous work, we report the mean Recall@K (R@K) and the median rank over 1000 images and 5000 sentences. Full details of the hyperparameters are in Appendix B.

Table 6 displays our results on this task.

[3]All reported results on SNLI are available at `https://nlp.stanford.edu/projects/snli/`

Model	AG	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	**98.7**	95.4	54.8	68.5	52.4	91.5
fastText (Joulin et al., 2017)	91.5	98.1	93.8	60.4	72.0	55.8	91.2
fastText-bigram (Joulin et al., 2017)	92.5	98.6	**95.7**	**63.9**	72.3	**60.2**	**94.6**
Glove (bow)	**94.0**	98.6	94.4	61.7	**74.1**	58.5	93.2
Picturebook (bow)	92.8	98.5	94.4	61.6	73.3	57.8	92.9
Glove + Picturebook (bow)	93.9	98.6	94.5	61.9	73.8	58.7	93.2

Table 5: Test accuracy [%] on topic and sentiment classification datasets. Best results per dataset are bolded, best results per section are underlined. We compare directly against other bag of ngram baselines.

	Image Annotation				Image Search			
Model	R@1	R@5	R@10	Med *r*	R@1	R@5	R@10	Med *r*
VSE++ (Faghri et al., 2017)	**64.6**		95.7	1	52.0		92.0	1
Glove	**64.6**	88.9	95.5	1	53.7	86.5	94.4	1
Picturebook	62.4	90.2	95.3	1	54.2	86.4	94.3	1
Glove + Picturebook	61.8	89.2	95.0	1	54.1	86.7	**94.7**	1
Glove + Picturebook + Contextual Gating	63.4	**90.3**	**96.5**	1	**55.2**	**87.2**	94.4	1

Table 6: COCO test-set results for image-sentence retrieval experiments. Our models use VSE++. R@K is Recall@K (high is good). Med *r* is the median rank (low is good).

Our Glove baseline was able to match or outperform the reported results in Faghri et al. (2017) with the exception of Recall@10 for image annotation, where it performs slightly worse. Glove+Picturebook improves over the Glove baseline for image search but falls short on image annotation. However, using contextual gating results in improvements over the baseline on all metrics except R@1 for image annotation. Our reported results have been recently outperformed by Gu et al. (2018); Huang et al. (2018b); Lee et al. (2018), which are more sophisticated methods that incorporate generative modelling, reinforcement learning and attention.

4.6 Machine Translation

We experiment with the Multi30k (Elliott et al., 2016, 2017) dataset for MT. We compare our Picturebook models with other text-only non-ensembled models on the Flickr Test2016, Flickr Test2017 and MSCOCO test sets from Caglayan et al. (2017), the winner of the WMT 17 Multimodal Machine Translation competition (Elliott et al., 2017). We use the standard seq2seq (Sutskever et al., 2015) with content-based attention (Bahdanau et al., 2015) model and we describe our hyperparmeters in Appendix B.

Table 7 summarizes our English → German results and Table 8 summarizes our English → French results. We find our models to perform better in BLEU than METEOR relatively compared to (Caglayan et al., 2017). We believe this is due to the fact we did not use Byte Pair Encoding (BPE) (Sennrich et al., 2016), and METEOR captures word stemming (Denkowski and Lavie, 2014). This is also highlighted where our French models perform better than our German models relatively, due to the compounding nature of German words. Since seq2seq MT models are typically trained without Glove embeddings, we also did not use Glove embeddings for this task, but rather we combine randomly initialized learnable embeddings with the fixed Picturebook embeddings. We find the gating mechanism not to help much with the MT task since the trainable embeddings are free to change their norm magnitudes. We did not experiment with regularizing the norm of the embeddings. On the English → German tasks, we find our Picturebook model to perform on average 0.8 BLEU or 0.7 METEOR over our baseline. On the German task, compared to the previously best published results (Caglayan et al., 2017) we do better in BLEU but slightly worse in METEOR. We suspect this is due to the fact that we did not use BPE. On the English → French task, the Picturebook models do on average 1.2 BLEU better or 1.0 METEOR over our baseline.

We also report results for the IWSLT 2014 German-English task (Cettolo et al., 2014) in Table 9. Compared to our baseline, we report a gain of 0.3 and 1.1 BLEU for German → English and English → German respectively. We

Model	Test2016		Test2017		MSCOCO	
	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
BPE (Caglayan et al., 2017)	38.1	**57.3**	30.8	**51.6**	26.4	**46.8**
Baseline	38.9	56.5	32.6	50.7	26.8	45.4
Picturebook	39.6	56.9	31.8	50.1	27.7	45.8
Picturebook + Inverse Picturebook	**40.2**	57.2	32.3	50.7	27.8	46.3
Picturebook + Inverse Picturebook + Gating	40.0	**57.3**	**33.0**	51.1	**27.9**	46.5

Table 7: Machine Translation results on the Multi30k English → German task. We note that our models do not use BPE, and we perform better in BLEU relative to METEOR.

Model	Test2016		Test2017		MSCOCO	
	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
BPE (Caglayan et al., 2017)	52.5	69.6	50.4	67.5	41.2	61.3
Baseline	60.7	74.1	52.3	67.4	42.8	60.6
Picturebook	61.0	74.2	52.4	67.5	43.1	61.0
Picturebook + Inverse Picturebook	61.8	75.0	52.6	67.7	42.8	61.2
Picturebook + Inverse Picturebook + Gating	**62.1**	**75.2**	**53.6**	**68.4**	**43.8**	**61.6**

Table 8: Machine Translation results on the Multi30k English → French task.

report new state-of-the-art results for the English → German task at 25.4 BLEU, while our German → English model achieves 29.6 BLEU which is slightly behind the recently proposed Neural Phrase-based Machine Translation (NPMT) model at 29.9 (Huang et al., 2018a). We note that the NPMT is not a seq2seq model and can be augmented with our Picturebook embeddings. We also note that our models may not be directly comparable to previously published seq2seq models from (Wiseman and Rush, 2016; Bahdanau et al., 2017) since we used a deeper encoder and decoder.

4.7 Limitations

We explored the use of Picturebook for larger machine translation tasks, including the popular WMT14 benchmarks. For these tasks, we found that models that incorporate Picturebook led to faster convergence. However, we were not able to improve upon BLEU scores from equivalent models that do not use Picturebook. This indicates that while our embeddings are useful for smaller MT experiments, further research is needed on how to best incorporate grounded representations in larger translation tasks.

4.8 Gate Analysis

In this section we perform an extensive analysis of the gating mechanism for models trained across datasets used in our experiments. In our first experiment, we aim to determine how well gate activations correlate to a) human judgments of concreteness and b) image dispersion (Kiela et al., 2014). For concreteness ratings, we use the dataset of Brysbaert et al. (2013) which provides ratings for 40,000 English lemmas. Image dispersion is the average distance between all pairs of images returned from a search query. It was shown in Kiela et al. (2014) that abstract words tend to have higher dispersion ratings, due to having much higher variety in the types of images returned from a query. On the other hand, low dispersion ratings were more associated with concrete words. For each word, we compute the mean gate activation value for Picturebook embeddings. [4] For concreteness ratings, we take the intersection of words that have ratings with the dataset vocabulary. We then compute the Spearman correlation of mean gate activations with a) concreteness ratings and b) image dispersion scores.

Table 10 illustrates the result of this analysis. We observe that gates have high correlations with concreteness ratings and strong negative correlations with image dispersion scores. Moreover, this result holds true across all datasets, even those that are not inherently visual. These results provide evidence that our gating mechanism actively prefers Glove embeddings for abstract words and Picturebook embeddings for concrete words. Appendix A contains examples of words that most strongly activate Glove and Picturebook gates.

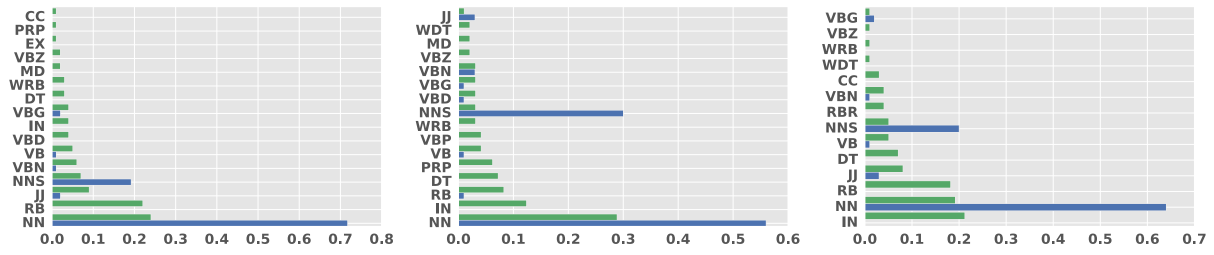
[4]We only consider non-contextualized gates.

Model	DE → EN BLEU	EN → DE BLEU
MIXER (Ranzato et al., 2016)	21.8	
Beam Search Optimization (Wiseman and Rush, 2016)	25.5	
Actor-Critic + Log Likelihood (Bahdanau et al., 2017)	28.5	
Neural Phrase-based Machine Translation (Huang et al., 2018a)	**29.9**	25.1
Baseline	29.3	24.3
Picturebook	29.6	**25.4**

Table 9: Machine Translation results on the IWSLT 2014 German-English task.

Rank	SNLI		MultiNLI		COCO		AG-News		DBpedia		Yelp		Amazon	
	ccorr	disp	ccorr	disp	ccorr	disp	ccorr	disp	ccorr	disp	ccorr	disp	ccorr	disp
top-1%	73	-41	39	-27	53	-22	60	-16	56	-30	47	-28	32	-17
top-10%	54	-39	48	-34	34	-23	52	-24	54	-32	49	-26	50	-30
all	35	-30	30	-27	21	-16	36	-17	39	-30	24	-20	33	-31

Table 10: Correlations (rounded, x100) of mean Picturebook gate activations to human judgements of concreteness ratings (ccorr) and image dispersion (disp) within the specified most frequent words.



(a) SNLI (b) MultiNLI (c) AG-News

Figure 1: POS analysis. Top bar for each tag is Glove, bottom is Picturebook. Tags are sorted by Glove frequencies. Results taken over the top 100 mean activation values within the 10K most frequent words.

Finally we analyze the parts-of-speech (POS) of the highest activated words. These results are shown in Figure 1. The highest scoring Picturebook words are almost all singular and plural nouns (NN / NNS). We also observe tags which are exclusively Glove oriented, namely adverbs (RB), prepositions (IN) and determiners (DT).

5 Conclusion

Traditionally, word representations have been built on co-occurrences of neighbouring words; and such representations only make use of the statistics of the text distribution. Picturebook embeddings offer an alternative approach to constructing word representations grounded in image search engines. In this work we demonstrated that Picturebook complements traditional embeddings on a wide variety of tasks. Through the use of multimodal gating, our models lead to interpretable weightings of abstract vs concrete words. In future work, we would like to explore other aspects of search engines for language grounding as well as the effect these embeddings may have on learning generic sentence representations (Kiros et al., 2015b; Hill et al., 2016; Conneau et al., 2017a; Logeswaran and Lee, 2018). Recently, contextualized word representations have shown promising improvements when combined with existing embeddings (Melamud et al., 2016; Peters et al., 2017; McCann et al., 2017; Peters et al., 2018). We expect that integrating Picturebook with these embeddings to lead to further performance improvements as well.

Acknowledgments

The authors would like to thank Chuck Rosenberg, Tom Duerig, Neil Alldrin, Zhen Li, Filipe Gonçalves, Mia Chen, Zhifeng Chen, Samy Bengio, Yu Zhang, Kevin Swersky, Felix Hill and the ACL anonymous reviewers for their valuable advice and feedback.

References

Andrew Anderson, Douwe Kiela, Stephen Clark, and Massimo Poesio. 2017. Visually Grounded and Textual Semantic Models Differentially Decode Brain Activity Associated with Concrete and Abstract Nouns. In *ACL*.

John Arevalo, Thamar Solorio, Manuel Montes y Gomez, and Fabio A. Gonzalez. 2017. Gated Multimodal Units for Information Fusion. In *arXiv:1702.01992*.

Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. 2016. Layer Normalization. In *arXiv:1607.06450*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An Actor-Critic Algorithm for Sequence Prediction. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Shane Bergsma and Benjamin Van Durme. 2011. Learning Bilingual Lexicons using the Visual Similarity of Labeled Web Images. In *IJCAI*.

Shane Bergsma and Randy Goebel. 2011. Using Visual Information to Predict Lexical Preference. In *RANLP*.

Sai Abishek Bhaskar, Maximilian Koper, Sabine Schulte Im Walde, and Diego Frassinelli. 2017. Exploring Multi-Modal Text+Image Models to Distinguish between Abstract and Concrete Nouns. In *IWCS Workshop on Foundations of Situated and Multimodal Communication*.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research* 49(1).

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods* 46(3).

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Speaking, Seeing, Understanding: Correlating semantic models with conceptual representation in the brain. In *EMNLP*.

Luana Bulat, Douwe Kiela, and Stephen Clark. 2015. Vision and Feature Norms: Improving automatic feature norm learning through cross-modal maps. In *NAACL*.

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes Garcia-Martinez, Marc Masana, Luis Herranz, and Joost van de Weijer. 2017. LIUM-CVC Submissions for WMT17 Multimodal Translation Task. In *Conference on Machine Translation*.

Mauro Cettolo, Jan Niehues, Sebastian Stuker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign, IWSLT 2014. In *IWSLT*.

Grzegorz Chrupala, Akos Kadar, and Afra Alishah. 2015. Learning language through pictures. In *EMNLP*.

Guillem Collell and Marie-Francine Moens. 2016. Is an Image Worth More than a Thousand Words? On the Fine-Grain Semantic Differences between Visual and Linguistic Representations. In *COLING*.

Guillem Collell, Ted Zhang, and Marie-Francine Moens. 2017. Imagined Visual Representations as Multimodal Embeddings. In *AAAI*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017a. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017b. Very deep convolutional networks for text classification. In *EACL*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL: Workshop on Statistical Machine Translation*.

Desmond Elliott, Stella Frank, Loic Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. In *Conference on Machine Translation*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German Image Description. In *ACL: Workshop on Vision and Language*.

Fartash Faghri, David Fleet, Jamie Kiros, and Sanja Fidler. 2017. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. In *arXiv:1707.05612*.

Goran Glavas, Ivan Vulic, and Simone Paolo Ponzetto. 2017. If Sentences Could See: Investigating Visual Information for Semantic Textual Similarity. In *IWCS*.

Jiuxiang Gu, Jianfei Cai, Shafiq Joty, Li Niu, and Gang Wang. 2018. Look, Imagine and Match: Improving Textual-Visual Cross-Modal Retrieval with Generative Models. In *CVPR*.

Mareike Hartmann and Anders Sogaard. 2017. Limitations of Cross-Lingual Learning from Image Search. In *arXiv:1709.05914*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL*.

Felix Hill and Anna Korhonen. 2014. Learning Abstract Concept Embeddings from Multi-Modal Data: Since You Probably Can't See What I Mean. In *EMNLP*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *Computational Linguistics* 41(4).

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8).

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018a. Towards Neural Phrase-based Machine Translation. In *ICLR*.

Yan Huang, Qi Wu, and Liang Wang. 2018b. Learning Semantic Concepts and Order for Image and Sentence Matching. In *CVPR*.

Jinbae Im and Sungzoon Cho. 2017. Distance-based self-attention network for natural language inference. In *arXiv:1712.02047*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Languag Modeling. In *ICLR*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *EACL*.

Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. 2016. Learning Visual Features from Large Weakly Supervised Data. In *ECCV*.

Douwe Kiela. 2016. MMFeat: A Toolkit for Extracting Multi-Modal Features. In *ACL: System Demonstrations*.

Douwe Kiela and Leon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *EMNLP*.

Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. 2017. Learning Visually Grounded Sentence Representations. In *arXiv:1707.06320*.

Douwe Kiela, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2018. Efficient Large-Scale Multi-Modal Classification. In *AAAI*.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving Multi-Modal Representations Using Image Dispersion: Why Less is Sometimes More. In *ACL*.

Douwe Kiela, Laura Rimell, Ivan Vulic, and Stephen Clark. 2015a. Exploiting Image Generality for Lexical Entailment Detection. In *ACL*.

Douwe Kiela, Anita Vero, and Stephen Clark. 2016. Comparing data sources and architectures for deep visual representation learning in semantics. In *EMNLP*.

Douwe Kiela, Ivan Vulic, and Stephen Clark. 2015b. Visual Bilingual Lexicon Induction with Transferred ConvNet Features. In *EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard Zemel. 2015a. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. In *arXiv:1411.2539*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015b. Skip-thought vectors. In *NIPS*.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining Language and Vision with a Multimodal Skip-gram Model. In *ACL*.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked Cross Attention for Image-Text Matching. In *arXiv:1803.08024*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft COCO: Common Objects in Context. In *ECCV*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Junhua Mao, Jiajing Xu, Yushi Jing, and Alan Yuille. 2016. Training and Evaluating Multimodal Word Embeddings with Large-scale Web Annotated Images. In *NIPS*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *CoNLL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *ICLR Workshop*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *arXiv:1802.05365*.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *EACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *ICLR*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2016. Recurrent Dropout without Memory Loss. In *COLING*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018. Reinforced Self-Attention Network: a Hybrid of Hard and Soft Attention for Sequence Modeling. In *arXiv:1801.10296*.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black Holes and White Rabbits: Metaphor Identification with Visual Features. In *NAACL*.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2017. Visually Grounded Meaning Representations. *PAMI* .

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2015. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *CVPR*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *ACL*.

Steffen Thoma, Achim Rettinger, and Fabian Both. 2017. Towards Holistic Concept Representations: Embedding Relational Knowledge, Visual Attributes, and Distributional Word Semantics. In *ISWC*.

Ivan Vulic, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-Modal Representations for Improved Bilingual Lexicon Learning. In *ACL*.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning Fine-grained Image Similarity with Deep Ranking. In *CVPR*.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2018. Learning Multimodal Word Representation via Dynamic Fusion Methods. In *AAAI*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *arXiv:1704.05426*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *EMNLP*.

Éloi Zablocki, Benjamin Piwowarski, Laure Soulier, and Patrick Gallinari. 2018. Learning Multi-Modal Word Representation Grounded in Visual Context. In *AAAI*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *NIPS*.

What Action Causes This? Towards Naive Physical Action-Effect Prediction

Qiaozi Gao[†] **Shaohua Yang**[†] **Joyce Y. Chai**[†] **Lucy Vanderwende**[‡]
[†]Department of Computer Science and Engineering, Michigan State University
[‡]Microsoft Research, Redmond, Washington
{gaoqiaoz, yangshao, jchai}@msu.edu
lucy_vanderwende@live.com

Abstract

Despite recent advances in knowledge representation, automated reasoning, and machine learning, artificial agents still lack the ability to understand basic action-effect relations regarding the physical world, for example, the action of cutting a cucumber most likely leads to the state where the cucumber is broken apart into smaller pieces. If artificial agents (e.g., robots) ever become our partners in joint tasks, it is critical to empower them with such action-effect understanding so that they can reason about the state of the world and plan for actions. Towards this goal, this paper introduces a new task on naive physical action-effect prediction, which addresses the relations between concrete actions (expressed in the form of verb-noun pairs) and their effects on the state of the physical world as depicted by images. We collected a dataset for this task and developed an approach that harnesses web image data through distant supervision to facilitate learning for action-effect prediction. Our empirical results have shown that web data can be used to complement a small number of seed examples (e.g., three examples for each action) for model learning. This opens up possibilities for agents to learn physical action-effect relations for tasks at hand through communication with humans with a few examples.

1 Introduction

Causation in the physical world has long been a central discussion to philosophers who study casual reasoning and explanation (Ducasse, 1926; Gopnik et al., 2007), to mathematicians or computer scientists who apply computational approaches to model cause-effect prediction (Pearl et al., 2009), and to domain experts (e.g., medical doctors) who attempt to understand the underlying cause-effect relations (e.g., disease and symptoms) for their particular inquires. Apart from this wide range of topics, this paper investigates a specific kind of causation, the very basic causal relations between a concrete action (expressed in the form of a verb-noun pair such as "cut-cucumber") and the change of the physical state caused by this action. We call such relations *naive* physical action-effect relations.

For example, given an image as shown in Figure 1, we would have no problem predicting what actions can cause the state of the world depicted in the image, e.g., slicing an apple will likely lead to the state. On the other hand, given a statement "slice an apple", it would not be hard for us to imagine what state change may happen to the apple. We can make such action-effect prediction because we have developed an understanding of this kind of basic action-effect relations at a very young age (Baillargeon, 2004). What happens to machines? Will artificial agents be able to make the same kind of predictions? The answer is not yet.

Despite tremendous progress in knowledge representation, automated reasoning, and machine learning, artificial agents still lack the understanding of naive causal relations regarding the physical world. This is one of the bottlenecks in machine intelligence. If artificial agents ever become capable of working with humans as partners, they will need to have this kind of physical action-effect understanding to help them reason, learn, and perform actions.

To address this problem, this paper introduces a new task on naive physical action-effect prediction. This task supports both *cause predic-*

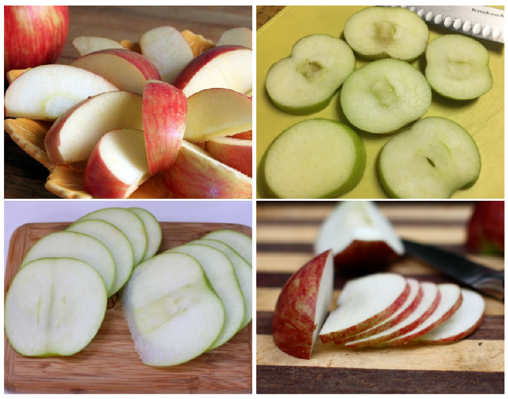


Figure 1: Images showing the effects of "slice an apple".

tion: given an image which describes a state of the world, identify the most likely action (in the form of a verb-noun pair, from a set of candidates) that can result in that state; and *effect prediction*: given an action in the form of a verb-noun pair, identify images (from a set of candidates) that depicts the most likely effects on the state of the world caused by that action. Note that there could be different ways to formulate this problem, for example, both causes and effects are in the form of language or in the form of images/videos. Here we intentionally frame the action as a language expression (i.e., a verb-noun pair) and the effect as depicted in an image in order to make a connection between language and perception. This connection is important for physical agents that not only can perceive and act, but also can communicate with humans in language.

As a first step, we collected a dataset of 140 verb-noun pairs. Each verb-noun pair is annotated with possible effects described in language and depicted in images (where language descriptions and image descriptions are collected separately). We have developed an approach that applies distant supervision to harness web data for bootstrapping action-effect prediction models. Our empirical results have shown that, using a simple bootstrapping strategy, our approach can combine the noisy web data with a small number of seed examples to improve action-effect prediction. In addition, for a new verb-noun pair, our approach can infer its effect descriptions and predict action-effect relations only based on 3 image examples.

The contributions of this paper are three folds. First, it introduces a new task on physical action-effect prediction, a first step towards an understanding of causal relations between physical actions and the state of the physical world. Such ability is central to robots which not only perceive from the environment, but also act to the environment through planning. To our knowledge, there is no prior work that attempts to connect actions (in language) and effects (in images) in this nature. Second, our approach harnesses the large amount of image data available on the web with minimum supervision. It has shown that physical action-effect models can be learned through a combination of a few annotated examples and a large amount of un-annotated web data. This opens up the possibility for humans to teach robots new tasks through language communication with a small number of examples. Third, we have created a dataset for this task, which is available to the community [1]. Our bootstrapping approach can serve as a baseline for future work on this topic.

In the following sections, we first describe our data collection effort, then introduce the bootstrapping approach for action-effect prediction, and finally present results from our experiments.

2 Related Work

In the NLP community, there has been extensive work that models cause-effect relations from text (Cole et al., 2005; Do et al., 2011; Yang and Mao, 2014). Most of these previous studies address high-level causal relations between events, for example, "the collapse of the housing bubble" causes the effect of "stock prices to fall" (Sharp et al., 2016). They do not concern the kind of naive physical action-effect relations in this paper. There is also an increasing amount of effort on capturing commonsense knowledge, for example, through knowledge base population. Except for few (Yatskar et al., 2016) that acquires knowledge from images, most of the previous effort apply information extraction techniques to extract facts from a large amount of web data (Dredze et al., 2010; Rajani and Mooney, 2016). DBPedia (Lehmann et al., 2015), Freebase (Bollacker et al., 2008), and YAGO (Suchanek et al., 2007) knowledge bases contain millions of facts about the world such as people and places. However, they do not contain basic cause-effect knowledge related to concrete actions and their effects to the world. Recent work started looking into phys-

[1]This dataset is available at `http://lair.cse.msu.edu/lair/projects/actioneffect.html`

ical causality of action verbs (Gao et al., 2016) and other physical properties of verbs (Forbes and Choi, 2017; Zellers and Choi, 2017; Chao et al., 2015). But they do not address action-effect prediction.

The idea of modeling object physical state change has also been studied in the computer vision community (Fire and Zhu, 2016). Computational models have been developed to infer object states from observations and to further predict future state changes (Zhou and Berg, 2016; Wu et al., 2016, 2017). The action recognition task can be treated as detecting the transformation on object states (Fathi and Rehg, 2013; Yang et al., 2013; Wang et al., 2016). However these previous works only focus on the visual presentation of motion effects. Recent years have seen an increasing amount of work integrating language and vision, for example, visual question answering (Antol et al., 2015; Fukui et al., 2016; Lu et al., 2016), image description generation (Xu et al., 2015; Vinyals et al., 2015), and grounding language to perception (Yang et al., 2016; Roy, 2005; Tellex et al., 2011; Misra et al., 2017). While many approaches require a large amount of training data, recent works have developed zero/few shot learning for language and vision (Mukherjee and Hospedales, 2016; Xu et al., 2016, 2017a,b; Tsai and Salakhutdinov, 2017). Different from these previous works, this paper introduces a new task that connects language with vision for physical action-effect prediction.

In the robotics community, an important task is to enable robots to follow human natural language instructions. Previous works (She et al., 2014; Misra et al., 2015; She and Chai, 2016, 2017) explicitly model verb semantics as desired goal states and thus linking natural language commands with underlying planning systems for action planning and execution. However, these studies were carried out either in a simulated world or in a carefully curated simple environment within the limitation of the robot's manipulation system. And they only focus on a very limited set of domain specific actions which often only involve the change of locations. In this work, we study a set of open-domain physical actions and a variety of effects perceived from the environment (i.e., from images).

3 Action-Effect Data Collection

We collected a dataset to support the investigation on physical action-effect prediction. This dataset consists of actions expressed in the form of verb-noun pairs, effects of actions described in language, and effects of actions depicted in images. Note that, as we would like to have a wide range of possible effects, language data and image data are collected separately.

Actions (verb-noun pairs). We selected 40 nouns that represent everyday life objects, most of them are from the COCO dataset (Lin et al., 2014), with a combination of food, kitchen ware, furniture, indoor objects, and outdoor objects. We also identified top 3000 most frequently used verbs from Google Syntactic N-gram dataset (Goldberg and Orwant, 2013) (Verbargs set). And we extracted top frequent verb-noun pairs containing a verb from the top 3000 verbs and a noun in the 40 nouns which hold a *dobj* (i.e., direct object) dependency relation. This resulted in 6573 candidate verb-noun pairs. As changes to an object can occur at various dimensions (e.g., size, color, location, attachment, etc.), we manually selected a subset of verb-noun pairs based on the following criteria: (1) changes to the objects are visible (as opposed to other types such as temperature change, etc.); and (2) changes reflect one particular dimension as opposed to multiple dimensions (as entailed by high-level actions such as "cook a meal", which correspond to multiple dimensions of change and can be further decomposed into basic actions). As a result, we created a subset of 140 verb-noun pairs (containing 62 unique verbs and 39 unique nouns) for our investigation.

Effects Described in Language. The basic knowledge about physical action-effect is so fundamental and shared among humans. It is often presupposed in our communication and not explicitly stated. Thus, it is difficult to extract naive action-effect relations from the existing textual data (e.g., web). This kind of knowledge is also not readily available in commonsense knowledge bases such as ConceptNet (Speer and Havasi, 2012). To overcome this problem, we applied crowd-sourcing (Amazon Mechanical Turk) and collected a dataset of language descriptions describing effects for each of the 140 verb-noun pairs. The workers were shown a verb-noun pair, and were asked to use their own words and imag-

Action	Effect Text
ignite paper	The paper is on fire.
soak shirt	The shirt is thoroughly wet.
fry potato	The potatoes become crisp and golden.
stain shirt	There is a visible mark on the shirt.

Table 1: Example action and effect text from our collected data.

inations to describe what changes might occur to the corresponding object as a result of the action. Each verb-noun pair was annotated by 10 different annotators, which has led to a total of 1400 effect descriptions. Table 1 shows some examples of collected effect descriptions. These effect language descriptions allow us to derive *seed effect knowledge* in a symbolic form.

Effects Depicted in Images. For each action, three students searched the web and collected a set of images depicting potential effects. Specifically, given a verb-noun pair, each of the three students was asked to collect at least 5 positive images and 5 negative images. Positive images are those deemed to capture the resulting world state of the action. And negative images are those deemed to capture some state of the related object (i.e., the nouns in the verb-noun pairs), but are not the resulting state of the corresponding action. Then, each student was also asked to provide positive or negative labels for the images collected by the other two students. As a result each image has three positive/negative labels. We only keep the images whose labels are agreed by all three students. In total, the dataset contains 4163 images. On average, each action has 15 positive images, and 15 negative images. Figure 2 shows several examples of positive images and negative images of the action *peel-orange*. The positive images show an orange in a *peeled* state, while the negative images show oranges in different states (orange as a whole, orange slices, orange juice, etc.).

4 Action-Effect Prediction

Action-effect prediction is to connect actions (as causes) to the effects of actions. Specifically, given an image which depicts a state of the world, our task is to predict what concrete actions could cause the state of the world. This task is different from traditional action recognition as the underlying actions (e.g., human body posture/movement) are not captured by the images. In this regard, it is also different from image description generation.

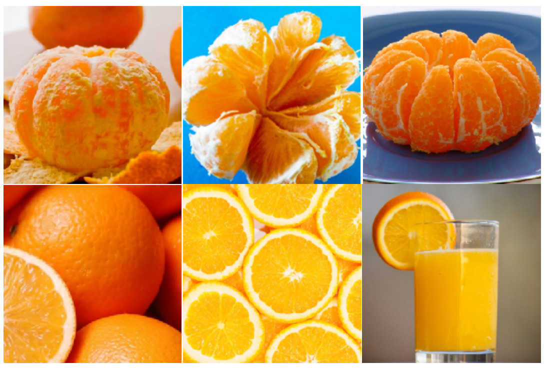


Figure 2: Positive images (top row) and negative images (bottom row) of the action *peel-orange*.

We frame the problem as a few-shot learning task, by only providing a few human-labelled images for each action at the training stage. Given the very limited training data, we attempt to make use of web-search images. Web search has been adopted by previous computer vision studies to acquire training data (Fergus et al., 2005; Kennedy et al., 2006; Berg et al., 2010; Otani et al., 2016). Compared with human annotations, web-search comes at a much lower cost, but with a trade-off of poor data quality. To address this issue, we apply a bootstrapping approach that aims to handle data with noisy labels.

The first question is what search terms should be used for image search. There are two options. The first option is to directly use the action terms (i.e., verb-noun pairs) to search images and the downloaded web images are referred to as *action web images*. As desired images should be depicting effects of an action, terms describing effects become a natural choice. The second option is to use the key phrases extracted from language effect descriptions to search the web. The downloaded web images are referred to as *effect web images*.

4.1 Extracting Effect Phrases from Language Data

We first apply chunking (shallow parsing) using the SENNA software (Collobert et al., 2011) to break an effect description into phrases such as noun phrases (NP), verb phrases (VP), prepositional phrases (PP), adjectives (ADJP), adverbs (ADVP), etc. After some examination, we found that most of the effect descriptions follow simple syntactic patterns. For a *verb-noun* pair, around 80% of its effect descriptions start with the same noun as the subject. In an effect description, the

Example patterns	Example *Effect Phrases* (bold) extracted from effect descriptions
VP with a verb $\in$ {be, become, turn, get}	The ship **is destroyed**.
VP + PRT	The wall is **knocked off**.
VP + ADVP	The door **swings forward**.
ADJP	The window would begin to get **clean**.
PP + NP	The eggs are divided **into whites and yolks**.

Table 2: Example patterns that are used to extract effect phrases (bold) from sample sentences.

change of state associated with the noun is mainly captured by some key phrases. For example, an adjective phrase usually describes a physical state; verbs like *be*, *become*, *turn*, *get* often indicate a description of change of the state. Based on these observations, we defined a set of patterns to identify phrases that describe physical states of an object. In total 1997 *effect phrases* were extracted from the language data. Table 2 shows some example patterns and example effect phrases that are extracted.

4.2 Downloading Web Images

The purpose of querying search engine is to retrieve images of objects in certain effect states. To form image searching keywords, the effect phrases are concatenated with the corresponding noun phrases, for example, "apple + into thin pieces". The image search results are downloaded and used as supplementary training data for the action-effect prediction models. However, web images can be noisy. First of all, not all of the automatically extracted effect phrases describe visible state of objects. Even if a phrase represents visible object states, the retrieved results may not be relevant. Figure 3 shows some example image search results using queries describing the object name "book", and describing the object state such as "book is on fire", "book is set aflame". These state phrases were used by human annotators to describe the effect of the action "burn a book". We can see that the images returned from the query "book is set aflame" are not depicting the physical effect state of "burn a book". Therefore, it's important to identify images with relevant effect states to train the model. To do that, we applied a bootstrapping method to handle the noisy web images as described in Section 4.3. For an action (i.e., a verb-noun pair), it has multiple corresponding effect phrases, and all of their image search results are treated as training images for this action.

Since both the human annotated image data (Section 3) and the web-search image data were obtained from Internet search engines, they may

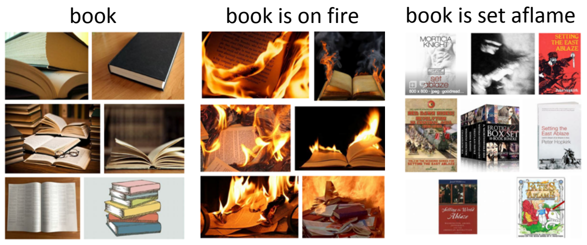


Figure 3: Examples of image search results.

have duplicates. As part of the annotated images are used as test data to evaluate the models, it is important to remove duplicates. We designed a simple method to remove any images from the web-search image set that has a duplicate in the human annotated set. We first embed all images into feature vectors using pre-trained CNNs. For each web-search image, we calculate its cosine similarity score with each of the annotated images. And we simply remove the web images that have a score larger than 0.95.

4.3 Models

We formulate the action-effect prediction task as a multi-class classification problem. Given an image, the model will output a probability distribution $\mathbf{q}$ over the candidate actions (i.e., verb-noun pairs) that can potentially cause the effect depicted in the image.

Specifically for model training, we are given a set of human annotated seeding image data $\{\mathbf{x}, \mathbf{t}\}$ and a set of web-search image data $\{\mathbf{x}', \mathbf{t}'\}$. Here $\mathbf{x}$ and $\mathbf{x}'$ are the images (depicting effect states), and $\mathbf{t}$ and $\mathbf{t}'$ are their classification targets (i.e., actions that cause the effects). Each target vector is the observed image label, $\mathbf{t} \in \{0, 1\}^C$, $\sum_i t_i = 1$, and C is the number of classes (i.e., actions). The human annotated targets $\mathbf{t}$ can be trusted. But the targets of web-search images $\mathbf{t}'$ are usually very noisy. Bootstrapping method has been shown to be an effective method to handle noisy labelled data (Rosenberg et al., 2005; Whitney and Sarkar, 2012; Reed et al., 2014). The objective of the

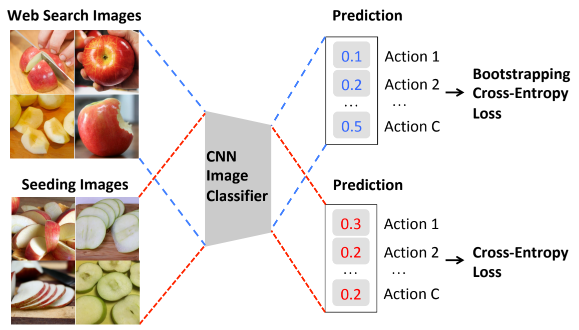


Figure 4: Architecture for the action-effect prediction model with bootstrapping.

cross-entropy loss is defined as follows:

$$\mathcal{L}(\mathbf{t}, \mathbf{q}) = \sum_{i=1}^{C} t_i \log(q_i), \quad (1)$$

where $\mathbf{q}$ are the predicted class probabilities, and C is the number of classes. To handle the noisy labels in the web-search data $\{\mathbf{x}', \mathbf{t}'\}$, we adopt a bootstrapping objective following Reed's work (Reed et al., 2014):

$$\mathcal{L}(\mathbf{t}', \mathbf{q}) = \sum_{i=1}^{C} [\beta t'_i + (1-\beta) z_i] \log(q_i), \quad (2)$$

where $\beta \in [0, 1]$ is a model parameter to be assigned, $\mathbf{z}$ is the one-hot vector of the prediction $\mathbf{q}$, $z_i = 1$, if $i = \text{argmax } q_k, k = 1 \ldots C$.

The model architecture is shown in Figure 4. After each training batch, the current model will be used to make predictions $\mathbf{q}$ on images in the next batch. And the target probabilities is calculated as a linear combination of the current predictions $\mathbf{q}$ and the observed noisy labels $\mathbf{t}'$. The idea behind this bootstrapping strategy is to ensure the consistency of the model's predictions. By first initializing the model on the seeding image data, the bootstrapping approach allows the model to trust more on the web images that are consistent with the seeding data.

4.4 Evaluation

We evaluate the models on the action-effect prediction task. Given an image that illustrates a state of the world, the goal is to predict what action could cause that state. Given an action in the form of a verb-noun pair, the goal is to identify images that depict the most likely effects on the state of the world caused by that action.

For each of the 140 verb-noun pairs, we use 10% of the human annotated images as the seeding image data for training, and use 30% for development and the rest 60% for test. The seeding image data set contains 408 images. On average, each verb-noun pair has less than 3 seeding images (including positive images and negative images). The development set contains 1252 images. The test set contains 2503 images. The model parameters were selected based on the performance on the development set.

As a given image may not be relevant to any effect, we add a background class to refer to images where effects are not caused by any action in the space of actions. So the total of classes for our evaluation model is 141. For each verb-noun pair and each of the effect phrases, around 40 images were downloaded from the Bing image search engine and used as candidate training examples. In total we have 6653 action web images and 59575 effect web images.

Methods for Comparison

All the methods compared are based on one neural network structure. We use ResNet (He et al., 2016) pre-trained on ImageNet (Deng et al., 2009) to extract image features. The extracted image features are fed to a fully connected layer with rectified linear units and then to a softmax layer to make predictions. More specifically, we compare the following configurations:

(1) *BS+Seed+Act+Eff*. The bootstrapping approach trained on the seeding images, the action web images, and the effect web images. During the training stage, the model was first trained on the seeding image data using vanilla cross-entropy objective (Equation 1). Then it was further trained on a combination of the seeding image data and web-search data using the bootstrapping objective (Equation 2). In the experiments we set $\beta = 0.3$.

(2) *BS+Seed+Act*. The bootstrapping approach trained in the same fashion as (1). The only difference is that this method does not use the effect web images.

(3) *Seed+Act+Eff*. A baseline method trained on a combination of the seeding images, the web action images, and the web effect images, using the vanilla cross-entropy objective.

(4) *Seed+Act*. A baseline method trained on a combination of the seeding images and the action web images, using the vanilla cross-entropy objective.

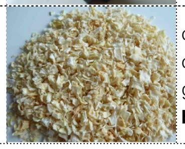
	Top Action Predictions	Top Effect Descriptions		Top Action Predictions	Top Effect Descriptions
	bake potato	potato crispy		wrap book	book is ripped
	peel potato	potato is crushed		**tear book**	paper become creased
	boil potato	eggs get beaten		fold paper	book into smaller pieces
	fry potato	potato browned		shave hair	meat is being prepped
	peel carrot	carrot into little sections		stain paper	bottle is pressed together
	cut wood	tree into pieces		close drawer	meat is exposed
	chop carrot	carrot into tiny pieces		**squeeze bottle**	paper around itself
	grate carrot	wood is being chopped		crack bottle	drawer is pushed back
	chop onion	onion is being cut		chop onion	onion is heated
	cut onion	onion in		cook onion	onion into small pieces
	slice onion	banana is made		grate potato	onion into multiple pieces
	background	banana is removed		**background**	onion is chopped

Figure 5: Several example test images and their predicted actions and predicted effect descriptions. The actions in bold are ground-truth labels.

	MAP	Top 1	Top 5	Top 20
BS+Seed+Act+Eff	**0.290**	**0.414**	**0.750**	**0.921**
BS+Seed+Act	0.252	**0.414**	0.721	0.893
Seed+Act+Eff	0.247	0.314	0.679	0.886
Seed+Act	0.241	0.371	0.650	0.814
Seed	0.182	0.329	0.629	0.807

Table 3: Results for the action-effect prediction task (given an action, rank all the candidate images).

	MAP	Top 1	Top 5	Top 20
BS+Seed+Act+Eff	**0.660**	**0.523**	**0.843**	**0.954**
BS+Seed+Act	0.642	0.508	0.802	0.924
Seed+Act+Eff	0.289	0.176	0.398	0.625
Seed+Act	0.481	0.301	0.724	0.926
Seed	0.634	0.520	0.765	0.892

Table 4: Results for the action-effect prediction task (given an image, rank all the actions).

(5) *Seed*. A baseline method that was only trained on the seeding image data, using the vanilla cross-entropy objective.

Evaluation Results

We apply the trained classification model to all of the test images. Based on the matrix of prediction scores, we can evaluate action-effect prediction from two angles: (1) given an action class, rank all the candidate images; (2) given an image, rank all the candidate action classes. Table 3 and 4 show the results for these two angels respectively. We report both mean average precision (MAP) and top prediction accuracy.

Overall, *BS+Seed+Act+Eff* gives the best performance. By comparing the bootstrap approach with baseline approaches (i.e., *BS+Seed+Act+Eff* vs. *Seed+Act+Eff*, and *BS+Seed+Act* vs. *Seed+Act*), the bootstrapping approaches clearly outperforms their counterparts, demonstrating its ability in handling noisy web data. Comparing *BS+Seed+Act+Eff* with *BS+Seed+Act*, we can see that *BS+Seed+Act+Eff* performs better. This indicates the use of effect descriptions can bring more relevant images to train better models for action-effect prediction.

In Table 4, the poor performance of *Seed+Act+Eff* and *Seed+Act* shows that it is risky to fully rely on the noisy web search results. These two methods had trouble in distinguishing the background class from the rest.

We further trained another multi-class classifier with web effect images, using their corresponding effect phrases as class labels. Given a test image, we apply this new classifier to predict the effect descriptions of this image. Figure 5 shows some example images, their predicted actions based on our bootstrapping approach and their predicted effect phrases based on the new classifier. These examples also demonstrate another advantage of incorporating seed effect knowledge from language data: it provides state descriptions that can be used to better explain the perceived state. Such explanation can be crucial in human-agent communication for action planning and reasoning.

5 Generalizing Effect Knowledge to New Verb-Noun Pairs

In real applications, it is very likely that we do not have the effect knowledge (i.e., language effect descriptions) for every verb-noun pair. And annotat-

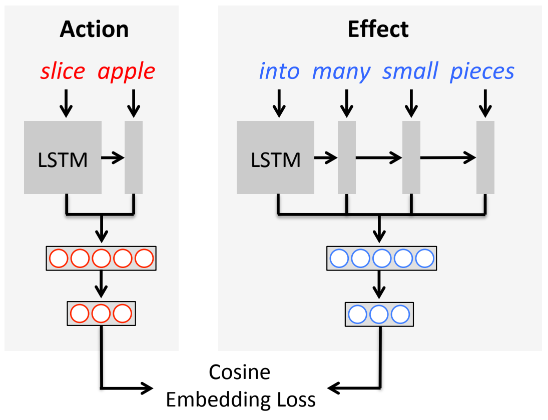


Figure 6: Architecture of the action-effect embedding model.

ing effect knowledge using language (as shown in Section 3) can be very expensive. In this section, we describe how to potentially generalize seed effect knowledge to new verb-noun pairs through an embedding model.

5.1 Action-Effect Embedding Model

The structure of our model is shown in Figure 6. It is composed of two sub-networks: one for verb-noun pairs (i.e., *action*) and the other one for effect phrases (i.e, *effect*). The *action* or *effect* is fed into an LSTM encoder and then to two fully-connected layers. The output is an action embedding $\mathbf{v}_c$ and effect embedding $\mathbf{v}_e$. The networks are trained by minimizing the following cosine embedding loss function:

$$\mathcal{L}(\mathbf{v}_c, \mathbf{v}_e) = \begin{cases} 1 - \mathrm{s}(\mathbf{v}_c, \mathbf{v}_e), & \text{if } (c, e) \in T \\ \max(0, \mathrm{s}(\mathbf{v_c}, \mathbf{v_e})), & \text{if } (c, e) \notin T \end{cases}$$

$\mathrm{s}(\cdot, \cdot)$ is the cosine similarity between vectors. T is a collection of action-effect pairs. Suppose c is an input for *action* and e is an input for *effect*, this loss function will learn an action and effect semantic space that maximizes the similarities between c and e if they have an action-effect relation (i.e., $(c, e) \in T$). During training, the negative action-effect pairs (i.e., $(c, e) \notin T$) are randomly sampled from data. In the experiments, the negative sampling ratio is set to 25. That is, for each positive action-effect pair, 25 negative pairs are created through random sampling.

At the inference step, given an unseen verb-noun pair, we embed it into the action and effect semantic space. Its embedding vector will be used to calculate similarities with all the embedding vectors of the candidate effect phrases.

	MAP	Top 1	Top 5
BS+Seed+Act+Eff	**0.529**	0.643	0.928
BS+Seed+Act+pEff	0.507	0.642	0.893
BS+Seed+Act	0.435	0.643	**0.964**
Seed	0.369	**0.678**	0.786

Table 5: Results for the action-effect prediction task (given an action, rank all the candidate images).

	MAP	Top 1	Top 5
BS+Seed+Act+Eff	**0.733**	**0.574**	0.947
BS+Seed+Act+pEff	0.729	0.551	**0.961**
BS+Seed+Act	0.724	0.557	0.933
Seed	0.705	0.557	0.898

Table 6: Results for the action-effect prediction task (given an image, rank all the actions).

5.2 Evaluation

We divided the 140 verb-noun pairs into 70% training set (98 verb-noun pairs), 10% development set (14) and 20% test set (28). For the action-effect embedding model, we use pre-trained GloVe word embeddings (Pennington et al., 2014) as input to the LSTM. The embedding model was trained using the language effect data corresponding to the training verb-noun pairs, and then it was applied to predict effect phrases for the unseen verb-noun pairs in the test set. For each unseen verb-noun pair, we collected its top five predicted effect phrases. Each predicted effect phrase was then used as query keywords to download web effect images. This set of web images are referred to as *pEff* and will be used in training the action-effect prediction model.

For each of the 28 test (i.e., new) verb-noun pairs, we use the same ratio 10% (about 3 examples) of the human annotated images as the seeding images, which were combined with downloaded web images to train the prediction model. The remaining 30% and 60% are used as the development set, and the test set. We compare the following different configurations:

(1) *BS+Seed+Act+pEff*. The bootstrapping approach trained on the seeding images, the action web images, and the web images downloaded using the predicted effect phrases.

(2) *BS+Seed+Act+Eff*. The bootstrapping approach trained on the seeding images, the action web images, and the effect web images (downloaded using ground-truth effect phrases).

(3) *BS+Seed+Act*. The bootstrapping approach trained on the seeding images and the action web

Action Text	Predicted Effect Text
chop **carrot**	carrot into sandwiches, carrot is sliced, carrot is cut thinly, carrot into different pieces, carrot is divided
ignite paper	paper is being charred , paper is being burned, paper is set, paper is being destroyed, paper is lit
mash potato	potato into chunks, potato into sandwiches, potato into slices, potato is chewed, potato into smaller pieces

Table 7: Example predicted effect phrases for new verb-noun pairs. Unseen verbs and nouns are shown in bold.

images.

(4) *Seed*. A baseline only trained on the seeding images.

Table 5 and 6 show the results for the action-effect prediction task for unseen verb-noun pairs. From the results we can see that *BS+Seed+Act+pEff* achieves close performance compared with *BS+Seed+Act+Eff*, which uses human annotated effect phrases. Although in most cases, *BS+Seed+Act+pEff* outperforms the baseline, which seems to point to the possibility that semantic embedding space can be employed to extend effect knowledge to new verb-noun pairs. However, the current results are not conclusive partly due to the small testing set. More in-depth evaluation is needed in the future.

Table 7 shows top predicted effect phrases for several new verb-noun pairs. After analyzing the action-effect prediction results we notice that generalizing the effect knowledge to a verb-noun pair that contains an unseen verb tends to be more difficult than generalizing to a verb-noun pair that contains an unseen noun. Among the 28 test verb-noun pairs, 12 of them contain unseen verbs and known nouns, 7 of them contain unseen nouns and known verbs. For the task of ranking images given an action, the mean average precision is 0.447 for the unseen verb cases and 0.584 for the unseen noun cases. Although not conclusive, this might indicate that, verbs tend to capture more information about the effect states of the world than nouns.

6 Discussion and Conclusion

When robots operate in the physical world, they not only need to perceive the world, but also need to act to the world. They need to understand the current state, to map their goals to the world state, and to plan for actions that can lead to the goals. All of these point to the importance of the ability to understand causal relations between actions and the state of the world. To address this issue, this paper introduces a new task on action-effect prediction.

Particularly, we focus on modeling the connection between an action (a verb-noun pair) and its effect as illustrated in an image and treat natural language effect descriptions as side knowledge to help acquiring web image data and bootstrap training. Our current model is very simple and performance is yet to be improved. We plan to apply more advanced approaches in the future, for example, attention models that jointly capture actions, image states, and effect descriptions. We also plan to incorporate action-effect prediction to human-robot collaboration, for example, to bridge the gap of commonsense knowledge about the physical world between humans and robots.

This paper presents an initial investigation on action-effect prediction. There are many challenges and unknowns, from problem formulation to knowledge representation; from learning and inference algorithms to methods and metrics for evaluations. Nevertheless, we hope this work can motivate more research in this area, enabling physical action-effect reasoning, towards agents which can perceive, act, and communicate with humans in the physical world.

Acknowledgments

This work was supported by the National Science Foundation (IIS-1617682) and the DARPA XAI program under a subcontract from UCLA (N66001-17-2-4029). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Renée Baillargeon. 2004. Infants' physical world. *Current directions in psychological science*, 13(3):89–94.

Tamara L Berg, Alexander C Berg, and Jonathan Shih. 2010. Automatic attribute discovery and characterization from noisy web data. In *European Conference on Computer Vision*, pages 663–676. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Yu-Wei Chao, Zhan Wang, Rada Mihalcea, and Jia Deng. 2015. Mining semantic affordances of visual object categories. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4259–4267. IEEE.

Stephen V Cole, Matthew D Royal, Marco G Valtorta, Michael N Huhns, and John B Bowles. 2005. A lightweight tool for automatically extracting causal relationships from text. In *SoutheastCon, 2006. Proceedings of the IEEE*, pages 125–129. IEEE.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics.

Curt J Ducasse. 1926. On the nature and the observability of the causal relation. *The Journal of Philosophy*, 23(3):57–68.

Alireza Fathi and James M Rehg. 2013. Modeling actions through state changes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2579–2586. IEEE.

Robert Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. 2005. Learning object categories from google's image search. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1816–1823. IEEE.

Amy Fire and Song-Chun Zhu. 2016. Learning perceptual causality from video. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(2):23.

Maxwell Forbes and Yejin Choi. 2017. Verb physics: Relative physical knowledge of actions and objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 266–276.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1814–1824.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 241–247.

Alison Gopnik, Laura Schulz, and Laura Elizabeth Schulz. 2007. *Causal learning: Psychology, philosophy, and computation*. Oxford University Press.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Lyndon S Kennedy, Shih-Fu Chang, and Igor V Kozintsev. 2006. To search or to label?: predicting the performance of search-based automatic image classifiers. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 249–258. ACM.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1015–1026. Association for Computational Linguistics.

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 992–1002.

Tanmoy Mukherjee and Timothy Hospedales. 2016. Gaussian visual-linguistic embedding for zero-shot recognition. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 912–918.

Mayu Otani, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Naokazu Yokoya. 2016. Learning joint representations of videos and sentences with web image search. In *European Conference on Computer Vision*, pages 651–667. Springer.

Judea Pearl et al. 2009. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Nazneen Fatema Rajani and Raymond J Mooney. 2016. Combining supervised and unsupervised ensembles for knowledge base population. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. In *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 29–36. IEEE.

Deb Roy. 2005. Grounding words in perception and action: computational insights. *Trends in cognitive sciences*, 9(8):389–396.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 138–148.

Lanbo She and Joyce Chai. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 108–117.

Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1634–1644.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, volume 1, page 2.

Yao-Hung Hubert Tsai and Ruslan Salakhutdinov. 2017. Improving one-shot learning through fusing side information. *arXiv preprint arXiv:1710.08347*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE.

Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. 2016. Actions˜ transformations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2658–2667.

Max Whitney and Anoop Sarkar. 2012. Bootstrapping via graph propagation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 620–628. Association for Computational Linguistics.

Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. 2016. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, page 7.

Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. 2017. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, pages 152–163.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Xun Xu, Timothy Hospedales, and Shaogang Gong. 2017a. Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision*, 123(3):309–333.

Xun Xu, Timothy M Hospedales, and Shaogang Gong. 2016. Multi-task zero-shot action recognition with prioritised data augmentation. In *European Conference on Computer Vision*, pages 343–359. Springer.

Zhongwen Xu, Linchao Zhu, and Yi Yang. 2017b. Few-shot object recognition from machine-labeled web images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y Chai. 2016. Grounded semantic role labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 149–159.

Xuefeng Yang and Kezhi Mao. 2014. Multi level causal relation identification using extended features. *Expert Systems with Applications*, 41(16):7171–7181.

Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. 2013. Detection of manipulation action consequences (mac). In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2563–2570. IEEE.

Mark Yatskar, Vicente Ordonez, and Ali Farhadi. 2016. Stating the obvious: Extracting visual common sense knowledge. In *Proceedings of NAACL-HLT*, pages 193–198.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yipin Zhou and Tamara L Berg. 2016. Learning temporal transformations from time-lapse videos. In *European Conference on Computer Vision*, pages 262–277. Springer.

Transformation Networks for Target-Oriented Sentiment Classification*

Xin Li[1], Lidong Bing[2], Wai Lam[1] and Bei Shi[1]
[1]Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong
[2]Tencent AI Lab, Shenzhen, China
{lixin,wlam,bshi}@se.cuhk.edu.hk
lyndonbing@tencent.com

Abstract

Target-oriented sentiment classification aims at classifying sentiment polarities over individual opinion targets in a sentence. RNN with attention seems a good fit for the characteristics of this task, and indeed it achieves the state-of-the-art performance. After re-examining the drawbacks of attention mechanism and the obstacles that block CNN to perform well in this classification task, we propose a new model to overcome these issues. Instead of attention, our model employs a CNN layer to extract salient features from the transformed word representations originated from a bi-directional RNN layer. Between the two layers, we propose a component to generate target-specific representations of words in the sentence, meanwhile incorporate a mechanism for preserving the original contextual information from the RNN layer. Experiments show that our model achieves a new state-of-the-art performance on a few benchmarks.[1]

1 Introduction

Target-oriented (also mentioned as "target-level" or "aspect-level" in some works) sentiment classification aims to determine sentiment polarities over "opinion targets" that explicitly appear in the sentences (Liu, 2012). For example, in the sentence *"I am pleased with the fast **log on**, and the long **battery life**"*, the user mentions two targets "***log on***" and "***better life***", and expresses positive sentiments over them. The task is usually formulated as predicting a sentiment category for a (target, sentence) pair.

Recurrent Neural Networks (RNNs) with attention mechanism, firstly proposed in machine translation (Bahdanau et al., 2014), is the most commonly-used technique for this task. For example, Wang et al. (2016); Tang et al. (2016b); Yang et al. (2017); Liu and Zhang (2017); Ma et al. (2017) and Chen et al. (2017) employ attention to measure the semantic relatedness between each context word and the target, and then use the induced attention scores to aggregate contextual features for prediction. In these works, the attention weight based combination of word-level features for classification may introduce noise and downgrade the prediction accuracy. For example, in "*This **dish** is my favorite and I always get it and never get tired of it.*", these approaches tend to involve irrelevant words such as "never" and "tired" when they highlight the opinion modifier "favorite". To some extent, this drawback is rooted in the attention mechanism, as also observed in machine translation (Luong et al., 2015) and image captioning (Xu et al., 2015).

Another observation is that the sentiment of a target is usually determined by key phrases such as "is my favorite". By this token, Convolutional Neural Networks (CNNs)—whose capability for extracting the informative n-gram features (also called "active local features") as sentence representations has been verified in (Kim, 2014; Johnson and Zhang, 2015)— should be a suitable model for this classification problem. However, CNN likely fails in cases where a sentence expresses different sentiments over multiple targets, such as *"great **food** but the **service** was dreadful!"*. One reason is that CNN cannot fully explore the target information as done by RNN-based meth-

*The work was done when Xin Li was an intern at Tencent AI Lab. This project is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

[1]Our code is open-source and available at https://github.com/lixin4ever/TNet

ods (Tang et al., 2016a).[2] Moreover, it is hard for vanilla CNN to differentiate opinion words of multiple targets. Precisely, multiple active local features holding different sentiments (e.g., "great food" and "service was dreadful") may be captured for a single target, thus it will hinder the prediction.

We propose a new architecture, named Target-Specific **T**ransformation **Net**works (TNet), to solve the above issues in the task of target sentiment classification. TNet firstly encodes the context information into word embeddings and generates the contextualized word representations with LSTMs. To integrate the target information into the word representations, TNet introduces a novel Target-Specific Transformation (TST) component for generating the target-specific word representations. Contrary to the previous attention-based approaches which apply the same target representation to determine the attention scores of individual context words, TST firstly generates different representations of the target conditioned on individual context words, then it consolidates each context word with its tailor-made target representation to obtain the transformed word representation. Considering the context word *"long"* and the target "*battery life*" in the above example, TST firstly measures the associations between "long" and individual target words. Then it uses the association scores to generate the target representation conditioned on "long". After that, TST transforms the representation of "long" into its target-specific version with the new target representation. Note that "long" could also indicate a negative sentiment (say for "startup time"), and the above TST is able to differentiate them.

As the context information carried by the representations from the LSTM layer will be lost after the non-linear TST, we design a context-preserving mechanism to contextualize the generated target-specific word representations. Such mechanism also allows deep transformation structure to learn abstract features[3]. To help the CNN feature extractor locate sentiment indicators more accurately, we adopt a proximity strategy to scale the input of convolutional layer with positional relevance between a word and the target.

[2]One method could be concatenating the target representation with each word representation, but the effect as shown in (Wang et al., 2016) is limited.

[3]Abstract features usually refer to the features ultimately useful for the task (Bengio et al., 2013; LeCun et al., 2015).

In summary, our contributions are as follows:

• TNet adapts CNN to handle target-level sentiment classification, and its performance dominates the state-of-the-art models on benchmark datasets.

• A novel Target-Specific Transformation component is proposed to better integrate target information into the word representations.

• A context-preserving mechanism is designed to forward the context information into a deep transformation architecture, thus, the model can learn more abstract contextualized word features from deeper networks.

2 Model Description

Given a target-sentence pair $(\mathbf{w}^\tau, \mathbf{w})$, where $\mathbf{w}^\tau = \{w_1^\tau, w_2^\tau, ..., w_m^\tau\}$ is a sub-sequence of $\mathbf{w} = \{w_1, w_2, ..., w_n\}$, and the corresponding word embeddings $\mathbf{x}^\tau = \{x_1^\tau, x_2^\tau, ..., x_m^\tau\}$ and $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, the aim of target sentiment classification is to predict the sentiment polarity $y \in \{P, N, O\}$ of the sentence $\mathbf{w}$ over the target $\mathbf{w}^\tau$, where P, N and O denote "positive", "negative" and "neutral" sentiments respectively.

The architecture of the proposed Target-Specific Transformation Networks (TNet) is shown in Fig. 1. The bottom layer is a BiLSTM which transforms the input $\mathbf{x} = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^{n\times \dim_w}$ into the contextualized word representations $\mathbf{h}^{(0)} = \{h_1^{(0)}, h_2^{(0)}, ..., h_n^{(0)}\} \in \mathbb{R}^{n\times 2\dim_h}$ (i.e. hidden states of BiLSTM), where $\dim_w$ and $\dim_h$ denote the dimensions of the word embeddings and the hidden representations respectively. The middle part, the core part of our TNet, consists of L Context-Preserving Transformation (CPT) layers. The CPT layer incorporates the target information into the word representations via a novel Target-Specific Transformation (TST) component. CPT also contains a context-preserving mechanism, resembling identity mapping (He et al., 2016a,b) and highway connection (Srivastava et al., 2015a,b), allows preserving the context information and learning more abstract word-level features using a deep network. The top most part is a position-aware convolutional layer which first encodes positional relevance between a word and a target, and then extracts informative features for classification.

2.1 Bi-directional LSTM Layer

As observed in Lai et al. (2015), combining contextual information with word embeddings is an

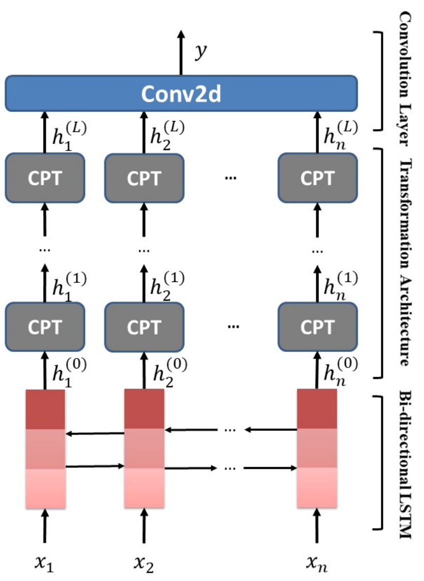


Figure 1: Architecture of TNet.

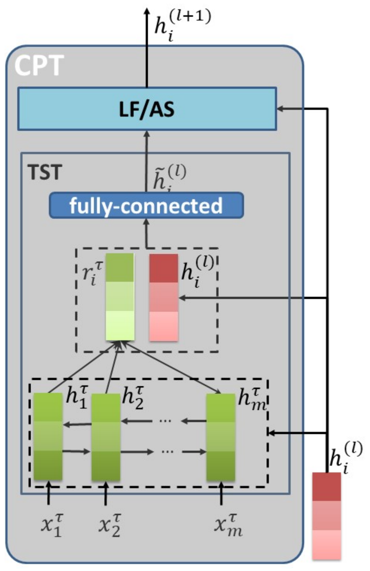


Figure 2: Details of a CPT module.

effective way to represent a word in convolution-based architectures. TNet also employs a BiLSTM to accumulate the context information for each word of the input sentence, i.e., the bottom part in Fig. 1. For simplicity and space issue, we denote the operation of an LSTM unit on x_i as $\mathrm{LSTM}(x_i)$. Thus, the contextualized word representation $h_i^{(0)} \in \mathbb{R}^{2\dim_h}$ is obtained as follows:

$$h_i^{(0)} = [\overrightarrow{\mathrm{LSTM}}(x_i); \overleftarrow{\mathrm{LSTM}}(x_i)], i \in [1, n]. \quad (1)$$

2.2 Context-Preserving Transformation

The above word-level representation has not considered the target information yet. Traditional attention-based approaches keep the word-level features static and aggregate them with weights as the final sentence representation. In contrast, as shown in the middle part in Fig. 1, we introduce multiple CPT layers and the detail of a single CPT is shown in Fig. 2. In each CPT layer, a tailor-made TST component that aims at better consolidating word representation and target representation is proposed. Moreover, we design a context-preserving mechanism enabling the learning of target-specific word representations in a deep neural architecture.

2.2.1 Target-Specific Transformation

TST component is depicted with the TST block in Fig. 2. The first task of TST is to generate the representation of the target. Previous methods (Chen et al., 2017; Liu and Zhang, 2017) average the embeddings of the target words as the target representation. This strategy may be inappropriate in some cases because different target words usually do not contribute equally. For example, in the target "*amd turin processor*", the word "processor" is more important than "amd" and "turin", because the sentiment is usually conveyed over the phrase head, i.e.,"processor", but seldom over modifiers (such as brand name "amd"). Ma et al. (2017) attempted to overcome this issue by measuring the importance score between each target word representation and the averaged sentence vector. However, it may be ineffective for sentences expressing multiple sentiments (e.g., *"Air has higher resolution but the fonts are small."*), because taking the average tends to neutralize different sentiments.

We propose to dynamically compute the importance of target words based on each sentence word rather than the whole sentence. We first employ another BiLSTM to obtain the target word representations $\mathbf{h}^\tau \in \mathbb{R}^{m \times 2\dim_h}$:

$$h_j^\tau = [\overrightarrow{\mathrm{LSTM}}(x_j^\tau); \overleftarrow{\mathrm{LSTM}}(x_j^\tau)], j \in [1, m]. \quad (2)$$

Then, we dynamically associate them with each word w_i in the sentence to tailor-make target representation r_i^τ at the time step i:

$$r_i^\tau = \sum_{j=1}^{m} h_j^\tau * \mathcal{F}(h_i^{(l)}, h_j^\tau), \quad (3)$$

where the function $\mathcal{F}$ measures the relatedness between the j-th target word representation h_j^τ and the i-th word-level representation $h_i^{(l)}$:

$$\mathcal{F}(h_i^{(l)}, h_j^\tau) = \frac{\exp{(h_i^{(l)\top} h_j^\tau)}}{\sum_{k=1}^m \exp{(h_i^{(l)\top} h_k^\tau)}}. \quad (4)$$

Finally, the concatenation of r_i^τ and $h_i^{(l)}$ is fed into a fully-connected layer to obtain the i-th target-specific word representation $\tilde{h}_i^{(l)}$:

$$\tilde{h}_i^{(l)} = g(W^\tau[h_i^{(l)} : r_i^\tau] + b^\tau), \quad (5)$$

where $g(*)$ is a non-linear activation function and ":" denotes vector concatenation. W^τ and b^τ are the weights of the layer.

2.2.2 Context-Preserving Mechanism

After the non-linear TST (see Eq. 5), the context information captured with contextualized representations from the BiLSTM layer will be lost since the mean and the variance of the features within the feature vector will be changed. To take advantage of the context information, which has been proved to be useful in (Lai et al., 2015), we investigate two strategies: Lossless Forwarding (LF) and Adaptive Scaling (AS), to pass the context information to each following layer, as depicted by the block "**LF/AS**" in Fig. 2. Accordingly, the model variants are named **TNet-LF** and **TNet-AS**.

Lossless Forwarding. This strategy preserves context information by directly feeding the features before the transformation to the next layer. Specifically, the input $h_i^{(l+1)}$ of the $(l+1)$-th CPT layer is formulated as:

$$h_i^{(l+1)} = h_i^{(l)} + \tilde{h}_i^{(l)}, i \in [1, n], l \in [0, L], \quad (6)$$

where $h_i^{(l)}$ is the input of the l-th layer and $\tilde{h}_i^{(l)}$ is the output of TST in this layer. We unfold the recursive form of Eq. 6 as follows:

$$h_i^{(l+1)} = h_i^{(0)} + \text{TST}(h_i^{(0)}) + \cdots + \text{TST}(h_i^{(l)}). \quad (7)$$

Here, we denote $\tilde{h}_i^{(l)}$ as $\text{TST}(h_i^{(l)})$. From Eq. 7, we can see that the output of each layer will contain the contextualized word representations (i.e., $h_i^{(0)}$), thus, the context information is encoded into the transformed features. We call this strategy "Lossless Forwarding" because the contextualized representations and the transformed representations (i.e., $\text{TST}(h_i^{(l)})$) are kept unchanged during the feature combination.

Adaptive Scaling. Lossless Forwarding introduces the context information by directly adding back the contextualized features to the transformed features, which raises a question: Can the weights of the input and the transformed features be adjusted dynamically? With this motivation, we propose another strategy, named "Adaptive Scaling". Similar to the gate mechanism in RNN variants (Jozefowicz et al., 2015), Adaptive Scaling introduces a gating function to control the passed proportions of the transformed features and the input features. The gate $\mathbf{t}^{(l)}$ as follows:

$$t_i^{(l)} = \sigma(W_{trans} h_i^{(l)} + b_{trans}), \quad (8)$$

where $t_i^{(l)}$ is the gate for the i-th input of the l-th CPT layer, and σ is the *sigmoid* activation function. Then we perform convex combination of $h_i^{(l)}$ and $\tilde{h}_i^{(l)}$ based on the gate:

$$h_i^{(l+1)} = t_i^{(l)} \odot \tilde{h}_i^{(l)} + (1 - t_i^{(l)}) \odot h_i^{(l)}. \quad (9)$$

Here, $\odot$ denotes element-wise multiplication. The non-recursive form of this equation is as follows (for clarity, we ignore the subscripts):

$$h^{(l+1)} = [\prod_{k=0}^{l}(1 - t^{(k)})] \odot h^{(0)}$$
$$+[t^{(0)} \prod_{k=1}^{l}(1 - t^{(k)})] \odot \text{TST}(h^{(0)}) + \cdots$$
$$+t^{(l-1)}(1 - t^{(l)}) \odot \text{TST}(h^{(l-1)}) + t^{(l)} \odot \text{TST}(h^{(l)}).$$

Thus, the context information is integrated in each upper layer and the proportions of the contextualized representations and the transformed representations are controlled by the computed gates in different transformation layers.

2.3 Convolutional Feature Extractor

Recall that the second issue that blocks CNN to perform well is that vanilla CNN may associate a target with unrelated general opinion words which are frequently used as modifiers for different targets across domains. For example, "***service***" in *"Great food but the service is dreadful"* may be associated with both "great" and "dreadful". To solve it, we adopt a proximity strategy, which is observed effective in (Chen et al., 2017; Li and Lam, 2017). The idea is a closer opinion word is more likely to be the actual modifier of the target.

		# Positive	# Negative	# Neutral
LAPTOP	Train	980	858	454
	Test	340	128	171
REST	Train	2159	800	632
	Test	730	195	196
TWITTER	Train	1567	1563	3127
	Test	174	174	346

Table 1: Statistics of datasets.

Specifically, we first calculate the position relevance v_i between the i-th word and the target[4]:

$$v_i = \begin{cases} 1 - \frac{(k+m-i)}{C} & i < k+m \\ 1 - \frac{i-k}{C} & k+m \le i \le n \\ 0 & i > n \end{cases} \quad (10)$$

where k is the index of the first target word, C is a pre-specified constant, and m is the length of the target $\mathbf{w}^\tau$. Then, we use v to help CNN locate the correct opinion w.r.t. the given target:

$$\hat{h}_i^{(l)} = h_i^{(l)} * v_i, i \in [1, n], l \in [1, L]. \quad (11)$$

Based on Eq. 10 and Eq. 11, the words close to the target will be highlighted and those far away will be downgraded. v is also applied on the intermediate output to introduce the position information into each CPT layer. Then we feed the weighted $\mathbf{h}^{(L)}$ to the convolutional layer, i.e., the top-most layer in Fig. 1, to generate the feature map $\mathbf{c} \in \mathbb{R}^{n-s+1}$ as follows:

$$c_i = \text{ReLU}(\boldsymbol{w}_{conv}^\top \mathbf{h}_{i:i+s-1}^{(L)} + b_{conv}), \quad (12)$$

where $\mathbf{h}_{i:i+s-1}^{(L)} \in \mathbb{R}^{s \cdot \dim_h}$ is the concatenated vector of $\hat{h}_i^{(L)}, \cdots, \hat{h}_{i+s-1}^{(L)}$, and s is the kernel size. $\boldsymbol{w}_{conv} \in \mathbb{R}^{s \cdot \dim_h}$ and $b_{conv} \in \mathbb{R}$ are learnable weights of the convolutional kernel. To capture the most informative features, we apply max pooling (Kim, 2014) and obtain the sentence representation $z \in \mathbb{R}^{n_k}$ by employing n_k kernels:

$$z = [\max(\mathbf{c}_1), \cdots, \max(\mathbf{c}_{n_k})]^\top. \quad (13)$$

Finally, we pass z to a fully connected layer for sentiment prediction:

$$p(y|\mathbf{w}^\tau, \mathbf{w}) = \text{Softmax}(W_f z + b_f). \quad (14)$$

where W_f and b_f are learnable parameters.

[4]As we perform sentence padding, it is possible that the index i is larger than the actual length n of the sentence.

3 Experiments

3.1 Experimental Setup

As shown in Table 1, we evaluate the proposed TNet on three benchmark datasets: LAPTOP and REST are from SemEval ABSA challenge (Pontiki et al., 2014), containing user reviews in laptop domain and restaurant domain respectively. We also remove a few examples having the "conflict label" as done in (Chen et al., 2017); TWITTER is built by Dong et al. (2014), containing twitter posts. All tokens are lowercased without removal of stop words, symbols or digits, and sentences are zero-padded to the length of the longest sentence in the dataset. Evaluation metrics are Accuracy and Macro-Averaged F1 where the latter is more appropriate for datasets with unbalanced classes. We also conduct pairwise t-test on both Accuracy and Macro-Averaged F1 to verify if the improvements over the compared models are reliable.

TNet is compared with the following methods.

- **SVM** (Kiritchenko et al., 2014): It is a traditional support vector machine based model with extensive feature engineering;
- **AdaRNN** (Dong et al., 2014): It learns the sentence representation toward target for sentiment prediction via semantic composition over dependency tree;
- **AE-LSTM**, and **ATAE-LSTM** (Wang et al., 2016): AE-LSTM is a simple LSTM model incorporating the target embedding as input, while ATAE-LSTM extends AE-LSTM with attention;
- **IAN** (Ma et al., 2017): IAN employs two LSTMs to learn the representations of the context and the target phrase interactively;
- **CNN-ASP**: It is a CNN-based model implemented by us which directly concatenates target representation to each word embedding;
- **TD-LSTM** (Tang et al., 2016a): It employs two LSTMs to model the left and right contexts of the target separately, then performs predictions based on concatenated context representations;
- **MemNet** (Tang et al., 2016b): It applies attention mechanism over the word embeddings multiple times and predicts sentiments

Hyper-params	**TNet-LF**			**TNet-AS**		
	LAPTOP	REST	TWITTER	LAPTOP	REST	TWITTER
dim_w		300			300	
dim_h		50			50	
dropout rates (p_{lstm}, p_{sent})		(0.3, 0.3)			(0.3, 0.3)	
L		2			2	
batch size	64	25	64	64	32	64
s		3			3	
n_k		50			100	
C		40.0			30.0	

Table 2: Settings of hyper-parameters.

based on the top-most sentence representations;

- **BILSTM-ATT-G** (Liu and Zhang, 2017): It models left and right contexts using two attention-based LSTMs and introduces gates to measure the importance of left context, right context, and the entire sentence for the prediction;

- **RAM** (Chen et al., 2017): RAM is a multilayer architecture where each layer consists of attention-based aggregation of word features and a GRU cell to learn the sentence representation.

We run the released codes of TD-LSTM and BILSTM-ATT-G to generate results, since their papers only reported results on TWITTER. We also rerun MemNet on our datasets and evaluate it with both accuracy and Macro-Averaged F1.[5]

We use pre-trained GloVe vectors (Pennington et al., 2014) to initialize the word embeddings and the dimension is 300 (i.e., $\text{dim}_w = 300$). For out-of-vocabulary words, we randomly sample their embeddings from the uniform distribution $\mathcal{U}(-0.25, 0.25)$, as done in (Kim, 2014). We only use one convolutional kernel size because it was observed that CNN with single optimal kernel size is comparable with CNN having multiple kernel sizes on small datasets (Zhang and Wallace, 2017). To alleviate overfitting, we apply dropout on the input word embeddings of the LSTM and the ultimate sentence representation z. All weight matrices are initialized with the uniform distribution $\mathcal{U}(-0.01, 0.01)$ and the biases are initialized as zeros. The training objective is cross-entropy, and Adam (Kingma and Ba, 2015) is adopted as the optimizer by following the learning rate and the decay rates in the original paper.

The hyper-parameters of TNet-LF and TNet-AS are listed in Table 2. Specifically, all hyper-parameters are tuned on 20% randomly held-out training data and the hyper-parameter collection producing the highest accuracy score is used for testing. Our model has comparable number of parameters compared to traditional LSTM-based models as we reuse parameters in the transformation layers and BiLSTM.[6]

3.2 Main Results

As shown in Table 3, both TNet-LF and TNet-AS consistently achieve the best performance on all datasets, which verifies the efficacy of our whole TNet model. Moreover, TNet can perform well for different kinds of user generated content, such as product reviews with relatively formal sentences in LAPTOP and REST, and tweets with more ungrammatical sentences in TWITTER. The reason is the CNN-based feature extractor arms TNet with more power to extract accurate features from ungrammatical sentences. Indeed, we can also observe that another CNN-based baseline, i.e., CNN-ASP implemented by us, also obtains good results on TWITTER.

On the other hand, the performance of those comparison methods is mostly unstable. For the tweet in TWITTER, the competitive BILSTM-ATT-G and RAM cannot perform as effective as they do for the reviews in LAPTOP and REST, due to the fact that they are heavily rooted in LSTMs and the ungrammatical sentences hinder their ca-

[5]The codes of TD-LSTM/MemNet and BILSTM-ATT-G are available at: `http://ir.hit.edu.cn/~dytang` and `http://leoncrashcode.github.io`. Note that MemNet was only evaluated with accuracy.

[6]All experiments are conducted on a single NVIDIA GTX 1080. The prediction cost of a sentence is about 2 ms.

	Models	LAPTOP		REST		TWITTER	
		ACC	Macro-F1	ACC	Macro-F1	ACC	Macro-F1
Baselines	SVM	70.49$^\natural$	-	80.16$^\natural$	-	63.40*	63.30*
	AdaRNN	-	-	-	-	66.30$^\natural$	65.90$^\natural$
	AE-LSTM	68.90$^\natural$	-	76.60$^\natural$	-	-	-
	ATAE-LSTM	68.70$^\natural$	-	77.20$^\natural$	-	-	-
	IAN	72.10$^\natural$	-	78.60$^\natural$	-	-	-
	CNN-ASP	72.46	65.31	77.82	65.11	73.27	71.77
	TD-LSTM	71.83	68.43	78.00	66.73	66.62	64.01
	MemNet	70.33	64.09	78.16	65.83	68.50	66.91
	BILSTM-ATT-G	74.37	69.90	80.38	70.78	72.70	70.84
	RAM	75.01	70.51	79.79	68.86	71.88	70.33
CPT Alternatives	LSTM-ATT-CNN	73.37	68.03	78.95	68.71	70.09	67.68
	LSTM-FC-CNN-LF	75.59	70.60	80.41	70.23	73.70	72.82
	LSTM-FC-CNN-AS	75.78	70.72	80.23	70.06	74.28	72.60
Ablated TNet	TNet w/o transformation	73.30	68.25	78.90	65.86	72.10	70.57
	TNet w/o context	73.91	68.87	80.07	69.01	74.51	73.05
	TNet-LF w/o position	75.13	70.63	79.86	69.69	73.83	72.49
	TNet-AS w/o position	75.27	70.03	79.79	69.78	73.84	72.47
TNet variants	TNet-LF	76.01†,‡	71.47†,‡	**80.79**†,‡	70.84‡	74.68†,‡	73.36†,‡
	TNet-AS	**76.54**†,‡	**71.75**†,‡	80.69†,‡	**71.27**†,‡	**74.97**†,‡	**73.60**†,‡

Table 3: Experimental results (%). The results with symbol"$\natural$" are retrieved from the original papers, and those starred ($*$) one are from Dong et al. (2014). The marker † refers to p-value < 0.01 when comparing with BILSTM-ATT-G, while the marker ‡ refers to p-value < 0.01 when comparing with RAM.

pability in capturing the context features. Another difficulty caused by the ungrammatical sentences is that the dependency parsing might be error-prone, which will affect those methods such as AdaRNN using dependency information.

From the above observations and analysis, some takeaway message for the task of target sentiment classification could be:

- LSTM-based models relying on sequential information can perform well for formal sentences by capturing more useful context features;
- For ungrammatical text, CNN-based models may have some advantages because CNN aims to extract the most informative n-gram features and is thus less sensitive to informal texts without strong sequential patterns.

3.3 Performance of Ablated TNet

To investigate the impact of each component such as deep transformation, context-preserving mechanism, and positional relevance, we perform comparison between the full TNet models and its ablations (the third group in Table 3). After removing the deep transformation (i.e., the techniques introduced in Section 2.2), both TNet-LF and TNet-AS are reduced to TNet w/o transformation (where position relevance is kept), and their results in both accuracy and F1 measure are incomparable with those of TNet. It shows that the integration of target information into the word-level representations is crucial for good performance.

Comparing the results of TNet and TNet w/o context (where TST and position relevance are kept), we observe that the performance of TNet w/o context drops significantly on LAPTOP and REST[7], while on TWITTER, TNet w/o context performs very competitive (p-values with TNet-LF and TNet-AS are 0.066 and 0.053 respectively for Accuracy). Again, we could attribute this phenomenon to the ungrammatical user generated content of twitter, because the context-preserving component becomes less important for such data. TNet w/o context performs consistently better than TNet w/o transformation, which verifies the efficacy of the target specific transformation (TST), before applying context-preserving.

As for the position information, we conduct statistical t-test between TNet-LF/AS and TNet-LF/AS w/o position together with performance comparison. All of the produced p-values are less than 0.05, suggesting that the improvements brought in by position information are significant.

[7]Without specification, the significance level is set to 0.05.

3.4 CPT versus Alternatives

The next interesting question is what if we replace the transformation module (i.e., the CPT layers in Fig.1) of TNet with other commonly-used components? We investigate two alternatives: attention mechanism and fully-connected (FC) layer, resulting in three pipelines as shown in the second group of Table 3 (position relevance is kept for them).

LSTM-ATT-CNN applies attention as the alternative[8], and it does not need the context-preserving mechanism. It performs unexceptionally worse than the TNet variants. We are surprised that LSTM-ATT-CNN is even worse than TNet w/o transformation (a pipeline simply removing the transformation module) on TWITTER. More concretely, applying attention results in negative effect on TWITTER, which is consistent with the observation that all those attention-based state-of-the-art methods (i.e., TD-LSTM, MemNet, BILSTM-ATT-G, and RAM) cannot perform well on TWITTER.

LSTM-FC-CNN-LF and LSTM-FC-CNN-AS are built by applying FC layer to replace TST and keeping the context-preserving mechanism (i.e., LF and AS). Specifically, the concatenation of word representation and the averaged target vector is fed to the FC layer to obtain target-specific features. Note that LSTM-FC-CNN-LF/AS are equivalent to TNet-LF/AS when processing single-word targets (see Eq. 3). They obtain competitive results on all datasets: comparable with or better than the state-of-the-art methods. The TNet variants can still outperform LSTM-FC-CNN-LF/AS with significant gaps, e.g., on LAPTOP and REST, the accuracy gaps between TNet-LF and LSTM-FC-CNN-LF are 0.42% ($p < 0.03$) and 0.38% ($p < 0.04$) respectively.

3.5 Impact of CPT Layer Number

As our TNet involves multiple CPT layers, we investigate the effect of the layer number L. Specifically, we conduct experiments on the held-out training data of LAPTOP and vary L from 2 to 10, increased by 2. The cases L=1 and L=15 are also included. The results are illustrated in Figure 3. We can see that both TNet-LF and TNet-AS achieve the best results when L=2. While increasing L, the performance is basically becoming worse. For large L, the performance of TNet-AS

[8]We tried different attention mechanisms and report the best one here, namely, *dot* attention (Luong et al., 2015).

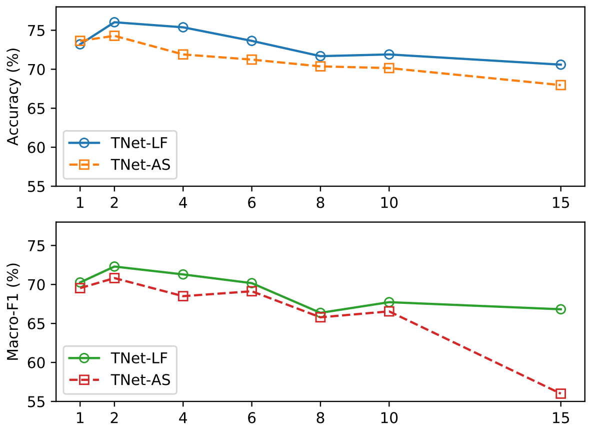


Figure 3: Effect of L.

generally becomes more sensitive, it is probably because AS involves extra parameters (see Eq 9) that increase the training difficulty.

3.6 Case Study

Table 4 shows some sample cases. The input targets are wrapped in the brackets with true labels given as subscripts. The notations P, N and O in the table represent positive, negative and neutral respectively. For each sentence, we underline the target with a particular color, and the text of its corresponding most informative n-gram feature[9] captured by TNet-AS (TNet-LF captures very similar features) is in the same color (so color printing is preferred). For example, for the target "resolution" in the first sentence, the captured feature is "Air has higher". Note that as discussed above, the CNN layer of TNet captures such features with the size-three kernels, so that the features are trigrams. Each of the last features of the second and seventh sentences contains a padding token, which is not shown.

Our TNet variants can predict target sentiment more accurately than RAM and BILSTM-ATT-G in the transitional sentences such as the first sentence by capturing correct trigram features. For the third sentence, its second and third most informative trigrams are "100% . PAD" and "' s not", being used together with "features make up", our models can make correct predictions. Moreover, TNet can still make correct prediction when the explicit opinion is target-specific. For example,

[9]For each convolutional filter, only one n-gram feature in the feature map will be kept after the max pooling. Among those from different filters, the n-gram with the highest frequency will be regarded as the most informative n-gram w.r.t. the given target.

Sentence	BILSTM-ATT-G	RAM	TNet-LF	TNet-AS
1. Air has higher **[resolution]**$_P$ but the **[fonts]**$_N$ are small .	(N$^{✗}$, N)	(N$^{✗}$, N)	(P, N)	(P, N)
2. Great **[food]**$_P$ but the **[service]**$_N$ is dreadful .	(P, N)	(P, N)	(P, N)	(P, N)
3. Sure it ' s not light and slim but the **[features]**$_P$ make up for it 100% .	N$^{✗}$	N$^{✗}$	P	P
4. Not only did they have amazing , **[sandwiches]**$_P$, **[soup]**$_P$, **[pizza]**$_P$ etc , but their **[homemade sorbets]**$_P$ are out of this world !	(P, O$^{✗}$, O$^{✗}$, P)	(P, P, O$^{✗}$, P)	(P, P, P, P)	(P, P, P, P)
5. **[startup times]**$_N$ are incredibly long : over two minutes .	P$^{✗}$	P$^{✗}$	N	N
6. I am pleased with the fast **[log on]**$_P$, speedy **[wifi connection]**$_P$ and the long **[battery life]**$_P$ (> 6 hrs) .	(P, P, P)	(P, P, P)	(P, P, P)	(P, P, P)
7. The **[staff]**$_N$ should be a bit more friendly .	P$^{✗}$	P$^{✗}$	P$^{✗}$	P$^{✗}$

Table 4: Example predictions, color printing is preferred. The input targets are wrapped in brackets with the true labels given as subscripts. ✗ indicates incorrect prediction.

"long" in the fifth sentence is negative for "startup time", while it could be positive for other targets such as "battery life" in the sixth sentence. The sentiment of target-specific opinion word is conditioned on the given target. Our TNet variants, armed with the word-level feature transformation w.r.t. the target, is capable of handling such case.

We also find that all these models cannot give correct prediction for the last sentence, a commonly used subjunctive style. In this case, the difficulty of prediction does not come from the detection of explicit opinion words but the inference based on implicit semantics, which is still quite challenging for neural network models.

4 Related Work

Apart from sentence level sentiment classification (Kim, 2014; Shi et al., 2018), aspect/target level sentiment classification is also an important research topic in the field of sentiment analysis. The early methods mostly adopted supervised learning approach with extensive hand-coded features (Blair-Goldensohn et al., 2008; Titov and McDonald, 2008; Yu et al., 2011; Jiang et al., 2011; Kiritchenko et al., 2014; Wagner et al., 2014; Vo and Zhang, 2015), and they fail to model the semantic relatedness between a target and its context which is critical for target sentiment analysis. Dong et al. (2014) incorporate the target information into the feature learning using dependency trees. As observed in previous works, the performance heavily relies on the quality of dependency parsing. Tang et al. (2016a) propose to split the context into two parts and associate target with contextual features separately. Similar to (Tang et al., 2016a), Zhang et al. (2016) develop a three-way gated neural network to model the interaction between the target and its surrounding contexts. Despite the advantages of jointly modeling target and context, they are not capable of capturing long-range information when some critical context information is far from the target. To overcome this limitation, researchers bring in the attention mechanism to model target-context association (Tang et al., 2016a,b; Wang et al., 2016; Yang et al., 2017; Liu and Zhang, 2017; Ma et al., 2017; Chen et al., 2017; Zhang et al., 2017; Tay et al., 2017). Compared with these methods, our TNet avoids using attention for feature extraction so as to alleviate the attended noise.

5 Conclusions

We re-examine the drawbacks of attention mechanism for target sentiment classification, and also investigate the obstacles that hinder CNN-based models to perform well for this task. Our TNet model is carefully designed to solve these issues. Specifically, we propose target specific transformation component to better integrate target information into the word representation. Moreover, we employ CNN as the feature extractor for this classification problem, and rely on the context-preserving and position relevance mechanisms to maintain the advantages of previous LSTM-based models. The performance of TNet consistently dominates previous state-of-the-art methods on different types of data. The ablation studies show the efficacy of its different modules, and thus verify the rationality of TNet's architecture.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*, pages 339–348.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of EMNLP*, pages 463–472.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 49–54.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *Proceedings of ECCV*, pages 630–645.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 151–160.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Proceedings of NIPS*, pages 919–927.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*, pages 2342–2350.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of SemEval*, pages 437–442.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of AAAI*, volume 333, pages 2267–2273.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of EMNLP*, pages 2876–2882.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of EACL*, pages 572–577.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of IJCAI*, pages 4068–4074.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of SemEval*, pages 27–35.

Bei Shi, Zihao Fu, Lidong Bing, and Wai Lam. 2018. Learning domain-sensitive and sentiment-aware word embeddings. In *Proceedings of ACL*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015a. Training very deep networks. In *Proceedings of NIPS*, pages 2377–2385.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015b. Highway networks. *arXiv preprint arXiv:1505.00387*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of EMNLP*, pages 214–224.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *arXiv preprint arXiv:1712.05403*.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*, pages 111–120.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of IJCAI*, pages 1347–1353.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of SemEval*, pages 223–229.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of EMNLP*, pages 606–615.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*, pages 2048–2057.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention based lstm for target dependent sentiment classification. In *Proceedings of AAAI*, pages 5013–5014.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. 2011. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proceedings of ACL*, pages 1496–1505.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proceedings of AAAI*, pages 3087–3093.

Ye Zhang and Byron Wallace. 2017. A sensitivity analysis of (and practitioners guide to) convolutional neural networks for sentence classification. In *Proceedings of IJCNLP*, pages 253–263.

Yue Zhang, Zhenghua Li, Jun Lang, Qingrong Xia, and Min Zhang. 2017. Dependency parsing with partial annotations: An empirical comparison. In *Proceedings of IJCNLP*, pages 49–58.

Target-Sensitive Memory Networks for Aspect Sentiment Classification

Shuai Wang[†], Sahisnu Mazumder[†], Bing Liu[†], Mianwei Zhou[‡], Yi Chang[§]
[†]Department of Computer Science, University of Illinois at Chicago, USA
[‡]Plus.AI, USA
[§]Artificial Intelligence School, Jilin University, China
shuaiwanghk@gmail.com, sahisnumazumder@gmail.com
liub@uic.edu, mianwei.zhou@gmail.com, yichang@acm.org

Abstract

Aspect sentiment classification (ASC) is a fundamental task in sentiment analysis. Given an aspect/target and a sentence, the task classifies the sentiment polarity expressed on the target in the sentence. Memory networks (MNs) have been used for this task recently and have achieved state-of-the-art results. In MNs, attention mechanism plays a crucial role in detecting the sentiment context for the given target. However, we found an important problem with the current MNs in performing the ASC task. Simply improving the attention mechanism will not solve it. The problem is referred to as *target-sensitive sentiment*, which means that the sentiment polarity of the (detected) context is dependent on the given target and it cannot be inferred from the context alone. To tackle this problem, we propose the *target-sensitive memory networks* (TMNs). Several alternative techniques are designed for the implementation of TMNs and their effectiveness is experimentally evaluated.

1 Introduction

Aspect sentiment classification (ASC) is a core problem of sentiment analysis (Liu, 2012). Given an aspect and a sentence containing the aspect, ASC classifies the sentiment polarity expressed in the sentence about the aspect, namely, positive, neutral, or negative. Aspects are also called *opinion targets* (or simply *targets*), which are usually product/service features in customer reviews. In this paper, we use aspect and target interchangeably. In practice, aspects can be specified by the user or extracted automatically using an aspect extraction technique (Liu, 2012). In this work, we assume the aspect terms are given and only focus on the classification task.

Due to their impressive results in many NLP tasks (Deng et al., 2014), neural networks have been applied to ASC (see the survey (Zhang et al., 2018)). Memory networks (MNs), a type of neural networks which were first proposed for question answering (Weston et al., 2015; Sukhbaatar et al., 2015), have achieved the state-of-the-art results in ASC (Tang et al., 2016). A key factor for their success is the attention mechanism. However, we found that using existing MNs to deal with ASC has an important problem and simply relying on attention modeling cannot solve it. That is, their performance degrades when the sentiment of a context word is sensitive to the given target.

Let us consider the following sentences:

(1) *The screen resolution is **excellent** but the price is **ridiculous**.*
(2) *The screen resolution is **excellent** but the price is **high**.*
(3) *The price is **high**.*
(4) *The screen resolution is **high**.*

In sentence (1), the sentiment expressed on aspect *screen resolution* (or *resolution* for short) is positive, whereas the sentiment on aspect *price* is negative. For the sake of predicting correct sentiment, a crucial step is to first detect the sentiment context about the given aspect/target. We call this step *targeted-context detection*. Memory networks (MNs) can deal with this step quite well because the sentiment context of a given aspect can be captured by the internal attention mechanism in MNs. Concretely, in sentence (1) the word "excellent" can be identified as the sentiment context when *resolution* is specified. Likewise, the context word "ridiculous" will be placed with a high attention when *price* is the target. With the correct targeted-context detected, a trained MN, which recognizes "excellent" as positive sentiment and "ridiculous" as negative sentiment, will infer correct sentiment polarity for the given target. This is relatively easy as "excellent" and "ridiculous" are both target-independent sentiment words, i.e., the words themselves already indicate clear sentiments.

As illustrated above, the attention mechanism addressing the targeted-context detection problem is very useful for ASC, and it helps classify many sentences like sentence (1) accurately. This also led to existing and potential research in improving attention modeling (discussed in Section 5). However, we observed that simply focusing on tackling the target-context detection problem and learning better attention are not sufficient to solve the problem found in sentences (2), (3) and (4).

Sentence (2) is similar to sentence (1) except that the (sentiment) context modifying aspect/target *price* is "high". In this case, when "high" is assigned the correct attention for the aspect *price*, the model also needs to capture the sentiment interaction between "high" and *price* in order to identify the correct sentiment polarity. This is not as easy as sentence (1) because "high" itself indicates no clear sentiment. Instead, its sentiment polarity is dependent on the given target.

Looking at sentences (3) and (4), we further see the importance of this problem and also why relying on attention mechanism alone is insufficient. In these two sentences, sentiment contexts are both "high" (i.e., same attention), but sentence (3) is negative and sentence (4) is positive simply because their target aspects are different. Therefore, focusing on improving attention will not help in these cases. We will give a theoretical insight about this problem with MNs in Section 3.

In this work, we aim to solve this problem. To distinguish it from the aforementioned targeted-context detection problem as shown by sentence (1), we refer to the problem in (2), (3) and (4) as the *target-sensitive sentiment* (or target-dependent sentiment) problem, which means that the sentiment polarity of a detected/attended context word is conditioned on the target and cannot be directly inferred from the context word alone, unlike "excellent" and "ridiculous". To address this problem, we propose *target-sensitive memory networks* (TMNs), which can capture the sentiment interaction between targets and contexts. We present several approaches to implementing TMNs and experimentally evaluate their effectiveness.

2 Memory Network for ASC

This section describes our basic memory network for ASC, also as a background knowledge. It does not include the proposed target-sensitive sentiment solutions, which are introduced in Section 4. The model design follows previous studies (Sukhbaatar et al., 2015; Tang et al., 2016) except that a different attention alignment function is used (shown in Eq. 1). Their original models will be compared in our experiments as well. The definitions of related notations are given in Table 1.

t	a target word, $t \in \mathbb{R}^{V \times 1}$
v_t	target embedding of t, $v_t \in \mathbb{R}^{d \times 1}$
x_i	a context word in a sentence, $x_i \in \mathbb{R}^{V \times 1}$
m_i, c_i	input, output context embedding of word x_i, and $m_i, c_i \in \mathbb{R}^{d \times 1}$
V	number of words in vocabulary
d	vector/embedding dimension
A	input embedding matrix $A \in \mathbb{R}^{d \times V}$
C	output embedding matrix $C \in \mathbb{R}^{d \times V}$
α	attention distribution in a sentence
α_i	attention of context word i, $\alpha_i \in (0, 1)$
o	output representation, $o \in \mathbb{R}^{d \times 1}$
K	number of sentiment classes
s	sentiment score, $s \in \mathbb{R}^{K \times 1}$
y	sentiment probability

Table 1: Definition of Notations

Input Representation: Given a target aspect t, an embedding matrix A is used to convert t into a vector representation, v_t ($v_t = At$). Similarly, each context word (non-aspect word in a sentence) $x_i \in \{x_1, x_2, ...x_n\}$ is also projected to the continuous space stored in memory, denoted by m_i ($m_i = Ax_i$) $\in \{m_1, m_2, ...m_n\}$. Here n is the number of words in a sentence and i is the word position/index. Both t and x_i are one-hot vectors. For an aspect expression with multiple words, its aspect representation v_t is the averaged vector of those words (Tang et al., 2016).

Attention: Attention can be obtained based on the above input representation. Specifically, an attention weight α_i for the context word x_i is computed based on the alignment function:

$$\alpha_i = softmax(v_t^T M m_i) \tag{1}$$

where $M \in \mathbb{R}^{d \times d}$ is the general learning matrix suggested by Luong et al. (2015). In this manner, attention $\alpha = \{\alpha_1, \alpha_2, ..\alpha_n\}$ is represented as a vector of probabilities, indicating the weight/importance of context words towards a given target. Note that $\alpha_i \in (0, 1)$ and $\sum_i \alpha_i = 1$.

Output Representation: Another embedding matrix C is used for generating the individual (output) continuous vector c_i ($c_i = Cx_i$) for each context word x_i. A final response/output vector o is produced by summing over these vectors weighted with the attention α, i.e., $o = \sum_i \alpha_i c_i$.

Sentiment Score (or Logit): The aspect sentiment scores (also called logits) for positive, neutral, and negative classes are then calculated, where a sentiment-specific weight matrix $W \in \mathbb{R}^{K\times d}$ is used. The sentiment scores are represented in a vector $s \in \mathbb{R}^{K\times 1}$, where K is the number of (sentiment) classes, which is 3 in ASC.

$$s = W(o + v_t) \tag{2}$$

The final sentiment probability y is produced with a $softmax$ operation, i.e., $y = softmax(s)$.

3 Problem of the above Model for Target-Sensitive Sentiment

This section analyzes the problem of target-sensitive sentiment in the above model. The analysis can be generalized to many existing MNs as long as their improvements are on attention α only. We first expand the sentiment score calculation from Eq. 2 to its individual terms:

$$\begin{aligned} s = W(o + v_t) &= W(\sum_i \alpha_i c_i + v_t) \\ &= \alpha_1 W c_1 + \alpha_2 W c_2 + ...\alpha_n W c_n + W v_t \end{aligned} \tag{3}$$

where "+" denotes element-wise summation. In Eq. 3, $\alpha_i W c_i$ can be viewed as the individual sentiment logit for a context word and Wv_t is the sentiment logit of an aspect. They are linearly combined to determine the final sentiment score s. This can be problematic in ASC. First, an aspect word often expresses no sentiment, for example, "screen". However, if the aspect term v_t is simply removed from Eq. 3, it also causes the problem that the model cannot handle target-dependent sentiment. For instance, the sentences (3) and (4) in Section 1 will then be treated as identical if their aspect words are not considered. Second, if an aspect word is considered and it directly bears some positive or negative sentiment, then when an aspect word occurs with different context words for expressing opposite sentiments, a contradiction can be resulted from them, especially in the case that the context word is a target-sensitive sentiment word. We explain it as follows.

Let us say we have two target words *price* and *resolution* (denoted as p and r). We also have two possible context words "high" and "low" (denoted as h and l). As these two sentiment words can modify both aspects, we can construct four snippets "high price", "low price", "high resolution" and "low resolution". Their sentiments are negative, positive, positive, and negative respectively. Let us set W to $\mathbb{R}^{1\times d}$ so that s becomes a 1-dimensional sentiment score indicator. $s > 0$ indicates a positive sentiment and $s < 0$ indicates a negative sentiment. Based on the above example snippets or phrases we have four corresponding inequalities: (a) $W(\alpha_h c_h + v_p) < 0$, (b) $W(\alpha_l c_l + v_p) > 0$, (c) $W(\alpha_h c_h + v_r) > 0$ and (d) $W(\alpha_l c_l + v_r) < 0$. We can drop all α terms here as they all equal to 1, i.e., they are the only context word in the snippets to attend to (the target words are not contexts). From (a) and (b) we can infer (e) $Wc_h < -Wv_p < Wc_l$. From (c) and (d) we can infer (f) $Wc_l < -Wv_r < Wc_h$. From (e) and (f) we have (g) $Wc_h < Wc_l < Wc_h$, which is a contradiction.

This contradiction means that MNs cannot learn a set of parameters W and C to correctly classify the above four snippets/sentences at the same time. This contradiction also generalizes to real-world sentences. That is, although real-world review sentences are usually longer and contain more words, since the attention mechanism makes MNs focus on the most important sentiment context (the context with high α_i scores), the problem is essentially the same. For example, in sentences (2) and (3) in Section 1, when *price* is targeted, the main attention will be placed on "high". For MNs, these situations are nearly the same as that for classifying the snippet "high price". We will also show real examples in the experiment section.

One may then ask whether improving attention can help address the problem, as α_i can affect the final results by adjusting the sentiment effect of the context word via $\alpha_i W c_i$. This is unlikely, if not impossible. First, notice that α_i is a scalar ranging in (0,1), which means it essentially assigns higher or lower weight to increase or decrease the sentiment effect of a context word. It cannot change the intrinsic sentiment orientation/polarity of the context, which is determined by Wc_i. For example, if Wc_i assigns the context word "high" a positive sentiment ($Wc_i > 0$), α_i will not make it negative (i.e., $\alpha_i W c_i < 0$ cannot be achieved by changing α_i). Second, other irrelevant/unimportant context words often carry no or little sentiment information, so increasing or decreasing their weights does not help. For example, in the sentence "the price is high", adjusting the weights of context words "the" and "is" will neither help solve the problem nor be intuitive to do so.

4 The Proposed Approaches

This section introduces six (6) alternative target-sensitive memory networks (TMNs), which all can deal with the target-sensitive sentiment problem. Each of them has its characteristics.

Non-linear Projection (NP): This is the first approach that utilizes a non-linear projection to capture the interplay between an aspect and its context. Instead of directly following the common linear combination as shown in Eq. 3, we use a non-linear projection (*tanh*) as the replacement to calculate the aspect-specific sentiment score.

$$s = W \cdot tanh(\sum_i \alpha_i c_i + v_t) \tag{4}$$

As shown in Eq. 4, by applying a non-linear projection over attention-weighted c_i and v_t, the context and aspect information are coupled in a way that the final sentiment score cannot be obtained by simply summing their individual contributions (compared with Eq. 3). This technique is also intuitive in neural networks. However, notice that by using the non-linear projection (or adding more sophisticated hidden layers) over them in this way, we sacrifice some interpretability. For example, we may have difficulty in tracking how each individual context word (c_i) affects the final sentiment score s, as all context and target representations are coupled. To avoid this, we can use the following five alternative techniques.

Contextual Non-linear Projection (CNP): Despite the fact that it also uses the non-linear projection, this approach incorporates the interplay between a context word and the given target into its (output) context representation. We thus name it Contextual Non-linear Projection (CNP).

$$s = W \sum_i \alpha_i \cdot tanh(c_i + v_t) \tag{5}$$

From Eq. 5, we can see that this approach can keep the linearity of attention-weighted context aggregation while taking into account the aspect information with non-linear projection, which works in a different way compared to NP. If we define $\tilde{c}_i = tanh(c_i + v_t)$, $\tilde{c}_i$ can be viewed as the target-aware context representation of context x_i and the final sentiment score is calculated based on the aggregation of such $\tilde{c}_i$. This could be a more reasonable way to carry the aspect information rather than simply summing the aspect representation (Eq. 3).

However, one potential disadvantage is that this setting uses the same set of vector representations (learned by embeddings C) for multiple purposes, i.e., to learn output (context) representations and to capture the interplay between contexts and aspects. This may degenerate its model performance when the computational layers in memory networks (called "hops") are deep, because too much information is required to be encoded in such cases and a sole set of vectors may fail to capture all of it.

To overcome this, we suggest the involvement of an additional new set of embeddings/vectors, which is exclusively designed for modeling the sentiment interaction between an aspect and its context. The key idea is to decouple different functioning components with different representations, but still make them work jointly. The following four techniques are based on this idea.

Interaction Term (IT): The third approach is to formulate explicit target-context sentiment interaction terms. Different from the targeted-context detection problem which is captured by attention (discussed in Section 1), here the ***target-context sentiment (TCS) interaction*** measures the sentiment-oriented interaction effect between targets and contexts, which we refer to as *TCS interaction* (or *sentiment interaction*) for short in the rest of this paper. Such sentiment interaction is captured by a new set of vectors, and we thus also call such vectors *TCS vectors*.

$$s = \sum_i \alpha_i (W_s c_i + w_I \langle d_i, d_t \rangle) \tag{6}$$

In Eq. 6, $W_s \in \mathbb{R}^{K \times d}$ and $w_I \in \mathbb{R}^{K \times 1}$ are used instead of W in Eq. 3. W_s models the direct sentiment effect from c_i while w_I works with d_i and d_t together for learning the TCS interaction. d_i and d_t are TCS vector representations of context x_i and aspect t, produced from a new embedding matrix D, i.e., $d_i = Dx_i, d_t = Dt$ ($D \in \mathbb{R}^{d \times V}$ and $d_i, d_t \in \mathbb{R}^{d \times 1}$).

Unlike input and output embeddings A and C, D is designed to capture the sentiment interaction. The vectors from D affect the final sentiment score through $w_I \langle d_i, d_t \rangle$, where w_I is a sentiment-specific vector and $\langle d_i, d_t \rangle \in \mathbb{R}$ denotes the dot product of the two TCS vectors d_i and d_t. Compared to the basic MNs, this model can better capture target-sensitive sentiment because the interactions between a context word h and different aspect words (say, p and r) can be different, i.e., $\langle d_h, d_p \rangle \neq \langle d_h, d_r \rangle$.

The key advantage is that now the sentiment effect is explicitly dependent on its target and context. For example, $\langle d_h, d_p \rangle$ can help shift the final sentiment to negative and $\langle d_h, d_r \rangle$ can help shift it to positive. Note that α is still needed to control the importance of different contexts. In this manner, targeted-context detection (attention) and TCS interaction are jointly modeled and work together for sentiment inference. The proposed techniques introduced below also follow this core idea but with different implementations or properties. We thus will not repeat similar discussions.

Coupled Interaction (CI): This proposed technique associates the TCS interaction with an additional set of context representation. This representation is for capturing the global correlation between context and different sentiment classes.

$$s = \sum_i \alpha_i (W_s c_i + W_I \langle d_i, d_t \rangle e_i) \qquad (7)$$

Specifically, e_i is another output representation for x_i, which is coupled with the sentiment interaction factor $\langle d_i, d_t \rangle$. For each context word x_i, e_i is generated as $e_i = E x_i$ where $E \in \mathbb{R}^{d \times V}$ is an embedding matrix. $\langle d_i, d_t \rangle$ and e_i function together as a target-sensitive context vector and are used to produce sentiment scores with W_I ($W_I \in \mathbb{R}^{K \times d}$).

Joint Coupled Interaction (JCI): A natural variant of the above model is to replace e_i with c_i, which means to learn a joint output representation. This can also reduce the number of learning parameters and simplify the CI model.

$$s = \sum_i \alpha_i (W_s c_i + W_I \langle d_i, d_t \rangle c_i) \qquad (8)$$

Joint Projected Interaction (JPI): This model also employs a unified output representation like JCI, but a context output vector c_i will be projected to two different continuous spaces before sentiment score calculation. To achieve the goal, two projection matrices W_1, W_2 and the non-linear projection function $tanh$ are used. The intuition is that, when we want to reduce the (embedding) parameters and still learn a joint representation, two different sentiment effects need to be separated in different vector spaces. The two sentiment effects are modeled as two terms:

$$\begin{aligned} s = & \sum_i \alpha_i W_J \tanh(W_1 c_i) \\ & + \sum_i \alpha_i W_J \langle d_i, d_t \rangle \tanh(W_2 c_i) \end{aligned} \qquad (9)$$

where the first term can be viewed as learning target-independent sentiment effect while the second term captures the TCS interaction. A joint sentiment-specific weight matrix $W_J (W_J \in \mathbb{R}^{K \times d})$ is used to control/balance the interplay between these two effects.

Discussions: (a) In IT, CI, JCI, and JPI, their first-order terms are still needed, because not in all cases sentiment inference needs TCS interaction. For some simple examples like "the battery is good", the context word "good" simply indicates clear sentiment, which can be captured by their first-order term. However, notice that the modeling of second-order terms offers additional help in both general and target-sensitive scenarios. (b) TCS interaction can be calculated by other modeling functions. We have tried several methods and found that using the dot product $\langle d_i, d_t \rangle$ or $d_i^T W d_t$ (with a projection matrix W) generally produces good results. (c) One may ask whether we can use fewer embeddings or just use one universal embedding to replace A, C and D (the definition of D can be found in the introduction of IT). We have investigated them as well. We found that merging A and C is basically workable. But merging D and A/C produces poor results because they essentially function with different purposes. While A and C handle targeted-context detection (attention), D captures the TCS interaction. (d) Except NP, we do not apply non-linear projection to the sentiment score layer. Although adding non-linear transformation to it may further improve model performance, the individual sentiment effect from each context will become untraceable, i.e., losing some interpretability. In order to show the effectiveness of learning TCS interaction and for analysis purpose, we do not use it in this work. But it can be flexibly added for specific tasks/analyses that do not require strong interpretability.

Loss function: The proposed models are all trained in an end-to-end manner by minimizing the cross entropy loss. Let us denote a sentence and a target aspect as x and t respectively. They appear together in a pair format (x, t) as input and all such pairs construct the dataset H. $g_{(x,t)}$ is a one-hot vector and $g^k_{(x,t)} \in \{0, 1\}$ denotes a gold sentiment label, i.e., whether (x, t) shows sentiment k. $y_{x,t}$ is the model-predicted sentiment distribution for (x, t). $y^k_{x,t}$ denotes its probability in class k. Based on them, the training loss is constructed as:

$$loss = - \sum_{(x,t) \in H} \sum_{k \in K} g^k_{(x,t)} \log y^k_{(x,t)} \tag{10}$$

5 Related Work

Aspect sentiment classification (ASC) (Hu and Liu, 2004), which is different from document or sentence level sentiment classification (Pang et al., 2002; Kim, 2014; Yang et al., 2016), has recently been tackled by neural networks with promising results (Dong et al., 2014; Nguyen and Shirai, 2015) (also see the survey (Zhang et al., 2018)). Later on, the seminal work of using attention mechanism for neural machine translation (Bahdanau et al., 2015) popularized the application of the attention mechanism in many NLP tasks (Hermann et al., 2015; Cho et al., 2015; Luong et al., 2015), including ASC.

Memory networks (MNs) (Weston et al., 2015; Sukhbaatar et al., 2015) are a type of neural models that involve such attention mechanisms (Bahdanau et al., 2015), and they can be applied to ASC. Tang et al. (2016) proposed an MN variant to ASC and achieved the state-of-the-art performance. Another common neural model using attention mechanism is the RNN/LSTM (Wang et al., 2016).

As discussed in Section 1, the attention mechanism is suitable for ASC because it effectively addresses the targeted-context detection problem. Along this direction, researchers have studied more sophisticated attentions to further help the ASC task (Chen et al., 2017; Ma et al., 2017; Liu and Zhang, 2017). Chen et al. (2017) proposed to use a recurrent attention mechanism. Ma et al. (2017) used multiple sets of attentions, one for modeling the attention of aspect words and one for modeling the attention of context words. Liu and Zhang (2017) also used multiple sets of attentions, one obtained from the left context and one obtained from the right context of a given target. Notice that our work does not lie in this direction. Our goal is to solve the target-sensitive sentiment and to capture the TCS interaction, which is a different problem. This direction is also finer-grained, and none of the above works addresses this problem. Certainly, both directions can improve the ASC task. We will also show in our experiments that our work can be integrated with an improved attention mechanism.

To the best of our knowledge, none of the existing studies addresses the target-sensitive sentiment problem in ASC under the purely data-driven and supervised learning setting. Other concepts like sentiment shifter (Polanyi and Zaenen, 2006) and sentiment composition (Moilanen and Pulman, 2007; Choi and Cardie, 2008; Socher et al., 2013) are also related, but they are not learned automatically and require rule/patterns or external resources (Liu, 2012). Note that our approaches do not rely on handcrafted patterns (Ding et al., 2008; Wu and Wen, 2010), manually compiled sentiment constraints and review ratings (Lu et al., 2011), or parse trees (Socher et al., 2013).

6 Experiments

We perform experiments on the datasets of SemEval Task 2014 (Pontiki et al., 2014), which contain online reviews from domain *Laptop* and *Restaurant*. In these datasets, aspect sentiment polarities are labeled. The training and test sets have also been provided. Full statistics of the datasets are given in Table 2.

Dataset	Positive		Neutral		Negative	
	Train	Test	Train	Test	Train	Test
Restaurant	2164	728	637	196	807	196
Laptop	994	341	464	169	870	128

Table 2: Statistics of Datasets

6.1 Candidate Models for Comparison

MN: The classic end-to-end memory network (Sukhbaatar et al., 2015).

AMN: A state-of-the-art memory network used for ASC (Tang et al., 2016). The main difference from MN is in its attention alignment function, which concatenates the distributed representations of the context and aspect, and uses an additional weight matrix for attention calculation, following the method introduced in (Bahdanau et al., 2015).

BL-MN: Our basic memory network presented in Section 2, which does not use the proposed techniques for capturing target-sensitive sentiments.

AE-LSTM: RNN/LSTM is another popular attention based neural model. Here we compare with a state-of-the-art attention-based LSTM for ASC, AE-LSTM (Wang et al., 2016).

ATAE-LSTM: Another attention-based LSTM for ASC reported in (Wang et al., 2016).

Target-sensitive Memory Networks (TMNs): The six proposed techniques, NP, CNP, IT, CI, JCI, and JPI give six target-sensitive memory networks.

Note that other non-neural network based models like SVM and neural models without attention mechanism like traditional LSTMs have been compared and reported with inferior performance in the ASC task (Dong et al., 2014; Tang et al., 2016; Wang et al., 2016), so they are excluded from comparisons here. Also, note that non-neural models like SVMs require feature engineering to manually encode aspect information, while this work aims to improve the aspect representation learning based approaches.

6.2 Evaluation Measure

Since we have a three-class classification task (positive, negative and neutral) and the classes are imbalanced as shown in Table 2, we use F1-score as our evaluation measure. We report both F1-Macro over all classes and all individual class-based F1 scores. As our problem requires fine-grained sentiment interaction, the class-based F1 provides more indicative information. In addition, we report the accuracy (same as F1-Micro), as it is used in previous studies. However, we suggest using F1-score because accuracy biases towards the majority class.

6.3 Training Details

We use the open-domain word embeddings[1] for the initialization of word vectors. We initialize other model parameters from a uniform distribution U(-0.05, 0.05). The dimension of the word embedding and the size of the hidden layers are 300. The learning rate is set to 0.01 and the dropout rate is set to 0.1. Stochastic gradient descent is used as our optimizer. The position encoding is also used (Tang et al., 2016). We also compare the memory networks in their multiple computational layers version (i.e., multiple hops) and the number of hops is set to 3 as used in the mentioned previous studies. We implemented all models in the TensorFlow environment using same input, embedding size, dropout rate, optimizer, etc. so as to test our hypotheses, i.e., to make sure the achieved improvements do not come from elsewhere. Meanwhile, we can also report all evaluation measures discussed above[2]. 10% of the training data is used as the development set. We report the best results for all models based on their F-1 Macro scores.

6.3.1 Result Analysis

The classification results are shown in Table 3. Note that the candidate models are all based on classic/standard attention mechanism, i.e., without sophisticated or multiple attentions involved. We compare the 1-hop and 3-hop memory networks as two different settings. The top three F1-Macro scores are marked in bold. Based on them, we have the following observations:

1. Comparing the 1-hop memory networks (first nine rows), we see significant performance gains achieved by CNP, CI, JCI, and JPI on both datasets, where each of them has $p < 0.01$ over the strongest baseline (BL-MN) from paired t-test using F1-Macro. IT also outperforms the other baselines while NP has similar performance to BL-MN. This indicates that TCS interaction is very useful, as BL-MN and NP do not model it.
2. In the 3-hop setting, TMNs achieve much better results on Restaurant. JCI, IT, and CI achieve the best scores, outperforming the strongest baseline AMN by 2.38%, 2.18%, and 2.03%. On Laptop, BL-MN and most TMNs (except CNP and JPI) perform similarly. However, BL-MN performs poorly on Restaurant (only better than two models) while TMNs show more stable performance.
3. Comparing all TMNs, we see that JCI works the best as it always obtains the top-three scores on two datasets and in two settings. CI and JPI also perform well in most cases. IT, NP, and CNP can achieve very good scores in some cases but are less stable. We also analyzed their potential issues in Section 4.
4. It is important to note that these improvements are quite large because in many cases sentiment interactions may not be necessary (like sentence (1) in Section 1). The overall good results obtained by TMNs demonstrate their capability of handling both general and target-sensitive sentiments, i.e., the proposed

[1]https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[2]Most related studies report accuracy only.

Restaurant						Laptop					
Model	**Macro**	**Neg.**	**Neu.**	**Pos.**	**Micro**	**Model**	**Macro**	**Neg.**	**Neu.**	**Pos.**	**Micro**
MN	58.91	57.07	36.81	82.86	71.52	MN	56.16	47.06	45.81	75.63	61.91
AMN	63.82	61.76	43.56	86.15	75.68	AMN	60.01	52.67	47.89	79.48	66.14
BL-MN	64.34	61.96	45.86	85.19	75.30	BL-MN	62.89	57.16	49.51	81.99	68.90
NP	64.62	64.89	43.21	85.78	75.93	NP	62.63	56.43	49.62	81.83	68.65
CNP	65.58	62.97	47.65	86.12	75.97	CNP	**64.38**	57.92	53.23	81.98	69.62
IT	65.37	65.22	44.44	86.46	76.98	IT	63.07	57.01	50.62	81.58	68.38
CI	**66.78**	65.49	48.32	86.51	76.96	CI	63.65	57.33	52.60	81.02	68.65
JCI	**66.21**	65.74	46.23	86.65	77.16	JCI	**64.19**	58.49	53.69	80.40	68.42
JPI	**66.58**	65.44	47.60	86.71	76.96	JPI	**64.53**	58.62	51.71	83.25	70.06
AE-LSTM	66.45	64.22	49.40	85.73	76.43	AE-LSTM	62.45	55.26	50.35	81.74	68.50
ATAE-LSTM	65.41	66.19	43.34	86.71	76.61	ATAE-LSTM	59.41	55.27	42.15	80.81	67.40
MN (hops)	62.68	60.35	44.57	83.11	72.86	MN (hops)	60.61	55.59	45.94	80.29	66.61
AMN (hops)	66.46	65.57	46.64	87.16	77.27	AMN (hops)	65.16	60.00	52.56	82.91	70.38
BL-MN (hops)	65.71	63.83	46.91	86.39	76.45	BL-MN (hops)	**67.11**	63.10	54.53	83.69	72.15
NP (hops)	65.98	64.18	47.86	85.90	75.73	NP (hops)	**67.79**	63.17	56.27	83.92	72.43
CNP (hops)	66.87	65.32	49.07	86.22	76.65	CNP (hops)	64.85	58.84	53.29	82.43	70.25
IT (hops)	**68.64**	67.11	51.47	87.33	78.55	IT (hops)	66.23	61.43	53.69	83.57	71.37
CI (hops)	**68.49**	64.83	53.03	87.60	78.69	CI (hops)	66.79	61.80	55.30	83.26	71.67
JCI (hops)	**68.84**	66.28	52.06	88.19	78.79	JCI (hops)	**67.23**	61.08	57.49	83.11	71.79
JPI (hops)	67.86	66.72	49.63	87.24	77.95	JPI (hops)	65.16	59.01	54.25	82.20	70.18

Table 3: Results of all models on two datasets. Top three F1-Macro scores are marked in bold. The first nine models are 1-hop memory networks. The last nine models are 3-hop memory networks.

techniques do not bring harm while capturing additional target-sensitive signals.

5. Micro-F1/accuracy is greatly affected by the majority class, as we can see the scores from Pos. and Micro are very consistent. TMNs, in fact, effectively improve the minority classes, which are reflected in Neg. and Neu., for example, JCI improves BL-MN by 3.78% in Neg. on Restaurant. This indicates their usefulness of capturing fine-grained sentiment signals. We will give qualitative examples in next section to show their modeling superiority for identifying target-sensitive sentiments.

Restaurant					
Model	**Macro**	**Neg.**	**Neu.**	**Pos.**	**Micro**
TRMN	69.00	68.66	50.66	87.70	78.86
RMN	67.48	66.48	49.11	86.85	77.14
Laptop					
Model	**Macro**	**Neg.**	**Neu.**	**Pos.**	**Micro**
TRMN	68.18	62.63	57.37	84.30	72.92
RMN	67.17	62.65	55.31	83.55	72.07

Table 4: Results with Recurrent Attention

Integration with Improved Attention: As discussed, the goal of this work is not for learning better attention but addressing the target-sensitive sentiment. In fact, solely improving attention does not solve our problem (see Sections 1 and 3). However, better attention can certainly help achieve an overall better performance for the ASC task, as it makes the targeted-context detection more accurate. Here we integrate our proposed technique JCI with a state-of-the-art sophisticated attention mechanism, namely, the recurrent attention framework, which involves multiple attentions learned iteratively (Kumar et al., 2016; Chen et al., 2017). We name our model with this integration as Target-sensitive Recurrent-attention Memory Network (TRMN) and the basic memory network with the recurrent attention as Recurrent-attention Memory Network (RMN). Their results are given in Table 4. TRMN achieves significant performance gain with $p < 0.05$ in paired t-test.

6.4 Effect of TCS Interaction for Identifying Target-Sensitive Sentiment

We now give some real examples to show the effectiveness of modeling TCS interaction for identifying target-sensitive sentiments, by comparing a regular MN and a TMN. Specifically, BL-MN and JPI are used. Other MNs/TMNs have similar performances to BL-MN/JPI qualitatively, so we do not list all of them here. For BL-MN and JPI, their sentiment scores of a single context word i are calculated by $\alpha_i W c_i$ (from Eq. 3) and $\alpha_i W_J tanh(W_1 c_i) + \alpha_i W_J \langle d_i, d_t \rangle tanh(W_2 c_i)$ (from Eq. 9), each of which results in a 3-dimensional vector.

Illustrative Examples: Table 5 shows two records in Laptop. In record 1, to identify the sentiment of target *price* in the presented sentence, the sentiment interaction between the context word "higher" and the target word *price* is the key. The

Record 1				Record 2			
Sentence	Price was higher when purchased on MAC..			**Sentence**	(MacBook) Air has higher resolution..		
Target	*Price*	**Sentiment**	*Negative*	**Target**	*Resolution*	**Sentiment**	*Positive*
Result	**Sentiment Logits on context "higher"**			**Result**	**Sentiment Logits on context "higher"**		
TMN	Negative	Neutral	Positive	**TMN**	Negative	Neutral	Positive
	0.2663 (Correct)	-0.2604	-0.0282		-0.4729	-0.3949	**0.9041 (Correct)**
MN	Negative	Neutral	Positive	**MN**	Negative	Neutral	Positive
	0.3641 (Correct)	-0.3275	-0.0750		**0.2562 (Wrong)**	-0.2305	- 0.0528

Table 5: Sample Records and Model Comparison between MN and TMN

specific sentiment scores of the word "higher" towards negative, neutral and positive classes in both models are reported. We can see both models accurately assign the highest sentiment scores to the negative class. We also observe that in MN the negative score (0.3641) in the 3-dimension vector $\{0.3641, -0.3275, -0.0750\}$ calculated by $\alpha_i W c_i$ is greater than neutral (-0.3275) and positive (-0.0750) scores. Notice that α_i is always positive (ranging in $(0, 1)$), so it can be inferred that the first value in vector Wc_i is greater than the other two values. Here c_i denotes the vector representation of "higher" so we use c_{higher} to highlight it and we have $\{Wc_{higher}\}^{Negative} > \{Wc_{higher}\}^{Neutral/Positive}$ as an inference.

In record 2, the target is *resolution* and its sentiment is positive in the presented sentence. Although we have the same context word "higher", different from record 1, it requires a positive sentiment interaction with the current target. Looking at the results, we see TMN assigns the highest sentiment score of word "higher" to positive class correctly, whereas MN assigns it to negative class. This error is expected if we consider the above inference $\{Wc_{higher}\}^{Negative} > \{Wc_{higher}\}^{Neutral/Positive}$ in MN. The cause of this unavoidable error is that Wc_i is not conditioned on the target. In contrast, $W_J\langle d_i, \cdot d_t\rangle tanh(W_2 c_i)$ can change the sentiment polarity with the aspect vector d_t encoded. Other TMNs also achieve it (like $W_I\langle d_i, d_t\rangle c_i$ in JCI).

One may notice that the aspect information (v_t) is actually also considered in the form of $\alpha_i W c_i + W v_t$ in MNs and wonder whether Wv_t may help address the problem given different v_t. Let us assume it helps, which means in the above example an MN makes $Wv_{resolution}$ favor the positive class and Wv_{price} favor the negative class. But then we will have trouble when the context word is "lower", where it requires $Wv_{resolution}$ to favor the negative class and Wv_{price} to favor the positive class. This contradiction reflects the theoretical problem discussed in Section 3.

Other Examples: We also found other interesting target-sensitive sentiment expressions like "*large bill*" and "*large portion*", "*small tip*" and "*small portion*" from Restaurant. Notice that TMNs can also improve the neutral sentiment (see Table 3). For instance, TMN generates a sentiment score vector of the context "over" for target aspect *price*: {**0.1373**, 0.0066, -0.1433} (negative) and for target aspect *dinner*: {0.0496, **0.0591**, -0.1128} (neutral) accurately. But MN produces both negative scores {**0.0069**, 0.0025, -0.0090} (negative) and {**0.0078**, 0.0028, -0.0102} (negative) for the two different targets. The latter one in MN is incorrect.

7 Conclusion and Future Work

In this paper, we first introduced the target-sensitive sentiment problem in ASC. After that, we discussed the basic memory network for ASC and analyzed the reason why it is incapable of capturing such sentiment from a theoretical perspective. We then presented six techniques to construct target-sensitive memory networks. Finally, we reported the experimental results quantitatively and qualitatively to show their effectiveness.

Since ASC is a fine-grained and complex task, there are many other directions that can be further explored, like handling sentiment negation, better embedding for multi-word phrase, analyzing sentiment composition, and learning better attention. We believe all these can help improve the ASC task. The work presented in this paper lies in the direction of addressing target-sensitive sentiment, and we have demonstrated the usefulness of capturing this signal. We believe that there will be more effective solutions coming in the near future.

Acknowledgments

This work was partially supported by National Science Foundation (NSF) under grant nos. IIS-1407927 and IIS-1650900, and by Huawei Technologies Co. Ltd with a research gift.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Empirical Methods in Natural Language Processing*, pages 452–461.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. In *IEEE Transactions on Multimedia*, pages 1875–1886. IEEE.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Empirical Methods in Natural Language Processing*, pages 793–801.

Li Deng, Dong Yu, et al. 2014. Deep learning: methods and applications. In *Foundations and Trends® in Signal Processing*, pages 197–387. Now Publishers, Inc.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *ACM International Conference on Web Search and Data Mining*, pages 231–240.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Annual Meeting of the Association for Computational Linguistics*, pages 49–54.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Empirical Methods in Natural Language Processing*, pages 1746–1751.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Bing Liu. 2012. Sentiment analysis and opinion mining. In *Synthesis lectures on human language technologies*. Morgan & Claypool Publishers.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *European Chapter of the Association for Computational Linguistics*, pages 572–577.

Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *ACM International Conference on World Wide Web*, pages 347–356.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*, pages 1412–1421.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *RANLP*, pages 378–382.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Empirical Methods in Natural Language Processing*, pages 2509–2514.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Empirical Methods in Natural Language Processing*, pages 79–86.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pages 1–10. Springer.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task4: Aspect based sentiment analysis. In *ProWorkshop on Semantic Evaluation (SemEval-2014)*. Association for Computational Linguistics.

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

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Empirical Methods in Natural Language Processing*, pages 214–224.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Empirical Methods in Natural Language Processing*, pages 606–615.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations*.

Yunfang Wu and Miaomiao Wen. 2010. Disambiguating dynamic sentiment ambiguous adjectives. In *International Conference on Computational Linguistics*, pages 1191–1199.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. In *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1253. Wiley Online Library.

Identifying Transferable Information Across Domains for Cross-domain Sentiment Classification

Raksha Sharma[1], Pushpak Bhattacharyya[1], Sandipan Dandapat[2] and Himanshu Sharad Bhatt[3]

[1]Department of Computer Science, Indian Institute of Technology Bombay
[2]Microsoft AI and Research, India
[3]American Express Big Data Labs, India
[1]{raksha,pb}@cse.iitb.ac.in
[2]sadandao@microsoft.com
[3]Himanshu.S.Bhatt@Aexp.Com

Abstract

Getting manually labeled data in each domain is always an expensive and a time consuming task. Cross-domain sentiment analysis has emerged as a demanding concept where a labeled source domain facilitates a sentiment classifier for an unlabeled target domain. However, polarity orientation (positive or negative) and the significance of a word to express an opinion often differ from one domain to another domain. Owing to these differences, cross-domain sentiment classification is still a challenging task. In this paper, we propose that words that do not change their polarity and significance represent the transferable (usable) information across domains for cross-domain sentiment classification. We present a novel approach based on χ^2 test and cosine-similarity between context vector of words to identify polarity preserving significant words across domains. Furthermore, we show that a weighted ensemble of the classifiers enhances the cross-domain classification performance.

1 Introduction

The choice of the words to express an opinion depends on the domain as users often use domain-specific words (Qiu et al., 2009; Sharma and Bhattacharyya, 2015). For example, *entertaining* and *boring* are frequently used in the movie domain to express an opinion; however, finding these words in the electronics domain is rare. Moreover, there are words which are likely to be used across domains in the same proportion, but may change their polarity orientation from one domain to another (Choi et al., 2009). For example, a word like *unpredictable* is positive in the movie domain (*unpredictable plot*), but negative in the automobile domain (*unpredictable steering*). Such a *polarity changing* word should be assigned positive orientation in the movie domain and negative orientation in the automobile domain.[1] Due to these differences across domains, a supervised algorithm trained on a labeled source domain, does not generalize well on an unlabeled target domain and the cross-domain performance degrades.

Generally, supervised learning algorithms have to be re-trained from scratch on every new domain using the manually annotated review corpus (Pang et al., 2002; Kanayama and Nasukawa, 2006; Pang and Lee, 2008; Esuli and Sebastiani, 2005; Breck et al., 2007; Li et al., 2009; Prabowo and Thelwall, 2009; Taboada et al., 2011; Cambria et al., 2013; Rosenthal et al., 2014). This is not practical as there are numerous domains and getting manually annotated data for every new domain is an expensive and time consuming task (Bhattacharyya, 2015). On the other hand, domain adaptation techniques work in contrast to traditional supervised techniques on the principle of transferring learned knowledge across domains (Blitzer et al., 2007; Pan et al., 2010; Bhatt et al., 2015). The existing transfer learning based domain adaptation algorithms for cross-domain classification have generally been proven useful in reducing the labeled data requirement, but they do not consider words like *unpredictable* that change polarity orientation across domains. Transfer (reuse) of changing polarity words affects the cross-domain performance negatively. Therefore, one cannot use transfer learning as the proverbial hammer, rather one needs to gauge what to transfer from the source domain to the target domain.

In this paper, we propose that the words which

[1]The word 'unpredictable' is a classic example of changing (inconsistent) polarity across domains (Turney, 2002; Fahrni and Klenner, 2008).

are equally significant with a consistent polarity across domains represent the usable information for cross-domain sentiment analysis. χ^2 is a popularly used and reliable statistical test to identify significance and polarity of a word in an annotated corpus (Oakes et al., 2001; Al-Harbi et al., 2008; Cheng and Zhulyn, 2012; Sharma and Bhattacharyya, 2013). However, for an unlabeled corpus no such statistical technique is applicable. Therefore, identification of words which are significant with a consistent polarity across domains is a non-trivial task. In this paper, we present a novel technique based on χ^2 test and cosine-similarity between context vector of words to identify Significant Consistent Polarity (SCP) words across domains.[2] The major contribution of this research is as follows.

1. *Extracting significant consistent polarity words across domains*: A technique which exploits cosine-similarity between context vector of words and χ^2 test is used to identify SCP words across labeled source and unlabeled target domains.

2. *An ensemble-based adaptation algorithm*: A classifier (C_s) trained on SCP words in the labeled source domain acts as a seed to initiate a classifier (C_t) on the target specific features. These classifiers are then combined in a weighted ensemble to further enhance the cross-domain classification performance.

Our results show that our approach gives a statistically significant improvement over Structured Correspondence Learning (SCL) (Bhatt et al., 2015) and common unigrams in identification of transferable words, which eventually facilitates a more accurate sentiment classifier in the target domain. The road-map for rest of the paper is as follows. Section 2 describes the related work. Section 3 describes the extraction of the SCP and the ensemble-based adaptation algorithm. Section 4 elaborates the dataset and the experimental protocol. Section 5 presents the results and section 6 reports the error analysis. Section 7 concludes the paper.[3]

[2]SCP words are words which are significant in both the domains with consistent polarity orientation.

[3]Majority of this work is done at Conduent Labs India till February 2016.

2 Related Work

The most significant efforts in the learning of transferable knowledge for cross-domain text classification are Structured Correspondence Learning (SCL) (Blitzer et al., 2007) and Structured Feature Alignment (SFA) (Pan et al., 2010). SCL aims to learn the co-occurrence between features from the two domains. It starts with learning pivot features that occur frequently in both the domains. It models correlation between pivots and all other features by training linear predictors to predict presence of pivot features in the unlabeled target domain data. SCL has shown significant improvement over a baseline (shift-unaware) model. SFA uses some domain-independent words as a bridge to construct a bipartite graph to model the co-occurrence relationship between domain-specific words and domain-independent words. Our approach also exploits the concept of co-occurrence (Pan et al., 2010), but we measure the co-occurrence in terms of similarity between context vector of words, unlike SCL and SFA, which literally look for the co-occurrence of words in the corpus. The use of context vector of words in place of words helps to overcome the data sparsity problem (Sharma et al., 2015).

Domain adaptation for sentiment classification has been explored by many researchers (Jiang and Zhai, 2007; Ji et al., 2011; Saha et al., 2011; Glorot et al., 2011; Xia et al., 2013; Zhou et al., 2014; Bhatt et al., 2015). Most of the works have focused on learning a shared low dimensional representation of features that can be generalized across different domains. However, none of the approaches explicitly analyses significance and polarity of words across domains. On the other hand, Glorot et al., (2011) proposed a deep learning approach which learns to extract a meaningful representation for each review in an unsupervised fashion. Zhou et al., (2014) also proposed a deep learning approach to learn a feature mapping between cross-domain heterogeneous features as well as a better feature representation for mapped data to reduce the bias issue caused by the cross-domain correspondences. Though deep learning based approaches perform reasonably good, they don't perform explicit identification and visualization of transferable features across domains unlike SFA and SCL, which output a set of words as transferable (reusable) features. Our approach explicitly determines the words which are equally significant with a consistent polarity across source and target domains. Our results show that the use of SCP words as features identified by our approach leads to a more accurate cross-domain sentiment classifier in the unlabeled target domain.

3 Approach: Cross-domain Sentiment Classification

The proposed approach identifies words which are equally significant for sentiment classification with a consistent polarity across source and target domains. These Significant Consistent Polarity (SCP) words make a set of transferable knowledge from the labeled source domain to the unlabeled target domain for cross-domain sentiment analysis. The algorithm further adapts to the unlabeled target domain by learning target domain specific features. The following sections elaborate SCP features extraction (3.1) and the ensemble-based cross-domain adaptation algorithm (3.2).

3.1 Extracting SCP Features

The words which are not significant for classification in the labeled source domain, do not transfer useful knowledge to the target domain through a supervised classifier trained in the source domain. Moreover, words that are significant in both the domains, but have different polarity orientation transfer the wrong information to the target domain through a supervised classifier trained in the labeled source domain, which also downgrade the cross-domain performance.

Our algorithm identifies the significance and the polarity of all the words individually in their respective domains. Then the words which are significant in both the domains with the consistent polarity orientation are used to initiate the cross-domain adaptation algorithm. The following sections elaborate how the significance and the polarity of the words are obtained in the labeled source and the unlabeled target domains.

3.1.1 Extracting Significant Words with the Polarity Orientation from the Labeled Source Domain

Since we have a polarity annotated dataset in the source domain, a statistical test like χ^2 test can be applied to find the significance of a word in the corpus for sentiment classification (Cheng and Zhulyn, 2012; Zheng et al., 2004). We have used *goodness of fit chi*^{2} *test* with equal number of reviews in positive and negative corpora. This test is generally used to determine whether sample data is consistent with a null hypothesis.[4] Here, the null hypothesis is that the word is equally used in the positive and the negative corpora. The χ^2 test is formulated as follows:

$$\chi^2(w) = ((c_p^w - \mu^w)^2 + (c_n^w - \mu^w)^2)/\mu^w \quad (1)$$

Where, c_p^w is the observed count of a word w in the positive documents and c_n^w is the observed count in the negative documents. μ^w represents an average of the word's count in the positive and the negative documents. Here, μ^w is the expected count or the value of the null-hypothesis. There is an inverse relation between χ^2 value and the p-value which is probability of the data given null hypothesis is true. In such a case where a word results in a p-value smaller than the critical p-value (0.05), we reject the null-hypothesis. Consequently, we assume that the word w belongs to a particular class (positive or negative) in the data, hence it is a significant word for classification (Sharma and Bhattacharyya, 2013).

Polarity of Words in the Labeled Source Domain: Chi-square test substantiates the statistically significant association of a word with a class label. Based on this association we assign a polarity orientation to a word in the domain. In other words, if a word is found significant by χ^2 test, then the exact class of the word is determined by comparing c_p^w and c_n^w. For instance, if c_p^w is higher than c_n^w, then the word is *positive*, else *negative*.

3.1.2 Extracting Significant Words with the Polarity Orientation from the Unlabeled Target Domain

Target domain data is unlabeled and hence, χ^2 test cannot be used to find significance of the words. However, to obtain SCP words across domains, we take advantage of the fact that we have to identify significance of only those words in the target domain which are already proven to be significant in the source domain. We presume that a word which is significant in the source domain as per χ^2 test and occurs with a frequency greater than a certain threshold (θ) in the target domain is significant in the target domain also.

$$count_t(significant_s(\mathrm{w})) > \theta \Rightarrow significant_t(\mathrm{w}) \quad (2)$$

[4]http://stattrek.com/chi-square-test/goodness-of-fit.aspx?Tutorial=AP.

Equation (2) formulates the significance test in the unlabeled target (t) domain. Here, function $significant_s$ assures the significance of the word w in the labeled source (s) domain and $count_t$ gives the normalized count of the w in t.[5] χ^2 test has one key assumption that the expected value of an observed variable should not be less than 5 to be significant. Considering this assumption as a base, we fix the value of θ as 10.[6]

Polarity of Words in the Unlabeled Target Domain: Generally, in a polar corpus, a positive word occurs more frequently in context of other positive words, while a negative word occurs in context of other negative words (Sharma et al., 2015).[7] Based on this hypothesis, we explore the contextual information of a word that is captured well by its context vector to assign polarity to words in the target domain (Rill et al., 2012; Rong, 2014). Mikolov et al., (2013) showed that similarity between context vector of words in vicinity such as 'go' and 'to' is higher compared to distant words or words that are not in the neighborhood of each other. Here, the observed concept is that if a word is positive, then its context vector learned from the polar review corpus will give higher cosine-similarity with a known positive polarity word in comparison to a known negative polarity word or vice versa. Therefore, based on the cosine-similarity scores we can assign the label of the known polarity word to the unknown polarity word. We term known polarity words as *Positive-pivot* and *Negative-pivot*.

Context Vector Generation: To compute context vector ($conVec$) of a word (w), we have used publicly available word2vec toolkit with the skip-gram model (Mikolov et al., 2013).[8] In this model, each word's Huffman code is used as an input to a log-linear classifier with a continuous projection layer and words within a given window are predicted (Faruqui et al., 2014). We construct a 100 dimensional vector for each candidate word from the unlabeled target domain data. The decision method given in Equation 3 defines the polarity assignment to the unknown polarity words of the target domain. If a word w gives a higher cosine-similarity with the PosPivot (Positive-pivot) than the NegPivot (Negative-pivot), the decision method assigns the positive polarity to the word w, else negative polarity to the word w.

$$\begin{aligned} &\text{If}(cosine(conVec(\text{w}), conVec(\text{PosPivot})) > \\ &\quad cosine(conVec(\text{w}), conVec(\text{NegPivot}))) \\ &\quad \Rightarrow \text{Positive} \\ &\text{If}(cosine(conVec(\text{w}), conVec(\text{PosPivot})) \\ &\quad < cosine(conVec(\text{w}), conVec(\text{NegPivot}))) \\ &\quad \Rightarrow \text{Negative} \end{aligned} \tag{3}$$

Pivot Selection Method: We empirically observed that a polar word which has the highest frequency in the corpus gives more coverage to estimate the polarity orientation of other words while using context vector. Essentially, the frequent occurrence of the word in the corpus allows it to be in context of other words frequently. Therefore a polar word having the *highest frequency* in the target domain is observed to be more accurate as *pivot* for identification of polarity of input words.[9] Table 1 shows the examples of a few words in the electronics domain whose polarity orientation is derived based on the similarity scores obtained with PosPivot and NegPivot words in the electronics domain.

Transferable Knowledge: The proposed algorithm uses the above mentioned techniques to identify the significance and the polarity of words in the labeled source data (cf. Section 3.1.1) and the unlabeled target data (cf. Section 3.1.2). The words which are found significant in both the domains with the same polarity orientation form a set of SCP features for cross-domain sentiment classification. The weights learned for the SCP features in the labeled source domain by the classification algorithm can be reused for sentiment classification in the unlabeled target domain as SCP features have consistent impacts in both the domains.

[5] Normalized count of w in t shows the proportion of occurrences of w in t.

[6] We tried with smaller values of $theta$ also, but they were not found as effective as $theta$ value of 10 for significant words identification.

[7] For example, 'excellent' will be used more often in positive reviews in comparison to negative reviews, hence, it would have more positive words in its context. Likewise, 'terrible' will be used more frequently in negative reviews in comparison to positive reviews, hence, it would have more negative words in its context.

[8] Available at: `https://radimrehurek.com/gensim/models/word2vec.html`

[9] To obtain the highest frequency based pivots, words in the target corpus (unlabeled) were ordered based on their frequency in the corpus, then a few top words were manually observed (by three human annotators) to pick out a positive word and a negative word. The positive and negative polarity of pivots were confirmed manually to get rid of random high frequency words (for example, *function words*). These highest frequency polar words were set as Positive-pivot and Negative-pivot.

Word	Great	Poor	Polarity
Noisy	0.03	**0.24**	Neg
Crap	0.04	**0.28**	Neg
Weak	0.05	**0.21**	Neg
Defective	0.21	**0.70**	Neg
Sturdy	**0.43**	0.04	Pos
Durable	**0.44**	0.00	Pos
Perfect	**0.48**	0.20	Pos
Handy	**0.60**	0.21	Pos

Table 1: Cosine-similarity scores with PosPivot (great) and NegPivot (poor), and inferred polarity orientation of the words.

Symbol	Description
s, t	Represent Source (s) and Target (t) respectively
l, u	Represent labeled and unlabeled respectively
D_s^l, D_t^u	Represent Dataset in s and t domains respectively
V_s, V_t	Vocabularies of words in the s and t respectively
r_s^i,r_t^i	i^{th} review in D_s^l and D_t^u respectively
$sigPol()$	Identifies significant words with their polarity
f	Set of features
SVM	Implemented classification algorithm
C_s	Classifier C_s is trained on D_s^l with SCP as features
R_t^n	Top-n reviews in t as per classification score by C_s
C_t	Classifier C_t is trained on R_n^t
$unigrams()$	Gives bag-of-words
W_s, W_t	Weights for C_s and C_t respectively
WSM	Weighted Sum Model

Table 2: Notations used in the paper

3.2 Ensemble-based Cross-domain Adaptation Algorithm

Apart from the transferable SCP words (Obtained in Section 3.1), each domain has specific discriminating words which can be discovered only from that domain data. The proposed cross-domain adaptation approach (Algorithm 1) attempts to learn such domain specific features from the target domain using a classifier trained on SCP words in the source domain. An ensemble of the classifiers trained on the SCP features (transferred from the source) and domain specific features (learned within the target) further enhances the cross-domain performance. Table 2 lists the notations used in the algorithm. The working of the cross-domain adaptation algorithm is as follows:

1. Identify SCP features from the labeled source and the unlabeled target domain data.

2. A SVM based classifier is trained on SCP words as features using labeled source domain data, named as C_s.

3. The classifier C_s is used to predict the labels for the unlabeled target domain instances D_t^u, and the confidently predicted instances of D_t^u form a set of pseudo labeled instances R_t^n.

4. A SVM based classifier is trained on the pseudo labeled target domain instances R_t^n, using unigrams in R_t^n as features to include the target specific words, this classifier is named as C_t .

5. Finally, a Weighted Sum Model (WSM) of C_s and C_t gives a classifier in the target domain.

The confidence in the prediction of D_t^u is measured in terms of the classification-score of the document, *i.e.*, the distance of the input document from the separating hyper-plane given by the SVM classifier (Hsu et al., 2003). The top n confidently predicted pseudo labeled instances (R_t^n) are used to train classifier C_t, where n depends on a threshold that is empirically set to $|\pm 0.2|$.[10] The classifier C_s trained on the SCP features (transferred knowledge) from the source domain and the classifier C_t trained on self-discovered target specific features from the pseudo labeled target domain instances bring in complementary information from the two domains. Therefore, combining C_s and C_t in a weighted ensemble (WSM) further enhances the cross-domain performance. Algorithm 1 gives the pseudo code of the proposed adaptation approach.

```
Input: $D_s^l = \{r_s^1, r_s^2, r_s^3, ....r_s^j\}$,
       $D_t^u = \{r_t^1, r_t^2, r_t^3, ....r_t^k\}$,
       $V_s = \{w_s^1, w_s^2, w_s^3, ....w_s^p\}$,
       $V_t = \{w_t^1, w_t^2, w_t^3, ....w_t^q\}$
Output: Sentiment Classifier in the Target Domain
1: SCP $= sigPol(D_s^l) \cap sigPol(D_t^u)$
2: $C_s =$ Train-SVM$(D_s^l)$, where $f =$ SCP
3: Predict Label: $C_s(D_t^u) \rightarrow D_t^l$
4: Select: $R_t^n \mid \forall r_t^i \in D_t^u$ , $C_s(r_t^i) > \phi$, where $i \in$
   $\{1, 2....k\}$ and $n <= k$
5: $C_t =$ Train-SVM$(R_t^n)$, where $f = \{unigrams(R_t^n)\}$
6: WSM $= (C_s * W_s + C_t * W_t)/(W_s + W_t)$
7: Sentiment Classifier in the Target Domain $=$ WSM
```

ALGORITHM 1: Building of target domain classifier from the source domain

Weighted Sum Model (WSM): The weighted ensemble of classifiers helps to overcome the er-

[10]Balamurali et al., (2013) have shown that 350 to 400 labeled documents are required to get a high accuracy classifier in a domain using supervised classification techniques, but beyond 400 labeled documents there is not much improvement in the classification accuracy. Hence, threshold on classification score is set such that it can give a sufficient number of documents for supervised classification. Threshold $|\pm 0.2|$ gives documents between 350 to 400.

rors produced by the individual classifier. The formulation of WSM is given in step-6 of the Algorithm 1. If C_s has wrongly predicted a document at boundary point and C_t has predicted the same document confidently, then weighted sum of C_s and C_t predicts the document correctly or vice versa. For example, a document is classified by C_s as negative (wrong prediction) with a classification-score of -0.07, while the same document is classified by C_t as positive (correct prediction) with a classification-score of 0.33, the WSM of C_s and C_t will classify the document as *positive* with a classification-score of 0.12 (Equation 4).

$$WSM = \frac{(-0.07 * 0.765 + 0.33 * 0.712)}{(0.765 + 0.712)} = 0.12 \tag{4}$$

Here 0.765 and 0.712 are the weights W_s and W_t to the classifiers C_s and C_t respectively.

Weights to the Classifiers in WSM: The weights W_s and W_t are the classification accuracies obtained by C_s and C_t respectively on the cross-validation data from the target domain. The weights W_s and W_t allow C_s and C_t to participate in the WSM in proportion of their accuracy on the cross-validation data. This restriction facilitates the domination of the classifier which is more accurate.

4 Dataset & Experimental Protocol

In this paper, we show comparison between SCP-based domain adaptation (our approach) and SCL-based domain adaptation approach proposed by Bhatt el al. (2015) using four domains, *viz.,* Electronics (E), Kitchen (K), Books (B), and DVD.[11] We use SVM algorithm with linear kernel (Tong and Koller, 2002) to train a classifier in all the mentioned classification systems in the paper. To implement SVM algorithm, we have used the publicly available Python based Scikit-learn package (Pedregosa et al., 2011).[12] Data in each domain is divided into three parts, *viz.,* train (60%), validation (20%) and test (20%). The SCP words are extracted from the training data. The weights W_S and W_t for the source and target classifiers are essentially accuracies obtained by C_s and C_t respectively on validation dataset from the *target domain*. We report the accuracy for all the systems on the test data. Table 3 shows the statistics of the dataset.

Domain	No. of Reviews	Avg. Length
Electronic (E)	2000	110 words
Kitchen (K)	2000	93 words
Books (B)	2000	173 words
DVD (D)	2000	197 words

Table 3: Dataset statistics

5 Results

In this paper, we compare our approach with Structured Correspondence Learning (SCL) and common unigrams. SCL is used by Bhatt et al., (2015) for identification of transferable information from the labeled source domain to the unlabeled target domain for cross-domain sentiment analysis. They showed that transferable features extracted by SCL provide a better cross-domain sentiment analysis system than the transferable features extracted by Structured Feature Alignment (Pan et al., 2010). The SCL-based sentiment classifier in the target domain proposed by Bhatt et. al., (2015) is state-of-the-art for cross-domain sentiment analysis. On the other hand, common unigrams of the source and target are the most visible transferable information.[13]

Gold standard SCP words: Chi-square test gives us significance and polarity of the word in the corpus by taking into account the polarity labels of the reviews. Application of chi-square test in both the domains, considering that the target domain is also labeled, gives us gold standard SCP words. There is no manual annotation involved.

F-score for SCP Words Identification Task: The set of SCP words represent the usable information across domains for cross-domain classification, hence we compare the F-score for the SCP words identification task obtained with our approach, SCL and common-unigrams in Figure 1. It demonstrates that our approach gives a huge improvement in the F-score over SCL and common unigrams for all the 12 pairs of the source and target domains. To measure the statistical significance of this improvement, we applied t-test on the F-score distribution obtained with our approach, SCL and common unigrams. t-test is a

[11]The same multi-domain dataset is used by Bhatt et al. (2015), available at: `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/index2.html`.

[12]Available at: `http://scikit-learn.org/stable/modules/svm.html`.

[13]Common unigrams is a set of unique words which appear in both the domains.

statistical significance test. It is used to determine whether two sets of data are significantly different or not.[14] Our approach performs significantly better than SCL and common unigrams, while SCL performs better than common unigrams as per t-test.

Comparison among C_s, C_t and WSM: Table 4 shows the comparison among classifiers obtained in the target domain using SCP given by our approach, SCL, common-unigrams, and gold standard SCP for electronics as the source and movie as the target domains. Since electronics and movie are two very dissimilar domains in terms of domain specific words, unlike, books and movie, getting a high accuracy classifier in the movie domain from the electronics domain is a challenging task (Pang et al., 2002). Therefore, in Table 4 results are reported with electronics as the source domain and movie as the target domain.[15] In all four cases, there is difference in the transferred information from the source to the target, but the ensemble-based classification algorithm (Section 3.2) is the same. Table 4 depicts sentiment classification accuracy obtained with C_s, C_t and WSM. The weights W_s and W_t in WSM are normalized accuracies by C_s and C_t respectively on the validation set from the target domain. The fourth column (size) represents the feature set size. We observed that WSM gives the highest accuracy, which validates our assumption that a weighted sum of two classifiers is better than the performance of individual classifiers. The WSM accuracy obtained with SCP words given by our approach is comparable to the accuracy obtained with gold standard SCP words.

The motivation of this research is to learn shared representation cognizant of significant and polarity changing words across domains. Hence, we report cross-domain classification accuracy obtained with three different types of shared representations (transferable knowledge), *viz.*, common-unigrams, SCL and our approach.[16] System-1, system-2 and system-3 in Table 5 show the final cross-domain sentiment classification accuracy obtained with WSM in the target domain for 12 pairs of source and target using common-unigrams, SCL and our approach respectively.

	W_s & W_t	**Features**	**Size**	**Acc**
C_s		SCP (Our approach)	296	75.0
C_t		Unigrams	4751	74.3
WSM	0.72 & 0.69	-	-	**77.5**
C_s		SCL	1000	66.8
C_t		Unigrams	4615	68.0
WSM	0.63 & 0.61	-	-	**69.3**
C_s		Common-Unigrams	2236	64.0
C_t		Unigrams	4236	64.0
WSM	0.62 & 0.58	-	-	**65**
C_s		SCP (Gold standard)	163	77.0
C_t		Unigrams	1183	78.5
WSM	0.73 & 0.75	-	-	**80.0**

Table 4: Classification accuracy in % given by C_s, C_t and WSM with different feature sets for electronics as source and movie as target.

System-1: This system considers common-unigrams of both the domains as shared representation. **System-2:** It differs from system-1 in the shared representation, which is learned using Structured Correspondence Learning (SCL) (Bhatt et al., 2015) to initiate the process. **System-3:** This system implements the proposed domain adaptation algorithm. Here, the shared representation is the *SCP words* and the ensemble-based domain adaptation algorithm (Section 3.2) gives the final classifier in the target domain. Table 5 depicts that the system-3 is better than system-1 and system-2 for all pairs, except K to B and B to D. For these two pairs, system-2 performs better than system-3, though the difference in accuracy is very low (below 1%).

To enhance the final accuracy in the target domain, Bhatt et al., (2015) performed iterations over the pseudo labeled target domain instances (R_t^n). In each iteration, they obtained a new C_t trained on increased number of pseudo labeled documents. This process is repeated till all the training instances of the target domain are considered. The C_t obtained in the last iteration makes WSM with C_s which is trained on the transferable features given by SCL. Bhatt et al., (2015) have shown that iteration-based domain adaptation technique is more effective than one-shot

[14] The detail about the test is available at: `http://www.socialresearchmethods.net/kb/stat_t.php`.

[15] The movie review dataset is a balanced corpus of 2000 reviews. Available at: `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

[16] The reported accuracy is the ratio of correctly predicted documents to that of the total number of documents in the test dataset.

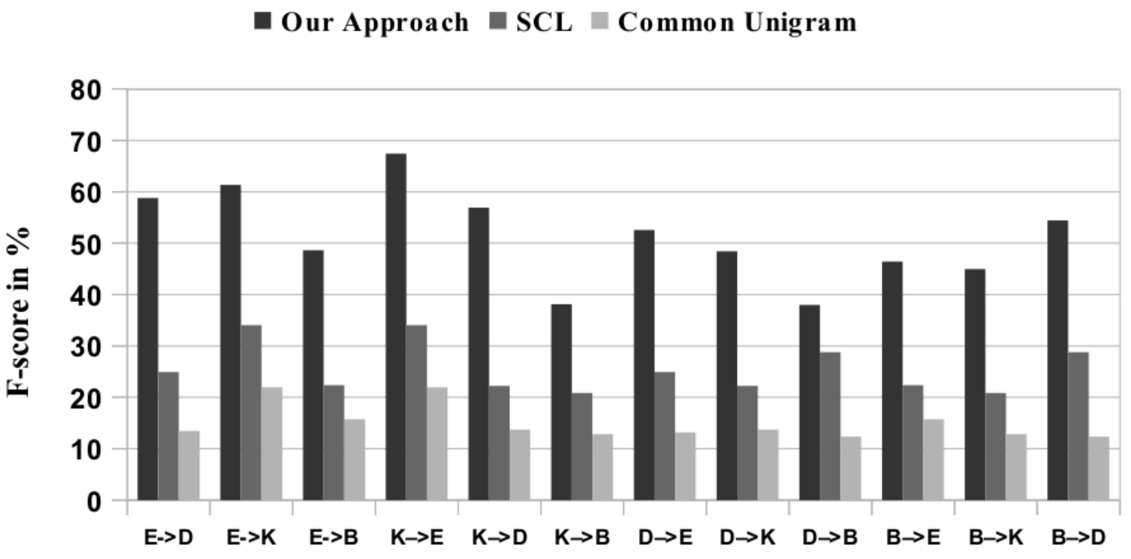


Figure 1: F-score for SCP words identification task (Source → Target) with respect to gold standard SCP words.

	System-1	System-2	System-3	System-4	System-5	System-6
D→B	62	64.2	**67**	66	76.5	**78.5**
E→B	63	58.9	**68.3**	67	75.6	**76.3**
K→B	67	**68.75**	67.85	69	71.2	**74**
B→D	76	**81**	80.5	77	81.5	**81.5**
E→D	68	71	**77.5**	71.5	74	**80.5**
K→D	69	69	**74**	71	75.2	**77**
B→E	68	66	**73**	69	79	**81.2**
D→E	61	62	**74**	66	73.2	**74.2**
K→E	76	75.75	**80**	78	81	**82**
B→K	66	67.5	**72**	69	79.2	**80.5**
D→K	65.75	67	**71**	66	80	**81**
E→K	74.25	75	**85.75**	76	84	**85.75**

Table 5: Cross-domain sentiment classification accuracy in the target domain (Source (S) → Target (T)).

Domain	Significant Words	Unigrams
Books (B)	76	89
DVD (D)	82.5	84
Electronics (E)	82.5	85
Kitchen (K)	82.5	86

Table 6: In-domain sentiment classification accuracy using significant words and unigrams.

adaptation approaches. **System-4, system-5**, and **system-6** in Table 5 incorporate the iterative process into **system-1, system-2,** and **system-3** respectively. We observed the same trend after the inclusion of the iterative process also, as the SCP-based system-6 performed the best in all 12 cases. On the other hand, SCL-based system-5 performs better than the common-unigrams based system-4. Table 7 shows the results of significance test (t-test) performed on the accuracy distributions produced by the six different systems. The noticeable point is that the iterations over SCL (system-5) and our approach (system-6) narrow down the difference in the accuracy between system-2 and system-3 as system-2 and system-3 have a statistically significant difference in accuracy with the p-value of 0.039 (Row-4 of Table 7), but the difference between system-5 and system-6 is not statistically significant. Essentially, system-3 does not give much improvement with iterations, unlike system-2. In other words, addition of the iterative process with the shared representation given by SCL overcomes the errors introduced by SCL. On the other hand, SCP given by our approach were able to produce a less erroneous system in one-shot. Table 6 shows the in-domain sentiment classification accuracy obtained with unigrams and significant words as features considering labeled data in the domain. System-6 tries to equalize the in-domain accuracy obtained with unigrams.

	P-value	Significant?
sys1 *vs* sys2	0.719	No
sys1 *vs* sys3	0.011	**Yes**
sys2 *vs* sys3	0.039	**Yes**
sys1 *vs* sys4	0.219	No
sys2 *vs* sys4	0.467	No
sys3 *vs* sys4	0.090	No
sys1 *vs* sys5	0	**Yes**
sys2 *vs* sys5	0	**Yes**
sys3 *vs* sys5	0.101	No
sys4 *vs* sys5	0	**Yes**
sys1 *vs* sys6	0	**Yes**
sys2 *vs* sys6	0	**Yes**
sys3 *vs* sys6	0.0130	**Yes**
sys4 *vs* sys6	0	**Yes**
sys5 *vs* sys6	0.231	No

Table 7: t-test (α = 0.05) results on the difference in accuracy produced by various systems (cf. Table 5).

To validate our assertion that polarity preserving significant words (SCP) across source and target domains make a less erroneous set of transferable knowledge from the source domain to the target domain, we computed Pearson product-moment correlation between F-score obtained for our approach (cf. Figure 1) and cross-domain accuracy obtained with SCP (System-3, cf. Table 5). We observed a *strong positive correlation* (r) of 0.78 between F-score and cross-domain accuracy. Essentially, an accurate set of SCP words positively stimulates an improved classifier in the unlabeled target domain.

6 Error Analysis

The pairs of domains which share a greater number of domain-specific words, result in a higher accuracy cross-domain classifier. For example, Electronics (E) and Kitchen (K) domains share many domain-specific words, hence pairing of such similar domains as the source and the target results into a higher accuracy classifier in the target domain. Table 5 shows that K→E outperforms B→E and D→E, and E→K outperforms B→K and D→K. On the other hand, DVD (D) and electronics are two very different domains unlike electronics and Kitchen, or DVD and books. The DVD dataset contains reviews about the music albums. This difference in types of reviews makes them to share less number of words. Table 8 shows the percent (%) of common words among the 4 domains. The percent of common unique words are common unique words divided by the summation of unique words in the domains individually.

E - D	E - K	E - B	D - K	D - B	K - B
15	22	17	14	22	17

Table 8: Common unique words between the domains in percent (%).

7 Conclusion

In this paper, we proposed that the Significant Consistent Polarity (SCP) words represent the transferable information from the labeled source domain to the unlabeled target domain for cross-domain sentiment classification. We showed a strong *positive correlation of* 0.78 *between the SCP words identified by our approach and the sentiment classification accuracy achieved in the unlabeled target domain*. Essentially, a set of less erroneous transferable features leads to a more accurate classification system in the unlabeled target domain. We have presented a technique based on χ^2 test and cosine-similarity between context vector of words to identify SCP words. Results show that the SCP words given by our approach represent more accurate transferable information in comparison to the Structured Correspondence Learning (SCL) algorithm and common-unigrams. Furthermore, we show that an ensemble of the classifiers trained on the SCP features and target specific features overcomes the errors of the individual classifiers.

References

S Al-Harbi, A Almuhareb, A Al-Thubaity, MS Khorsheed, and A Al-Rajeh. 2008. Automatic arabic text classification.

AR Balamurali, Mitesh M Khapra, and Pushpak Bhattacharyya. 2013. Lost in translation: viability of machine translation for cross language sentiment analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 38–49. Springer.

Himanshu S. Bhatt, Deepali Semwal, and S. Roy. 2015. An iterative similarity based adaptation technique for cross-domain text classification. In *Proceedings of Conference on Natural Language Learning*, pages 52–61.

Pushpak Bhattacharyya. 2015. *Multilingual Projections*. Springer International Publishing, Cham.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of Association for Computational Linguistics*, pages 440–447.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2683–2688.

Erik Cambria, Bjorn Schuller, Yunqing Xia, and Catherine Havasi. 2013. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, (2):15–21.

Alex Cheng and Oles Zhulyn. 2012. A system for multilingual sentiment learning on large data sets. In *Proceedings of International Conference on Computational Linguistics*, pages 577–592.

Yoonjung Choi, Youngho Kim, and Sung-Hyon Myaeng. 2009. Domain-specific sentiment analysis using contextual feature generation. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 37–44. ACM.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *Proceedings of International Conference on Information and Knowledge Management*, pages 617–624.

Angela Fahrni and Manfred Klenner. 2008. Old wine or warm beer: Target-specific sentiment analysis of adjectives. In *Proc. of the Symposium on Affective Language in Human and Machine, AISB*, pages 60–63.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2003. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Yang-Sheng Ji, Jia-Jun Chen, Gang Niu, Lin Shang, and Xin-Yu Dai. 2011. Transfer learning via multi-view principal component analysis. *Journal of Computer Science and Technology*, 26(1):81–98.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of Association for Computational Linguistics*, pages 264–271.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 355–363.

Tao Li, Yi Zhang, and Vikas Sindhwani. 2009. A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In *Proceedings of International Joint Conference on Natural Language Processing*, pages 244–252.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*.

Michael Oakes, Robert Gaaizauskas, Helene Fowkes, Anna Jonsson, Vincent Wan, and Micheline Beaulieu. 2001. A method based on the chi-square test for document classification. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 440–441. ACM.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of International Conference on World Wide Web*, pages 751–760.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 79–86.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Rudy Prabowo and Mike Thelwall. 2009. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *IJCAI*, volume 9, pages 1199–1204.

Sven Rill, Jörg Scheidt, Johannes Drescher, Oliver Schütz, Dirk Reinel, and Florian Wogenstein. 2012. A generic approach to generate opinion lists of phrases for opinion mining applications. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 7. ACM.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Sara Rosenthal, Preslav Nakov, Alan Ritter, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of SemEval*, pages 73–80.

Avishek Saha, Piyush Rai, Hal Daumé III, Suresh Venkatasubramanian, and Scott L DuVall. 2011. Active supervised domain adaptation. In *Machine Learning and Knowledge Discovery in Databases*, pages 97–112.

Raksha Sharma and Pushpak Bhattacharyya. 2013. Detecting domain dedicated polar words. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 661–666.

Raksha Sharma and Pushpak Bhattacharyya. 2015. Domain sentiment matters: A two stage sentiment analyzer. In *Proceedings of the International Conference on Natural Language Processing*.

Raksha Sharma, Mohit Gupta, Astha Agarwal, and Pushpak Bhattacharyya. 2015. Adjective intensity and sentiment analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

Peter D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Association for Computational Linguistics*, pages 417–424.

Rui Xia, Chengqing Zong, Xuelei Hu, and Erik Cambria. 2013. Feature ensemble plus sample selection: domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18.

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. 2004. Feature selection for text categorization on imbalanced data. *ACM Sig KDD Explorations Newsletter*, 6(1):80–89.

Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. 2014. Hybrid heterogeneous transfer learning through deep learning. In *AAAI*, pages 2213–2220.

Unpaired Sentiment-to-Sentiment Translation: A Cycled Reinforcement Learning Approach

Jingjing Xu[1]*, **Xu Sun**[1]*, **Qi Zeng**[1], **Xuancheng Ren**[1],
Xiaodong Zhang[1], **Houfeng Wang**[1], **Wenjie Li**[2]
[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]Department of Computing, Hong Kong Polytechnic University
{jingjingxu,xusun,pkuzengqi,renxc,zxdcs,wanghf}@pku.edu.cn
cswjli@comp.polyu.edu.hk

Abstract

The goal of sentiment-to-sentiment "translation" is to change the underlying sentiment of a sentence while keeping its content. The main challenge is the lack of parallel data. To solve this problem, we propose a cycled reinforcement learning method that enables training on unpaired data by collaboration between a neutralization module and an emotionalization module. We evaluate our approach on two review datasets, Yelp and Amazon. Experimental results show that our approach significantly outperforms the state-of-the-art systems. Especially, the proposed method substantially improves the content preservation performance. The BLEU score is improved from 1.64 to 22.46 and from 0.56 to 14.06 on the two datasets, respectively.[1]

1 Introduction

Sentiment-to-sentiment "translation" requires the system to change the underlying sentiment of a sentence while preserving its non-emotional semantic content as much as possible. It can be regarded as a special style transfer task that is important in Natural Language Processing (NLP) (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018). It has broad applications, including review sentiment transformation, news rewriting, etc. Yet the lack of parallel training data poses a great obstacle to a satisfactory performance.

Recently, several related studies for language style transfer (Hu et al., 2017; Shen et al., 2017) have been proposed. However, when applied to the sentiment-to-sentiment "translation" task, most existing studies only change the underlying sentiment and fail in keeping the semantic content. For example, given "The food is delicious" as the source input, the model generates "What a bad movie" as the output. Although the sentiment is successfully transformed from positive to negative, the output text focuses on a different topic. The reason is that these methods attempt to implicitly separate the emotional information from the semantic information in the same dense hidden vector, where all information is mixed together in an uninterpretable way. Due to the lack of supervised parallel data, it is hard to only modify the underlying sentiment without any loss of the non-emotional semantic information.

To tackle the problem of lacking parallel data, we propose a cycled reinforcement learning approach that contains two parts: a neutralization module and an emotionalization module. The neutralization module is responsible for extracting non-emotional semantic information by explicitly filtering out emotional words. The advantage is that only emotional words are removed, which does not affect the preservation of non-emotional words. The emotionalization module is responsible for adding sentiment to the neutralized semantic content for sentiment-to-sentiment translation.

In cycled training, given an emotional sentence with sentiment s, we first neutralize it to the non-emotional semantic content, and then force the emotionalization module to reconstruct the original sentence by adding the sentiment s. Therefore, the emotionalization module is taught to add sentiment to the semantic context in a supervised way. By adding opposite sentiment, we can achieve the goal of sentiment-to-sentiment translation. Because of the discrete choice of neutral words, the gradient is no longer differentiable over the neutralization module. Thus, we use policy gradient,

*Equal Contribution.

[1]The released code can be found in https://github.com/lancopku/unpaired-sentiment-translation

one of the reinforcement learning methods, to reward the output of the neutralization module based on the feedback from the emotionalization module. We add different sentiment to the semantic content and use the quality of the generated text as reward. The quality is evaluated by two useful metrics: one for identifying whether the generated text matches the target sentiment; one for evaluating the content preservation performance. The reward guides the neutralization module to better identify non-emotional words. In return, the improved neutralization module further enhances the emotionalization module.

Our contributions are concluded as follows:

- For sentiment-to-sentiment translation, we propose a cycled reinforcement learning approach. It enables training with unpaired data, in which only reviews and sentiment labels are available.
- Our approach tackles the bottleneck of keeping semantic information by explicitly separating sentiment information from semantic content.
- Experimental results show that our approach significantly outperforms the state-of-the-art systems, especially in content preservation.

2 Related Work

Style transfer in computer vision has been studied (Johnson et al., 2016; Gatys et al., 2016; Liao et al., 2017; Li et al., 2017; Zhu et al., 2017). The main idea is to learn the mapping between two image domains by capturing shared representations or correspondences of higher-level structures.

There have been some studies on unpaired language style transfer recently. Hu et al. (2017) propose a new neural generative model that combines variational auto-encoders (VAEs) and holistic attribute discriminators for effective imposition of style semantic structures. Fu et al. (2018) propose to use an adversarial network to make sure that the input content does not have style information. Shen et al. (2017) focus on separating the underlying content from style information. They learn an encoder that maps the original sentence to style-independent content and a style-dependent decoder for rendering. However, their evaluations only consider the transferred style accuracy. We argue that content preservation is also an indispensable evaluation metric. However, when applied to the sentiment-to-sentiment translation task, the previously mentioned models share the same problem. They have the poor preservation of non-emotional semantic content.

In this paper, we propose a cycled reinforcement learning method to improve sentiment-to-sentiment translation in the absence of parallel data. The key idea is to build supervised training pairs by reconstructing the original sentence. A related study is "back reconstruction" in machine translation (He et al., 2016; Tu et al., 2017). They couple two inverse tasks: one is for translating a sentence in language A to a sentence in language B; the other is for translating a sentence in language B to a sentence in language A. Different from the previous work, we do not introduce the inverse task, but use collaboration between the neutralization module and the emotionalization module.

Sentiment analysis is also related to our work (Socher et al., 2011; Pontiki et al., 2015; Rosenthal et al., 2017; Chen et al., 2017; Ma et al., 2017, 2018b). The task usually involves detecting whether a piece of text expresses positive, negative, or neutral sentiment. The sentiment can be general or about a specific topic.

3 Cycled Reinforcement Learning for Unpaired Sentiment-to-Sentiment Translation

In this section, we introduce our proposed method. An overview is presented in Section 3.1. The details of the neutralization module and the emotionalization module are shown in Section 3.2 and Section 3.3. The cycled reinforcement learning mechanism is introduced in Section 3.4.

3.1 Overview

The proposed approach contains two modules: a neutralization module and an emotionalization module, as shown in Figure 1. The neutralization module first extracts non-emotional semantic content, and then the emotionalization module attaches sentiment to the semantic content. Two modules are trained by the proposed cycled reinforcement learning method. The proposed method requires the two modules to have initial learning ability. Therefore, we propose a novel pre-training method, which uses a self-attention based sentiment classifier (SASC). A sketch of cycled reinforcement learning is shown in Algorithm 1. The

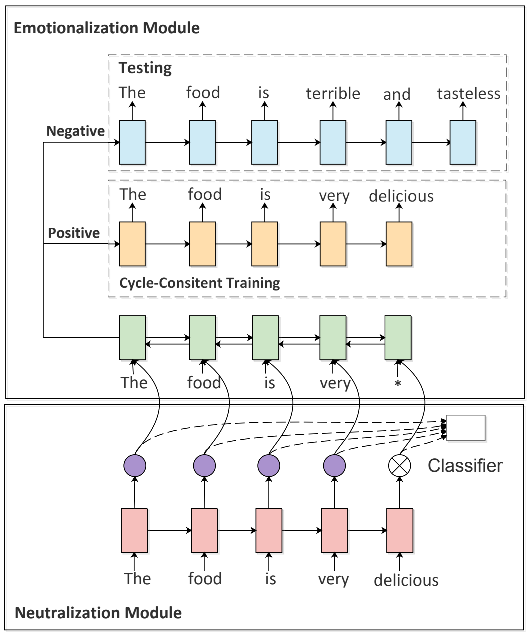


Figure 1: An illustration of the two modules. Lower: The neutralization module removes emotional words and extracts non-emotional semantic information. Upper: The emotionalization module adds sentiment to the semantic content. The proposed self-attention based sentiment classifier is used to guide the pre-training.

details are introduced as follows.

3.2 Neutralization Module

The neutralization module N_θ is used for explicitly filtering out emotional information. In this paper, we consider this process as an extraction problem. The neutralization module first identifies non-emotional words and then feeds them into the emotionalization module. We use a single Long-short Term Memory Network (LSTM) to generate the probability of being neutral or being polar for every word in a sentence. Given an emotional input sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$ of T words from Γ, the vocabulary of words, this module is responsible for producing a neutralized sequence.

Since cycled reinforcement learning requires the modules with initial learning ability, we propose a novel pre-training method to teach the neutralization module to identify non-emotional words. We construct a self-attention based sentiment classifier and use the learned attention weight as the supervisory signal. The motivation comes from the fact that, in a well-trained sentiment classification model, the attention weight reflects the sentiment contribution of each word to

Algorithm 1 The cycled reinforcement learning method for training the neutralization module N_θ and the emotionalization module E_ϕ.

```
 1: Initialize the neutralization module $N_\theta$, the emotionalization module $E_\phi$ with random weights $\theta$, $\phi$
 2: Pre-train $N_\theta$ using MLE based on Eq. 6
 3: Pre-train $E_\phi$ using MLE based on Eq. 7
 4: for each iteration $i = 1, 2, ..., M$ do
 5:     Sample a sequence $\boldsymbol{x}$ with sentiment $s$ from $X$
 6:     Generate a neutralized sequence $\hat{\boldsymbol{x}}$ based on $N_\theta$
 7:     Given $\hat{\boldsymbol{x}}$ and $s$, generate an output based on $E_\phi$
 8:     Compute the gradient of $E_\phi$ based on Eq. 8
 9:     Compute the reward $R_1$ based on Eq. 11
10:     $\bar{s}$ = the opposite sentiment
11:     Given $\hat{\boldsymbol{x}}$ and $\bar{s}$, generate an output based on $E_\phi$
12:     Compute the reward $R_2$ based on Eq. 11
13:     Compute the combined reward $R_c$ based on Eq. 10
14:     Compute the gradient of $N_\theta$ based on Eq. 9
15:     Update model parameters $\theta$, $\phi$
16: end for
```

some extent. Emotional words tend to get higher attention weights while neutral words usually get lower weights. The details of sentiment classifier are described as follows.

Given an input sequence $\boldsymbol{x}$, a sentiment label y is produced as

$$y = softmax(W \cdot \boldsymbol{c}) \tag{1}$$

where W is a parameter. The term $\boldsymbol{c}$ is computed as a weighted sum of hidden vectors:

$$\boldsymbol{c} = \sum_{i=0}^{T} \alpha_i \boldsymbol{h}_i \tag{2}$$

where α_i is the weight of h_i. The term $\boldsymbol{h}_i$ is the output of LSTM at the i-th word. The term α_i is computed as

$$\alpha_i = \frac{\exp(e_i)}{\sum_{i=0}^{T} \exp(e_i)} \tag{3}$$

where $e_i = f(\boldsymbol{h}_i, \boldsymbol{h}_T)$ is an alignment model. We consider the last hidden state $\boldsymbol{h}_T$ as the context vector, which contains all information of an input sequence. The term e_i evaluates the contribution of each word for sentiment classification.

Our experimental results show that the proposed sentiment classifier achieves the accuracy of 89% and 90% on two datasets. With high classification accuracy, the attention weight produced by the classifier is considered to adequately capture the sentiment information of each word.

To extract non-emotional words based on continuous attention weights, we map attention weights to discrete values, 0 and 1. Since the discrete method is not the key part is this paper, we only use the following method for simplification.

We first calculate the averaged attention value in a sentence as

$$\bar{\alpha} = \frac{1}{T}\sum_{i=0}^{T}\alpha_i \tag{4}$$

where $\bar{\alpha}$ is used as the threshold to distinguish non-emotional words from emotional words. The discrete attention weight is calculated as

$$\hat{\alpha_i} = \begin{cases} 1, & \text{if } \alpha_i \leq \bar{\alpha} \\ 0, & \text{if } \alpha_i > \bar{\alpha} \end{cases} \tag{5}$$

where $\hat{\alpha_i}$ is treated as the identifier.

For pre-training the neutralization module, we build the training pair of input text $\boldsymbol{x}$ and a discrete attention weight sequence $\hat{\boldsymbol{\alpha}}$. The cross entropy loss is computed as

$$L_\theta = -\sum_{i=1}^{T} P_{N_\theta}(\hat{\alpha_i}|x_i) \tag{6}$$

3.3 Emotionalization Module

The emotionalization module E_ϕ is responsible for adding sentiment to the neutralized semantic content. In our work, we use a bi-decoder based encoder-decoder framework, which contains one encoder and two decoders. One decoder adds the positive sentiment and the other adds the negative sentiment. The input sentiment signal determines which decoder to use. Specifically, we use the seq2seq model (Sutskever et al., 2014) for implementation. Both the encoder and decoder are LSTM networks. The encoder learns to compress the semantic content into a dense vector. The decoder learns to add sentiment based on the dense vector. Given the neutralized semantic content and the target sentiment, this module is responsible for producing an emotional sequence.

For pre-training the emotionalization module, we first generate a neutralized input sequence $\hat{\boldsymbol{x}}$ by removing emotional words identified by the proposed sentiment classifier. Given the training pair of a neutralized sequence $\hat{\boldsymbol{x}}$ and an original sentence $\boldsymbol{x}$ with sentiment s, the cross entropy loss is computed as

$$L_\phi = -\sum_{i=1}^{T} P_{E_\phi}(x_i|\hat{x}_i, s) \tag{7}$$

where a positive example goes through the positive decoder and a negative example goes through the negative decoder.

We also explore a simpler method for pre-training the emotionalization module, which uses the product between a continuous vector $1 - \boldsymbol{\alpha}$ and a word embedding sequence as the neutralized content where $\boldsymbol{\alpha}$ represents an attention weight sequence. Experimental results show that this method achieves much lower results than explicitly removing emotional words based on discrete attention weights. Thus, we do not choose this method in our work.

3.4 Cycled Reinforcement Learning

Two modules are trained by the proposed cycled method. The neutralization module first neutralizes an emotional input to semantic content and then the emotionalization module is forced to reconstruct the original sentence based on the source sentiment and the semantic content. Therefore, the emotionalization module is taught to add sentiment to the semantic content in a supervised way. Because of the discrete choice of neutral words, the loss is no longer differentiable over the neutralization module. Therefore, we formulate it as a reinforcement learning problem and use policy gradient to train the neutralization module. The detailed training process is shown as follows.

We refer the neutralization module N_θ as the first agent and the emotionalization module E_ϕ as the second one. Given a sentence $\boldsymbol{x}$ associated with sentiment s, the term $\hat{\boldsymbol{x}}$ represents the middle neutralized context extracted by $\hat{\boldsymbol{\alpha}}$, which is generated by $P_{N_\theta}(\hat{\boldsymbol{\alpha}}|\boldsymbol{x})$.

In cycled training, the original sentence can be viewed as the supervision for training the second agent. Thus, the gradient for the second agent is

$$\nabla_\phi J(\phi) = \nabla_\phi \log(P_{E_\phi}(\boldsymbol{x}|\hat{\boldsymbol{x}}, s)) \tag{8}$$

We denote $\bar{\boldsymbol{x}}$ as the output generated by $P_{E_\phi}(\bar{\boldsymbol{x}}|\hat{\boldsymbol{x}}, s)$. We also denote $\boldsymbol{y}$ as the output generated by $P_{E_\phi}(\boldsymbol{y}|\hat{\boldsymbol{x}}, \bar{s})$ where $\bar{s}$ represents the opposite sentiment. Given $\bar{\boldsymbol{x}}$ and $\boldsymbol{y}$, we first calculate rewards for training the neutralized module, R_1 and R_2. The details of calculation process will be introduced in Section 3.4.1. Then, we optimize parameters through policy gradient by maximizing the expected reward to train the neutralization module. It guides the neutralization module to identify non-emotional words better. In return, the improved neutralization module further enhances the emotionalization module.

According to the policy gradient theorem (Williams, 1992), the gradient for the first agent is

$$\nabla_\theta J(\theta) = \mathbb{E}[R_c \cdot \nabla_\theta \log(P_{N_\theta}(\hat{\boldsymbol{\alpha}}|\boldsymbol{x}))] \quad (9)$$

where R_c is calculated as

$$R_c = R_1 + R_2 \quad (10)$$

Based on Eq. 8 and Eq. 9, we use the sampling approach to estimate the expected reward. This cycled process is repeated until converge.

3.4.1 Reward

The reward consists of two parts, sentiment confidence and BLEU. Sentiment confidence evaluates whether the generated text matches the target sentiment. We use a pre-trained classifier to make the judgment. Specially, we use the proposed self-attention based sentiment classifier for implementation. The BLEU (Papineni et al., 2002) score is used to measure the content preservation performance. Considering that the reward should encourage the model to improve both metrics, we use the harmonic mean of sentiment confidence and BLEU as reward, which is formulated as

$$R = (1 + \beta^2)\frac{2 \cdot BLEU \cdot Confid}{(\beta^2 \cdot BLEU) + Confid} \quad (11)$$

where β is a harmonic weight.

4 Experiment

In this section, we evaluate our method on two review datasets. We first introduce the datasets, the training details, the baselines, and the evaluation metrics. Then, we compare our approach with the state-of-the-art systems. Finally, we show the experimental results and provide the detailed analysis of the key components.

4.1 Unpaired Datasets

We conduct experiments on two review datasets that contain user ratings associated with each review. Following previous work (Shen et al., 2017), we consider reviews with rating above three as positive reviews and reviews below three as negative reviews. The positive and negative reviews are not paired. Since our approach focuses on sentence-level sentiment-to-sentiment translation where sentiment annotations are provided at the document level, we process the two datasets with the following steps. First, following previous work (Shen et al., 2017), we filter out the reviews that exceed 20 words. Second, we construct text-sentiment pairs by extracting the first sentence in a review associated with its sentiment label, because the first sentence usually expresses the core idea. Finally, we train a sentiment classifier and filter out the text-sentiment pairs with the classifier confidence below 0.8. Specially, we use the proposed self-attention based sentiment classifier for implementation. The details of the processed datasets are introduced as follows.

Yelp Review Dataset (Yelp): This dataset is provided by Yelp Dataset Challenge.[2] The processed Yelp dataset contains 400K, 10K, and 3K pairs for training, validation, and testing, respectively.

Amazon Food Review Dataset (Amazon): This dataset is provided by McAuley and Leskovec (2013). It consists of amounts of food reviews from Amazon.[3] The processed Amazon dataset contains 230K, 10K, and 3K pairs for training, validation, and testing, respectively.

4.2 Training Details

We tune hyper-parameters based on the performance on the validation sets. The self-attention based sentiment classifier is trained for 10 epochs on two datasets. We set β for calculating reward to 0.5, hidden size to 256, embedding size to 128, vocabulary size to 50K, learning rate to 0.6, and batch size to 64. We use the Adagrad (Duchi et al., 2011) optimizer. All of the gradients are clipped when the norm exceeds 2. Before cycled training, the neutralization module and the emotionalization module are pre-trained for 1 and 4 epochs on the yelp dataset, for 3 and 5 epochs on the Amazon dataset.

4.3 Baselines

We compare our proposed method with the following state-of-the-art systems.

Cross-Alignment Auto-Encoder (CAAE): This method is proposed by Shen et al. (2017). They propose a method that uses refined alignment of latent representations in hidden layers to

[2]`https://www.yelp.com/dataset/challenge`

[3]`http://amazon.com`

perform style transfer. We treat this model as a baseline and adapt it by using the released code.

Multi-Decoder with Adversarial Learning (MDAL): This method is proposed by Fu et al. (2018). They use a multi-decoder model with adversarial learning to separate style representations and content representations in hidden layers. We adapt this model by using the released code.

4.4 Evaluation Metrics

We conduct two evaluations in this work, including an automatic evaluation and a human evaluation. The details of evaluation metrics are shown as follows.

4.4.1 Automatic Evaluation

We quantitatively measure sentiment transformation by evaluating the accuracy of generating designated sentiment. For a fair comparison, we do not use the proposed sentiment classification model. Following previous work (Shen et al., 2017; Hu et al., 2017), we instead use a state-of-the-art sentiment classifier (Vieira and Moura, 2017), called TextCNN, to automatically evaluate the transferred sentiment accuracy. TextCNN achieves the accuracy of 89% and 88% on two datasets. Specifically, we generate sentences given sentiment s, and use the pre-trained sentiment classifier to assign sentiment labels to the generated sentences. The accuracy is calculated as the percentage of the predictions that match the sentiment s.

To evaluate the content preservation performance, we use the BLEU score (Papineni et al., 2002) between the transferred sentence and the source sentence as an evaluation metric. BLEU is a widely used metric for text generation tasks, such as machine translation, summarization, etc. The metric compares the automatically produced text with the reference text by computing overlapping lexical n-gram units.

To evaluate the overall performance, we use the geometric mean of ACC and BLEU as an evaluation metric. The G-score is one of the most commonly used “single number” measures in Information Retrieval, Natural Language Processing, and Machine Learning.

4.4.2 Human Evaluation

While the quantitative evaluation provides indication of sentiment transfer quality, it can not evaluate the quality of transferred text accurately. Therefore, we also perform a human evaluation on the test set. We randomly choose 200 items for the human evaluation. Each item contains the transformed sentences generated by different systems given the same source sentence. The items are distributed to annotators who have no knowledge about which system the sentence is from. They are asked to score the transformed text in terms of sentiment and semantic similarity. Sentiment represents whether the sentiment of the source text is transferred correctly. Semantic similarity evaluates the context preservation performance. The score ranges from 1 to 10 (1 is very bad and 10 is very good).

Yelp	ACC	BLEU	G-score
CAAE (Shen et al., 2017)	93.22	1.17	10.44
MDAL (Fu et al., 2018)	85.65	1.64	11.85
Proposed Method	80.00	22.46	**42.38**
Amazon	**ACC**	**BLEU**	**G-score**
CAAE (Shen et al., 2017)	84.19	0.56	6.87
MDAL (Fu et al., 2018)	70.50	0.27	4.36
Proposed Method	70.37	14.06	**31.45**

Table 1: Automatic evaluations of the proposed method and baselines. ACC evaluates sentiment transformation. BLEU evaluates content preservation. G-score is the geometric mean of ACC and BLEU.

4.5 Experimental Results

Automatic evaluation results are shown in Table 1. ACC evaluates sentiment transformation. BLEU evaluates semantic content preservation. G-score represents the geometric mean of ACC and BLEU. CAAE and MDAL achieve much lower BLEU scores, 1.17 and 1.64 on the Yelp dataset, 0.56 and 0.27 on the Amazon dataset. The low BLEU scores indicate the worrying content preservation performance to some extent. Even with the desired sentiment, the irrelevant generated text leads to worse overall performance. In general, these two systems work more like sentiment-aware language models that generate text only based on the target sentiment and neglect the source input. The main reason is that these two systems attempt to separate emotional information from non-emotional content in a hidden layer, where all information is complicatedly mixed together. It is difficult to only modify emotional information without any loss of non-emotional semantic content.

In comparison, our proposed method achieves the best overall performance on the two datasets,

Yelp	Sentiment	Semantic	G-score
CAAE (Shen et al., 2017)	7.67	3.87	5.45
MDAL (Fu et al., 2018)	7.12	3.68	5.12
Proposed Method	6.99	5.08	**5.96**
Amazon	**Sentiment**	**Semantic**	**G-score**
CAAE (Shen et al., 2017)	8.61	3.15	5.21
MDAL (Fu et al., 2018)	7.93	3.22	5.05
Proposed Method	7.92	4.67	**6.08**

Table 2: Human evaluations of the proposed method and baselines. *Sentiment* evaluates sentiment transformation. *Semantic* evaluates content preservation.

demonstrating the ability of learning knowledge from unpaired data. This result is attributed to the improved BLEU score. The BLEU score is largely improved from 1.64 to 22.46 and from 0.56 to 14.06 on the two datasets. The score improvements mainly come from the fact that we separate emotional information from semantic content by explicitly filtering out emotional words. The extracted content is preserved and fed into the emotionalization module. Given the overall quality of transferred text as the reward, the neutralization module is taught to extract non-emotional semantic content better.

Table 2 shows the human evaluation results. It can be clearly seen that the proposed method obviously improves semantic preservation. The semantic score is increased from 3.87 to 5.08 on the Yelp dataset, and from 3.22 to 4.67 on the Amazon dataset. In general, our proposed model achieves the best overall performance. Furthermore, it also needs to be noticed that with the large improvement in content preservation, the sentiment accuracy of the proposed method is lower than that of CAAE on the two datasets. It shows that simultaneously promoting sentiment transformation and content preservation remains to be studied further.

By comparing two evaluation results, we find that there is an agreement between the human evaluation and the automatic evaluation. It indicates the usefulness of automatic evaluation metrics. However, we also notice that the human evaluation has a smaller performance gap between the baselines and the proposed method than the automatic evaluation. It shows the limitation of automatic metrics for giving accurate results. For evaluating sentiment transformation, even with a high accuracy, the sentiment classifier sometimes generates noisy results, especially for those neutral sentences (e.g., "I ate a cheese sandwich"). For evaluating content preservation, the BLEU score is computed based on the percentage of overlapping n-grams between the generated text and the reference text. However, the overlapping n-grams contain not only content words but also function words, bringing the noisy results. In general, accurate automatic evaluation metrics are expected in future work.

Input: *I would strongly advise against using this company.* **CAAE**: *I love this place for a great experience here.* **MDAL**: *I have been a great place was great.* **Proposed Method**: *I would love using this company.*
Input: *The service was nearly non-existent and extremely rude.* **CAAE**: *The best place in the best area in vegas.* **MDAL**: *The food is very friendly and very good.* **Proposed Method**: *The service was served and completely fresh.*
Input: *Asked for the roast beef and mushroom sub, only received roast beef.* **CAAE**: *We had a great experience with.* **MDAL**: *This place for a great place for a great food and best.* **Proposed Method**: *Thanks for the beef and spring bbq.*
Input: *Worst cleaning job ever!* **CAAE**: *Great food and great service!* **MDAL**: *Great food, food!* **Proposed Method**: *Excellent outstanding job ever!*
Input: *Most boring show I've ever been.* **CAAE**: *Great place is the best place in town.* **MDAL**: *Great place I've ever ever had.* **Proposed Method**: *Most amazing show I've ever been.*
Input: *Place is very clean and the food is delicious.* **CAAE**: *Don't go to this place.* **MDAL**: *This place wasn't worth the worst place is horrible.* **Proposed Method**: *Place is very small and the food is terrible.*
Input: *Really satisfied with experience buying clothes.* **CAAE**: *Don't go to this place.* **MDAL**: *Do not impressed with this place.* **Proposed Method**: *Really bad experience.*

Table 3: Examples generated by the proposed approach and baselines on the Yelp dataset. The two baselines change not only the polarity of examples, but also the semantic content. In comparison, our approach changes the sentiment of sentences with higher semantic similarity.

Table 3 presents the examples generated by different systems on the Yelp dataset. The two baselines change not only the polarity of examples, but also the semantic content. In comparison, our method precisely changes the sentiment of sentences (and paraphrases slightly to ensure fluency), while keeping the semantic content unchanged.

Yelp	ACC	BLEU	G-score
Emotionalization Module	41.84	25.66	32.77
+ NM + Cycled RL	85.71	1.08	9.62
+ NM + Pre-training	70.61	17.02	34.66
+ NM + Cycled RL + Pre-training	80.00	22.46	**42.38**
Amazon	ACC	BLEU	G-score
Emotionalization Module	57.28	12.22	26.46
+ NM + Cycled RL	64.16	8.03	22.69
+ NM + Pre-training	69.61	11.16	27.87
+ NM + Cycled RL + Pre-training	70.37	14.06	**31.45**

Table 4: Performance of key components in the proposed approach. "NM" denotes the neutralization module. "Cycled RL" represents cycled reinforcement learning.

4.6 Incremental Analysis

In this section, we conduct a series of experiments to evaluate the contributions of our key components. The results are shown in Table 4.

We treat the emotionalization module as a baseline where the input is the original emotional sentence. The emotionalization module achieves the highest BLEU score but with much lower sentiment transformation accuracy. The encoding of the original sentiment leads to the emotional hidden vector that influences the decoding process and results in worse sentiment transformation performance.

It can be seen that the method with all components achieves the best performance. First, we find that the method that only uses cycled reinforcement learning performs badly because it is hard to guide two randomly initialized modules to teach each other. Second, the pre-training method brings a slight improvement in overall performance. The G-score is improved from 32.77 to 34.66 and from 26.46 to 27.87 on the two datasets. The bottleneck of this method is the noisy attention weight because of the limited sentiment classification accuracy. Third, the method that combines cycled reinforcement learning and pre-training achieves the better performance than using one of them. Pre-training gives the two modules initial learning ability. Cycled training teaches the two modules to improve each other based on the feedback signals. Specially, the G-score is improved from 34.66 to 42.38 and from 27.87 to 31.45 on the two datasets. Finally, by comparing the methods with and without the neutralization module, we find that the neutralization mechanism improves a lot in sentiment transformation with a slight reduction on content preservation. It proves the effectiveness of explicitly separating sentiment information from semantic content.

Michael is absolutely wonderful.
I would strongly advise against using this company.
Horrible experience!
Worst cleaning job ever!
Most boring show i 've ever been.
Hainan chicken was really good.
I really don't understand all the negative reviews for this dentist.
Smells so weird in there.
The service was nearly non-existent and extremely rude.

Table 5: Analysis of the neutralization module. Words in red are removed by the neutralization module.

Furthermore, to analyze the neutralization ability in the proposed method, we randomly sample several examples, as shown in Table 5. It can be clearly seen that emotional words are removed accurately almost without loss of non-emotional information.

4.7 Error Analysis

Although the proposed method outperforms the state-of-the-art systems, we also observe several failure cases, such as sentiment-conflicted sentences (e.g., "Outstanding and bad service"), neutral sentences (e.g., "Our first time here"). Sentiment-conflicted sentences indicate that the original sentiment is not removed completely. This problem occurs when the input contains emotional words that are unseen in the training data, or the sentiment is implicitly expressed. Handling complex sentiment expressions is an important problem for future work. Neutral sentences demonstrate that the decoder sometimes fails in adding the target sentiment and only generates text based on the semantic content. A better sentiment-aware decoder is expected to be explored in future work.

5 Conclusions and Future Work

In this paper, we focus on unpaired sentiment-to-sentiment translation and propose a cycled reinforcement learning approach that enables training in the absence of parallel training data. We conduct experiments on two review datasets. Experimental results show that our method substantially outperforms the state-of-the-art systems, especially in terms of semantic preservation. For future work, we would like to explore a fine-grained version of sentiment-to-sentiment translation that not only reverses sentiment, but also changes the strength of sentiment.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028), National High Technology Research and Development Program of China (863 Program, No. 2015AA015404), and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving sentiment analysis via sentence type classification using bilstm-crf and CNN. *Expert Syst. Appl.*, 72:221–230.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 623–632.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI 2018*.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2414–2423.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 820–828.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. In *ICML 2017*.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *ECCV, 2016*, pages 694–711.

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. 2017. Demystifying neural style transfer. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2230–2236.

Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. 2017. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 36(4):120:1–120:15.

Junyang Lin, Shuming Ma, Qi Su, and Xu Sun. 2018. Decoding-history-based adaptive control of attention for neural machine translation. *CoRR*, abs/1802.01812.

Dehong Ma, Sujian Li, Xiaodong Zhang, Houfeng Wang, and Xu Sun. 2017. Cascading multiway attentions for document-level sentiment classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 634–643.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018a. Query and output: Generating words by querying distributed word representations for paraphrase generation. *CoRR*, abs/1803.01465.

Shuming Ma, Xu Sun, Junyang Lin, and Xuancheng Ren. 2018b. A hierarchical end-to-end model for jointly improving text summarization and sentiment classification. *CoRR*, abs/1805.01089.

Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 897–908.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 486–495.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 502–518.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS 2017*.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 129–136.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3097–3103.

Joao Paulo Albuquerque Vieira and Raimundo Santos Moura. 2017. An analysis of convolutional neural networks for sentence classification. In *XLIII 2017*, pages 1–5.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Jingjing Xu, Xu Sun, Xuancheng Ren, Junyang Lin, Bingzhen Wei, and Wei Li. 2018. DP-GAN: diversity-promoting generative adversarial network for generating informative and diversified text. *CoRR*, abs/1802.01345.

Hongyu Zang and Xiaojun Wan. 2017. Towards automatic generation of product reviews from aspect-sentiment scores. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 168–177.

Zhiyuan Zhang, Wei Li, and Xu Sun. 2018. Automatic transferring between ancient chinese and contemporary chinese. *CoRR*, abs/1803.01557.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251.

Discourse Marker Augmented Network with Reinforcement Learning for Natural Language Inference

Boyuan Pan[†], **Yazheng Yang**[‡], **Zhou Zhao**[‡], **Yueting Zhuang**[‡], **Deng Cai**[†♯*], **Xiaofei He**[⋆†]
[†]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
[‡]College of Computer Science, Zhejiang University, Hangzhou, China
[♯]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
[⋆]Fabu Inc., Hangzhou, China
{panby, yazheng_yang, zhaozhou, yzhuang}@zju.edu.cn
{dengcai, xiaofeihe}@cad.zju.edu.com

Abstract

Natural Language Inference (NLI), also known as Recognizing Textual Entailment (RTE), is one of the most important problems in natural language processing. It requires to infer the logical relationship between two given sentences. While current approaches mostly focus on the interaction architectures of the sentences, in this paper, we propose to transfer knowledge from some important discourse markers to augment the quality of the NLI model. We observe that people usually use some discourse markers such as "so" or "but" to represent the logical relationship between two sentences. These words potentially have deep connections with the meanings of the sentences, thus can be utilized to help improve the representations of them. Moreover, we use reinforcement learning to optimize a new objective function with a reward defined by the property of the NLI datasets to make full use of the labels information. Experiments show that our method achieves the state-of-the-art performance on several large-scale datasets.

1 Introduction

In this paper, we focus on the task of Natural Language Inference (NLI), which is known as a significant yet challenging task for natural language understanding. In this task, we are given two sentences which are respectively called **premise** and **hypothesis**. The goal is to determine whether the logical relationship between them is *entailment*, *neutral*, or *contradiction*.

Recently, performance on NLI(Chen et al., 2017b; Gong et al., 2018; Chen et al., 2017c) has been significantly boosted since the release of some high quality large-scale benchmark datasets such as SNLI(Bowman et al., 2015) and MultiNLI(Williams et al., 2017). Table 1 shows some examples in SNLI. Most state-of-the-art works focus on the interaction architectures between the premise and the hypothesis, while they rarely concerned the discourse relations of the sentences, which is a core issue in natural language understanding.

Premise: A soccer game with multiple males playing. **Hypothesis**: Some men are playing a sport. **Label**: *Entailment*
Premise: An older and younger man smiling. **Hypothesis**: Two men are smiling and laughing at the cats playing on the floor. **Label**: *Neutral*
Premise: A black race car starts up in front of a crowd of people **Hypothesis**: A man is driving down a lonely road. **Label**: *Contradiction*

Table 1: Three examples in SNLI dataset.

People usually use some certain set of words to express the discourse relation between two sentences[1]. These words, such as "but" or "and", are denoted as *discourse markers*. These discourse markers have deep connections with the intrinsic relations of two sentences and intuitively correspond to the intent of NLI, such as "but" to "contradiction", "so" to "entailment", *etc.*

Very few NLI works utilize this information revealed by discourse markers. Nie et al. (2017) proposed to use discourse markers to help rep-

[*]corresponding author

[1]Here sentences mean either the whole sentences or the main clauses of a compound sentence.

resent the meanings of the sentences. However, they represent each sentence by a single vector and directly concatenate them to predict the answer, which is too simple and not ideal for the large-scale datasets.

In this paper, we propose a *Discourse Marker Augmented Network* for natural language inference, where we transfer the knowledge from the existing supervised task: Discourse Marker Prediction (DMP)(Nie et al., 2017), to an integrated NLI model. We first propose a sentence encoder model that learns the representations of the sentences from the DMP task and then inject the encoder to the NLI network. Moreover, because our NLI datasets are manually annotated, each example from the datasets might get several different labels from the annotators although they will finally come to a consensus and also provide a certain label. In consideration of that different confidence level of the final labels should be discriminated, we employ reinforcement learning with a reward defined by the uniformity extent of the original labels to train the model. The contributions of this paper can be summarized as follows.

- Unlike previous studies, we solve the task of the natural language inference via transferring knowledge from another supervised task. We propose the Discourse Marker Augmented Network to combine the learned encoder of the sentences with the integrated NLI model.
- According to the property of the datasets, we incorporate reinforcement learning to optimize a new objective function to make full use of the labels' information.
- We conduct extensive experiments on two large-scale datasets to show that our method achieves better performance than other state-of-the-art solutions to the problem.

2 Task Description

2.1 Natural Language Inference (NLI)

In the natural language inference tasks, we are given a pair of sentences (P, H), which respectively means the premise and hypothesis. Our goal is to judge whether their logical relationship between their meanings by picking a label from a small set: *entailment* (The hypothesis is definitely a true description of the premise), *neutral* (The hypothesis might be a true description of the premise), and *contradiction* (The hypothesis is definitely a false description of the premise).

2.2 Discourse Marker Prediction (DMP)

For DMP, we are given a pair of sentences (S_1, S_2), which is originally the first half and second half of a complete sentence. The model must predict which discourse marker was used by the author to link the two ideas from a set of candidates.

3 Sentence Encoder Model

Following (Nie et al., 2017; Kiros et al., 2015), we use BookCorpus(Zhu et al., 2015) as our training data for discourse marker prediction, which is a dataset of text from unpublished novels, and it is large enough to avoid bias towards any particular domain or application. After preprocessing, we obtain a dataset with the form (S_1, S_2, m), which means the first half sentence, the last half sentence, and the discourse marker that connected them in the original text. Our goal is to predict the m given S_1 and S_2.

We first use *Glove*(Pennington et al., 2014) to transform $\{S_t\}_{t=1}^{2}$ into vectors word by word and subsequently input them to a bi-directional LSTM:

$$\begin{aligned} \overrightarrow{\mathbf{h}_t^i} &= \overrightarrow{\text{LSTM}}(Glove(S_t^i)), i = 1, ..., |S_t| \\ \overleftarrow{\mathbf{h}_t^i} &= \overleftarrow{\text{LSTM}}(Glove(S_t^i)), i = |S_t|, ..., 1 \end{aligned} \tag{1}$$

where $Glove(w)$ is the embedding vector of the word w from the *Glove* lookup table, $|S_t|$ is the length of the sentence S_t. We apply max pooling on the concatenation of the hidden states from both directions, which provides regularization and shorter back-propagation paths(Collobert and Weston, 2008), to extract the features of the whole sequences of vectors:

$$\begin{aligned} \overrightarrow{\mathbf{r}_t} &= \text{Max}_{dim}([\overrightarrow{\mathbf{h}_t^1}; \overrightarrow{\mathbf{h}_t^2}; ...; \overrightarrow{\mathbf{h}_t^{|S_t|}}]) \\ \overleftarrow{\mathbf{r}_t} &= \text{Max}_{dim}([\overleftarrow{\mathbf{h}_t^1}; \overleftarrow{\mathbf{h}_t^2}; ...; \overleftarrow{\mathbf{h}_t^{|S_t|}}]) \end{aligned} \tag{2}$$

where Max_{dim} means that the max pooling is performed across each dimension of the concatenated vectors, $[;]$ denotes concatenation. Moreover, we combine the last hidden state from both directions and the results of max pooling to represent our sentences:

$$\mathbf{r}_t = [\overrightarrow{\mathbf{r}_t}; \overleftarrow{\mathbf{r}_t}; \overrightarrow{\mathbf{h}_t^{|S_t|}}; \overleftarrow{\mathbf{h}_t^1}] \tag{3}$$

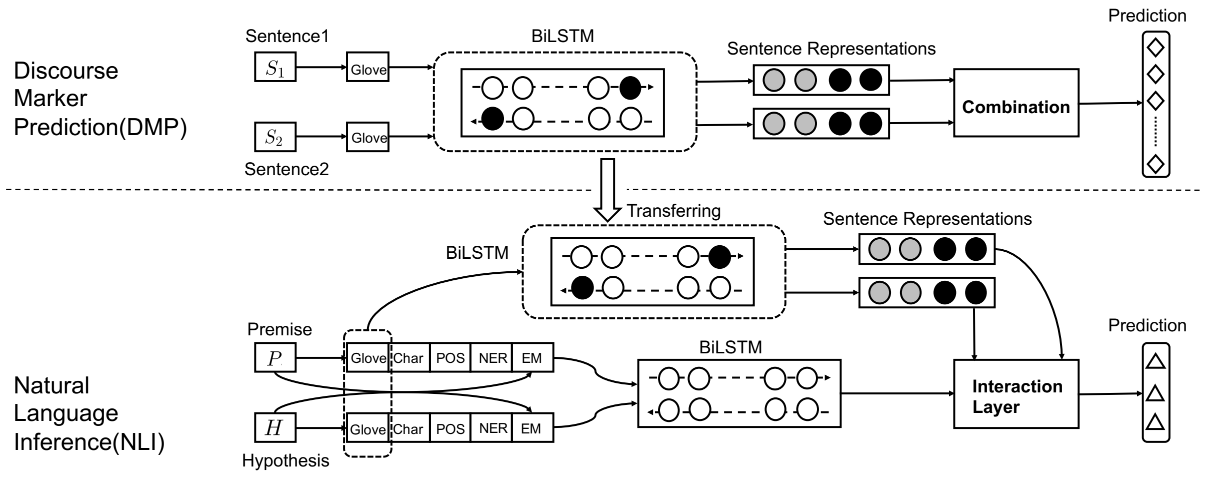


Figure 1: Overview of our Discource Marker Augmented Network, comprising the part of Discourse Marker Prediction (upper) for pre-training and Natural Language Inferance (bottom) to which the learned knowledge will be transferred.

where $\mathbf{r}_t$ is the representation vector of the sentence S_t. To predict the discourse marker between S_1 and S_2, we combine the representations of them with some linear operation:

$$\mathbf{r} = [\mathbf{r}_1; \mathbf{r}_2; \mathbf{r}_1 + \mathbf{r}_2; \mathbf{r}_1 \odot \mathbf{r}_2] \tag{4}$$

where $\odot$ is elementwise product. Finally we project $\mathbf{r}$ to a vector of label size (the total number of discourse markers in the dataset) and use softmax function to normalize the probability distribution.

4 Discourse Marker Augmented Network

As presented in Figure 1, we show how our Discourse Marker Augmented Network incorporates the learned encoder into the NLI model.

4.1 Encoding Layer

We denote the premise as P and the hypothesis as H. To encode the words, we use the concatenation of following parts:

Word Embedding: Similar to the previous section, we map each word to a vector space by using pre-trained word vectors *GloVe*.

Character Embedding: We apply Convolutional Neural Networks (CNN) over the characters of each word. This approach is proved to be helpful in handling out-of-vocab (OOV) words(Yang et al., 2017).

POS and NER tags: We use the part-of-speech (POS) tags and named-entity recognition (NER) tags to get syntactic information and entity label of the words. Following (Pan et al., 2017b), we apply the skip-gram model(Mikolov et al., 2013) to train two new lookup tables of POS tags and NER tags respectively. Each word can get its own POS embedding and NER embedding by these lookup tables. This approach represents much better geometrical features than common used one-hot vectors.

Exact Match: Inspired by the machine comprehension tasks(Chen et al., 2017a), we want to know whether every word in P is in H (and H in P). We use three binary features to indicate whether the word can be exactly matched to any question word, which respectively means original form, lowercase and lemma form.

For encoding, we pass all sequences of vectors into a bi-directional LSTM and obtain:

$$\begin{aligned} \mathbf{p}_i &= \text{BiLSTM}(f_{rep}(P_i), \mathbf{p}_{i-1}), i = 1, ..., n \\ \mathbf{u}_j &= \text{BiLSTM}(f_{rep}(H_j), \mathbf{u}_{j-1}), j = 1, ..., m \end{aligned} \tag{5}$$

where $f_{rep}(\mathbf{x}) = [\text{Glove}(\mathbf{x}); \text{Char}(\mathbf{x}); \text{POS}(\mathbf{x}); \text{NER}(\mathbf{x}); \text{EM}(\mathbf{x})]$ is the concatenation of the embedding vectors and the feature vectors of the word $\mathbf{x}$, $n = |P|$, $m = |H|$.

4.2 Interaction Layer

In this section, we feed the results of the encoding layer and the learned sentence encoder into the attention mechanism, which is responsible for linking and fusing information from the premise and the hypothesis words.

We first obtain a similarity matrix $\mathbf{A} \in R^{n \times m}$ between the premise and hypothesis by

$$\mathbf{A}_{ij} = \mathbf{v}_1^\top [\mathbf{p}_i; \mathbf{u}_j; \mathbf{p}_i \circ \mathbf{u}_j; \mathbf{r}_p; \mathbf{r}_h] \tag{6}$$

where $\mathbf{v}_1$ is the trainable parameter, $\mathbf{r}_p$ and $\mathbf{r}_h$ are sentences representations from the equation (3) learned in the Section 3, which denote the premise and hypothesis respectively. In addition to previous popular similarity matrix, we incorporate the relevance of each word of $P(H)$ to the whole sentence of $H(P)$. Now we use $\mathbf{A}$ to obtain the attentions and the attended vectors in both directions.

To signify the attention of the i-th word of P to every word of H, we use the weighted sum of $\mathbf{u}_j$ by $\mathbf{A}_{i:}$:

$$\tilde{\mathbf{u}}_i = \sum_j \mathbf{A}_{ij} \cdot \mathbf{u}_j \tag{7}$$

where $\tilde{\mathbf{u}}_i$ is the attention vector of the i-th word of P for the entire H. In the same way, the $\tilde{\mathbf{p}}_j$ is obtained via:

$$\tilde{\mathbf{p}}_j = \sum_i \mathbf{A}_{ij} \cdot \mathbf{p}_i \tag{8}$$

To model the local inference between aligned word pairs, we integrate the attention vectors with the representation vectors via:

$$\begin{aligned} \hat{\mathbf{p}}_i &= f([\mathbf{p}_i; \tilde{\mathbf{u}}_i; \mathbf{p}_i - \tilde{\mathbf{u}}_i; \mathbf{p}_i \odot \tilde{\mathbf{u}}_i]) \\ \hat{\mathbf{u}}_j &= f([\mathbf{u}_j; \tilde{\mathbf{p}}_j; \mathbf{u}_j - \tilde{\mathbf{p}}_j; \mathbf{u}_j \odot \tilde{\mathbf{p}}_j]) \end{aligned} \tag{9}$$

where f is a 1-layer feed-forward neural network with the ReLU activation function, $\hat{\mathbf{p}}_i$ and $\hat{\mathbf{u}}_j$ are local inference vectors. Inspired by (Seo et al., 2016) and (Chen et al., 2017b), we use a modeling layer to capture the interaction between the premise and the hypothesis. Specifically, we use bi-directional LSTMs as building blocks:

$$\begin{aligned} \mathbf{p}_i^M &= \mathrm{BiLSTM}(\hat{\mathbf{p}}_i, \mathbf{p}_{i-1}^M) \\ \mathbf{u}_j^M &= \mathrm{BiLSTM}(\hat{\mathbf{u}}_j, \mathbf{u}_{j-1}^M) \end{aligned} \tag{10}$$

Here, $\mathbf{p}_i^M$ and $\mathbf{u}_j^M$ are the modeling vectors which contain the crucial information and relationship among the sentences.

We compute the representation of the whole sentence by the weighted average of each word:

$$\begin{aligned} \mathbf{p}^M &= \sum_i \frac{\exp(\mathbf{v}_2^\top \mathbf{p}_i^M)}{\sum_{i'} \exp(\mathbf{v}_2^\top \mathbf{p}_{i'}^M)} \mathbf{p}_i^M \\ \mathbf{u}^M &= \sum_j \frac{\exp(\mathbf{v}_3^\top \mathbf{u}_j^M)}{\sum_{j'} \exp(\mathbf{v}_3^\top \mathbf{u}_{j'}^M)} \mathbf{u}_j^M \end{aligned} \tag{11}$$

Label Number	SNLI		MultiNLI	
	Correct	Total	Correct	Total
1	510711	510711	392702	392702
2	0	0	0	0
3	8748	0	3045	0
4	16395	2199	4859	0
5	33179	56123	11743	19647

Table 2: Statistics of the labels of SNLI and MuliNLI. **Total** means the number of examples whose number of annotators is in the left column. **Correct** means the number of examples whose number of correct labels from the annotators is in the left column.

where $\mathbf{v}_2, \mathbf{v}_3$ are trainable vectors. We don't share these parameter vectors in this seemingly parallel strucuture because there is some subtle difference between the premise and hypothesis, which will be discussed later in Section 5.

4.3 Output Layer

The NLI task requires the model to predict the logical relation from the given set: *entailment*, *neutral* or *contradiction*. We obtain the probability distribution by a linear function with softmax function:

$$\mathbf{d} = \mathrm{softmax}(\mathbf{W}[\mathbf{p}^M; \mathbf{u}^M; \mathbf{p}^M \odot \mathbf{u}^M; \mathbf{r}_p \odot \mathbf{r}_h]) \tag{12}$$

where $\mathbf{W}$ is a trainable parameter. We combine the representations of the sentences computed above with the representations learned from DMP to obtain the final prediction.

4.4 Training

As shown in Table 2, many examples from our datasets are labeled by several people, and the choices of the annotators are not always consistent. For instance, when the label number is 3 in SNLI, "total=0" means that no examples have 3 annotators (maybe more or less); "correct=8748" means that there are 8748 examples whose number of correct labels is 3 (the number of annotators maybe 4 or 5, but some provided wrong labels). Although all the labels for each example will be unified to a final (correct) label, diversity of the labels for a single example indicates the low confidence of the result, which is not ideal to only use the final label to optimize the model.

We propose a new objective function that combines both the log probabilities of the ground-truth label and a reward defined by the property of the datasets for the reinforcement learning. The most widely used objective function for the natural language inference is to minimize the negative log cross-entropy loss:

$$J_{CE}(\Theta) = -\frac{1}{N}\sum_{k}^{N} log(\mathbf{d}_l^k) \tag{13}$$

where Θ are all the parameters to optimize, N is the number of examples in the dataset, $\mathbf{d}_l$ is the probability of the ground-truth label l.

However, directly using the final label to train the model might be difficult in some situations, where the example is confusing and the labels from the annotators are different. For instance, consider an example from the SNLI dataset:

- P: "A smiling costumed woman is holding an umbrella."
- H: "A happy woman in a fairy costume holds an umbrella."

The final label is *neutral*, but the original labels from the five annotators are *neural*, *neural*, *entailment*, *contradiction*, *neural*, in which case the relation between "smiling" and "happy" might be under different comprehension. The final label's confidence of this example is obviously lower than an example that all of its labels are the same. To simulate the thought of human being more closely, in this paper, we tackle this problem by using the REINFORCE algorithm(Williams, 1992) to minimize the negative expected reward, which is defined as:

$$J_{RL}(\Theta) = -\mathbb{E}_{l\sim\pi(l|P,H)}[R(l, \{l^*\})] \tag{14}$$

where $\pi(l|P, H)$ is the previous action policy that predicts the label given P and H, $\{l^*\}$ is the set of annotated labels, and

$$R(l, \{l^*\}) = \frac{\text{number of } l \text{ in } \{l^*\}}{|\{l^*\}|} \tag{15}$$

is the reward function defined to measure the distance to all the ideas of the annotators.

To avoid of overwriting its earlier results and further stabilize training, we use a linear function to integrate the above two objective functions:

$$J(\Theta) = \lambda J_{CE}(\Theta) + (1 - \lambda)J_{RL}(\Theta) \tag{16}$$

where λ is a tunable hyperparameter.

Discourse Marker	Percentage(%)
but	57.12
because	9.41
if	29.78
when	25.32
so	31.01
although	1.76
before	15.52
still	11.29

Table 3: Statistics of discouse markers in our dataset from BookCorpus.

5 Experiments

5.1 Datasets

BookCorpus: We use the dataset from BookCorpus(Zhu et al., 2015) to pre-train our sentence encoder model. We preprocessed and collected discourse markers from BookCorpus as (Nie et al., 2017). We finally curated a dataset of 6527128 pairs of sentences for 8 discourse markers, whose statistics are shown in Table 3.

SNLI: Stanford Natural Language Inference(Bowman et al., 2015) is a collection of more than 570k human annotated sentence pairs labeled for entailment, contradiction, and semantic independence. SNLI is two orders of magnitude larger than all other resources of its type. The premise data is extracted from the captions of the Flickr30k corpus(Young et al., 2014), the hypothesis data and the labels are manually annotated. The original SNLI corpus contains also the other category, which includes the sentence pairs lacking consensus among multiple human annotators. We remove this category and use the same split as in (Bowman et al., 2015) and other previous work.

MultiNLI: Multi-Genre Natural Language Inference(Williams et al., 2017) is another large-scale corpus for the task of NLI. MultiNLI has 433k sentences pairs and is in the same format as SNLI, but it includes a more diverse range of text, as well as an auxiliary test set for cross-genre transfer evaluation. Half of these selected genres appear in training set while the rest are not, creating in-domain (matched) and cross-domain (mismatched) development/test sets.

Method	SNLI	MultiNLI	
		Matched	Mismatched
300D LSTM encoders(Bowman et al., 2016)	80.6	–	–
300D Tree-based CNN encoders(Mou et al., 2016)	82.1	–	–
4096D BiLSTM with max-pooling(Conneau et al., 2017)	84.5	–	–
600D Gumbel TreeLSTM encoders(Choi et al., 2017)	86.0	–	–
600D Residual stacked encoders(Nie and Bansal, 2017)	86.0	74.6	73.6
Gated-Att BiLSTM(Chen et al., 2017d)	–	73.2	73.6
100D LSTMs with attention(Rocktäschel et al., 2016)	83.5	–	–
300D re-read LSTM(Sha et al., 2016)	87.5	–	–
DIIN(Gong et al., 2018)	88.0	78.8	77.8
Biattentive Classification Network(McCann et al., 2017)	88.1	–	–
300D CAFE(Tay et al., 2017)	88.5	78.7	77.9
KIM(Chen et al., 2017b)	88.6	–	–
600D ESIM + 300D Syntactic TreeLSTM(Chen et al., 2017c)	88.6	–	–
DMAN	**88.8**	**78.9**	**78.2**
BiMPM(Ensemble)(Wang et al., 2017)	88.8	–	–
DIIN(Ensemble)(Gong et al., 2018)	88.9	80.0	78.7
KIM(Ensemble)(Chen et al., 2017b)	89.1	–	–
300D CAFE(Ensemble)(Tay et al., 2017)	89.3	80.2	79.0
DMAN(Ensemble)	**89.6**	**80.3**	**79.4**

Table 4: Performance on the SNLI dataset and the MultiNLI dataset. In the top part, we show sentence encoding-based models; In the medium part, we present the performance of integrated neural network models; In the bottom part, we show the results of ensemble models.

5.2 Implementation Details

We use the Stanford CoreNLP toolkit(Manning et al., 2014) to tokenize the words and generate POS and NER tags. The word embeddings are initialized by 300d *Glove*(Pennington et al., 2014), the dimensions of POS and NER embeddings are 30 and 10. The dataset we use to train the embeddings of POS tags and NER tags are the training set given by SNLI. We apply Tensorflow r1.3 as our neural network framework. We set the hidden size as 300 for all the LSTM layers and apply dropout(Srivastava et al., 2014) between layers with an initial ratio of 0.9, the decay rate as 0.97 for every 5000 step. We use the AdaDelta for optimization as described in (Zeiler, 2012) with ρ as 0.95 and ϵ as 1e-8. We set our batch size as 36 and the initial learning rate as 0.6. The parameter λ in the objective function is set to be 0.2. For DMP task, we use stochastic gradient descent with initial learning rate as 0.1, and we anneal by half each time the validation accuracy is lower than the previous epoch. The number of epochs is set to be 10, and the feedforward dropout rate is 0.2. The learned encoder in subsequent NLI task is trainable.

5.3 Results

In table 4, we compare our model to other competitive published models on SNLI and MultiNLI. As we can see, our method Discourse Marker Augmented Network (DMAN) clearly outperforms all the baselines and achieves the state-of-the-art results on both datasets.

The methods in the top part of the table are sentence encoding based models. Bowman et al. (2016) proposed a simple baseline that uses LSTM to encode the whole sentences and feed them into a MLP classifier to predict the final inference relationship, they achieve an accuracy of 80.6% on SNLI. Nie and Bansal (2017) test their model on both SNLI and MiltiNLI, and achieves competitive results.

In the medium part, we show the results of other neural network models. Obviously, the performance of most of the integrated methods are better than the sentence encoding based models above. Both DIIN(Gong et al., 2018) and

Ablation Model	Accuracy
Only Sentence Encoder Model	83.37
No Sentence Encoder Model	87.24
No Char Embedding	87.95
No POS Embedding	88.76
No NER Embedding	88.71
No Exact Match	88.26
No REINFORCE	88.41
DMAN	88.83

Table 5: Ablations on the SNLI development dataset.

CAFE(Tay et al., 2017) exceed other methods by more than 4% on MultiNLI dataset. However, our DMAN achieves 88.8% on SNLI, 78.9% on matched MultiNLI and 78.2% on mismatched MultiNLI, which are all best results among the baselines.

We present the ensemble results on both datasets in the bottom part of the table 4. We build an ensemble model which consists of 10 single models with the same architecture but initialized with different parameters. The performance of our model achieves 89.6% on SNLI, 80.3% on matched MultiNLI and 79.4% on mismatched MultiNLI, which are all state-of-the-art results.

5.4 Ablation Analysis

As shown in Table 5, we conduct an ablation experiment on SNLI development dataset to evaluate the individual contribution of each component of our model. Firstly we only use the results of the sentence encoder model to predict the answer, in other words, we represent each sentence by a single vector and use dot product with a linear function to do the classification. The result is obviously not satisfactory, which indicates that only using sentence embedding from discourse markers to predict the answer is not ideal in large-scale datasets. We then remove the sentence encoder model, which means we don't use the knowledge transferred from the DMP task and thus the representations $\mathbf{r}_p$ and $\mathbf{r}_h$ are set to be zero vectors in the equation (6) and the equation (12). We observe that the performance drops significantly to 87.24%, which is nearly 1.5% to our DMAN model, which indicates that the discourse markers have deep connections with the logical relations between two sentences they links. When

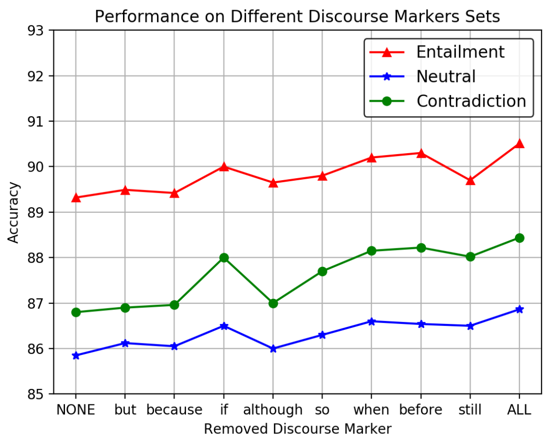


Figure 2: Performance when the sentence encoder is pretrained on different discourse markers sets. "NONE" means the model doesn't use any discourse markers; "ALL" means the model use all the discourse markers.

we remove the character-level embedding and the POS and NER features, the performance drops a lot. We conjecture that those feature tags help the model represent the words as a whole while the char-level embedding can better handle the out-of-vocab (OOV) or rare words. The exact match feature also demonstrates its effectiveness in the ablation result. Finally, we ablate the reinforcement learning part, in other words, we only use the original loss function to optimize the model (set $\lambda = 1$). The result drops about 0.5%, which proves that it is helpful to utilize all the information from the annotators.

5.5 Semantic Analysis

In Figure 2, we show the performance on the three relation labels when the model is pre-trained on different discourse markers sets. In other words, we removed discourse marker from the original set each time and use the rest 7 discourse markers to pre-train the sentence encoder in the DMP task and then train the DMAN. As we can see, there is a sharp decline of accuracy when removing "but", "because" and "although". We can intuitively speculate that "but" and "although" have direct connections with the contradiction label (which drops most significantly) while "because" has some links with the entailment label. We observe that some discourse markers such as "if" or "before" contribute much less than other words which have strong logical hints, although they

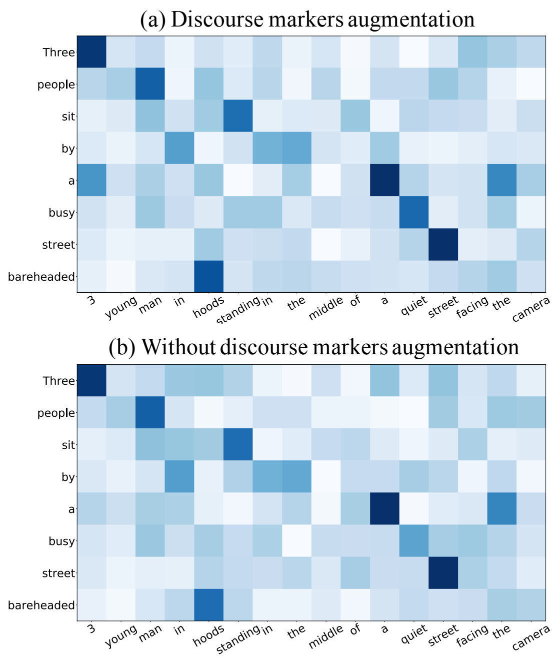


Figure 3: Comparison of the visualized similarity relations.

actually improve the performance of the model. Compared to the other two categories, the "contradiction" label examples seem to benefit the most from the pre-trained sentence encoder.

5.6 Visualization

In Figure 3, we also provide a visualized analysis of the hidden representation from similarity matrix $\mathbf{A}$ (computed in the equation (6)) in the situations that whether we use the discourse markers or not. We pick a sentence pair whose premise is "*3 young man in hoods standing in the middle of a quiet street facing the camera.*" and hypothesis is "*Three people sit by a busy street bareheaded.*" We observe that the values are highly correlated among the synonyms like "people" with "man", "three" with "3" in both situations. However, words that might have contradictory meanings like "hoods" with "bareheaded", "quiet" with "busy" perform worse without the discourse markers augmentation, which conforms to the conclusion that the "contradiction" label examples benefit a lot which is observed in the Section 5.5.

6 Related Work

6.1 Discourse Marker Applications

This work is inspired most directly by the DisSent model and Discourse Prediction Task of Nie et al. (2017), which introduce the use of the discourse markers information for the pretraining of sentence encoders. They follow (Kiros et al., 2015) to collect a large sentence pairs corpus from BookCorpus(Zhu et al., 2015) and propose a sentence representation based on that. They also apply their pre-trained sentence encoder to a series of natural language understanding tasks such as sentiment analysis, question-type, entailment, and relatedness. However, all those datasets are provided by Conneau et al. (2017) for evaluating sentence embeddings and are almost all small-scale and are not able to support more complex neural network. Moreover, they represent each sentence by a single vector and directly combine them to predict the answer, which is not able to interact among the words level.

In closely related work, Jernite et al. (2017) propose a model that also leverage discourse relations. However, they manually group the discourse markers into several categories based on human knowledge and predict the category instead of the explicit discourse marker phrase. However, the size of their dataset is much smaller than that in (Nie et al., 2017), and sometimes there has been disagreement among annotators about what exactly is the correct categorization of discourse relations(Hobbs, 1990).

Unlike previous works, we insert the sentence encoder into an integrated network to augment the semantic representation for NLI tasks rather than directly combining the sentence embeddings to predict the relations.

6.2 Natural Language Inference

Earlier research on the natural language inference was based on small-scale datasets(Marelli et al., 2014), which relied on traditional methods such as shallow methods(Glickman et al., 2005), natural logic methods(MacCartney and Manning, 2007), *etc.* These datasets are either not large enough to support complex deep neural network models or too easy to challenge natural language.

Large and complicated networks have been successful in many natural language processing tasks(Zhu et al., 2017; Chen et al., 2017e; Pan et al., 2017a). Recently, Bowman et al. (2015) released Stanford Natural language Inference (SNLI) dataset, which is a high-quality and large-scale benchmark, thus inspired many significant works(Bowman et al., 2016; Mou et al., 2016; Vendrov et al., 2016; Conneau et al., 2017; Wang et al., 2017; Gong et al., 2018; McCann et al., 2017; Chen et al., 2017b; Choi et al., 2017; Tay et al., 2017). Most of them focus on the improvement of the interaction architectures and obtain competitive results, while transfer learning from external knowledge is popular as well. Vendrov et al. (2016) incorpated Skipthought(Kiros et al., 2015), which is an unsupervised sequence model that has been proven to generate useful sentence embedding. McCann et al. (2017) proposed to transfer the pre-trained encoder from the neural machine translation (NMT) to the NLI tasks.

Our method combines a pre-trained sentence encoder from the DMP task with an integrated NLI model to compose a novel framework. Furthermore, unlike previous studies, we make full use of the labels provided by the annotators and employ policy gradient to optimize a new objective function in order to simulate the thought of human being.

7 Conclusion

In this paper, we propose *Discourse Marker Augmented Network* for the task of the natural language inference. We transfer the knowledge learned from the discourse marker prediction task to the NLI task to augment the semantic representation of the model. Moreover, we take the various views of the annotators into consideration and employ reinforcement learning to help optimize the model. The experimental evaluation shows that our model achieves the state-of-the-art results on SNLI and MultiNLI datasets. Future works involve the choice of discourse markers and some other transfer learning sources.

8 Acknowledgements

This work was supported in part by the National Nature Science Foundation of China (Grant Nos: 61751307), in part by the grant ZJU Research 083650 of the ZJUI Research Program from Zhejiang University and in part by the National Youth Top-notch Talent Support Program. The experiments are supported by Chengwei Yao in the Experiment Center of the College of Computer Science and Technology, Zhejiang university.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1466–1477.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading wikipedia to answer open-domain questions. In *Meeting of the Association for Computational Linguistics (ACL)*, pages 1870–1879.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2017b. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017c. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1657–1668.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017d. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 36–40.

Zheqian Chen, Ben Gao, Huimin Zhang, Zhou Zhao, Haifeng Liu, and Deng Cai. 2017e. User personalized satisfaction prediction via multiple instance deep learning. In *International Conference on World Wide Web (WWW)*, pages 907–915.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2017. Learning to compose task-specific tree structures. *The Association for the Advancement of Artificial Intelligence (AAAI)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. Web based probabilistic textual entailment.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *International Conference on Learning Representations (ICLR)*.

Jerry R Hobbs. 1990. *Literature and cognition*. 21. Center for the Study of Language (CSLI).

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems (NIPS)*, pages 3294–3302.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6297–6308.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 130–136.

Allen Nie, Erin D Bennett, and Noah D Goodman. 2017. Dissent: Sentence representation learning from explicit discourse relations. *arXiv preprint arXiv:1710.04334*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Boyuan Pan, Hao Li, Zhou Zhao, Deng Cai, and Xiaofei He. 2017a. Keyword-based query comprehending via multiple optimized-demand augmentation. *arXiv preprint arXiv:1711.00179*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017b. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. *International Conference on Learning Representations (ICLR)*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2870–2879.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. *International Conference on Learning Representations (ICLR)*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? fine-grained gating for reading comprehension. *International Conference on Learning Representations (ICLR)*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to do next: modeling user behaviors by time-lstm. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3602–3608.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 19–27.

Working Memory Networks: Augmenting Memory Networks with a Relational Reasoning Module

Juan Pavez*, Héctor Allende
Department of Informatics
Federico Santa María
Technical University
Valparaíso, Chile
juan.pavezs@alumnos.usm.cl
hallende@inf.utfsm.cl

Héctor Allende-Cid
Escuela de Ingeniería Informática
Pontífica Universidad Católica
de Valparaíso
Valparaíso, Chile
hector.allende@pucv.cl

Abstract

During the last years, there has been a lot of interest in achieving some kind of complex reasoning using deep neural networks. To do that, models like Memory Networks (MemNNs) have combined external memory storages and attention mechanisms. These architectures, however, lack of more complex reasoning mechanisms that could allow, for instance, relational reasoning. Relation Networks (RNs), on the other hand, have shown outstanding results in relational reasoning tasks. Unfortunately, their computational cost grows quadratically with the number of memories, something prohibitive for larger problems. To solve these issues, we introduce the Working Memory Network, a MemNN architecture with a novel working memory storage and reasoning module. Our model retains the relational reasoning abilities of the RN while reducing its computational complexity from quadratic to linear. We tested our model on the text QA dataset bAbI and the visual QA dataset NLVR. In the jointly trained bAbI-10k, we set a new state-of-the-art, achieving a mean error of less than 0.5%. Moreover, a simple ensemble of two of our models solves all 20 tasks in the joint version of the benchmark.

1 Introduction

A central ability needed to solve daily tasks is complex reasoning. It involves the capacity to comprehend and represent the environment, retain information from past experiences, and solve problems based on the stored information. Our ability to solve those problems is supported by multiple specialized components, including short-term memory storage, long-term semantic and procedural memory, and an executive controller that, among others, controls the attention over memories (Baddeley, 1992).

Many promising advances for achieving complex reasoning with neural networks have been obtained during the last years. Unlike symbolic approaches to complex reasoning, deep neural networks can learn representations from perceptual information. Because of that, they do not suffer from the symbol grounding problem (Harnad, 1999), and can generalize better than classical symbolic approaches. Most of these neural network models make use of an explicit memory storage and an attention mechanism. For instance, Memory Networks (MemNN), Dynamic Memory Networks (DMN) or Neural Turing Machines (NTM) (Weston et al., 2014; Kumar et al., 2016; Graves et al., 2014) build explicit memories from the perceptual inputs and access these memories using learned attention mechanisms. After that some memories have been attended, using a multi-step procedure, the attended memories are combined and passed through a simple output layer that produces a final answer. While this allows some multi-step inferential process, these networks lack a more complex reasoning mechanism, needed for more elaborated tasks such as inferring relations among entities (relational reasoning). On the contrary, Relation Networks (RNs), proposed in Santoro et al. (2017), have shown outstanding performance in relational reasoning tasks. Nonetheless, a major drawback of RNs is that they consider each of the input objects in pairs, having to process a quadratic number of relations. That limits the usability of the model on large problems and makes forward and backward computations quite expensive. To solve these problems we propose a novel Memory Network

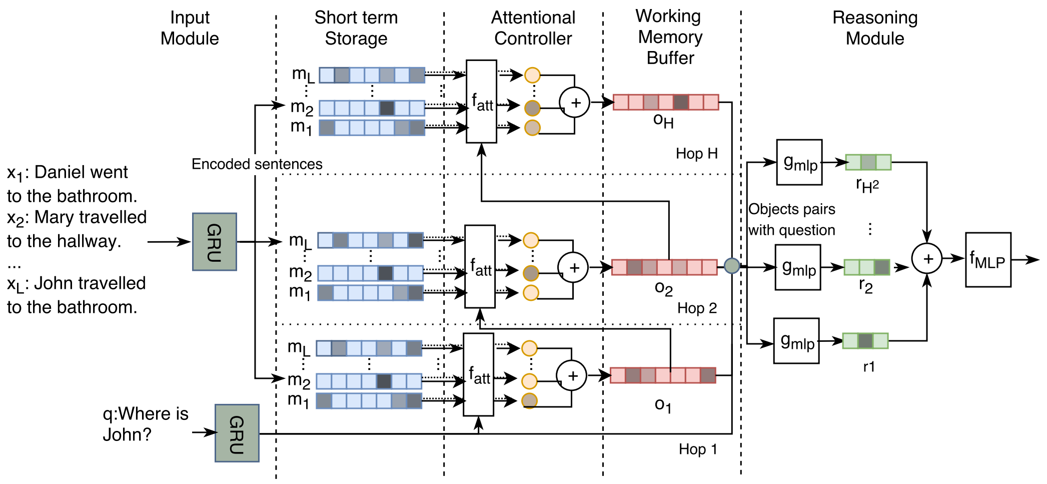


Figure 1: The W-MemNN model applied to textual question answering. Each input fact is processed using a GRU, and the output representation is stored in the short-term memory storage. Then, the attentional controller computes an output vector that summarizes relevant parts of the memories. This process is repeated H hops (a dotted line delimits each hop), and each output is stored in the working memory buffer. Finally, the output of each hop is passed to the reasoning module that produces the final output.

architecture called the Working Memory Network (W-MemNN). Our model augments the original MemNN with a relational reasoning module and a new working memory buffer.

The attention mechanism of the Memory Network allows the filtering of irrelevant inputs, reducing a lot of the computational complexity while keeping the relational reasoning capabilities of the RN. Three main components compose the W-MemNN: An input module that converts the perceptual inputs into an internal vector representation and save these representations into a short-term storage, an attentional controller that attend to these internal representations and update a working memory buffer, and a reasoning module that operates on the set of objects stored in the working memory buffer in order to produce a final answer. This component-based architecture is inspired by the well-known model from cognitive sciences called the multi-component working memory model, proposed in Baddeley and Hitch (1974).

We studied the proposed model on the text-based QA benchmark bAbI (Weston et al., 2015) which consists of 20 different toy tasks that measure different reasoning skills. While models such as EntNet (Henaff et al., 2016) have focused on the per-task training version of the benchmark (where a different model is trained for each task), we decided to focus on the jointly trained version of the task, where the model is trained on all tasks simultaneously. In the jointly trained bAbI-10k benchmark we achieved state-of-the-art performance, improving the previous state-of-the-art on more than 2%. Moreover, a simple ensemble of two of our models can solve all 20 tasks simultaneously. Also, we tested our model on the visual QA dataset NLVR. In that dataset, we obtained performance at the level of the Module Neural Networks (Andreas et al., 2016). Our model, however, achieves these results using the raw input statements, without the extra text processing used in the Module Networks.

Finally, qualitative and quantitative analysis shows that the inclusion of the Relational Reasoning module is crucial to improving the performance of the MemNN on tasks that involve relational reasoning. We can achieve this performance by also reducing the computation times of the RN considerably. Consequently, we hope that this contribution may allow applying RNs to larger problems.

2 Model

Our model is based on the Memory Network architecture. Unlike MemNN we have included a reasoning module that helps the network to solve more complex tasks. The proposed model consists of three main modules: An input module, an attentional controller, and a reasoning module. The model processes the input information in multiple passes or hops. At each pass the output of the previous hop can condition the current pass, allowing some incremental refinement.

Input module: The input module converts the perceptual information into an internal feature representation. The input information can be processed in chunks, and each chunk is saved into a short-term storage. The definition of what is a chunk of information depends on each task. For instance, for textual question answering, we define each chunk as a sentence. Other options might be n-grams or full documents. This short-term storage can only be accessed during the hop.

Attentional Controller: The attentional controller decides in which parts of the short-term storage the model should focus. The attended memories are kept during all the hops in a working memory buffer. The attentional controller is conditioned by the task at hand, for instance, in question answering the question can condition the attention. Also, it may be conditioned by the output of previous hops, allowing the model to change its focus to new portions of the memory over time. Many models compute the attention for each memory using a compatibility function between the memory and the question. Then, the output is calculated as the weighted sum of the memory values, using the attention as weight. A simple way to compute the attention for each memory is to use dot-product attention. This kind of mechanism is used in the original Memory Network and computes the attention value as the dot product between each memory and the question. Although this kind of attention is simple, it may not be enough for more complex tasks. Also, since there are no learned weights in the attention mechanism, the attention relies entirely on the learned embeddings. That is something that we want to avoid in order to separate the learning of the input and attention module. One way to allow learning in the dot-product attention is to project the memories and query vectors linearly. That is done by multiplying each vector by a learned projection matrix (or equivalently a feed-forward neural network). In this way, we can set apart the attention and input embeddings learning, and also allow more complex patterns of attention.

Reasoning Module: The memories stored in the working memory buffer are passed to the reasoning module. The choice of reasoning mechanism is left open and may depend on the task at hand. In this work, we use a Relation Network as the reasoning module. The RN takes the attended memories in pairs to infer relations among the memories. That can be useful, for example, in tasks that include comparisons.

A detailed description of the full model is shown in Figure 1.

2.1 W-MemN2N for Textual Question Answering

We proceed to describe an implementation of the model for textual question answering. In textual question answering the input consists of a set of sentences or facts, a question, and an answer. The goal is to answer the question correctly based on the given facts.

Let (s, q, a) represents an input sample, consisting of a set of sentences $s = \{x_i\}_{i=1}^{L}$, a query q and an answer a. Each sentence contains M words, $\{w_i\}_{i=1}^{M}$, where each word is represented as a one-hot vector of length $|V|$, being $|V|$ the vocabulary size. The question contains Q words, represented as in the input sentences.

Input Module

Each word in each sentence is encoded into a vector representation v_i using an embedding matrix $W \in \mathbb{R}^{|V|\times d}$, where d is the embedding size. Then, the sentence is converted into a memory vector m_i using the final output of a gated recurrent neural network (GRU) (Chung et al., 2014):

$$m_i = \text{GRU}([v_1, v_2, ..., v_M])$$

Each memory $\{m_i\}_{i=1}^{L}$, where $m_i \in \mathbb{R}^d$, is stored into the short-term memory storage. The question is encoded into a vector u in a similar way, using the output of a gated recurrent network.

Attentional Controller

Our attention module is based on the Multi-Head attention mechanism proposed in Vaswani et al. (2017). First, the memories are projected using a projection matrix $W_m \in \mathbb{R}^{d\times d}$, as $m'_i = W_m m_i$. Then, the similarity between the projected memory and the question is computed using the Scaled Dot-Product attention:

$$\alpha_i = \text{Softmax}\left(\frac{u^T m'_i}{\sqrt{d}}\right) \quad (1)$$

$$= \frac{\exp((u^T m'_i)/\sqrt{d})}{\sum_j \exp((u^T m'_j)/\sqrt{d})}. \quad (2)$$

Next, the memories are combined using the attention weights α_i, obtaining an output vector $h = \sum_j \alpha_j m_j$.
In the Multi-Head mechanism, the memories are projected S times using different projection matrices $\{W_m^s\}_{s=1}^S$. For each group of projected memories, an output vector $\{h_i\}_{i=1}^S$ is obtained using the Scaled Dot-Product attention (eq. 2). Finally, all vector outputs are concatenated and projected again using a different matrix:

$$o_k = [h_1; h_2; ...; h_S]W_o,$$

where ; is the concatenation operator and $W_o \in \mathbb{R}^{Sd \times d}$. The o_k vector is the final response vector for the hop k. This vector is stored in the working memory buffer. The attention procedure can be repeated many times (or hops). At each hop, the attention can be conditioned on the previous hop by replacing the question vector u by the output of the previous hop. To do that we pass the output through a simple neural network f_t. Then, we use the output of the network as the new conditioner:

$$o_k^n = f_t(o_k). \tag{3}$$

This network allows some learning in the transition patterns between hops.
We found Multi-Head attention to be very useful in the joint bAbI task. This can be a product of the intrinsic multi-task nature of the bAbI dataset. A possibility is that each attention head is being adapted for different groups of related tasks. However, we did not investigate this further.
Also, note that while in this section we use the same set of memories at each hop, this is not necessary. For larger sequences each hop can operate in different parts of the input sequence, allowing the processing of the input in various steps.

Reasoning Module

The outputs stored in the working memory buffer are passed to the reasoning module. The reasoning module used in this work is a Relation Network (RN). In the RN the output vectors are concatenated in pairs together with the question vector. Each pair is passed through a neural network g_θ and all the outputs of the network are added to produce a single vector. Then, the sum is passed to a final neural network f_ϕ:

$$r = f_\phi\left(\sum_{i,j} g_\theta([o_i; o_j; u])\right), \tag{4}$$

The output of the Relation Network is then passed through a final weight matrix and a softmax to produce the predicted answer:

$$\hat{a} = \text{Softmax}(Vr), \tag{5}$$

where $V \in \mathbb{R}^{|A| \times d_\phi}$, $|A|$ is the number of possible answers and d_ϕ is the dimension of the output of f_ϕ. The full network is trained end-to-end using standard cross-entropy between $\hat{a}$ and the true label a.

3 Related Work

3.1 Memory Augmented Neural Networks

During the last years, there has been plenty of work on achieving complex reasoning with deep neural networks. An important part of these developments has used some kind of explicit memory and attention mechanisms. One of the earliest recent work is that of Memory Networks (Weston et al., 2014). Memory Networks work by building an addressable memory from the inputs and then accessing those memories in a series of reading operations. Another, similar, line of work is the one of Neural Turing Machines. They were proposed in Graves et al. (2014) and are the basis for recent neural architectures including the Differentiable Neural Computer (DNC) and the Sparse Access Memory (SAM) (Graves et al., 2016; Rae et al., 2016). The NTM model also uses a content addressable memory, as in the Memory Network, but adds a write operation that allows updating the memory over time. The management of the memory, however, is different from the one of the MemNN. While the MemNN model pre-load the memories using all the inputs, the NTM writes and read the memory one input at a time.

An additional model that makes use of explicit external memory is the Dynamic Memory Network (DMN) (Kumar et al., 2016; Xiong et al., 2016). The model shares some similarities with the Memory Network model. However, unlike the MemNN model, it operates in the input sequentially (as in the NTM model). The model defines an Episodic Memory module that makes use of a Gated Recurrent Neural Network (GRU) to store and update an internal state that represents the episodic storage.

3.2 Memory Networks

Since our model is based on the MemNN architecture, we proceed to describe it in more detail. The

Memory Network model was introduced in Weston et al. (2014). In that work, the authors proposed a model composed of four components: The input feature map that converts the input into an internal vector representation, the generalization module that updates the memories given the input, the output feature map that produces a new output using the stored memories, and the response module that produces the final answer. The model, as initially proposed, needed some strong supervision that explicitly tells the model which memories to attend. In order to solve that limitation, the End-To-End Memory Network (MemN2N) was proposed in Sukhbaatar et al. (2015).

The model replaced the hard-attention mechanism used in the original MemNN by a soft-attention mechanism that allowed to train it end-to-end without strong supervision. In our model, we use a component-based approach, as in the original MemNN architecture. However, there are some differences: First, our model makes use of two external storages: a short-term storage, and a working memory buffer. The first is equivalent to the one updated by the input and generalization module of the MemNN. The working memory buffer, on the other hand, does not have a counterpart in the original model. Second, our model replaces the response module by a reasoning module. Unlike the original MemNN, our reasoning module is intended to make more complex work than the response module, that was only designed to produce a final answer.

3.3 Relation Networks

The ability to infer and learn relations between entities is fundamental to solve many complex reasoning problems. Recently, a number of neural network models have been proposed for this task. These include Interaction Networks, Graph Neural Networks, and Relation Networks (Battaglia et al., 2016; Scarselli et al., 2009; Santoro et al., 2017). In specific, Relation Networks (RNs) have shown excellent results in solving textual and visual question answering tasks requiring relational reasoning. The model is relatively simple: First, all the inputs are grouped in pairs and each pair is passed through a neural network. Then, the outputs of the first network are added, and another neural network processes the final vector. The role of the first network is to infer relations among each pair of objects. In Palm et al. (2017) the authors propose a recurrent extension to the RN. By allowing multiple steps of relational reasoning, the model can learn to solve more complex tasks. The main issue with the RN architecture is that its scale very poorly for larger problems. That is because it operates on $O(n^2)$ pairs, where n is the number of input objects (for instance, sentences in the case of textual question answering). This becomes quickly prohibitive for tasks involving many input objects.

3.4 Cognitive Science

The concept of working memory has been extensively developed in cognitive psychology. It consists of a limited capacity system that allows temporary storage and manipulation of information and is crucial to any reasoning task. One of the most influential models of working memory is the multi-component model of working memory proposed by Baddeley and Hitch (1974). This model is composed both of a supervisory attentional controller (the central executive) and two short-term storage systems: The phonological loop, capable of holding speech-based information, and the visuospatial sketchpad, concerned with visual storage. The central executive plays various functions, including the capacity to focus attention, to divide attention and to control access to long-term memory. Later modifications to the model (Baddeley, 2000) include an episodic buffer that is capable of integrating and holding information from different sources. Connections of the working memory model to memory augmented neural networks have been already studied in Graves et al. (2014). We follow this effort and subdivide our model into components that resemble (in a basic way) the multi-component model of working memory. Note, however, that we use the term working memory buffer instead of episodic buffer. That is because the episodic buffer has an integration function that our model does not cover. However, that can be an interesting source of inspiration for next versions of the model that integrate both visual and textual information for question answering.

4 Experiments

4.1 Textual Question Answering

To evaluate our model on textual question answering we used the Facebook bAbI-10k dataset (Weston et al., 2015). The bAbI dataset is a textual

	LSTM	MN-S	MN	SDNC	WMN	WMN†
1: 1 supporting fact	0.0	0.0	0.0	0.0	0.0	0.0
2: 2 supporting facts	81.9	0.0	1.0	0.6	0.7	0.3
3: 3 supporting facts	83.1	0.0	6.8	0.7	5.3	4.6
4: 2 argument relations	0.2	0.0	0.0	0.0	0.0	0.0
5: 3 argument relations	1.2	0.3	6.1	0.3	0.6	0.4
6: yes/no questions	51.8	0.0	0.1	0.0	0.0	0.0
7: counting	24.9	3.3	6.6	0.2	0.6	0.5
8: lists/sets	34.1	1.0	2.7	0.2	0.2	0.3
9: simple negation	20.2	0.0	0.0	0.0	0.0	0.0
10: indefinite knowledge	30.1	0.0	0.5	0.2	0.5	0.0
11: basic coreference	10.3	0.0	0.0	0.0	0.3	0.0
12: conjunction	23.4	0.0	0.1	0.1	0.0	0.0
13: compound coreference	6.1	0.0	0.0	0.1	0.0	0.0
14: time reasoning	81.0	0.0	0.0	0.1	0.0	0.0
15: basic deduction	78.7	0.0	0.2	0.0	0.0	0.0
16: basic induction	51.9	0.0	0.2	54.1	0.0	0.3
17: positional reasoning	50.1	24.6	41.8	0.3	0.3	0.1
18: size reasoning	6.8	2.1	8.0	0.1	0.1	0.4
19: path finding	90.3	31.9	75.7	1.2	0.6	0.0
20: agent's motivations	2.1	0.	0.0	0.0	0.0	0.0
Mean Error (%)	36.4	3.2	7.5	2.8	0.4	**0.3**
Failed tasks (err. $> 5\%$)	16	2	6	1	1	**0**

Table 1: Test accuracies on the jointly trained bAbI-10k dataset. `MN-S` stands for strongly supervised Memory Network, `MN-U` for end-to-end Memory Network without supervision, and `WMN` for Working Memory Network. Results for `LSTM`, `MN-U`, and `MN-S` are took from Sukhbaatar et al. (2015). Results for `SDNC` are took from Rae et al. (2016). `WMN`† is an ensemble of two Working Memory Networks.

QA benchmark composed of 20 different tasks. Each task is designed to test a different reasoning skill, such as deduction, induction, and coreference resolution. Some of the tasks need relational reasoning, for instance, to compare the size of different entities. Each sample is composed of a question, an answer, and a set of facts. There are two versions of the dataset, referring to different dataset sizes: bAbI-1k and bAbI-10k. In this work, we focus on the bAbI-10k version of the dataset which consists of $10,000$ training samples per task. A task is considered solved if a model achieves greater than 95% accuracy. Note that training can be done per-task or joint (by training the model on all tasks at the same time). Some models (Liu and Perez, 2017) have focused in the per-task training performance, including the EntNet model (Henaff et al., 2016) that solves all the tasks in the per-task training version. We choose to focus on the joint training version since we think is more indicative of the generalization properties of the model. A detailed analysis of the dataset can be found in Lee et al. (2015).

Model Details

To encode the input facts we used a word embedding that projected each word in a sentence into a real vector of size d. We defined $d = 30$ and used a GRU with 30 units to process each sentence. We used the 30 sentences in the support set that were immediately prior to the question. The question was processed using the same configuration but with a different GRU. We used 8 heads in the Multi-Head attention mechanism. For the transition networks f_t, which operates in the output of each hop, we used a two-layer MLP consisting of 15 and 30 hidden units (so the output preserves the memory dimension). We used $H = 4$ hops (or equivalently, a working memory buffer of size 4). In the reasoning module, we used a 3-layer MLP consisting of 128 units in each layer and with ReLU non-linearities for g_θ. We omitted the f_ϕ network since we did not observe improvements when using it. The final layer was a linear layer that produced logits for a softmax over the answer vocabulary.

Training Details

We trained our model end-to-end with a cross-entropy loss function and using the Adam optimizer (Kingma and Ba, 2014). We used a learning rate of $\nu = 1e^{-3}$. We trained the model during 400 epochs. For training, we used a batch size of 32. As in Sukhbaatar et al. (2015) we did not average the loss over a batch. Also, we clipped gradients with norm larger than 40 (Pascanu et al., 2013). For all the dense layers we used ℓ_2 regularization with value $1e^{-3}$. All weights were initialized using Glorot normal initialization (Glorot and Bengio, 2010). 10% of the training set was held-out to form a validation set that we used to select the architecture and for hyperparameter tunning. In some cases, we found useful to restart training after the 400 epochs with a smaller learning rate of $1e^{-5}$ and anneals every 5 epochs by $\nu/2$ until 20 epochs were reached.

bAbI-10k Results

On the jointly trained bAbI-10k dataset our best model (out of 10 runs) achieves an accuracy of **99.58%**. That is a **2.38%** improvement over the previous state-of-the-art that was obtained by the Sparse Differential Neural Computer (SDNC) (Rae et al., 2016). The best model of the 10 runs solves almost all tasks of the bAbI-10k dataset (by a 0.3% margin). However, a simple ensemble of the best two models solves all 20 tasks and achieves an almost perfect accuracy of **99.7%**. We list the results for each task in Table 1. Other authors have reported high variance in the results, for instance, the authors of the SDNC report a mean accuracy and standard deviation over 15 runs of 93.6 ± 2.5 (with 15.9 ± 1.6 passed tasks). In contrast, our model achieves a mean accuracy of 98.3 ± 1.2 (with 18.6 ± 0.4 passed tasks), which is better and more stable than the average results obtained by the SDNC.
The Relation Network solves 18/20 tasks. We achieve even better performance, and with considerably fewer computations, as is explained in Section 4.3. We think that by including the attention mechanism, the relation reasoning module can focus on learning the relation among relevant objects, instead of learning spurious relations among irrelevant objects. For that, the Multi-Head attention mechanism was very helpful.

The Effect of the Relational Reasoning Module

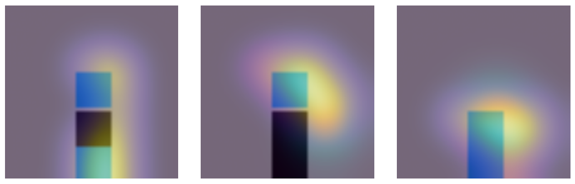
When compared to the original Memory Network, our model substantially improves the accuracy of tasks 17 (positional reasoning) and 19 (path finding). Both tasks require the analysis of multiple relations (Lee et al., 2015). For instance, the task 19 needs that the model reasons about the relation of different positions of the entities, and in that way find a path to arrive from one to another. The accuracy improves in **75.1%** for task 19 and in **41.5%** for task 17 when compared with the MemN2N model. Since both tasks require reasoning about relations, we hypothesize that the relational reasoning module of the W-MemNN was of great help to improve the performance on both tasks.
The Relation Network, on the other hand, fails in the tasks 2 (2 supporting facts) and 3 (3 supporting facts). Both tasks require handling a significant number of facts, especially in task 3. In those cases, the attention mechanism is crucial to filter out irrelevant facts.

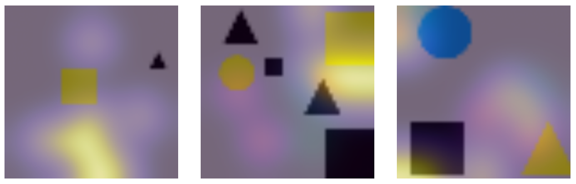
4.2 Visual Question Answering

To further study our model we evaluated its performance on a visual question answering dataset. For that, we used the recently proposed NLVR dataset (Suhr et al., 2017). Each sample in the NLVR dataset is composed of an image with three sub-images and a statement. The task consists in judging if the statement is true or false for that image. Evaluating the statement requires reasoning about the sets of objects in the image, comparing objects properties, and reasoning about spatial relations. The dataset is interesting for us for two reasons. First, the statements evaluation requires complex relational reasoning about the objects in the image. Second, unlike the bAbI dataset, the statements are written in natural language. Because of that, each statement displays a range of syntactic and semantic phenomena that are not present in the bAbI dataset.

Model details

Our model can be easily adapted to deal with visual information. Following the idea from Santoro et al. (2017), instead of processing each input using a recurrent neural network, we use a Convolutional Neural Network (CNN). The CNN takes as input each sub-image and convolved them through convolutional layers. The output of the CNN consists of k feature maps (where k is the number





Story (2 supporting facts)	Support	Hop 1	Hop 2	Hop 3	Hop 4
Mary moved to the office.		0.79	0.30	0.15	0.15
Sandra travelled to the bedroom.	True	0.02	2.64	2.75	0.39
Daniel dropped the football.		0.03	0.13	0.16	0.41
Sandra left the milk there.	True	1.01	0.07	0.16	0.38
Daniel grabbed the football there.		0.08	0.31	0.07	0.27
Question: Where is the milk? Answer: bedroom, Pred: bedroom					

Story (2 supporting facts)	Support	Hop 1	Hop 2	Hop 3	Hop 4
Brian is white.		0.46	0.36	0.35	0.89
Bernhard is white.		0.07	0.13	0.19	0.81
Julius is a frog.	True	0.16	2.03	0.39	0.26
Julius is white.	True	0.09	0.23	2.42	1.32
Greg is a frog.	True	1.95	1.60	0.77	0.25
Question: What color is greg? Answer: white, Pred: white					

Table 2: Examples of visualizations of attention for textual and visual QA. Top: Visualization of attention values for the NLVR dataset. To get more aesthetic figures we applied a gaussian blur to the attention matrix. Bottom: Attention values for the bAbI dataset. In each cell, the sum of the attention for all heads is shown.

of kernels in the final convolutional layer) of size $d \times d$. Then, each memory is built from the vector composed by the concatenation of the cells in the same position of each feature map. Consequently, $d \times d$ memories of size k are stored in the short-term storage. The statement is processed using a GRU neural network as in the textual reasoning task. Then, we can proceed using the same architecture for the reasoning and attention module that the one used in the textual QA model. However, for the visual QA task, we used an additive attention mechanism. The additive attention computes the attention weight using a feed-forward neural network applied to the concatenation of the memory vector and statement vector.

Results

Our model achieves a validation / test accuracy of $65.6\%/65.8\%$. Notably, we achieved a performance comparable to the results of the Module Neural Networks (Andreas et al., 2016) that make use of standard NLP tools to process the statements into structured representations. Unlike the Module Neural Networks, we achieved our results using only raw input statements, allowing the model to learn how to process the textual input by itself. Note that given the more complex nature of the language used in the NLVR dataset we needed to use a larger embedding size and GRU hidden layer than in the bAbI dataset (100 and 128 respectively). That, however, is a nice feature of separating the input from the reasoning and attention component: One way to process more complex language statements is increasing the capacity of the input module.

4.3 From $O(n^2)$ to $O(n)$

One of the major limitations of RNs is that they need to process each one of the memories in pairs. To do that, the RN must perform $O(n^2)$ forward and backward passes (where n is the number of memories). That becomes quickly prohibitive for a larger number of memories. In contrast, the dependence of the W-MemNN run times on the number of memories is linear. Note, however, that computation times in the W-MemNN depend quadratically on the size of the working memory buffer. Nonetheless, this number is expected to be much smaller than the number of memories. To compare both models we measured the wall-clock time for a forward and backward pass for a single batch of size 32. We performed these experiments on a GPU NVIDIA K80. Figure 2 shows the results.

4.4 Memory Visualizations

One nice feature from Memory Networks is that they allow some interpretability of the reasoning procedure by looking at the attention weights. At each hop, the attention weights show which parts of the memory the model found relevant to produce the output. RNs, on the contrary, lack of this feature. Table 2 shows the attention values for visual and textual question answering.

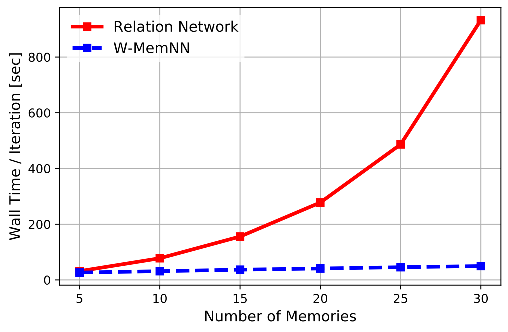


Figure 2: Wall-clock times for a forward and backward pass for a single batch. The batch size used is 32. While for 5 memories the times are comparable, for 30 memories the W-MemNN takes around 50s while the RN takes 930s, a speedup of almost 20×.

5 Conclusion

We have proposed a novel Working Memory Network architecture that introduces improved reasoning abilities to the original MemNN model. We demonstrated that by augmenting the MemNN architecture with a Relation Network, the computational complexity of the RN can be reduced, without loss of performance. That opens the opportunity for using RNs in larger problems, something that may be very useful, given the many tasks requiring a significant amount of memories.

Although we have used RN as the reasoning module in this work, other options can be tested. It might be interesting to analyze how other reasoning modules can improve different weaknesses of the model.

We presented results on the jointly trained bAbI-10k dataset, where we achieve a new state-of-the-art, with an average error of less than 0.5%. Also, we showed that our model can be easily adapted for visual question answering.

Our architecture combines perceptual input processing, short-term memory storage, an attention mechanism, and a reasoning module. While other models have focused on different parts of these components, we think that is important to find ways to combine these different mechanisms if we want to build models capable of complex reasoning. Evidence from cognitive sciences seems to show that all these abilities are needed in order to achieve human-level complex reasoning.

Acknowledgments

JP was supported by the Scientific and Technological Center of Valparaíso (CCTVal) under Fondecyt grant BASAL FB0821. HA was supported through the research project Fondecyt-Conicyt 1170123. The work of HAC was supported by the research project Fondecyt Initiation into Research 11150248.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 39–48.

Alan Baddeley. 1992. Working memory. *Science* 255(5044):556–559.

Alan Baddeley. 2000. The episodic buffer: a new component of working memory? *Trends in cognitive sciences* 4(11):417–423.

Alan D Baddeley and Graham Hitch. 1974. Working memory. In *Psychology of learning and motivation*, Elsevier, volume 8, pages 47–89.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*. pages 4502–4510.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pages 249–256.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471.

Stevan Harnad. 1999. The symbol grounding problem. *CoRR* cs.AI/9906002. http://arxiv.org/abs/cs.AI/9906002.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2015. Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426* .

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. volume 1, pages 1–10.

Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. 2017. Recurrent relational networks for complex relational reasoning. *CoRR* abs/1711.08028. http://arxiv.org/abs/1711.08028.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. pages 1310–1318.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. 2016. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*. pages 3621–3629.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*. pages 4974–4983.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1):61–80.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 217–223. https://doi.org/10.18653/v1/P17-2034.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. https://arxiv.org/pdf/1706.03762.pdf.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916. http://arxiv.org/abs/1410.3916.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. pages 2397–2406.

Reasoning with Sarcasm by Reading In-between

Yi Tay[†], **Luu Anh Tuan**[ψ], **Siu Cheung Hui**[ϕ], **Jian Su**[δ]
[†]ytay017@e.ntu.edu.sg
[ψ]at.luu@i2r.a-star.edu.sg
[ϕ]asschui@ntu.edu.sg
[δ]sujian@i2r.a-star.edu.sg
[†,ϕ]School of Computer Science and Engineering, Nanyang Technological University
[ψ,δ]A*Star, Institute for Infocomm Research, Singapore

Abstract

Sarcasm is a sophisticated speech act which commonly manifests on social communities such as Twitter and Reddit. The prevalence of sarcasm on the social web is highly disruptive to opinion mining systems due to not only its tendency of polarity flipping but also usage of figurative language. Sarcasm commonly manifests with a contrastive theme either between positive-negative sentiments or between literal-figurative scenarios. In this paper, we revisit the notion of modeling *contrast* in order to reason with sarcasm. More specifically, we propose an attention-based neural model that looks *in-between* instead of *across*, enabling it to explicitly model contrast and incongruity. We conduct extensive experiments on six benchmark datasets from Twitter, Reddit and the Internet Argument Corpus. Our proposed model not only achieves state-of-the-art performance on all datasets but also enjoys improved interpretability.

1 Introduction

Sarcasm, commonly defined as *'An ironical taunt used to express contempt'*, is a challenging NLP problem due to its highly figurative nature. The usage of sarcasm on the social web is prevalent and can be frequently observed in reviews, microblogs (*tweets*) and online forums. As such, the battle against sarcasm is also regularly cited as one of the key challenges in sentiment analysis and opinion mining applications (Pang et al., 2008). Hence, it is both imperative and intuitive that effective sarcasm detectors can bring about numerous benefits to opinion mining applications.

Sarcasm is often associated to several linguistic phenomena such as (1) an explicit contrast between sentiments or (2) disparity between the conveyed emotion and the author's situation (context). Prior work has considered sarcasm to be a contrast between a positive and negative sentiment (Riloff et al., 2013). Consider the following examples:

1. I absolutely *love* to be *ignored*!
2. Yay!!! The best thing to wake up to is my neighbor's drilling.
3. Perfect movie for people who can't fall asleep.

Given the examples, we make a crucial observation - Sarcasm relies a lot on the semantic relationships (and contrast) between individual words and phrases in a sentence. For instance, the relationships between phrases {*love*, *ignored*}, {*best*, *drilling*} and {*movie*, *asleep*} (in the examples above) richly characterize the nature of sarcasm conveyed, i.e., word pairs tend to be contradictory and more often than not, express a juxtaposition of positive and negative terms. This concept is also explored in (Joshi et al., 2015) in which the authors refer to this phenomena as *'incongruity'*. Hence, it would be useful to capture the relationships between selected word pairs in a sentence, i.e., *looking in-between*.

State-of-the-art sarcasm detection systems mainly rely on deep and *sequential* neural networks (Ghosh and Veale, 2016; Zhang et al., 2016). In these works, compositional encoders such as gated recurrent units (GRU) (Cho et al., 2014) or long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) are often employed, with the input document being parsed one word at a time. This has several shortcomings for the sarcasm detection task. Firstly, there is no explicit interaction between word pairs, which hampers its ability to explicitly model contrast, incongruity or juxtaposition of situations. Secondly, it is difficult to capture long-range dependencies. In this case, contrastive situations (or sentiments) which are commonplace in sarcastic language may be hard to detect with simple sequential models.

To overcome the weaknesses of standard sequential models such as recurrent neural networks, our work is based on the intuition that modeling intra-sentence relationships can not only improve classification performance but also pave the way for more explainable neural sarcasm detection methods. In other words, our key intuition manifests itself in the form of an attention-based neural network. While the key idea of most neural attention mechanisms is to focus on relevant words and sub-phrases, it merely looks *across* and does not explicitly capture word-word relationships. Hence, it suffers from the same shortcomings as sequential models.

In this paper, our aim is to combine the effectiveness of state-of-the-art recurrent models while harnessing the intuition of *looking in-between*. We propose a multi-dimensional intra-attention recurrent network that models intricate similarities between each word pair in the sentence. In other words, our novel deep learning model aims to capture *'contrast'* (Riloff et al., 2013) and *'incongruity'* (Joshi et al., 2015) within end-to-end neural networks. Our model can be thought of self-targeted co-attention (Xiong et al., 2016), which allows our model to not only capture word-word relationships but also long-range dependencies. Finally, we show that our model produces interpretable attention maps which aid in the explainability of model outputs. To the best of our knowledge, our model is the first attention model that can produce explainable results in the sarcasm detection task.

Briefly, the prime contributions of this work can be summarized as follows:

- We propose a new state-of-the-art method for sarcasm detection. Our proposed model, the Multi-dimensional Intra-Attention Recurrent Network (MIARN) is strongly based on the intuition of compositional learning by leveraging intra-sentence relationships. To the best of our knowledge, none of the existing state-of-the-art models considered exploiting intra-sentence relationships, solely relying on sequential composition.
- We conduct extensive experiments on multiple benchmarks from Twitter, Reddit and the Internet Argument Corpus. Our proposed MIARN achieves highly competitive performance on all benchmarks, outperforming existing state-of-the-art models such as GRNN (Zhang et al., 2016) and CNN-LSTM-DNN (Ghosh and Veale, 2016).

2 Related Work

Sarcasm is a complex linguistic phenomena that have long fascinated both linguists and NLP researchers. After all, a better computational understanding of this complicated speech act could potentially bring about numerous benefits to existing opinion mining applications. Across the rich history of research on sarcasm, several theories such as the Situational Disparity Theory (Wilson, 2006) and the Negation Theory (Giora, 1995) have emerged. In these theories, a common theme is a motif that is strongly grounded in contrast, whether in sentiment, intention, situation or context. (Riloff et al., 2013) propagates this premise forward, presenting an algorithm strongly based on the intuition that sarcasm arises from a juxtaposition of positive and negative situations.

2.1 Sarcasm Detection

Naturally, many works in this area have treated the sarcasm detection task as a standard text classification problem. An extremely comprehensive overview can be found at (Joshi et al., 2017). Feature engineering approaches were highly popular, exploiting a wide diverse range of features such as syntactic patterns (Tsur et al., 2010), sentiment lexicons (González-Ibánez et al., 2011), n-gram (Reyes et al., 2013), word frequency (Barbieri et al., 2014), word shape and pointedness features (Ptáček et al., 2014), readability and flips (Rajadesingan et al., 2015), etc. Notably, there have been quite a reasonable number of works that propose features based on similarity and contrast. (Hernández-Farías et al., 2015) measured the Wordnet based semantic similarity between words. (Joshi et al., 2015) proposed a framework based on explicit and implicit incongruity, utilizing features based on positive-negative patterns. (Joshi et al., 2016) proposed similarity features based on word embeddings.

2.2 Deep Learning for Sarcasm Detection

Deep learning based methods have recently garnered considerable interest in many areas of NLP research. In our problem domain, (Zhang et al., 2016) proposed a recurrent-based model with a gated pooling mechanism for sarcasm detection on Twitter. (Ghosh and Veale, 2016) proposed a convolutional long-short-term memory network (CNN-LSTM-DNN) that achieves state-of-the-art performance.

While our work focuses on document-only sarcasm detection, several notable works have proposed models that exploit personality information (Ghosh and Veale, 2017) and user context (Amir et al., 2016). Novel methods for sarcasm detection such as gaze / cognitive features (Mishra et al., 2016, 2017) have also been explored. (Peled and Reichart, 2017) proposed a novel framework based on neural machine translation to convert a sequence from sarcastic to non-sarcastic. (Felbo et al., 2017) proposed a layer-wise training scheme that utilizes emoji-based distant supervision for sentiment analysis and sarcasm detection tasks.

2.3 Attention Models for NLP

In the context of NLP, the key idea of neural attention is to soft select a sequence of words based on their relative importance to the task at hand. Early innovations in attentional paradigms mainly involve neural machine translation (Luong et al., 2015; Bahdanau et al., 2014) for aligning sequence pairs. Attention is also commonplace in many NLP applications such as sentiment classification (Chen et al., 2016; Yang et al., 2016), aspect-level sentiment analysis (Tay et al., 2018s, 2017b; Chen et al., 2017) and entailment classification (Rocktäschel et al., 2015). Co-attention / Bi-Attention (Xiong et al., 2016; Seo et al., 2016) is a form of pairwise attention mechanism that was proposed to model query-document pairs. Intra-attention can be interpreted as a self-targetted co-attention and is seeing a lot promising results in many recent works (Vaswani et al., 2017; Parikh et al., 2016; Tay et al., 2017a; Shen et al., 2017). The key idea is to model a sequence against itself, learning to attend while capturing long term dependencies and word-word level interactions. To the best of our knowledge, our work is not only the first work that only applies intra-attention to sarcasm detection but also the first attention model for sarcasm detection.

3 Our Proposed Approach

In this section, we describe our proposed model. Figure 1 illustrates our overall model architecture.

3.1 Input Encoding Layer

Our model accepts a sequence of one-hot encoded vectors as an input. Each one-hot encoded vector corresponds to a single word in the vocabulary. In the input encoding layer, each one-hot vector is converted into a low-dimensional vector representation (word embedding). The word embeddings are parameterized by an embedding layer $\mathbf{W} \in \mathbb{R}^{n \times |V|}$. As such, the output of this layer is a sequence of word embeddings, i.e., $\{w_1, w_2, \cdots w_\ell\}$ where ℓ is a predefined maximum sequence length.

3.2 Multi-dimensional Intra-Attention

In this section, we describe our multi-dimensional intra-attention mechanism for sarcasm detection. We first begin by describing the standard single-dimensional intra-attention. The multi-dimensional adaptation will be introduced later in this section. The key idea behind this layer is to *look in-between*, i.e., modeling the semantics between each word in the input sequence. We first begin by modeling the relationship of each word pair in the input sequence. A simple way to achieve this is to use a linear[1] transformation layer to project the concatenation of each word embedding **pair** into a scalar score as follows:

$$s_{ij} = W_a([w_i; w_j]) + b_a \tag{1}$$

where $W_a \in \mathbb{R}^{2n \times 1}, b_a \in \mathbb{R}$ are the parameters of this layer. $[.;.]$ is the vector concatenation operator and s_{ij} is a scalar representing the affinity score between word pairs (w_i, w_j). We can easily observe that s is a symmetrical matrix of $\ell \times \ell$ dimensions. In order to learn attention vector a, we apply a row-wise max-pooling operator on matrix s.

$$a = softmax(\max_{row} s) \tag{2}$$

where $a \in \mathbb{R}^\ell$ is a vector representing the learned intra-attention weights. Then, the vector a is employed to learn weighted representation of $\{w_1, w_2 \cdots w_\ell\}$ as follows:

$$v_a = \sum_{i=1}^{\ell} w_i a_i \tag{3}$$

[1]Early experiments found that adding nonlinearity here may degrade performance.

where $v \in \mathbb{R}^n$ is the intra-attentive representation of the input sequence. While other choices of pooling operators may be also employed (e.g., mean-pooling over max-pooling), the choice of max-pooling is empirically motivated. Intuitively, this attention layer learns to pay attention based on a word's *largest* contribution to all words in the sequence. Since our objective is to highlight words that might contribute to the contrastive theories of sarcasm, a more discriminative pooling operator is desirable. Notably, we also mask values of s where $i = j$ such that we do not allow the relationship scores of a word with respect to itself to influence the overall attention weights.

Furthermore, our network can be considered as an *'inner'* adaptation of neural attention, modeling intra-sentence relationships between the raw word representations instead of representations that have been compositionally manipulated. This allows word-to-word similarity to be modeled *'as it is'* and not be influenced by composition. For example, when using the outputs of a compositional encoder (e.g., LSTM), matching words n and $n + 1$ might not be meaningful since they would be relatively similar in terms of semantic composition. For relatively short documents (such as tweets), it is also intuitive that attention typically focuses on the last hidden representation.

Intuitively, the relationships between two words is often not straightforward. Words are complex and often hold more than one meanings (or word senses). As such, it might be beneficial to model *multiple views* between two words. This can be modeled by representing the word pair interaction with a vector instead of a scalar. As such, we propose a multi-dimensional adaptation of the intra-attention mechanism. The key idea here is that each word pair is projected down to a low-dimensional vector before we compute the affinity score, which allows it to not only capture one view (one scalar) but also multiple views. A modification to Equation (1) constitutes our Multi-Dimensional Intra-Attention variant.

$$s_{ij} = W_p(ReLU(W_q([w_i; w_j]) + b_q)) + b_p \quad (4)$$

where $W_q \in \mathbb{R}^{n \times k}, W_p \in \mathbb{R}^{k \times 1}, b_q \in \mathbb{R}^k, b_p \in \mathbb{R}$ are the parameters of this layer. The final intra-attentive representation is then learned with Equation (2) and Equation (3) which we do not repeat here for the sake of brevity.

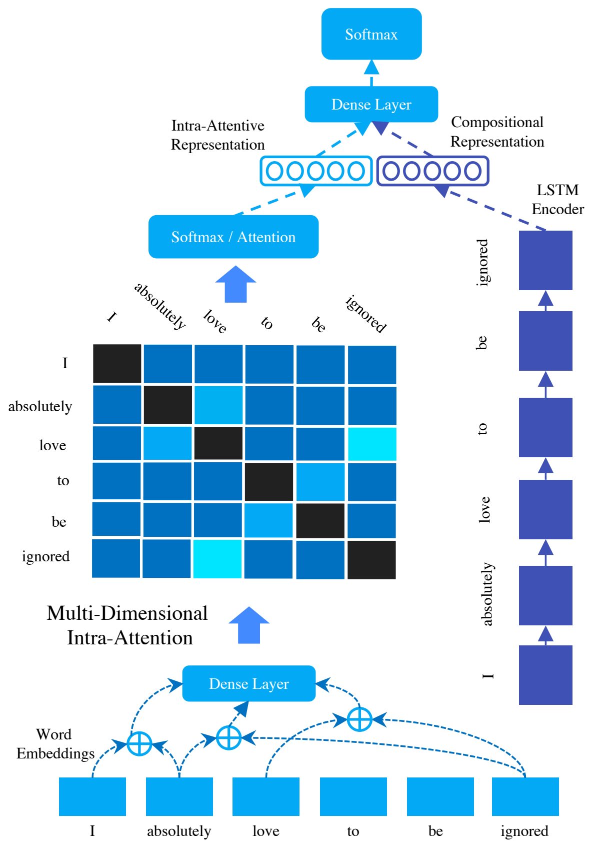


Figure 1: High level overview of our proposed MIARN architecture. MIARN learns two representations, one based on intra-sentence relationships (intra-attentive) and another based on sequential composition (LSTM). Both views are used for prediction.

3.3 Long Short-Term Memory Encoder

While we are able to simply use the learned representation v for prediction, it is clear that v does not encode compositional information and may miss out on important compositional phrases such as *'not happy'*. Clearly, our intra-attention mechanism simply considers a word-by-word interaction and does not model the input document sequentially. As such, it is beneficial to use a separate compositional encoder for this purpose, i.e., learning compositional representations. To this end, we employ the standard Long Short-Term Memory (LSTM) encoder. The output of an LSTM encoder at each time-step can be briefly defined as:

$$h_i = \text{LSTM}(w, i), \quad \forall i \in [1, \dots \ell] \quad (5)$$

where ℓ represents the maximum length of the sequence and $h_i \in \mathbb{R}^d$ is the hidden output of the LSTM encoder at time-step i. d is the size of the hidden units of the LSTM encoder. LSTM encoders are parameterized by gating mechanisms learned via nonlinear transformations. Since

LSTMs are commonplace in standard NLP applications, we omit the technical details for the sake of brevity. Finally, to obtain a compositional representation of the input document, we use $v_c = h_\ell$ which is the last hidden output of the LSTM encoder. Note that the inputs to the LSTM encoder are the word embeddings right after the input encoding layer and not the output of the intra-attention layer. We found that applying an LSTM on the intra-attentively scaled representations do not yield any benefits.

3.4 Prediction Layer

The inputs to the final prediction layer are two representations, namely (1) the intra-attentive representation ($v_a \in \mathbb{R}^n$) and (2) the compositional representation ($v_c \in \mathbb{R}^d$). This layer learns a joint representation of these two views using a nonlinear projection layer.

$$v = ReLU(W_z([v_a; v_c]) + b_z) \tag{6}$$

where $W_z \in \mathbb{R}^{(d+n)\times d}$ and $b_z \in \mathbb{R}^d$. Finally, we pass v into a Softmax classification layer.

$$\hat{y} = Softmax(W_f\, v + b_f) \tag{7}$$

where $W_f \in \mathbb{R}^{d\times 2}, b_f \in \mathbb{R}^2$ are the parameters of this layer. $\hat{y} \in \mathbb{R}^2$ is the output layer of our proposed model.

3.5 Optimization and Learning

Our network is trained end-to-end, optimizing the standard binary cross-entropy loss function.

$$J = -\sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)] + R \tag{8}$$

where J is the cost function, $\hat{y}$ is the output of the network, $R = ||\theta||_{L2}$ is the L2 regularization and λ is the weight of the regularizer.

4 Empirical Evaluation

In this section, we describe our experimental setup and results. Our experiments were designed to answer the following research questions (**RQ**s).

- **RQ1** - Does our proposed approach outperform existing state-of-the-art models?
- **RQ2** - What are the impacts of some of the architectural choices of our model? How much does intra-attention contribute to the model performance? Is the Multi-Dimensional adaptation better than the Single-Dimensional adaptation?
- **RQ3** - What can we interpret from the intra-attention layers? Does this align with our hypothesis about *looking in-between* and modeling contrast?

4.1 Datasets

We conduct our experiments on six publicly available benchmark datasets which span across three well-known sources.

- **Tweets** - Twitter[2] is a microblogging platform which allows users to post statuses of less than 140 characters. We use two collections for sarcasm detection on tweets. More specifically, we use the dataset obtained from (1) (Ptáček et al., 2014) in which tweets are trained via hashtag based semi-supervised learning, i.e., hashtags such as #not, #sarcasm and #irony are marked as sarcastic tweets and (2) (Riloff et al., 2013) in which Tweets are hand annotated and manually checked for sarcasm. For both datasets, we retrieve. Tweets using the Twitter API using the provided tweet IDs.
- **Reddit** - Reddit[3] is a highly popular social forum and community. Similar to Tweets, sarcastic posts are obtained via the tag '/s' which are marked by the authors themselves. We use two Reddit datasets which are obtained from the subreddits */r/movies* and */r/technology* respectively. Datasets are subsets from (Khodak et al., 2017).
- **Debates** - We use two datasets[4] from the Internet Argument Corpus (IAC) (Lukin and Walker, 2017) which have been hand annotated for sarcasm. This dataset, unlike the first two, is mainly concerned with long text and provides a diverse comparison from the other datasets. The IAC corpus was designed for research on political debates on online forums. We use the V1 and V2 versions of the sarcasm corpus which are denoted as IAC-V1 and IAC-V2 respectively.

The statistics of the datasets used in our experiments is reported in Table 1.

[2]https://twitter.com
[3]https://reddit.com
[4]https://nlds.soe.ucsc.edu/sarcasm1

Dataset	Train	Dev	Test	Avg ℓ
Tweets (Ptáček et al.)	44017	5521	5467	18
Tweets (Riloff et al.)	1369	195	390	14
Reddit (/r/movies)	5895	655	1638	12
Reddit (/r/technology)	16146	1793	4571	11
Debates IAC-V1	3716	464	466	54
Debates IAC-V2	1549	193	193	64

Table 1: Statistics of datasets used in our experiments.

4.2 Compared Methods

We compare our proposed model with the following algorithms.

- **NBOW** is a simple neural bag-of-words baseline that sums all the word embeddings and passes the summed vector into a simple logistic regression layer.
- **CNN** is a vanilla Convolutional Neural Network with max-pooling. CNNs are considered as compositional encoders that capture n-gram features by parameterized sliding windows. The filter width is 3 and number of filters $f = 100$.
- **LSTM** is a vanilla Long Short-Term Memory Network. The size of the LSTM cell is set to $d = 100$.
- **ATT-LSTM (Attention-based LSTM)** is a LSTM model with a neural attention mechanism applied to all the LSTM hidden outputs. We use a similar adaptation to (Yang et al., 2016), albeit only at the document-level.
- **GRNN (Gated Recurrent Neural Network)** is a Bidirectional Gated Recurrent Unit (GRU) model that was proposed for sarcasm detection by (Zhang et al., 2016). GRNN uses a gated pooling mechanism to aggregate the hidden representations from a standard BiGRU model. Since we only compare on document-level sarcasm detection, we do not use the variant of GRNN that exploits user context.
- **CNN-LSTM-DNN (Convolutional LSTM + Deep Neural Network)**, proposed by (Ghosh and Veale, 2016), is the state-of-the-art model for sarcasm detection. This model is a combination of a CNN, LSTM and Deep Neural Network via stacking. It stacks two layers of 1D convolution with 2 LSTM layers. The output passes through a deep neural network (DNN) for prediction.

Both CNN-LSTM-DNN (Ghosh and Veale, 2016) and GRNN (Zhang et al., 2016) are state-of-the-art models for document-level sarcasm detection and have outperformed numerous neural and non-neural baselines. In particular, both works have well surpassed feature-based models (Support Vector Machines, etc.), as such we omit comparisons for the sake of brevity and focus comparisons with recent neural models instead. Moreover, since our work focuses only on document-level sarcasm detection, we do not compare against models that use external information such as user profiles, context, personality information (Ghosh and Veale, 2017) or emoji-based distant supervision (Felbo et al., 2017).

For our model, we report results on both multi-dimensional and single-dimensional intra-attention. The two models are named as MIARN and SIARN respectively.

4.3 Implementation Details and Metrics

We adopt standard the evaluation metrics for the sarcasm detection task, i.e., macro-averaged F1 and accuracy score. Additionally, we also report precision and recall scores. All deep learning models are implemented using TensorFlow (Abadi et al., 2015) and optimized on a NVIDIA GTX1070 GPU. Text is preprocessed with NLTK[5]'s Tweet tokenizer. Words that only appear once in the entire corpus are removed and marked with the UNK token. Document lengths are truncated at $40, 20, 80$ tokens for Twitter, Reddit and Debates dataset respectively. Mentions of other users on the Twitter dataset are replaced by '@USER'. Documents with URLs (i.e., containing 'http') are removed from the corpus. Documents with less than 5 tokens are also removed. The learning optimizer used is the RMSProp with an initial learning rate of 0.001. The L2 regularization is set to 10^{-8}. We initialize the word embedding layer with GloVe (Pennington et al., 2014). We use the GloVe model trained on 2B Tweets for the Tweets and Reddit dataset. The Glove model trained on Common Crawl is used for the Debates corpus. The size of the word embeddings is fixed at $d = 100$ and are fine-tuned during training. In all experiments, we use a development set to select the best hyperparameters. Each model is trained for a total of 30 epochs and the model is saved each time the performance

[5]https://nltk.org

	Tweets (Ptáček et al., 2014)				Tweets (Riloff et al., 2013)			
Model	P	R	F1	Acc	P	R	F1	Acc
NBOW	80.02	79.06	79.43	80.39	*71.28*	62.37	64.13	79.23
Vanilla CNN	82.13	79.67	80.39	81.65	71.04	67.13	68.55	*79.48*
Vanilla LSTM	*84.62*	83.21	83.67	*84.50*	67.33	67.20	67.27	76.27
Attention LSTM	84.16	*85.10*	83.67	84.40	68.78	*68.63*	*68.71*	77.69
GRNN (Zhang et al.)	84.06	83.02	83.43	84.20	66.32	64.74	65.40	76.41
CNN-LSTM-DNN (Ghosh and Veale)	84.06	83.45	*83.74*	84.39	69.76	66.62	67.81	78.72
SIARN (this paper)	85.02	84.27	84.59	85.24	**73.82**	**73.26**	**73.24**	**82.31**
MIARN (this paper)	**86.13**	**85.79**	**86.00**	**86.47**	73.34	68.34	70.10	80.77

Table 2: Experimental Results on Tweets datasets. Best result in is boldface and second best is underlined. Best performing baseline is in *italics*.

	Reddit (/r/movies)				Reddit (/r/technology)			
Model	P	R	F1	Acc	P	R	F1	Acc
NBOW	67.33	66.56	66.82	67.52	65.45	65.62	65.52	66.55
Vanilla CNN	65.97	65.97	65.97	66.24	65.88	62.90	62.85	66.80
Vanilla LSTM	67.57	67.67	67.32	67.34	66.94	67.22	67.03	*67.92*
Attention LSTM	68.11	67.87	67.94	68.37	*68.20*	*68.78*	*67.44*	67.22
GRNN (Zhang et al.)	66.16	66.16	66.16	66.42	66.56	66.73	66.66	67.65
CNN-LSTM-DNN (Ghosh and Veale)	*68.27*	*67.87*	*67.95*	*68.50*	66.14	66.73	65.74	66.00
SIARN (this paper)	69.59	**69.48**	69.52	69.84	**69.35**	**70.05**	**69.22**	69.57
MIARN (this paper)	**69.68**	69.37	**69.54**	**69.90**	68.97	69.30	69.09	**69.91**

Table 3: Experimental results on Reddit datasets. Best result in is boldface and second best is underlined. Best performing baseline is in *italics*.

	Debates (IAC-V1)				Debates (IAC-V2)			
Model	P	R	F1	Acc	P	R	F1	Acc
NBOW	57.17	57.03	57.00	57.51	66.01	66.03	66.02	66.09
Vanilla CNN	58.21	58.00	57.95	58.55	68.45	68.18	68.21	68.56
Vanilla LSTM	54.87	54.89	54.84	54.92	68.30	63.96	60.78	62.66
Attention LSTM	58.98	57.93	57.23	59.07	70.04	69.62	*69.63*	*69.96*
GRNN (Zhang et al.)	56.21	56.21	55.96	55.96	62.26	61.87	61.21	61.37
CNN-LSTM-DNN (Ghosh and Veale)	55.50	54.60	53.31	55.96	64.31	64.33	64.31	64.38
SIARN (this paper)	**63.94**	63.45	62.52	62.69	72.17	71.81	71.85	72.10
MIARN (this paper)	63.88	**63.71**	**63.18**	**63.21**	**72.92**	**72.93**	**72.75**	**72.75**

Table 4: Experimental results on Debates datasets. Best result in is boldface and second best is underlined. Best performing baseline is in *italics*.

on the development set is topped. The batch size is tuned amongst $\{128, 256, 512\}$ for all datasets. The only exception is the Tweets dataset from (Riloff et al., 2013), in which a batch size of 16 is used in lieu of the much smaller dataset size. For fair comparison, all models have the same hidden representation size and are set to 100 for both recurrent and convolutional based models (i.e., number of filters). For MIARN, the size of intra-attention hidden representation is tuned amongst $\{4, 8, 10, 20\}$.

4.4 Experimental Results

Table 2, Table 3 and Table 4 reports a performance comparison of all benchmarked models on the Tweets, Reddit and Debates datasets respectively. We observe that our proposed SIARN and MIARN models achieve the best results across all six datasets. The relative improvement differs across domain and datasets. On the Tweets dataset from (Ptáček et al., 2014), MIARN achieves about $\approx 2\% - 2.2\%$ improvement in terms of F1 and accuracy score when compared against the best baseline. On the other Tweets dataset from (Riloff et al., 2013), the performance gain of our proposed model is larger, i.e., $3\% - 5\%$ improvement on average over most baselines. Our proposed SIARN and MIARN models achieve very competitive performance on the Reddit datasets, with an average of $\approx 2\%$ margin improvement over the best baselines. Notably, the baselines we compare against are extremely competitive state-of-the-art neural network models. This further reinforces the effectiveness of our proposed approach. Additionally, the performance improvement on Debates (long text) is significantly larger than short text (i.e., Twitter and Reddit). For example, MIARN outperforms GRNN and CNN-LSTM-DNN by $\approx 8\% - 10\%$ on both IAC-V1 and IAC-V2. At this note, we can safely put **RQ1** to rest.

Overall, the performance of MIARN is often marginally better than SIARN (with some exceptions, e.g., *Tweets* dataset from (Riloff et al., 2013)). We believe that this is attributed to the fact that more complex word-word relationships can be learned by using multi-dimensional values instead of single-dimensional scalars. The performance brought by our additional intra-attentive representations can be further observed by comparing against the vanilla LSTM model. Clearly, removing the intra-attention network reverts our model to the standard LSTM. The performance improvements are encouraging, leading to almost 10% improvement in terms of F1 and accuracy. On datasets with short text, the performance improvement is often a modest $\approx 2\% - 3\%$ (**RQ2**). Notably, our proposed models also perform much better on long text, which can be attributed to the intra-attentive representations explicitly modeling long range dependencies. Intuitively, this is problematic for models that only capture sequential dependencies (e.g., word by word).

Finally, the relative performance of competitor methods are as expected. NBOW performs the worse, since it is just a naive bag-of-words model without any compositional or sequential information. On short text, LSTMs are overall better than CNNs. However, this trend is reversed on long text (i.e., Debates) since the LSTM model may be overburdened by overly long sequences. On short text, we also found that attention (or the gated pooling mechanism from GRNN) did not really help make any significant improvements over the vanilla LSTM model and a qualitative explanation to why this is so is deferred to the next section. However, attention helps for long text (such as debates), resulting in Attention LSTMs becoming the strongest baseline on the Debates datasets. However, our proposed intra-attentive model is both effective on short text and long text, outperforming Attention LSTMs consistently on all datasets.

4.5 In-depth Model Analysis

In this section, we present an in-depth analysis of our proposed model. More specifically, we not only aim to showcase the interpretability of our model but also explain how representations are formed. More specifically, we test our model (trained on Tweets dataset by (Ptáček et al., 2014)) on two examples. We extract the attention maps of three models, namely MIARN, Attention LSTM (ATT-LSTM) and applying Attention mechanism directly on the word embeddings without using a LSTM encoder (ATT-RAW). Table 5 shows the visualization of the attention maps.

Label	Model	Sentence
True	MIARN	I totally love being ignored !!
	ATT-LSTM	I totally love being ignored !!
	ATT-RAW	I totally love being ignored !!
False	MIARN	Being ignored sucks big time
	ATT-LSTM	Being ignored sucks big time
	ATT-RAW	Being ignored sucks big time

Table 5: Visualization of normalized attention weights on three different attention models (*Best viewed in color*). The intensity denotes the strength of the attention weight on the word.

In the first example (*true* label), we notice that the attention maps of MIARN are focusing on the words *'love'* and *'ignored'*. This is in concert with our intuition about modeling contrast and incongruity. On the other hand, both ATT-LSTM and ATT-RAW learn very different attention maps. As for ATT-LSTM, the attention weight is focused completely on the last representation - the token *'!!'*. Additionally, we also observed that this is true for many examples in the Tweets and Reddit dataset. We believe that this is the reason why standard neural attention does not help as what the attention mechanism is learning is to select the last representation (i.e., vanilla LSTM). Without the LSTM encoder, the attention weights focus on *'love'* but not *'ignored'*. This fails to capture any concept of contrast or incongruity.

Next, we consider the *false* labeled example. This time, the attention maps of MIARN are not as distinct as before. However, they focus on sentiment-bearing words, composing the words *'ignored sucks'* to form the majority of the intra-attentive representation. This time, passing the vector made up of *'ignored sucks'* allows the subsequent layers to recognize that there is no contrasting situation or sentiment. Similarly, ATT-LSTM focuses on the last word *time* which is totally non-interpretable. On the other hand, ATT-RAW focuses on relatively non-meaningful words such as *'big'*.

Overall, we analyzed two cases (positive and negative labels) and found that MIARN produces very explainable attention maps. In general, we found that MIARN is able to identify contrast and incongruity in sentences, allowing our model to better detect sarcasm. This is facilitated by modeling intra-sentence relationships. Notably, the standard vanilla attention is not explainable or interpretable.

5 Conclusion

Based on the intuition of intra-sentence similarity (i.e., *looking in-between*), we proposed a new neural network architecture for sarcasm detection. Our network incorporates a multi-dimensional intra-attention component that learns an intra-attentive representation of the sentence, enabling it to detect contrastive sentiment, situations and incongruity. Extensive experiments over six public benchmarks confirm the empirical effectiveness of our proposed model. Our proposed MIARN model outperforms strong state-of-the-art baselines such as GRNN and CNN-LSTM-DNN. Analysis of the intra-attention scores shows that our model learns highly interpretable attention weights, paving the way for more explainable neural sarcasm detection methods.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. pages 50–58.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1650–1659.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 452–461.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524* .

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@NAACL-HLT 2016, June 16, 2016, San Diego, California, USA*. pages 161–169. http://aclweb.org/anthology/W/W16/W16-0425.pdf.

Aniruddha Ghosh and Tony Veale. 2017. Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 482–491.

Rachel Giora. 1995. On irony and negation. *Discourse processes* 19(2):239–264.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*. Association for Computational Linguistics, pages 581–586.

Irazú Hernández-Farías, José-Miguel Benedí, and Paolo Rosso. 2015. Applying basic features from sentiment analysis for automatic irony detection. In *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, pages 337–344.

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

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)* 50(5):73.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. volume 2, pages 757–762.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? *arXiv preprint arXiv:1610.00883* .

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579* .

Stephanie Lukin and Marilyn Walker. 2017. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. *arXiv preprint arXiv:1708.08572* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Abhijit Mishra, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 377–387. https://doi.org/10.18653/v1/P17-1035.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016. Harnessing cognitive features for sarcasm detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1104.pdf.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 2249–2255.

Lotem Peled and Roi Reichart. 2017. Sarcasm SIGN: interpreting sarcasm with sentiment based monolingual machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1690–1700. https://doi.org/10.18653/v1/P17-1155.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. pages 213–223.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, pages 97–106.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation* 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 704–714. http://aclweb.org/anthology/D/D13/D13-1066.pdf.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696* .

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018s. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *In Proceedings of the AAAI 2018, 5956-5963*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017a. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102* .

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017b. Dyadic memory networks for aspect-based sentiment analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. pages 107–116. https://doi.org/10.1145/3132847.3132936.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Deirdre Wilson. 2006. The pragmatics of verbal irony: Echo or pretence? *Lingua* 116(10):1722–1743.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2449–2460. http://aclweb.org/anthology/C/C16/C16-1231.pdf.

Adversarial Contrastive Estimation

Avishek Joey Bose[1,2,*,†] **Huan Ling**[1,2,*,†] **Yanshuai Cao**[1,*]
[1]Borealis AI [2]University of Toronto
{joey.bose,huan.ling}@mail.utoronto.ca
{yanshuai.cao}@borealisai.com

Abstract

Learning by contrasting positive and negative samples is a general strategy adopted by many methods. Noise contrastive estimation (NCE) for word embeddings and translating embeddings for knowledge graphs are examples in NLP employing this approach. In this work, we view contrastive learning as an abstraction of all such methods and augment the negative sampler into a mixture distribution containing an adversarially learned sampler. The resulting adaptive sampler finds harder negative examples, which forces the main model to learn a better representation of the data. We evaluate our proposal on learning word embeddings, order embeddings and knowledge graph embeddings and observe both faster convergence and improved results on multiple metrics.

1 Introduction

Many models learn by contrasting losses on observed positive examples with those on some fictitious negative examples, trying to decrease some score on positive ones while increasing it on negative ones. There are multiple reasons why such contrastive learning approach is needed. Computational tractability is one. For instance, instead of using softmax to predict a word for learning word embeddings, noise contrastive estimation (NCE) (Dyer, 2014; Mnih and Teh, 2012) can be used in skip-gram or CBOW word embedding models (Gutmann and Hyvärinen, 2012; Mikolov et al., 2013; Mnih and Kavukcuoglu, 2013; Vaswani et al., 2013). Another reason is modeling need, as certain assumptions are best expressed as some score or energy in margin based or un-normalized probability models (Smith and Eisner, 2005). For example, modeling entity relations as translations or variants thereof in a vector space naturally leads to a distance-based score to be minimized for observed entity-relation-entity triplets (Bordes et al., 2013).

Given a scoring function, the gradient of the model's parameters on observed positive examples can be readily computed, but the negative phase requires a design decision on how to sample data. In noise contrastive estimation for word embeddings, a negative example is formed by replacing a component of a positive pair by randomly selecting a sampled word from the vocabulary, resulting in a fictitious word-context pair which would be unlikely to actually exist in the dataset. This negative sampling by corruption approach is also used in learning knowledge graph embeddings (Bordes et al., 2013; Lin et al., 2015; Ji et al., 2015; Wang et al., 2014; Trouillon et al., 2016; Yang et al., 2014; Dettmers et al., 2017), order embeddings (Vendrov et al., 2016), caption generation (Dai and Lin, 2017), etc.

Typically the corruption distribution is the same for all inputs like in skip-gram or CBOW NCE, rather than being a conditional distribution that takes into account information about the input sample under consideration. Furthermore, the corruption process usually only encodes a human prior as to what constitutes a hard negative sample, rather than being learned from data. For these two reasons, the simple fixed corruption process often yields only easy negative examples. Easy negatives are sub-optimal for learning discriminative representation as they do not force the model to find critical characteristics of observed positive data, which has been independently discovered in applications outside NLP previously (Shrivastava

[*] authors contributed equally
[†] Work done while author was an intern at Borealis AI

et al., 2016). Even if hard negatives are occasionally reached, the infrequency means slow convergence. Designing a more sophisticated corruption process could be fruitful, but requires costly trial-and-error by a human expert.

In this work, we propose to augment the simple corruption noise process in various embedding models with an adversarially learned conditional distribution, forming a mixture negative sampler that adapts to the underlying data and the embedding model training progress. The resulting method is referred to as adversarial contrastive estimation (ACE). The adaptive conditional model engages in a minimax game with the primary embedding model, much like in Generative Adversarial Networks (GANs) (Goodfellow et al., 2014a), where a discriminator net (D), tries to distinguish samples produced by a generator (G) from real data (Goodfellow et al., 2014b). In ACE, the main model learns to distinguish between a real positive example and a negative sample selected by the mixture of a fixed NCE sampler and an adversarial generator. The main model and the generator takes alternating turns to update their parameters. In fact, our method can be viewed as a conditional GAN (Mirza and Osindero, 2014) on discrete inputs, with a mixture generator consisting of a learned and a fixed distribution, with additional techniques introduced to achieve stable and convergent training of embedding models.

In our proposed ACE approach, the conditional sampler finds harder negatives than NCE, while being able to gracefully fall back to NCE whenever the generator cannot find hard negatives. We demonstrate the efficacy and generality of the proposed method on three different learning tasks, word embeddings (Mikolov et al., 2013), order embeddings (Vendrov et al., 2016) and knowledge graph embeddings (Ji et al., 2015).

2 Method

2.1 Background: contrastive learning

In the most general form, our method applies to supervised learning problems with a contrastive objective of the following form:

$$L(\omega) = \mathbb{E}_{p(x^+,y^+,y^-)}\, l_\omega(x^+, y^+, y^-) \qquad (1)$$

where $l_\omega(x^+, y^+, y^-)$ captures both the model with parameters ω and the loss that scores a positive tuple (x^+, y^+) against a negative one (x^+, y^-). $\mathbb{E}_{p(x^+,y^+,y^-)}(.)$ denotes expectation with respect to some joint distribution over positive and negative samples. Furthermore, by the law of total expectation, and the fact that given x^+, the negative sampling is not dependent on the positive label, i.e. $p(y^+, y^-|x^+) = p(y^+|x^+)p(y^-|x^+)$, Eq. 1 can be re-written as

$$\mathbb{E}_{p(x^+)}[\mathbb{E}_{p(y^+|x^+)p(y^-|x^+)}\, l_\omega(x^+, y^+, y^-)] \qquad (2)$$

Separable loss

In the case where the loss decomposes into a sum of scores on positive and negative tuples such as $l_\omega(x^+, y^+, y^-) = s_\omega\,(x^+, y^+) - \tilde{s}_\omega\,(x^+, y^-)$, then Expression. 2 becomes

$$\mathbb{E}_{p^+(x)}[\mathbb{E}_{p^+(y|x)}\, s_\omega\,(x, y) - \mathbb{E}_{p^-(y|x)}\, \tilde{s}_\omega\,(x, y)] \qquad (3)$$

where we moved the $+$ and $-$ to p for notational brevity. Learning by stochastic gradient descent aims to adjust ω to pushing down $s_\omega\,(x, y)$ on samples from p^+ while pushing up $\tilde{s}_\omega\,(x, y)$ on samples from p^-. Note that for generality, the scoring function for negative samples, denoted by $\tilde{s}_\omega$, could be slightly different from s_ω. For instance, $\tilde{s}$ could contain a margin as in the case of Order Embeddings in Sec. 4.2.

Non separable loss

Eq. 1 is the general form that we would like to consider because for certain problems, the loss function cannot be separated into sums of terms containing only positive (x^+, y^+) and terms with negatives (x^+, y^-). An example of such a non-separable loss is the triplet ranking loss (Schroff et al., 2015): $l_\omega = \max(0, \eta + s_\omega\,(x^+, y^+) - s_\omega\,(x^+, y^-))$, which does not decompose due to the rectification.

Noise contrastive estimation

The typical NCE approach in tasks such as word embeddings (Mikolov et al., 2013), order embeddings (Vendrov et al., 2016), and knowledge graph embeddings can be viewed as a special case of Eq. 2 by taking $p(y^-|x^+)$ to be some unconditional $p_{nce}(y)$.

This leads to efficient computation during training, however, $p_{nce}(y)$ sacrifices the sampling efficiency of learning as the negatives produced using a fixed distribution are not tailored toward x^+, and as a result are not necessarily hard negative examples. Thus, the model is not forced to discover discriminative representation of observed positive data. As training progresses, more and more negative examples are correctly learned, the probability of drawing a hard negative example diminishes further, causing slow convergence.

2.2 Adversarial mixture noise

To remedy the above mentioned problem of a fixed unconditional negative sampler, we propose to augment it into a mixture one, $\lambda p_{nce}(y) + (1 - \lambda) g_\theta(y|x)$, where g_θ is a conditional distribution with a learnable parameter θ and λ is a hyperparameter. The objective in Expression. 2 can then be written as (conditioned on x for notational brevity):

$$L(\omega, \theta; x) = \lambda\, \mathbb{E}_{p(y^+|x) p_{nce}(y^-)}\, l_\omega(x, y^+, y^-) \\ + (1 - \lambda)\, \mathbb{E}_{p(y^+|x) g_\theta(y^-|x)}\, l_\omega(x, y^+, y^-) \quad (4)$$

We learn (ω, θ) in a GAN-style minimax game:

$$\min_\omega \max_\theta V(\omega, \theta) = \min_\omega \max_\theta \mathbb{E}_{p^+(x)}\, L(\omega, \theta; x) \quad (5)$$

The embedding model behind $l_\omega(x, y^+, y^-)$ is similar to the discriminator in (conditional) GAN (or critic in Wasserstein (Arjovsky et al., 2017) or Energy-based GAN (Zhao et al., 2016), while $g_\theta(y|x)$ acts as the generator. Henceforth, we will use the term discriminator (D) and embedding model interchangeably, and refer to g_θ as the generator.

2.3 Learning the generator

There is one important distinction to typical GAN: $g_\theta(y|x)$ defines a categorical distribution over possible y values, and samples are drawn accordingly; in contrast to typical GAN over continuous data space such as images, where samples are generated by an implicit generative model that warps noise vectors into data points. Due to the discrete sampling step, g_θ cannot learn by receiving gradient through the discriminator. One possible solution is to use the Gumbel-softmax reparametrization trick (Jang et al., 2016; Maddison et al., 2016), which gives a differentiable approximation. However, this differentiability comes at the cost of drawing N Gumbel samples per each categorical sample, where N is the number of categories. For word embeddings, N is the vocabulary size, and for knowledge graph embeddings, N is the number of entities, both leading to infeasible computational requirements.

Instead, we use the REINFORCE (Williams, 1992) gradient estimator for $\nabla_\theta L(\theta, x)$:

$$(1-\lambda)\, \mathbb{E}\left[-l_\omega(x, y^+, y^-) \nabla_\theta \log(g_\theta(y^-|x))\right] \quad (6)$$

where the expectation $\mathbb{E}$ is with respect to $p(y^+, y^-|x) = p(y^+|x) g_\theta(y^-|x)$, and the discriminator loss $l_\omega(x, y^+, y^-)$ acts as the reward.

With a separable loss, the (conditional) value function of the minimax game is:

$$L(\omega, \theta; x) = \mathbb{E}_{p^+(y|x)}\, s_\omega(x, y) \\ - \mathbb{E}_{p_{nce}(y)}\, \tilde{s}_\omega(x, y) - \mathbb{E}_{g_\theta(y|x)}\, \tilde{s}_\omega(x, y) \quad (7)$$

and only the last term depends on the generator parameter ω. Hence, with a separable loss, the reward is $-\tilde{s}(x^+, y^-)$. This reduction does not happen with a non-separable loss, and we have to use $l_\omega(x, y^+, y^-)$.

2.4 Entropy and training stability

GAN training can suffer from instability and degeneracy where the generator probability mass collapses to a few modes or points. Much work has been done to stabilize GAN training in the continuous case (Arjovsky et al., 2017; Gulrajani et al., 2017; Cao et al., 2018). In ACE, if the generator g_θ probability mass collapses to a few candidates, then after the discriminator successfully learns about these negatives, g_θ cannot adapt to select new hard negatives, because the REINFORCE gradient estimator Eq. 6 relies on g_θ being able to explore other candidates during sampling. Therefore, if the g_θ probability mass collapses, instead of leading to oscillation as in typical GAN, the min-max game in ACE reaches an equilibrium where the discriminator wins and g_θ can no longer adapt, then ACE falls back to NCE since the negative sampler has another mixture component from NCE.

This behavior of gracefully falling back to NCE is more desirable than the alternative of stalled training if $p^-(y|x)$ does not have a simple p_{nce} mixture component. However, we would still like to avoid such collapse, as the adversarial samples provide greater learning signals than NCE samples. To this end, we propose to use a regularizer to encourage the categorical distribution $g_\theta(y|x)$ to have high entropy. In order to make the the regularizer interpretable and its hyperparameters easy to tune, we design the following form:

$$R_{ent}(x) = min(0, c - H(g_\theta(y|x))) \quad (8)$$

where $H(g_\theta(y|x))$ is the entropy of the categorical distribution $g_\theta(y|x)$, and $c = \log(k)$ is the entropy of a uniform distribution over k choices, and k is a hyper-parameter. Intuitively, R_{ent} expresses the prior that the generator should spread its mass over more than k choices for each x.

2.5 Handling false negatives

During negative sampling, $p^-(y|x)$ could actually produce y that forms a positive pair that exists in the training set, i.e., a false negative. This possibility exists in NCE already, but since p_{nce} is not adaptive, the probability of sampling a false negative is low. Hence in NCE, the score on this false negative (true observation) pair is pushed up less in the negative term than in the positive term.

However, with the adaptive sampler, $g_\omega(y|x)$, false negatives become a much more severe issue. $g_\omega(y|x)$ can learn to concentrate its mass on a few false negatives, significantly canceling the learning of those observations in the positive phase. The entropy regularization reduces this problem as it forces the generator to spread its mass, hence reducing the chance of a false negative.

To further alleviate this problem, whenever computationally feasible, we apply an additional two-step technique. First, we maintain a hash map of the training data in memory, and use it to efficiently detect if a negative sample (x^+, y^-) is an actual observation. If so, its contribution to the loss is given a zero weight in ω learning step. Second, to upate θ in the generator learning step, the reward for false negative samples are replaced by a large penalty, so that the REINFORCE gradient update would steer g_θ away from those samples. The second step is needed to prevent null computation where g_θ learns to sample false negatives which are subsequently ignored by the discriminator update for ω.

2.6 Variance Reduction

The basic REINFORCE gradient estimator is poised with high variance, so in practice one often needs to apply variance reduction techniques. The most basic form of variance reduction is to subtract a baseline from the reward. As long as the baseline is not a function of actions (i.e., samples y^- being drawn), the REINFORCE gradient estimator remains unbiased. More advanced gradient estimators exist that also reduce variance (Grathwohl et al., 2017; Tucker et al., 2017; Liu et al., 2018), but for simplicity we use the self-critical baseline method (Rennie et al., 2016), where the baseline is $b(x) = l_\omega(y^+, y^\star, x)$, or $b(x) = -\tilde{s}_\omega(y^\star, x)$ in the separable loss case, and $y^\star = \text{argmax}_i g_\theta(y_i|x)$. In other words, the baseline is the reward of the most likely sample according to the generator.

2.7 Improving exploration in g_θ by leveraging NCE samples

In Sec. 2.4 we touched on the need for sufficient exploration in g_θ. It is possible to also leverage negative samples from NCE to help the generator learn. This is essentially off-policy exploration in reinforcement learning since NCE samples are not drawn according to $g_\theta(y|x)$. The generator learning can use importance re-weighting to leverage those samples. The resulting REINFORCE gradient estimator is basically the same as Eq. 6 except that the rewards are reweighted by $g_\theta(y^-|x)/p_{nce}(y^-)$, and the expectation is with respect to $p(y^+|x)p_{nce}(y^-)$. This additional off-policy learning term provides gradient information for generator learning if $g_\theta(y^-|x)$ is not zero, meaning that for it to be effective in helping exploration, the generator cannot be collapsed at the first place. Hence, in practice, this term is only used to further help on top of the entropy regularization, but it does not replace it.

3 Related Work

Smith and Eisner (2005) proposed contrastive estimation as a way for unsupervised learning of log-linear models by taking implicit evidence from user-defined neighborhoods around observed datapoints. Gutmann and Hyvärinen (2010) introduced NCE as an alternative to the hierarchical softmax. In the works of Mnih and Teh (2012) and Mnih and Kavukcuoglu (2013), NCE is applied to log-bilinear models and Vaswani et al. (2013) applied NCE to neural probabilistic language models (Yoshua et al., 2003). Compared to these previous NCE methods that rely on simple fixed sampling heuristics, ACE uses an adaptive sampler that produces harder negatives.

In the domain of max-margin estimation for structured prediction (Taskar et al., 2005), loss augmented MAP inference plays the role of finding hard negatives (the hardest). However, this inference is only tractable in a limited class of models such structured SVM (Tsochantaridis et al., 2005). Compared to those models that use exact maximization to find the hardest negative configuration each time, the generator in ACE can be viewed as learning an approximate amortized inference network. Concurrently to this work, Tu and Gimpel (2018) proposes a very similar framework, using a learned inference network for Structured prediction energy networks (SPEN) (Belanger and McCallum, 2016).

Concurrent with our work, there have been other interests in applying the GAN to NLP problems (Fedus et al., 2018; Wang et al., 2018; Cai and Wang, 2017). Knowledge graph models naturally lend to a GAN setup, and has been the subject of study in Wang et al. (2018) and Cai and Wang (2017). These two concurrent works are most closely related to one of the three tasks on which we study ACE in this work. Besides a more general formulation that applies to problems beyond those considered in Wang et al. (2018) and Cai and Wang (2017), the techniques introduced in our work on handling false negatives and entropy regularization lead to improved experimental results as shown in Sec. 5.4.

4 Application of ACE on three tasks

4.1 Word Embeddings

Word embeddings learn a vector representation of words from co-occurrences in a text corpus. NCE casts this learning problem as a binary classification where the model tries to distinguish positive word and context pairs, from negative noise samples composed of word and false context pairs. The NCE objective in Skip-gram (Mikolov et al., 2013) for word embeddings is a separable loss of the form:

$$\begin{aligned} L = -\sum_{w_t \in V} [\log p(y=1|w_t, w_c^+) \\ + \sum_{c=1}^{K} \log p(y=0|w_t, w_c^-)] \end{aligned} \tag{9}$$

Here, w_c^+ is sampled from the set of true contexts and $w_c^- \sim Q$ is sampled k times from a fixed noise distribution. Mikolov et al. (2013) introduced a further simplification of NCE, called "Negative Sampling" (Dyer, 2014). With respect to our ACE framework, the difference between NCE and Negative Sampling is inconsequential, so we continue the discussion using NCE. A drawback of this sampling scheme is that it favors more common words as context. Another issue is that the negative context words are sampled in the same way, rather than tailored toward the actual target word. To apply ACE to this problem we first define the value function for the minimax game, $V(D, G)$, as follows:

$$\begin{aligned} V(D,G) &= \mathbb{E}_{p^+(w_c)}[\log D(w_c, w_t)] \\ &- \mathbb{E}_{p_{nce}(w_c)}[-\log(1 - D(w_c, w_t))] \\ &- \mathbb{E}_{g_\theta(w_c|w_t)}[-\log(1 - D(w_c, w_t))] \end{aligned} \tag{10}$$

with $D = p(y = 1|w_t, w_c)$ and $G = g_\theta(w_c|w_t)$.

Implementation details

For our experiments, we train all our models on a single pass of the May 2017 dump of the English Wikipedia with lowercased unigrams. The vocabulary size is restricted to the top $150k$ most frequent words when training from scratch while for finetuning we use the same vocabulary as Pennington et al. (2014), which is $400k$ of the most frequent words. We use 5 NCE samples for each positive sample and 1 adversarial sample in a window size of 10 and the same positive subsampling scheme proposed by Mikolov et al. (2013). Learning for both G and D uses Adam (Kingma and Ba, 2014) optimizer with its default parameters. Our conditional discriminator is modeled using the Skip-Gram architecture, which is a two layer neural network with a linear mapping between the layers. The generator network consists of an embedding layer followed by two small hidden layers, followed by an output softmax layer. The first layer of the generator shares its weights with the second embedding layer in the discriminator network, which we find really speeds up convergence as the generator does not have to relearn its own set of embeddings. The difference between the discriminator and generator is that a sigmoid nonlinearity is used after the second layer in the discriminator, while in the generator, a softmax layer is used to define a categorical distribution over negative word candidates. We find that controlling the generator entropy is critical for finetuning experiments as otherwise the generator collapses to its favorite negative sample. The word embeddings are taken to be the first dense matrix in the discriminator.

4.2 Order Embeddings Hypernym Prediction

As introduced in Vendrov et al. (2016), ordered representations over hierarchy can be learned by order embeddings. An example task for such ordered representation is hypernym prediction. A hypernym pair is a pair of concepts where the first concept is a specialization or an instance of the second.

For completeness, we briefly describe order embeddings, then analyze ACE on the hypernym prediction task. In order embeddings, each entity is represented by a vector in $\mathbb{R}^N$, the score for a positive ordered pair of entities (x, y) is defined by $s_\omega(x, y) = ||max(0, y - x)||^2$ and, score for a negative ordered pair (x^+, y^-) is defined by $\tilde{s}_\omega(x^+, y^-) = \max\{0, \eta - s(x^+, y^-)\}$, where is η is the margin. Let $f(u)$ be the embedding function which takes an entity as input and outputs en embedding vector. We define P as a set of positive pairs and N as negative pairs, the separable loss function for order embedding task is defined by:

$$L = \sum_{(u,v)\in P} s_\omega(f(u), f(v))) + \sum_{(u,v)\in N} \tilde{s}(f(u), f(v)) \tag{11}$$

Implementation details

Our generator for this task is just a linear fully connected softmax layer, taking an embedding vector from discriminator as input and outputting a categorical distribution over the entity set. For the discriminator, we inherit all model setting from Vendrov et al. (2016): we use 50 dimensions hidden state and bash size 1000, a learning rate of 0.01 and the Adam optimizer. For the generator, we use a batch size of 1000, a learning rate 0.01 and the Adam optimizer. We apply weight decay with rate 0.1 and entropy loss regularization as described in Sec. 2.4. We handle false negative as described in Sec. 2.5. After cross validation, variance reduction and leveraging NCE samples does not greatly affect the order embedding task.

4.3 Knowledge Graph Embeddings

Knowledge graphs contain entity and relation data of the form *(head entity, relation, tail entity)*, and the goal is to learn from observed positive entity relations and predict missing links (a.k.a. link prediction). There have been many works on knowledge graph embeddings, e.g. TransE (Bordes et al., 2013), TransR (Lin et al., 2015), TransH (Wang et al., 2014), TransD (Ji et al., 2015), Complex (Trouillon et al., 2016), DistMult (Yang et al., 2014) and ConvE (Dettmers et al., 2017). Many of them use a contrastive learning objective. Here we take TransD as an example, and modify its noise contrastive learning to ACE, and demonstrate significant improvement in sample efficiency and link prediction results.

Implementation details

Let a positive entity-relation-entity triplet be denoted by $\xi^+ = (h^+, r^+, t^+)$, and a negative triplet could either have its head or tail be a negative sample, i.e. $\xi^- = (h^-, r^+, t^+)$ or $\xi^- = (h^+, r^+, t^-)$. In either case, the general formulation in Sec. 2.1 still applies. The non-separable loss function takes on the form:

$$l = \max(0, \eta + s_\omega(\xi^+) - s_\omega(\xi^-)) \tag{12}$$

The scoring rule is:

$$s = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\| \tag{13}$$

where $\mathbf{r}$ is the embedding vector for r, and $\mathbf{h}_\perp$ is projection of the embedding of h onto the space of $\mathbf{r}$ by $\mathbf{h}_\perp = \mathbf{h} + \mathbf{r}_p\mathbf{h}_p^\top\mathbf{h}$, where $\mathbf{r}_p$ and $\mathbf{h}_p$ are projection parameters of the model. $\mathbf{t}_\perp$ is defined in a similar way through parameters $\mathbf{t}$, $\mathbf{t}_p$ and $\mathbf{r}_p$.

The form of the generator $g_\theta(t^-|r^+, h^+)$ is chosen to be $f_\theta(\mathbf{h}_\perp, \mathbf{h}_\perp + \mathbf{r})$, where f_θ is a feedforward neural net that concatenates its two input arguments, then propagates through two hidden layers, followed by a final softmax output layer. As a function of (r^+, h^+), g_θ shares parameter with the discriminator, as the inputs to f_θ are the embedding vectors. During generator learning, only θ is updated and the TransD model embedding parameters are frozen.

5 Experiments

We evaluate ACE with experiments on word embeddings, order embeddings, and knowledge graph embeddings tasks. In short, whenever the original learning objective is contrastive (all tasks except Glove fine-tuning) our results consistently show that ACE improves over NCE. In some cases, we include additional comparisons to the state-of-art results on the task to put the significance of such improvements in context: the generic ACE can often make a reasonable baseline competitive with SOTA methods that are optimized for the task.

For word embeddings, we evaluate models trained from scratch as well as fine-tuned Glove models (Pennington et al., 2014) on word similarity tasks that consist of computing the similarity

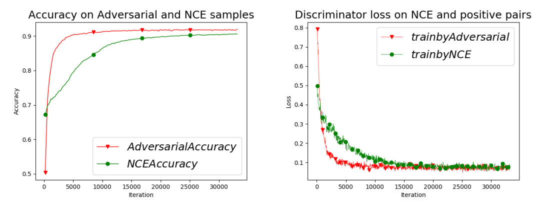


Figure 1: **Left**: Order embedding Accuracy plot. **Right**: Order embedding discriminator Loss plot on NCE sampled negative pairs and positive pairs.

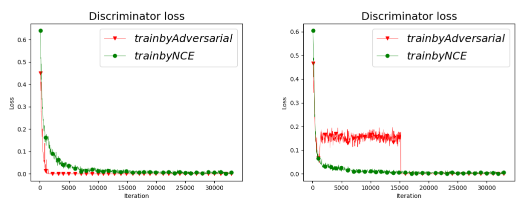


Figure 2: loss curve on NCE negative pairs and ACE negative pairs. **Left**: without entropy and weight decay. **Right**: with entropy and weight decay

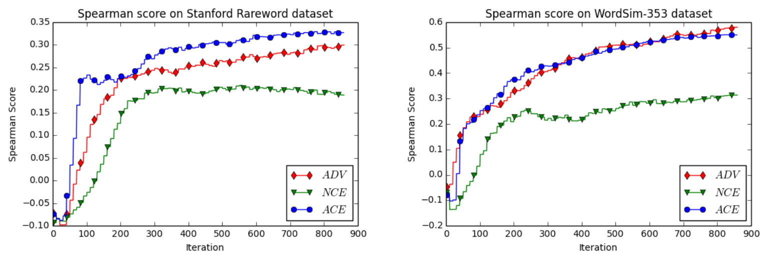


Figure 3: **Left**: Rare Word, **Right**: WS353 similarity scores during the first epoch of training.

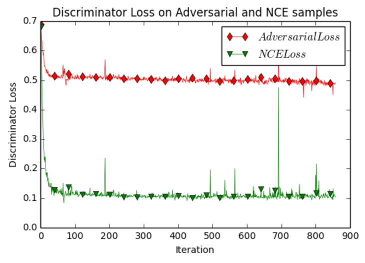


Figure 4: Training from scratch losses on the Discriminator

between word pairs where the ground truth is an average of human scores. We choose the Rare word dataset (Luong et al., 2013) and WordSim-353 (Finkelstein et al., 2001) by virtue of our hypothesis that ACE learns better representations for both rare and frequent words. We also qualitatively evaluate ACE word embeddings by inspecting the nearest neighbors of selected words.

For the hypernym prediction task, following Vendrov et al. (2016), hypernym pairs are created from the WordNet hierarchy's transitive closure. We use the released random development split and test split from Vendrov et al. (2016), which both contain 4000 edges.

For knowledge graph embeddings, we use TransD (Ji et al., 2015) as our base model, and perform ablation study to analyze the behavior of ACE with various add-on features, and confirm that entropy regularization is crucial for good performance in ACE. We also obtain link prediction results that are competitive or superior to the state-of-arts on the WN18 dataset (Bordes et al., 2014).

5.1 Training Word Embeddings from scratch

In this experiment, we empirically observe that training word embeddings using ACE converges significantly faster than NCE after one epoch. As shown in Fig. 3 both ACE (a mixture of p_{nce} and g_θ) and just g_θ (denoted by ADV) significantly outperforms the NCE baseline, with an absolute improvement of 73.1% and 58.5% respectively on RW score. We note similar results on WordSim-353 dataset where ACE and ADV outperforms NCE by 40.4% and 45.7%. We also evaluate our model qualitatively by inspecting the nearest neighbors of selected words in Table. 1. We first present the five nearest neighbors to each word to show that both NCE and ACE models learn sensible embeddings. We then show that ACE embeddings have much better semantic relevance in a larger neighborhood (nearest neighbor 45-50).

5.2 Finetuning Word Embeddings

We take off-the-shelf pre-trained Glove embeddings which were trained using 6 billion tokens (Pennington et al., 2014) and fine-tune them using our algorithm. It is interesting to note that the original Glove objective does not fit into the contrastive learning framework, but nonetheless we find that they benefit from ACE. In fact, we observe that training such that 75% of the words appear as positive contexts is sufficient to beat the largest dimensionality pre-trained Glove model on word similarity tasks. We evaluate our performance on the Rare Word and WordSim353 data. As can be seen from our results in Table 2, ACE on RW is not always better and for the 100d and 300d Glove embeddings is marginally worse. However, on WordSim353 ACE does considerably better across the board to the point where 50d Glove embeddings outperform the 300d baseline Glove model.

5.3 Hypernym Prediction

As shown in Table 3, with ACE training, our method achieves a 1.5% improvement on accu-

	Queen	King	Computer	Man	Woman
Skip-Gram NCE Top 5	princess	prince	computers	woman	girl
	king	queen	computing	boy	man
	empress	kings	software	girl	prostitute
	pxqueen	emperor	microcomputer	stranger	person
	monarch	monarch	mainframe	person	divorcee
Skip-Gram NCE Top 45-50	sambiria	eraric	hypercard	angiomata	suitor
	phongsri	mumbere	neurotechnology	someone	nymphomaniac
	safrit	empress	lgp	bespectacled	barmaid
	mcelvoy	saxonvm	pcs	hero	redheaded
	tsarina	pretender	keystroke	clown	jew
Skip-Gram ACE Top 5	princess	prince	software	woman	girl
	prince	vi	computers	girl	herself
	elizabeth	kings	applications	tells	man
	duke	duke	computing	dead	lover
	consort	iii	hardware	boy	tells
Skip-Gram ACE Top 45-50	baron	earl	files	kid	aunt
	abbey	holy	information	told	maid
	throne	cardinal	device	revenge	wife
	marie	aragon	design	magic	lady
	victoria	princes	compatible	angry	bride

Table 1: Top 5 Nearest Neighbors of Words followed by Neighbors 45-50 for different Models.

	RW	WS353
Skipgram Only NCE baseline	18.90	31.35
Skipgram + Only ADV	29.96	58.05
Skipgram + ACE	32.71	55.00
Glove-50 (Recomputed based on(Pennington et al., 2014))	34.02	49.51
Glove-100 (Recomputed based on(Pennington et al., 2014))	36.64	52.76
Glove-300 (Recomputed based on(Pennington et al., 2014))	**41.18**	60.12
Glove-50 + ACE	35.60	60.46
Glove-100 + ACE	36.51	63.29
Glove-300 + ACE	40.57	**66.50**

Table 2: Spearman score ($\rho * 100$) on RW and WS353 Datasets. We trained a skipgram model from scratch under various settings for only 1 epoch on wikipedia. For finetuned models we re-computed the scores based on the publicly available 6B tokens Glove models and we finetuned until roughly 75% of the vocabulary was seen.

Method	Accuracy (%)
order-embeddings	90.6
order-embeddings + Our ACE	**92.0**

Table 3: Order Embedding Performance

racy over Vendrov et al. (2016) without tunning any of the discriminator's hyperparameters. We further report training curve in Fig. 1, we report loss curve on randomly sampled pairs. We stress that in the ACE model, we train random pairs and generator generated pairs jointly, as shown in Fig. 2, hard negatives help the order embedding model converges faster.

5.4 Ablation Study and Improving TransD

To analyze different aspects of ACE, we perform an ablation study on the knowledge graph embedding task. As described in Sec. 4.3, the base model (discriminator) we apply ACE to is TransD (Ji et al., 2015). Fig. 5 shows validation performance as training progresses. All variants of ACE converges to better results than base NCE. Among ACE variants, all methods that include entropy regularization significantly outperform without entropy regularization. Without the self critical baseline variance reduction, learning could progress faster at the beginning but the final performance suffers slightly. The best performance is obtained without the additional off-policy learning of the generator.

Table. 4 shows the final test results on WN18 link prediction task. It is interesting to note that ACE improves MRR score more significantly than hit@10. As MRR is a lot more sensitive to the top rankings, i.e., how the correct configuration ranks among the competitive alternatives, this is consistent with the fact that ACE samples hard negatives and forces the base model to learn a more discriminative representation of the positive examples.

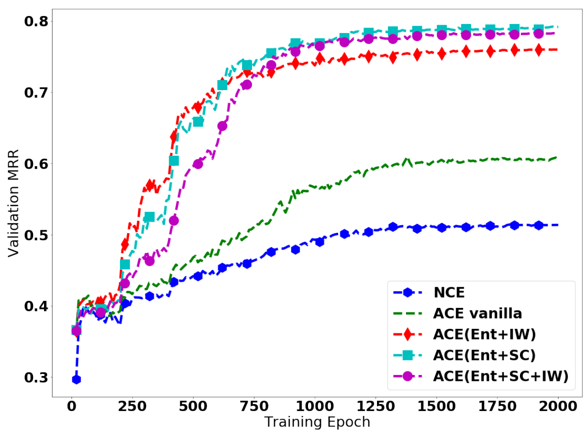


Figure 5: Ablation study: measuring validation Mean Reciprocal Rank (MRR) on WN18 dataset as training progresses.

	MRR	hit@10
ACE(Ent+SC)	0.792	0.945
ACE(Ent+SC+IW)	0.768	**0.949**
NCE TransD (ours)	0.527	0.947
NCE TransD ((Ji et al., 2015))	-	0.925
KBGAN(DISTMULT) ((Cai and Wang, 2017))	0.772	0.948
KBGAN(COMPLEX) ((Cai and Wang, 2017))	0.779	0.948
Wang et al. ((Wang et al., 2018))	-	0.93
COMPLEX ((Trouillon et al., 2016))	**0.941**	0.947

Table 4: WN18 experiments: the first portion of the table contains results where the base model is TransD, the last separated line is the COMPLEX embedding model (Trouillon et al., 2016), which achieves the SOTA on this dataset. Among all TransD based models (the best results in this group is underlined), ACE improves over basic NCE and another GAN based approach KBGAN. The gap on MRR is likely due to the difference between TransD and COMPLEX models.

5.5 Hard Negative Analysis

To better understand the effect of the adversarial samples proposed by the generator we plot the discriminator loss on both p_{nce} and g_θ samples. In this context, a harder sample means a higher loss assigned by the discriminator. Fig. 4 shows that discriminator loss for the word embedding task on g_θ samples are always higher than on p_{nce} samples, confirming that the generator is indeed sampling harder negatives.
For Hypernym Prediction task, Fig.2 shows discriminator loss on negative pairs sampled from NCE and ACE respectively. The higher the loss the harder the negative pair is. As indicated in the left plot, loss on the ACE negative terms collapses faster than on the NCE negatives. After adding entropy regularization and weight decay, the generator works as expected.

6 Limitations

When the generator softmax is large, the current implementation of ACE training is computationally expensive. Although ACE converges faster per iteration, it may converge more slowly on wall-clock time depending on the cost of the softmax. However, embeddings are typically used as pre-trained building blocks for subsequent tasks. Thus, their learning is usually the pre-computation step for the more complex downstream models and spending more time is justified, especially with GPU acceleration. We believe that the computational cost could potentially be reduced via some existing techniques such as the "augment and reduce" variational inference of (Ruiz et al., 2018), adaptive softmax (Grave et al., 2016), or the "sparsely-gated" softmax of Shazeer et al. (2017), but leave that to future work.

Another limitation is on the theoretical front. As noted in Goodfellow (2014), GAN learning does not implement maximum likelihood estimation (MLE), while NCE has MLE as an asymptotic limit. To the best of our knowledge, more distant connections between GAN and MLE training are not known, and tools for analyzing the equilibrium of a min-max game where players are parametrized by deep neural nets are currently not available to the best of our knowledge.

7 Conclusion

In this paper, we propose Adversarial Contrastive Estimation as a general technique for improving supervised learning problems that learn by contrasting observed and fictitious samples. Specifically, we use a generator network in a conditional GAN like setting to propose hard negative examples for our discriminator model. We find that a mixture distribution of randomly sampling negative examples along with an adaptive negative sampler leads to improved performances on a variety of embedding tasks. We validate our hypothesis that hard negative examples are critical to optimal learning and can be proposed via our ACE framework. Finally, we find that controlling the entropy of the generator through a regularization term and properly handling false negatives is crucial for successful training.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Liwei Cai and William Yang Wang. 2017. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*.

Yanshuai Cao, Gavin Weiguang Ding, Kry Yik-Chau Lui, and Ruitong Huang. 2018. Improving GAN training via binarized representation entropy (BRE) regularization. In *International Conference on Learning Representations*.

Bo Dai and Dahua Lin. 2017. Contrastive learning for image captioning. In *Advances in Neural Information Processing Systems*, pages 898–907.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*.

Chris Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. MaskGAN: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014a. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014b. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ian J Goodfellow. 2014. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515*.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. 2017. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2016. Efficient softmax approximation for GPUs. *arXiv preprint arXiv:1609.04309*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. 2018. Action-dependent control variates for policy optimization via stein identity. In *International Conference on Learning Representations*.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

M. Mirza and S. Osindero. 2014. Conditional Generative Adversarial Nets. *ArXiv e-prints*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563*.

Francisco JR Ruiz, Michalis K Titsias, Adji B Dieng, and David M Blei. 2018. Augment and reduce: Stochastic inference for large categorical distributions. *arXiv preprint arXiv:1802.04220*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484.

Lifu Tu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *International Conference on Learning Representations*.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2627–2636. Curran Associates, Inc.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *International Conference on Learning Representations*.

Peifeng Wang, Shuangyin Li, and Rong Pan. 2018. Incorporating GAN for negative sampling in knowledge representation learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. AAAI Press.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Bengio Yoshua, Ducharme Rejean, Vincent Pascal, and Jauvin Christian. 2003. A neural probabilistic language model. *Journal of Machine Learning Research.*

Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126.*

Adaptive Scaling for Sparse Detection in Information Extraction

Hongyu Lin[1,2], **Yaojie Lu**[1,2], **Xianpei Han**[1], **Le Sun**[1]
[1]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{hongyu2016,yaojie2017,xianpei,sunle}@iscas.ac.cn

Abstract

This paper focuses on detection tasks in information extraction, where positive instances are sparsely distributed and models are usually evaluated using F-measure on positive classes. These characteristics often result in deficient performance of neural network based detection models. In this paper, we propose *adaptive scaling*, an algorithm which can handle the positive sparsity problem and directly optimize over F-measure via dynamic cost-sensitive learning. To this end, we borrow the idea of marginal utility from economics and propose a theoretical framework for instance importance measuring without introducing any additional hyper-parameters. Experiments show that our algorithm leads to a more effective and stable training of neural network based detection models.

1 Introduction

Detection problems, aiming to identify occurrences of specific kinds of information (e.g., events, relations, or entities) in documents, are fundamental and widespread in information extraction (IE). For instance, an event detection (Walker et al., 2006) system may want to detect triggers for "*Attack*" events, such as "*shot*" in sentence "*He was shot*". In relation detection (Hendrickx et al., 2009), we may want to identify all instances of a specific relation, such as "*Jane joined Google*" for "*Employment*" relation.

Recently, a number of researches have employed neural network models to solve detection problems, and have achieved significant improvement in many tasks, such as event detection (Chen et al., 2015; Nguyen and Grishman, 2015), relation detection (Zeng et al., 2014; Santos et al., 2015) and named entity recognition (Huang et al., 2015; Chiu and Nichols, 2015; Lample et al., 2016). These methods usually regard detection problems as standard classification tasks, with several positive classes for targets to detect and one negative class for irrelevant (background) instances. For example, an event detection model will identify event triggers in sentence "*He was shot*" by classifying word "*shot*" into positive class "*Attack*", and classifying all other words into the negative class "*NIL*". To optimize classifiers, cross-entropy loss function is commonly used in this paradigm.

	Classification	**Detection**
Target Instances	All instances	Sparse positive instances
Evaluation	Accuracy or F-measure **on all classes**	F-measure **on only positive classes**
Typical Tasks	Text Classification, Sentiment Classification	Event Detection, Relation Detection

Table 1: Comparison between standard classification tasks and detection problems.

However, different from standard classification tasks, detection tasks have unique *class inequality* characteristic, which stems from both data distribution and applied evaluation metric. Table 1 shows their differences. First, positive instances are commonly sparsely distributed in detection tasks. For example, in event detection, less than 2% of words are a trigger of an event in RichERE dataset (Song et al., 2015). Furthermore, detection tasks are commonly evaluated using F-measure on positive classes, rather than accuracy or F-measure on all classes. Therefore positive and negative classes play different roles in the evaluation: the performance is evaluated by only considering how well we can detect positive instances, while correct predictions of negative instances are ignored.

Due to the class inequality characteristic, reported results indicate that simply applying standard classification paradigm to detection tasks will result in deficient performance (Anand et al., 1993; Carvajal et al., 2004; Lin et al., 2017). This is because minimizing cross-entropy loss function corresponds to maximize the accuracy of neural networks on all training instances, rather than F-measure on positive classes. Furthermore, due to the positive sparsity problem, training procedure will easily achieve a high accuracy on negative class, but is difficult to converge on positive classes and often leads to a low recall rate. Although simple sampling heuristics can alleviate this problem to some extent, they either suffer from losing inner class information or over-fitting positive instances (He and Garcia, 2009; Fernández-Navarro et al., 2011), which often result in instability during the training procedure.

Some previous approaches (Joachims, 2005; Jansche, 2005, 2007; Dembczynski et al., 2011; Chinta et al., 2013; Narasimhan et al., 2014; Natarajan et al., 2016) tried to solve this problem by directly optimizing F-measure. Parambath et al. (2014) proved that it is sufficient to solve F-measure optimization problem via cost-sensitive learning, where class-specific cost factors are applied to indicate the importance of different classes to F-measure. However, optimal factors are not known a priori so ε-search needs to be applied, which is too time consuming for the optimization of neural networks.

To solve the class inequality problem for sparse detection model optimization, this paper proposes a theoretical framework to quantify the importance of positive/negative instances during training. We borrow the idea of marginal utility from Economics (Stigler, 1950), and regard the evaluation metric (i.e., F-measure commonly) as the utility to optimize. Based on the above idea, the importance of an instance is measured using the marginal utility of correctly predicting it. For standard classification tasks evaluated using accuracy, our framework proves that correct predictions of positive and negative instances will have equal and unchanged marginal utility, i.e., all instances are with the same importance. For detection problems evaluated using F-measure, our framework proves that the utility of correctly predicting one more positive instance (*marginal positive utility*) and that of correctly predicting one more negative instance (*marginal negative utility*) are different and dynamically changed during model training. That is, the importance of instances of each class is not only determined by their data distribution, but also affected by how well the current model can converge on different classes.

Based on the above framework, we propose *adaptive scaling*, a dynamic cost-sensitive learning algorithm which adaptively scales costs of instances of different classes with above quantified importance during the training procedure, and thus can make the optimization criteria consistent with the evaluation metric. Furthermore, a batch-wise version of our adaptive scaling algorithm is proposed to make it directly applicable as a plug-in of conventional neural network training algorithms. Compared with previous methods, adaptive scaling is designed based on marginal utility framework and doesn't introduce any additional hyper-parameter, and therefore is more efficient and stable to transfer among datasets and models.

Generally, the main contributions of this paper are:

- We propose a marginal utility based framework for detection model optimization, which can dynamically quantify instance importance to different evaluation metrics.
- Based on the above framework, we present adaptive scaling, a plug-in algorithm which can effectively resolve the class inequality problem in neural detection model optimization via dynamic cost-sensitive learning.

We conducted experimental studies[1] on event detection, a typical sparse detection task in IE. We thoroughly compared various methods for adapting classical neural network models into detection problems. Experiment results show that our adaptive scaling algorithm not only achieves a better performance, but also is more stable and more adaptive for training neural networks on various models and datasets.

2 Background

Relation between Accuracy Metric and Cross-Entropy Loss. Recent neural network methods usually regard detection problems as standard classification tasks, with several positive classes to detect, and one negative class for other irrelevant

[1]Our source code is openly available at `github.com/sanmusunrise/AdaScaling`.

instances. Formally, given P positive training instances $\mathcal{P} = \{(x_i, y_i)_{i=1}^{P}\}$, and N negative instances $\mathcal{N} = \{(x_i, y_i)_{i=1}^{N}\}$ (due to positive sparsity, $P \ll N$), the training of neural network classifiers usually involves in minimizing the softmax cross-entropy loss function regarding to model parameters θ:

$$\mathcal{L}_{CE}(\theta) = -\frac{1}{P+N} \sum_{(x_i,y_i)\in\mathcal{P}\bigcup\mathcal{N}} \log p(y_i|x_i;\theta) \quad (1)$$

and if $P, N \to \infty$, we have

$$\lim_{P,N\to\infty} \mathcal{L}_{CE}(\theta) = -\mathbb{E}[\log p(y|x;\theta)] = -\log(\text{Accuracy}) \quad (2)$$

which reveals that minimizing cross-entropy loss corresponds to maximize the expected accuracy of the classifier on training data.

Divergence between F-Measure and Cross-Entropy Loss. However, detection tasks are mostly evaluated using F-measure computed on positive classes, which makes it unsuitable to optimize classifiers using cross-entropy loss. For instance, due to the positive sparsity, simply classifying all instances into negative class will achieve a high accuracy but zero F-measure.

To show where this divergence comes from, let $c_1, c_2, ..., c_{k-1}$ denote $k-1$ positive classes and c_k is the negative class, we define $TP = \sum_{i=1}^{k-1} TP_i$, where TP_i is the population of correctly predicted instances of positive class c_i. TN denotes the number of correctly predicted negative instances. PE represents positive-positive error, where an instance is classified into one positive class c_i but its golden label is another positive class c_j. Then we have following metrics[2]:

$$\text{Accuracy} = \frac{TP+TN}{P+N} \quad (3)$$

$$\text{Precision} = \frac{TP}{N-TN+PE+TP} \quad (4)$$

$$\text{Recall} = \frac{TP}{P} \quad (5)$$

$$\begin{aligned} F_\beta &= (1+\beta^2)\frac{\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}} \\ &= (1+\beta^2)\frac{TP}{\beta^2 P + N - TN + PE + TP} \end{aligned} \quad (6)$$

where β in F_β is a factor indicating the metric attaches β times as much importance to recall as precision. We can easily see that for accuracy metric, correct predictions of positive and negative instances are equally regarded (i.e., TP and TN are symmetric), which is consistent with cross-entropy loss function. However, when measuring using F-measure, this condition is no longer holding. The importance varies from different classes (i.e., TP and TN are asymmetric). Therefore, to make the training procedure consistent with F-measure, it is critical to take this importance difference into consideration.

F-measure Optimization via Cost-sensitive Learning. Parambath et al. (2014) have shown that F-measure can be optimized via cost-sensitive learning, where a cost (importance) is set for each class for adjusting their impact on model learning. However, most previous studies set such costs manually (Anand et al., 1993; Domingos, 1999; Krawczyk et al., 2014) or search them on large scale dataset (Nan et al., 2012; Parambath et al., 2014), whose best settings are not transferable and very time-consuming to find for neural network models. This motivates us to develop a theoretical framework for measuring such importance.

3 Adaptive Scaling for Sparse Detection

This section describes how to effectively optimize neural network detection models via dynamic cost-sensitive learning. Specifically, we first propose a marginal utility based theoretical framework for measuring the importance of positive/negative instances. Then we present our adaptive scaling algorithm, which can leverage the importance of each class for effective and robust training of neural network detection models. Finally, a batch-wise version of our algorithm is proposed to make it can be applied as a plug-in of batch-based neural network training algorithms.

3.1 Marginal Utility based Importance Measuring

Conventional methods commonly deal with the class inequality problem in sparse detection by deemphasizing the importance of negative instances during training. This raises two questions: 1) How to quantify the importance of instances of each class? As mentioned by Parambath et al. (2014), that importance is related to the convergence ability of models, which means that this problem cannot be solved by only considering the distribution of training data. 2) Is the im-

[2]This paper considers micro-averaged metrics. But our conclusions can be easily extended to macro-averaged metrics by scaling above-mentioned coefficients with sample sizes of each class.

portance of positive/negative instances remaining unchanged during the entire training process? If not, how it changes according to the convergence of the model?

To this end, we borrow the idea of *marginal utility* from economics, which means the change of utility from consuming one more unit of product. In detection tasks, we regard its evaluation metric (F-measure) as the utility function. The increment of utility from correctly predicting one more positive instance (marginal positive utility) can be regarded as the relative importance of positive classes, and that from correctly predicting one more negative instance (marginal negative utility) is look upon as the relative importance of the negative class. If marginal positive utility overweighs marginal negative utility, positive instances should be considered more important during optimization because it can lead to more improvement on the evaluation metric. In contrast, if marginal negative utility is higher, training procedure should incline to negative instances since it is more effective for optimizing the evaluation metric.

Formally, we derive marginal positive utility $MU(TP)$ and marginal negative utility $MU(TN)$ by computing the partial derivative of the evaluation metric with respect to TP and TN respectively. For instance, the marginal positive utility $MU_{acc}(TP)$ and the marginal negative utility $MU_{acc}(TN)$ regarding to accuracy metric are:

$$MU_{acc}(TP) = \frac{\partial(\text{Accuracy})}{\partial(TP)} = \frac{1}{P+N} \tag{7}$$

$$MU_{acc}(TN) = \frac{\partial(\text{Accuracy})}{\partial(TN)} = \frac{1}{P+N} \tag{8}$$

We can see that $MU_{acc}(TP)$ and $MU_{acc}(TN)$ are equal and constant regardless of the values of TP and TN. This indicates that, to optimize accuracy, we can simply treat positive and negative instances equally during the training phase, and this is what we exactly do when optimizing cross-entropy loss in Equation 1. For detection problems evaluated using F-measure, we can obtain the marginal utilities from Equation 6 as:

$$MU_{F_\beta}(TP) = \frac{(1+\beta^2)(\beta^2 P + N - TN + PE)}{(\beta^2 P + N - TN + PE + TP)^2} \tag{9}$$

$$MU_{F_\beta}(TN) = \frac{(1+\beta^2)\cdot TP}{(\beta^2 P + N - TN + PE + TP)^2} \tag{10}$$

This result is different from that of accuracy metric. First, $MU_{F_\beta}(TP)$ and $MU_{F_\beta}(TN)$ is no longer equal, indicating that the importance of positive/negative instances to F-measure are different. Besides, it is notable that $MU_{F_\beta}(TP)$ and $MU_{F_\beta}(TN)$ are dynamically changed during the training phase and are highly related to how well current model can fit positive instances and negative instances, i.e., TP and TN.

3.2 Adaptive Scaling Algorithm

In this section, we describe how to incorporate the above importance measures into the training procedure of neural networks, so that it can dynamically adjust weights of positive and negative instances regarding to F-measure.

Specifically, given the current model of neural networks parameterized by θ, let $w_\beta(\theta)$ denote the relative importance of negative instances to positive instances for F_β-measure. Then $w_\beta(\theta)$ can be computed as the ratio of marginal negative utility $MU_{F_\beta}(TN(\theta))$ to the marginal positive utility $MU_{F_\beta}(TP(\theta))$, where $TP(\theta)$ and $TN(\theta)$ are TP and TN on training data with respect to θ-parameterized model:

$$w_\beta(\theta) = \frac{MU_{F_\beta}(TN(\theta))}{MU_{F_\beta}(TP(\theta))} = \frac{TP(\theta)}{\beta^2 P + N - TN(\theta) + PE} \tag{11}$$

Then at each iteration of the model optimization (i.e., each step of gradient descending), we want the model to take next update step proportional to the gradient of the w_β-scaled cross-entropy loss function $\mathcal{L}_{AS}(\theta)$ at the current point:

$$\begin{aligned} \mathcal{L}_{AS}(\theta) = &- \sum_{(x_i,y_i)\in\mathcal{P}} \log p(y_i|x_i;\theta) \\ &- \sum_{(x_i,y_i)\in\mathcal{N}} w_\beta(\theta)\cdot \log p(y_i|x_i;\theta) \end{aligned} \tag{12}$$

Consequently, based on the contributions that correctly predicting one more instances of each class bringing to F-measure, the training procedure dynamically adjusts its attention between positive and negative instances. Thus our adaptive scaling algorithm can take the class inequality characteristic of detection problems into consideration without introducing any additional hyper-parameter[3].

3.3 Properties and Relations to Previous Empirical Conclusions

In this section, we investigate the properties of our adaptive scaling algorithm. By investigating the

[3]Note that β is set according to the applied F_β evaluation metric and therefore is not a hyper-parameter.

change of scaling coefficient $w_\beta(\theta)$ during training, we find that our method has a tight relation to previous empirical conclusions on solving the class inequality problem.

Property 1. *The relative importance of positive/negative instances is related to the ratio of the instance number of each class, as well as how well current model can fit each class.* It is easy to derive that if we fix the accuracies of each classes, $w_\beta(\theta)$ will be smaller if the ratio of the size of negative instances to that of the positive instances (i.e., $\frac{N}{P}$) increases. This indicates that the training procedure should pay more attention to positive instances if the empirical distribution inclines more severely towards negative class, which is identical to conventional practice that we should deemphasize more on negative instances if the positive sparsity problem is more severe (Japkowicz and Stephen, 2002). Besides, $w_\beta(\theta)$ highly depends on TP and TN, which is identical to previous conclusion that the best cost factors are related to the convergence ability of models (Parambath et al., 2014).

Property 2. *For micro-averaged F-measure, all positive instances are equally weighted regardless of the sample size of its class.* Let $MU(TP_i)$ be the marginal utility of positive class c_i, we have:

$$MU_{F_\beta}(TP_i) = \frac{\partial(F_\beta)}{\partial(TP)} \cdot \frac{\partial(TP)}{\partial(TP_i)} = MU_{F_\beta}(TP) \tag{13}$$

This corresponds to the applied micro-averaged F-measure, in which all positive instances are equally considered regardless of the sample size of its class. Thus correctly predicting one more positive instance of any class will result in the same increment of micro-averaged F-measure.

Property 3. *The importance of negative instances increases with the rise of accuracy on positive classes.* This is a straightforward consequence because if the model has higher accuracy on positive instances then it should shift more of its attention to negative ones. Besides, if the accuracy of positive class is close to zero, F-measure will also be close to zero no matter how high the accuracy on negative class is, i.e., correctly predicting negative instances can result in little F-measure increment. Therefore negative instances are inconsequential when the accuracy on positive class is low. And with the increment of positive accuracy, the importance of negative class also increases.

Property 4. *The importance of negative instances increased with the rise of accuracy on the negative class.* This can make the training procedure incline to hard negative instances, which is similar to Focal Loss (Lin et al., 2017). During model convergence, easy negative instances can be correctly classified at the very beginning of training and its loss (negative log probability) will reduce very quickly. This is analogical to removing easy negative instances out of the training procedure and the hard negative instances remaining become more balanced proportional to positive instances. Therefore the importance w_β of remaining hard negative instances are increased to make the model fit them better.

Property 5. *The importance of negative instances increases when more attention is paid to precision than recall.* We can see that w_β decreases with the rise of β, which indicates we focus more on recall than precision. This is identical to practice in sampling heuristics that models should attach more attention to negative instances and sub-sample more of them if evaluation metrics incline more to precision than recall.

3.4 Batch-wise Adaptive Scaling

In large-scale machine learning, batch-wise gradient based algorithm is more popular and efficient for neural network training. This section presents a batch-wise version of our adaptive scaling algorithm, which uses batch-based estimator $\hat{w_\beta}(\theta)$ to replace $w_\beta(\theta)$ in Equation 12.

First, because the main challenge of detection tasks is to identify positive instances from background ones, rather than distinguish between positive classes, we ignore the positive-positive error PE in our experiments. In fact, we found that compared with P and $N - TN$, PE is much smaller and has very limited impact on the final result. Besides, for TP and TN, we approximate them using their expectation on the current batch, which can produce a robust estimation even when the batch size is not large enough. Specifically, let $\mathcal{P}^\mathcal{B} = \{(x_i, y_i)_{i=1}^{P^B}\}$ denotes P^B positive instances and $\mathcal{N}^\mathcal{B} = \{(x_i, y_i)_{i=1}^{N^B}\}$ is N^B negative instances in the batch, we estimate $TP(\theta)$ and $TN(\theta)$ as:

$$TP^B(\theta) = \sum_{(x_i,y_i)\in\mathcal{P}^\mathcal{B}} p(y_i|x_i;\theta) \tag{14}$$

$$TN^B(\theta) = \sum_{(x_i,y_i)\in\mathcal{N}^\mathcal{B}} p(y_i|x_i;\theta) \tag{15}$$

Then we can compute the estimator $\hat{w_\beta}(\theta)$ for $w_\beta(\theta)$ as:

$$\hat{w_\beta}(\theta) = \frac{TP^B(\theta)}{\beta^2 P^B + N^B - TN^B(\theta)} \tag{16}$$

where $\hat{w_\beta}(\theta)$ is computed using only the instances in a batch, which makes it can be directly applied as a plug-in of conventional batch-based neural network optimization algorithm where the loss of negative instances in batch are scaled by $\hat{w_\beta}(\theta)$.

4 Experiments

4.1 Data Preparation

To assess the effectiveness of our method, we conducted experiments on event detection, which is a typical detection task in IE. We used the official evaluation datasets of TAC KBP 2017 Event Nugget Detection Evaluation (LDC2017E55) as test sets, which contains 167 English documents and 167 Chinese documents annotated with Rich ERE annotation standard. For English, we used previously annotated RichERE datasets, including LDC2015E29, LDC2015E68, LDC2016E31 and TAC KBP 2015-2016 Evaluation datasets in LDC2017E02 as the training set. For Chinese, the training set includes LDC2015E105, LDC2015E112, LDC2015E78 and the Chinese part of LDC2017E02. For both Chinese and English, we sampled 20 documents from the evaluation dataset of 2016 year as the development set. Finally, there are 866/20/167 documents in English train/development/test set and 506/20/167 documents in Chinese train/development/test set respectively. We used Stanford CoreNLP toolkit (Manning et al., 2014) for sentence splitting and word segmentation in Chinese.

4.2 Baselines

To verify the effectiveness of our adaptive scaling algorithm, we conducted experiments on two state-of-the-art neural network event detection models. The first one is Dynamic Multipooling Convolutional Neural network (DMCNN) proposed by Chen et al. (2015), a one-layer CNN model with a dynamic multi-pooling operation over convolutional feature maps. The second one is BiLSTM used by Feng et al. (2016) and Yang and Mitchell (2017), where a bidirectional LSTM layer is firstly applied to the input sentence and then word-wise classification is directly conducted on the output of the BiLSTM layer of each word.

We compared our method with following baselines upon above-mentioned two models:

1) **Vanilla models (Vanilla)**, which used the original cross-entropy loss function without any additional treatment for class inequality problem.

2) **Under-sampling (Sampling)**, which samples only part of negative instances as the training data. This is the most widely used solution in event detection (Chen et al., 2015).

3) **Static scaling (Scaling)**, which scales loss of negative instances with a constant. This is a simple but effective cost-sensitive learning method.

4) **Focal Loss (Focal)** (Lin et al., 2017), which scales loss of an instance with a factor proportional to the probability of incorrectly predicting it. This method proves to be effective in some detection problems such as Object Detection.

5) **Softmax-Margin Loss (CLUZH)** (Makarov and Clematide, 2017), which sets additional costs for false-negative error and positive-positive error. This method was used in the 5-model ensembling CLUZH system in TAC KBP 2017 Evaluation. Besides, it also introduced several strong handcraft features, which makes it achieve the best performance on Chinese and very competitive performance on English in the evaluation.

We evaluated all systems with micro-F1 metric computed using the official evaluation toolkit[4]. We reported the average performance of 10 runs (Mean) of each system on the official *type classification* task.[5] We also reported the variance (Var) of the performance to evaluate the stabilities of different methods. As TAC KBP2017 allowed each team to submit 3 different runs, to make our results comparable with evaluation results, we selected 3 best runs of each system on the development set and reported the best test set performance among them, which is referred as *Best3* in this paper. We applied grid search (Hsu et al., 2003) to find best hyper-parameters for all methods.

4.3 Overall results

Table 2 shows the overall results on both English and Chinese. From this table, we can see that:

1) **The class inequality problem is crucial for sparse detection tasks and requires special consideration.** Compared with vanilla models, all

[4]`github.com/hunterhector/EvmEval/tarball/master`

[5]Realis classification, another task in the evaluation, can be regarded as a standard classification task without background class, so we didn't include it here.

Model	English			Chinese		
	Mean	Var	Best3	Mean	Var	Best3
CLUZH*	-	-	**48.60**	-	-	50.14
BiLSTM						
Vanilla	41.91	1.40	43.27	44.23	1.88	47.13
Focal	43.23	0.52	44.65	44.37	4.45	46.90
Sampling	46.66	0.27	47.70	48.97	0.97	50.24
Scaling	46.61	0.35	47.71	48.87	0.83	49.99
A-Scaling	47.48	0.20	48.11	49.19	0.46	50.40
DMCNN						
Vanilla	44.41	2.21	47.12	44.85	5.63	48.16
Focal	45.24	1.38	47.33	44.61	7.59	49.74
Sampling	46.83	0.23	47.65	50.77	2.34	52.50
Scaling	47.06	1.92	48.07	51.38	0.74	52.49
A-Scaling	**47.60**	**0.16**	48.31	**51.87**	**0.39**	**52.99**

Table 2: Experiment results on TAC KBP 2017 evaluation datasets. * indicates the best (ensembling) results reported in the original paper. "A-Scaling" is batch-wise adaptive scaling algorithm.

other methods trying to tackle this problem have shown significant improvements on both models and both languages, especially on Chinese dataset where the positive sparsity problem is more severe (Makarov and Clematide, 2017).

2) **It is critical to take the different roles of classes into consideration for F-measure optimization.** Even down-weighting the loss assigned to well-classified examples can alleviate the positive sparsity problem by deemphasizing easy negative instances during optimization, Focal Loss cannot achieve competitive performance because it does not distinguish between different classes.

3) **Marginal Utility based framework provides a solid foundation for measuring instance importance, thus makes our adaptive scaling algorithm steadily outperform all heuristic baselines.** No matter on mean or Best3 metric, adaptive scaling steadily outperforms other baselines on both BiLSTM and DMCNN model. Furthermore, we can see that simple models with adaptive scaling outperform the state-of-the-art CLUZH system on Chinese (which has more severe positive sparsity problem) and achieve comparable results with it on English. Please note that CLUZH is an ensemble of five models and uses extra hand-crafted features. This verified the effectiveness of our adaptive scaling algorithm.

4) **Our adaptive scaling algorithm doesn't need additional hyper-parameters and the importance of instances is dynamically estimated. This leads to a more stable and transferable solution for detection model optimization.** First, we can see that adaptive scaling has the lowest

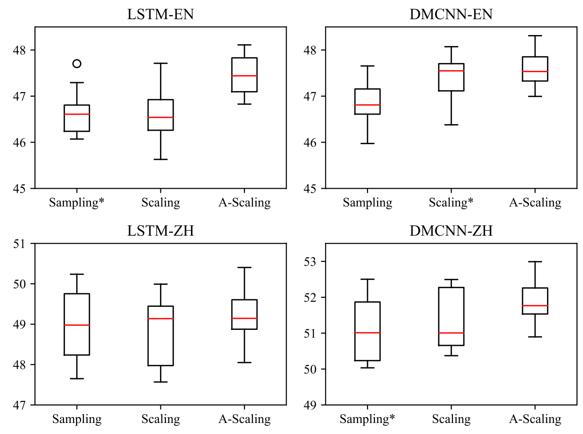


Figure 1: Box plots of three different methods. * indicates outliers not shown in the figure exist.

variance among all methods, which means that it is more stable than other methods. Besides, adaptive scaling doesn't introduce any additional hyper-parameters. In contrast, in experiment we found that the best hyper-parameters for undersampling (the ratio of sampled negative instances to positive instances) and static scaling (the prior cost for negative instances) remarkably varied from models and datasets.

4.4 Stability Analysis

This section investigated the stability of different methods. Table 2 have shown that adaptive scaling has a much smaller variance than other baselines. To investigate its reason, Figure 1 shows the box plots of adaptive scaling and other heuristic methods on both models and both languages.

We can see that interquartile ranges (i.e., the difference between 75th and 25th percentiles of data) of the performances of adaptive scaling are smaller than other methods. In all groups of experiments, the performances of our adaptive scaling algorithm are with a smaller fluctuation. This demonstrates the stability of adaptive scaling algorithm. Furthermore, we found that conventional methods are more instable on Chinese dataset where the data distribution is more skewed. We believe that this is because:

1) Under-Sampling might undermine the inner sub-concept structure of negative class by simply dropping negative instances, and its performance depends on the quality of sampled data, which can result in the instability.

2) Static scaling sets the importance of negative instances statically in the entire training procedure. However, as shown in Section 3, the rel-

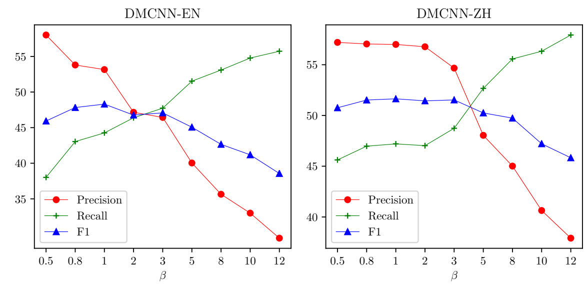


Figure 2: Change of Precision, Recall and F1 regarding to β using adaptive scaling on DMCNN.

ative importance between different classes is dynamically changed during the training procedure, which makes static scaling incapable of achieving stable performance in different phases of training.

3) Adaptive scaling achieves more stable performance during the entire training procedure. First, it doesn't drop any instances, so it can maintain the inner structure of negative class without any information loss. Besides, our algorithm can dynamically adjust the scaling factor during training, therefore can automatically shift attention between positive and negative classes according to the convergence state of the model.

4.5 Adaptability on Different β

Figure 2 shows the change of Precision, Recall and F1 measures regarding to different β. We can see that when β increases, the precision decreased and the recall increased by contrast. This is identical to the nature of F_β where β represents the relative importance of precision and recall. Furthermore, adaptive scaling with $\beta = 1$ achieved the best performance on F_1 measure. This further demonstrates that w_β derived from our marginal utility framework is a good and adaptive estimator for the relative importance of the negative class to positive classes of F_β measure.

5 Related Work

This paper proposes adaptive scaling algorithm for sparse detection problem. Related work to this paper mainly includes:

Classification on Imbalanced Data. Conventional approaches addressed data imbalance from either data-level or algorithm-level. Data-level approaches resample the training data to maintain the balance between different classes (Japkowicz and Stephen, 2002; Drummond et al., 2003). Further improvements on this direction involve how to better sampling data with minimum information loss (Carvajal et al., 2004; Estabrooks et al., 2004; Han et al., 2005; Fernández-Navarro et al., 2011). Algorithm-level approaches attempt to choose an appropriate inductive bias on models or algorithms to make them more suitable on data imbalance condition, including instance weighting (Ting, 2002; Lin et al., 2017), cost-sensitive learning (Anand et al., 1993; Domingos, 1999; Sun et al., 2007; Krawczyk et al., 2014) and active learning approaches (Ertekin et al., 2007a,b; Zhu and Hovy, 2007).

F-Measure Optimization. Previous research on F-measure optimization mainly fell into two paradigms (Nan et al., 2012): 1) Decision-theoretic approaches (DTA), which first estimate a probability model and find the optimal predictions according to that model (Joachims, 2005; Jansche, 2005, 2007; Dembczynski et al., 2011; Busa-Fekete et al., 2015; Natarajan et al., 2016). The main drawback of these methods is that they need to estimate the joint probability with exponentially many combinations, thus make them hard to use in practice; 2) Empirical utility maximization (EUM) approaches, which adapt approximate methods to find a best classifier in hypothesises (Musicant et al., 2003; Chinta et al., 2013; Parambath et al., 2014; Narasimhan et al., 2014). However, EUM methods depend on thresholds or costs that are not known a priori so time-consuming searching on large development set is required. Our adaptive scaling algorithm is partially inspired by EUM approaches, but is based on the marginal utility framework, which doesn't introduce any additional hyper-parameter or searching procedure.

Neural Network based Event Detection. Event detection is a typical task of detection problems. Recently neural network based methods have achieved significant progress in Event Detection. CNNs (Chen et al., 2015; Nguyen and Grishman, 2015) and Bi-LSTMs (Zeng et al., 2016; Yang and Mitchell, 2017) are two effective and widely used models. Some improvements have been made by jointly predicting triggers and arguments (Nguyen et al., 2016) or introducing more complicated architectures to capture larger scale of contexts (Feng et al., 2016; Nguyen and Grishman, 2016; Ghaeini et al., 2016).

6 Conclusions

This paper proposes adaptive scaling algorithm for detection tasks, which can deal with its positive sparsity problem and directly optimize F-measure by adaptively scaling the influence of negative instances in loss function. Based on the marginal utility theory framework, our method leads to more effective, stable and transferable optimization of neural networks without introducing additional hyper-parameters. Experiments on event detection verified the effectiveness and stability of our adaptive scaling algorithm.

The divergence between loss functions and evaluation metrics is common in NLP and machine learning. In the future we want to apply our marginal utility based framework to other metrics, such as Mean Average Precision (MAP).

Acknowledgments

We sincerely thank the reviewers for their valuable comments. Moreover, this work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505, and the Young Elite Scientists Sponsorship Program no. YESS20160177.

References

Rangachari Anand, Kishan G Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. 1993. An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*, 4(6):962–969.

Róbert Busa-Fekete, Balázs Szörényi, Krzysztof Dembczynski, and Eyke Hüllermeier. 2015. Online f-measure optimization. In *Advances in Neural Information Processing Systems*, pages 595–603.

K Carvajal, M Chacón, D Mery, and G Acuna. 2004. Neural network method for failure detection with skewed class distribution. *Insight-Non-Destructive Testing and Condition Monitoring*, 46(7):399–402.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL 2015*.

Punya Murthy Chinta, P Balamurugan, Shirish Shevade, and M Narasimha Murty. 2013. Optimizing f-measure with non-convex loss and sparse linear classifiers. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Krzysztof J Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2011. An exact algorithm for f-measure maximization. In *Advances in neural information processing systems*, pages 1404–1412.

Pedro M. Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In *KDD*.

Chris Drummond, Robert C Holte, et al. 2003. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer.

Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. 2007a. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM.

Seyda Ertekin, Jian Huang, and C Lee Giles. 2007b. Active learning for class imbalance problem. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 823–824. ACM.

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. 2004. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of ACL 2016*.

Francisco Fernández-Navarro, César Hervás-Martínez, and Pedro Antonio Gutiérrez. 2011. A dynamic over-sampling procedure based on sensitivity for multi-class problems. *Pattern Recognition*, 44(8):1821–1833.

Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of ACL 2016*.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer.

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification.

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

Martin Jansche. 2005. Maximum expected f-measure training of logistic regression models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 692–699. Association for Computational Linguistics.

Martin Jansche. 2007. A maximum expected utility framework for binary sequence labeling. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 736–743.

Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM.

Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. 2014. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*.

Peter Makarov and Simon Clematide. 2017. UZH at TAC KBP 2017: Event nugget detection via joint learning with softmax-margin objective. In *Proceedings of TAC 2017*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *In Proceedings of ACL 2014*.

David R Musicant, Vipin Kumar, Aysel Ozgur, et al. 2003. Optimizing f-measure with support vector machines. In *FLAIRS conference*, pages 356–360.

Ye Nan, Kian Ming Chai, Wee Sun Lee, and Hai Leong Chieu. 2012. Optimizing f-measure: A tale of two approaches. *arXiv preprint arXiv:1206.4625*.

Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. 2014. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Advances in Neural Information Processing Systems*, pages 1493–1501.

Nagarajan Natarajan, Oluwasanmi Koyejo, Pradeep Ravikumar, and Inderjit Dhillon. 2016. Optimal classification with multivariate losses. In *International Conference on Machine Learning*, pages 1530–1538.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT 2016*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL 2015*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of EMNLP 2016*.

Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. 2014. Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, pages 2123–2131.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98.

George J Stigler. 1950. The development of utility theory. i. *Journal of Political Economy*, 58(4):307–327.

Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Kai Ming Ting. 2002. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Ying Zeng, Honghui Yang, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2016. A convolution bilstm neural network model for chinese event extraction. In *Proceedings of NLPCC-ICCPOL 2016*.

Jingbo Zhu and Eduard Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Strong Baselines for Neural Semi-Supervised Learning under Domain Shift

Sebastian Ruder[♠♣] **Barbara Plank**[♡◇]
[♠]Insight Research Centre, National University of Ireland, Galway, Ireland
[♣]Aylien Ltd., Dublin, Ireland
[♡]Center for Language and Cognition, University of Groningen, The Netherlands
[◇]Department of Computer Science, IT University of Copenhagen, Denmark
sebastian@ruder.io,bplank@itu.dk

Abstract

Novel neural models have been proposed in recent years for learning under domain shift. Most models, however, only evaluate on a single task, on proprietary datasets, or compare to weak baselines, which makes comparison of models difficult. In this paper, we re-evaluate classic general-purpose bootstrapping approaches in the context of neural networks under domain shifts vs. recent neural approaches and propose a novel *multi-task tri-training* method that reduces the time and space complexity of classic tri-training. Extensive experiments on two benchmarks are negative: while our novel method establishes a new state-of-the-art for sentiment analysis, it does not fare consistently the best. More importantly, we arrive at the somewhat surprising conclusion that classic tri-training, with some additions, outperforms the state of the art. We conclude that classic approaches constitute an important and strong baseline.

1 Introduction

Deep neural networks (DNNs) excel at learning from labeled data and have achieved state of the art in a wide array of supervised NLP tasks such as dependency parsing (Dozat and Manning, 2017), named entity recognition (Lample et al., 2016), and semantic role labeling (He et al., 2017).

In contrast, learning from unlabeled data, especially under domain shift, remains a challenge. This is common in many real-world applications where the distribution of the training and test data differs. Many state-of-the-art domain adaptation approaches leverage task-specific characteristics such as sentiment words (Blitzer et al., 2006; Wu and Huang, 2016) or distributional features (Schnabel and Schütze, 2014; Yin et al., 2015) which do not generalize to other tasks. Other approaches that are in theory more general only evaluate on proprietary datasets (Kim et al., 2017) or on a single benchmark (Zhou et al., 2016), which carries the risk of overfitting to the task. In addition, most models only compare against weak baselines and, strikingly, almost none considers evaluating against approaches from the extensive semi-supervised learning (SSL) literature (Chapelle et al., 2006).

In this work, we make the argument that such algorithms make strong baselines for any task in line with recent efforts highlighting the usefulness of classic approaches (Melis et al., 2017; Denkowski and Neubig, 2017). We re-evaluate bootstrapping algorithms in the context of DNNs. These are general-purpose semi-supervised algorithms that treat the model as a black box and can thus be used easily—with a few additions—with the current generation of NLP models. Many of these methods, though, were originally developed with in-domain performance in mind, so their effectiveness in a domain adaptation setting remains unexplored.

In particular, we re-evaluate three traditional bootstrapping methods, self-training (Yarowsky, 1995), tri-training (Zhou and Li, 2005), and tri-training with disagreement (Søgaard, 2010) for neural network-based approaches on *two* NLP tasks with different characteristics, namely, a sequence prediction and a classification task (POS tagging and sentiment analysis). We evaluate the methods across multiple domains on two well-established benchmarks, without taking any further task-specific measures, and compare to the best results published in the literature.

We make the somewhat surprising observation that classic tri-training outperforms task-agnostic state-of-the-art semi-supervised learning (Laine and Aila, 2017) and recent neural adaptation approaches (Ganin et al., 2016; Saito et al., 2017).

In addition, we propose *multi-task tri-training*, which reduces the main deficiency of tri-training, namely its time and space complexity. It establishes a new state of the art on unsupervised domain adaptation for sentiment analysis but it is outperformed by classic tri-training for POS tagging.

Contributions Our contributions are: a) We propose a novel multi-task tri-training method. b) We show that tri-training can serve as a strong and robust semi-supervised learning baseline for the current generation of NLP models. c) We perform an extensive evaluation of bootstrapping[1] algorithms compared to state-of-the-art approaches on two benchmark datasets. d) We shed light on the task and data characteristics that yield the best performance for each model.

2 Neural bootstrapping methods

We first introduce three classic bootstrapping methods, self-training, tri-training, and tri-training with disagreement and detail how they can be used with neural networks. For in-depth details we refer the reader to (Abney, 2007; Chapelle et al., 2006; Zhu and Goldberg, 2009). We introduce our novel multi-task tri-training method in §2.3.

2.1 Self-training

Self-training (Yarowsky, 1995; McClosky et al., 2006b) is one of the earliest and simplest bootstrapping approaches. In essence, it leverages the model's own predictions on unlabeled data to obtain additional information that can be used during training. Typically the most confident predictions are taken at face value, as detailed next.

Self-training trains a model m on a labeled training set L and an unlabeled data set U. At each iteration, the model provides predictions $m(x)$ in the form of a probability distribution over classes for all unlabeled examples x in U. If the probability assigned to the most likely class is higher than a predetermined threshold τ, x is added to the labeled examples with $p(x) = \arg\max m(x)$ as pseudo-label. This instantiation is the most widely used and shown in Algorithm 1.

Calibration It is well-known that output probabilities in neural networks are poorly calibrated (Guo et al., 2017). Using a fixed threshold τ is thus not the best choice. While the *absolute* confidence value is inaccurate, we can expect that the *relative* order of confidences is more robust.

For this reason, we select the top n unlabeled examples that have been predicted with the highest confidence after every epoch and add them to the labeled data. This is one of the many variants for self-training, called *throttling* (Abney, 2007). We empirically confirm that this outperforms the classic selection in our experiments.

Algorithm 1 Self-training (Abney, 2007)

```
1: repeat
2:     $m \leftarrow train\_model(L)$
3:     for $x \in U$ do
4:         if $\max m(x) > \tau$ then
5:             $L \leftarrow L \cup \{(x, p(x))\}$
6: until no more predictions are confident
```

Online learning In contrast to many classic algorithms, DNNs are trained online by default. We compare training setups and find that training until convergence on labeled data and then training until convergence using self-training performs best.

Classic self-training has shown mixed success. In parsing it proved successful only with small datasets (Reichart and Rappoport, 2007) or when a generative component is used together with a reranker in high-data conditions (McClosky et al., 2006b; Suzuki and Isozaki, 2008). Some success was achieved with careful task-specific data selection (Petrov and McDonald, 2012), while others report limited success on a variety of NLP tasks (Plank, 2011; Van Asch and Daelemans, 2016; van der Goot et al., 2017). Its main downside is that the model is not able to correct its own mistakes and errors are amplified, an effect that is increased under domain shift.

2.2 Tri-training

Tri-training (Zhou and Li, 2005) is a classic method that reduces the bias of predictions on unlabeled data by utilizing the agreement of three independently trained models. Tri-training (cf. Algorithm 2) first trains three models m_1, m_2, and m_3 on bootstrap samples of the labeled data L. An unlabeled data point is added to the training set of a model m_i if the other two models m_j and m_k agree on its label. Training stops when the classifiers do not change anymore.

Tri-training *with disagreement* (Søgaard, 2010)

[1]We use the term bootstrapping as used in the semi-supervised learning literature (Zhu, 2005), which should not be confused with the statistical procedure of the same name (Efron and Tibshirani, 1994).

Algorithm 2 Tri-training (Zhou and Li, 2005)

```
1:  for $i \in \{1..3\}$ do
2:      $S_i \leftarrow bootstrap\_sample(L)$
3:      $m_i \leftarrow train\_model(S_i)$
4:  repeat
5:      for $i \in \{1..3\}$ do
6:          $L_i \leftarrow \emptyset$
7:          for $x \in U$ do
8:              if $p_j(x) = p_k(x)(j, k \neq i)$ then
9:                  $L_i \leftarrow L_i \cup \{(x, p_j(x))\}$
            $m_i \leftarrow train\_model(L \cup L_i)$
10: until none of $m_i$ changes
11: apply majority vote over $m_i$
```

is based on the intuition that a model should only be strengthened in its weak points and that the labeled data should not be skewed by easy data points. In order to achieve this, it adds a simple modification to the original algorithm (altering line 8 in Algorithm 2), requiring that for an unlabeled data point on which m_j and m_k *agree*, the other model m_i *disagrees* on the prediction. Tri-training with disagreement is more data-efficient than tri-training and has achieved competitive results on part-of-speech tagging (Søgaard, 2010).

Sampling unlabeled data Both tri-training and tri-training with disagreement can be very expensive in their original formulation as they require to produce predictions for each of the three models on all unlabeled data samples, which can be in the millions in realistic applications. We thus propose to sample a number of unlabeled examples at every epoch. For all traditional bootstrapping approaches we sample 10k candidate instances in each epoch. For the neural approaches we use a linearly growing candidate sampling scheme proposed by (Saito et al., 2017), increasing the candidate pool size as the models become more accurate.

Confidence thresholding Similar to self-training, we can introduce an additional requirement that pseudo-labeled examples are only added if the probability of the prediction of at least one model is higher than some threshold τ. We did not find this to outperform prediction without threshold for traditional tri-training, but thresholding proved essential for our method (§2.3).

The most important condition for tri-training and tri-training with disagreement is that the models are diverse. Typically, bootstrap samples are used to create this diversity (Zhou and Li, 2005; Søgaard, 2010). However, training separate models on bootstrap samples of a potentially large amount of training data is expensive and takes a lot of time. This drawback motivates our approach.

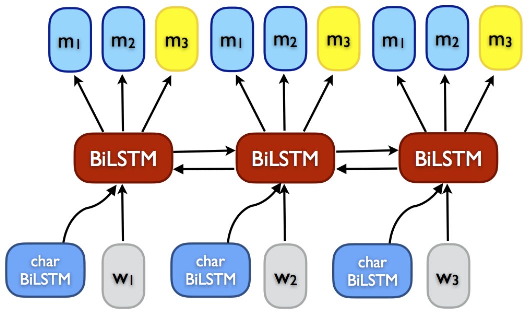


Figure 1: Multi-task tri-training (MT-Tri).

2.3 Multi-task tri-training

In order to reduce both the time and space complexity of tri-training, we propose Multi-task Tri-training (MT-Tri). MT-Tri leverages insights from multi-task learning (MTL) (Caruana, 1993) to share knowledge across models and accelerate training. Rather than storing and training each model separately, we propose to share the parameters of the models and train them jointly using MTL.[2] All models thus collaborate on learning a joint representation, which improves convergence.

The output softmax layers are model-specific and are only updated for the input of the respective model. We show the model in Figure 1 (as instantiated for POS tagging). As the models leverage a joint representation, we need to ensure that the features used for prediction in the softmax layers of the different models are as diverse as possible, so that the models can still learn from each other's predictions. In contrast, if the parameters in all output softmax layers were the same, the method would degenerate to self-training.

To guarantee diversity, we introduce an orthogonality constraint (Bousmalis et al., 2016) as an additional loss term, which we define as follows:

$$\mathcal{L}_{orth} = \|W_{m_1}^{\top} W_{m_2}\|_F^2 \tag{1}$$

where $|\cdot\|_F^2$ is the squared Frobenius norm and W_{m_1} and W_{m_2} are the softmax output parameters

[2]Note: we use the term multi-task learning here albeit all tasks are of the same kind, similar to work on multi-lingual modeling treating each language (but same label space) as separate task e.g., (Fang and Cohn, 2017). It is interesting to point out that our model is further doing implicit multi-view learning by way of the orthogonality constraint.

of the two source and pseudo-labeled output layers m_1 and m_2, respectively. The orthogonality constraint encourages the models not to rely on the same features for prediction. As enforcing pair-wise orthogonality between three matrices is not possible, we only enforce orthogonality between the softmax output layers of m_1 and m_2,[3] while m_3 is gradually trained to be more target-specific. We parameterize $\mathcal{L}_{orth}$ by γ=0.01 following (Liu et al., 2017). We do not further tune γ.

More formally, let us illustrate the model by taking the sequence prediction task (Figure 1) as illustration. Given an utterance with labels $y_1, .., y_n$, our Multi-task Tri-training loss consists of three task-specific (m_1, m_2, m_3) tagging loss functions (where $\vec{h}$ is the uppermost Bi-LSTM encoding):

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i}\sum_{1,..,n} \log P_{m_i}(y|\vec{h}) + \gamma\mathcal{L}_{orth} \quad (2)$$

In contrast to classic tri-training, we can train the multi-task model with its three model-specific outputs jointly and *without* bootstrap sampling on the labeled source domain data until convergence, as the orthogonality constraint enforces different representations between models m_1 and m_2. From this point, we can leverage the pair-wise agreement of two output layers to add pseudo-labeled examples as training data to the third model. We train the third output layer m_3 only on pseudo-labeled target instances in order to make tri-training more robust to a domain shift. For the final prediction, majority voting of all three output layers is used, which resulted in the best instantiation, together with confidence thresholding ($\tau = 0.9$, except for high-resource POS where $\tau = 0.8$ performed slightly better). We also experimented with using a domain-adversarial loss (Ganin et al., 2016) on the jointly learned representation, but found this not to help. The full pseudo-code is given in Algorithm 3.

Computational complexity The motivation for MT-Tri was to reduce the space and time complexity of tri-training. We thus give an estimate of its efficiency gains. MT-Tri is ~3× more space-efficient than regular tri-training; tri-training stores one set of parameters for each of the three models, while MT-Tri only stores one set of parameters (we use three output layers, but these make up a comparatively small part of the total parameter budget). In terms of time efficiency, tri-training first requires to train each of the models from scratch. The actual tri-training takes about the same time as training from scratch and requires a separate forward pass for each model, effectively training three independent models simultaneously. In contrast, MT-Tri only necessitates one forward pass as well as the evaluation of the two additional output layers (which takes a negligible amount of time) and requires about as many epochs as tri-training until convergence (see Table 3, second column) while adding fewer unlabeled examples per epoch (see Section 3.4). In our experiments, MT-Tri trained about 5-6× faster than traditional tri-training.

Algorithm 3 Multi-task Tri-training

```
1:  $m \leftarrow train\_model(L)$
2:  **repeat**
3:      **for** $i \in \{1..3\}$ **do**
4:          $L_i \leftarrow \emptyset$
5:          **for** $x \in U$ **do**
6:              **if** $p_j(x) = p_k(x)(j, k \neq i)$ **then**
7:                  $L_i \leftarrow L_i \cup \{(x, p_j(x))\}$
8:          **if** $i = 3$ **then** $m_i = train\_model(L_i)$
9:          **else** $m_i \leftarrow train\_model(L \cup L_i)$
10: **until** end condition is met
11: apply majority vote over $m_i$
```

MT-Tri can be seen as a self-ensembling technique, where different variations of a model are used to create a stronger ensemble prediction. Recent approaches in this line are *snapshot ensembling* (Huang et al., 2017) that ensembles models converged to different minima during a training run, *asymmetric tri-training* (Saito et al., 2017) (ASYM) that leverages agreement on two models as information for the third, and *temporal ensembling* (Laine and Aila, 2017), which ensembles predictions of a model at different epochs. We tried to compare to temporal ensembling in our experiments, but were not able to obtain consistent results.[4] We compare to the closest most recent method, asymmetric tri-training (Saito et al., 2017). It differs from ours in two aspects: a) ASYM leverages only pseudo-labels from data points on which m_1 and m_2 agree, and b) it uses only one task (m_3) as final predictor. In essence, our formulation of MT-Tri is closer to the original tri-training formulation (agreements on two provide pseudo-labels to the third) thereby incorporating more diversity.

[3]We also tried enforcing orthogonality on a hidden layer rather than the output layer, but this did not help.

[4]We suspect that the sparse features in NLP and the domain shift might be detrimental to its unsupervised consistency loss.

	Domain	# labeled	# unlabeled
POS tagging	Answers	3,489	27,274
	Emails	4,900	1,194,173
	Newsgroups	2,391	1,000,000
	Reviews	3,813	1,965,350
	Weblogs	2,031	524,834
	WSJ	30,060	100,000
Sentiment	Book	2,000	4,465
	DVD	2,000	3,586
	Electronics	2,000	5,681
	Kitchen	2,000	5,945

Table 1: Number of labeled and unlabeled sentences for each domain in the SANCL 2012 dataset (Petrov and McDonald, 2012) for POS tagging (above) and the Amazon Reviews dataset (Blitzer et al., 2006) for sentiment analysis (below).

3 Experiments

In order to ascertain which methods are robust across different domains, we evaluate on two widely used unsupervised domain adaptation datasets for two tasks, a sequence labeling and a classification task, cf. Table 1 for data statistics.

3.1 POS tagging

For POS tagging we use the SANCL 2012 shared task dataset (Petrov and McDonald, 2012) and compare to the top results in both low and high-data conditions (Schnabel and Schütze, 2014; Yin et al., 2015). Both are strong baselines, as the FLORS tagger has been developed for this challenging dataset and it is based on contextual distributional features (excluding the word's identity), and hand-crafted suffix and shape features (including some language-specific morphological features). We want to gauge to what extent we can adopt a nowadays fairly standard (but more lexicalized) general neural tagger.

Our POS tagging model is a state-of-the-art Bi-LSTM tagger (Plank et al., 2016) with word and 100-dim character embeddings. Word embeddings are initialized with the 100-dim Glove embeddings (Pennington et al., 2014). The BiLSTM has one hidden layer with 100 dimensions. The base POS model is trained on WSJ with early stopping on the WSJ development set, using patience 2, Gaussian noise with $\sigma = 0.2$ and word dropout with $p = 0.25$ (Kiperwasser and Goldberg, 2016).

Regarding data, the source domain is the Ontonotes 4.0 release of the Penn treebank Wall Street Journal (WSJ) annotated for 48 fine-grained POS tags. This amounts to 30,060 labeled sentences. We use 100,000 WSJ sentences from 1988 as unlabeled data, following Schnabel and Schütze (2014).[5] As target data, we use the five SANCL domains (answers, emails, newsgroups, reviews, weblogs). We restrict the amount of unlabeled data for each SANCL domain to the first 100k sentences, and do not do any pre-processing. We consider the development set of ANSWERS as our only target dev set to set hyperparameters. This may result in suboptimal per-domain settings but better resembles an unsupervised adaptation scenario.

3.2 Sentiment analysis

For sentiment analysis, we evaluate on the Amazon reviews dataset (Blitzer et al., 2006). Reviews with 1 to 3 stars are ranked as negative, while reviews with 4 or 5 stars are ranked as positive. The dataset consists of four domains, yielding 12 adaptation scenarios. We use the same pre-processing and architecture as used in (Ganin et al., 2016; Saito et al., 2017): 5,000-dimensional tf-idf weighted unigram and bigram features as input; 2k labeled source samples and 2k unlabeled target samples for training, 200 labeled target samples for validation, and between 3k-6k samples for testing. The model is an MLP with one hidden layer with 50 dimensions, sigmoid activations, and a softmax output. We compare against the Variational Fair Autoencoder (VFAE) (Louizos et al., 2015) model and domain-adversarial neural networks (DANN) (Ganin et al., 2016).

3.3 Baselines

Besides comparing to the top results published on both datasets, we include the following baselines:

a) the task model trained on the source domain;
b) self-training (Self);
c) tri-training (Tri);
d) tri-training with disagreement (Tri-D); and
e) asymmetric tri-training (Saito et al., 2017).

Our proposed model is multi-task tri-training (MT-Tri). We implement our models in DyNet (Neubig et al., 2017). Reporting single evaluation scores might result in biased results (Reimers and Gurevych, 2017). Throughout the paper, we report mean accuracy and standard deviation over five runs for POS tagging and over ten runs for

[5]Note that our unlabeled data might slightly differ from theirs. We took the first 100k sentences from the 1988 WSJ dataset from the BLLIP 1987-89 WSJ Corpus Release 1.

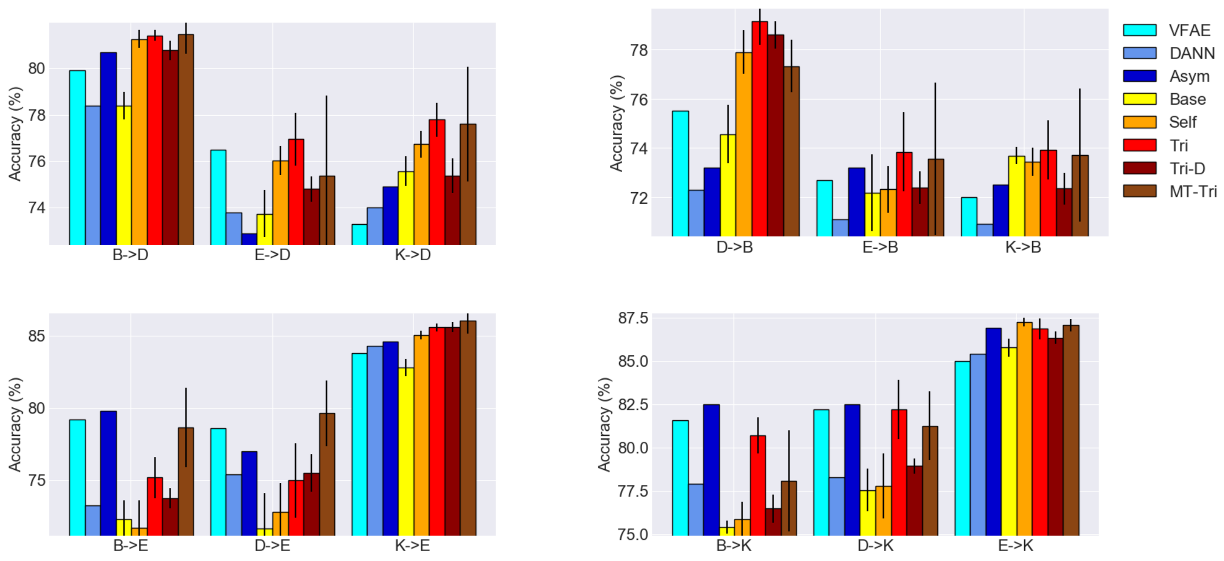


Figure 2: Average results for unsupervised domain adaptation on the Amazon dataset. Domains: B (Book), D (DVD), E (Electronics), K (Kitchen). Results for VFAE, DANN, and Asym are from Saito et al. (2017).

sentiment analysis. Significance is computed using bootstrap test. The code for all experiments is released at: `https://github.com/bplank/semi-supervised-baselines`.

3.4 Results

Sentiment analysis We show results for sentiment analysis for all 12 domain adaptation scenarios in Figure 2. For clarity, we also show the accuracy scores averaged across each target domain as well as a global macro average in Table 2.

Model	D	B	E	K	Avg
VFAE*	76.57	73.40	80.53	82.93	78.36
DANN*	75.40	71.43	77.67	80.53	76.26
Asym*	76.17	72.97	80.47	**83.97**	78.39
Src	75.91	73.47	75.61	79.58	76.14
Self	78.00	74.55	76.54	80.30	77.35
Tri	**78.72**	**75.64**	78.60	83.26	79.05
Tri-D	76.99	74.44	78.30	80.59	77.58
MT-Tri	78.14	74.86	**81.45**	82.14	**79.15**

Table 2: Average accuracy scores for each SA target domain. *: result from Saito et al. (2017).

Self-training achieves surprisingly good results but is not able to compete with tri-training. Tri-training with disagreement is only slightly better than self-training, showing that the disagreement component might not be useful when there is a strong domain shift. Tri-training achieves the best average results on two target domains and clearly outperforms the state of the art on average.

MT-Tri finally outperforms the state of the art on 3/4 domains, and even slightly traditional tri-training, resulting in the overall best method. This improvement is mainly due to the B->E and D->E scenarios, on which tri-training struggles. These domain pairs are among those with the highest $\mathcal{A}$-distance (Blitzer et al., 2007), which highlights that tri-training has difficulty dealing with a strong shift in domain. Our method is able to mitigate this deficiency by training one of the three output layers only on pseudo-labeled target domain examples.

In addition, MT-Tri is more efficient as it adds a smaller number of pseudo-labeled examples than tri-training at every epoch. For sentiment analysis, tri-training adds around 1800-1950/2000 unlabeled examples at every epoch, while MT-Tri only adds around 100-300 in early epochs. This shows that the orthogonality constraint is useful for inducing diversity. In addition, adding fewer examples poses a smaller risk of swamping the learned representations with useless signals and is more akin to fine-tuning, the standard method for supervised domain adaptation (Howard and Ruder, 2018).

We observe an asymmetry in the results between some of the domain pairs, e.g. B->D and D->B. We hypothesize that the asymmetry may be due to properties of the data and that the domains are relatively far apart e.g., in terms of $\mathcal{A}$-distance. In fact, asymmetry in these domains is already reflected

		Target domains							
Model	*ep*	Answers	Emails	Newsgroups	Reviews	Weblogs	Avg	WSJ	μ_{pseudo}
Src (+glove)		87.63 ±.37	86.49 ±.35	**88.60** ±.22	90.12 ±.32	92.85 ±.17	89.14 ±.28	95.49 ±.09	—
Self	(5)	87.64 ±.18	86.58 ±.30	88.42 ±.24	90.03 ±.11	92.80 ±.19	89.09 ±.20	95.36 ±.07	.5k
Tri	(4)	88.42 ±.16	87.46 ±.20	87.97 ±.09	90.72 ±.14	93.40 ±.15	89.56 ±.16	95.94 ±.07	20.5k
Tri-D	(7)	**88.50** ±.04	**87.63** ±.15	88.12 ±.05	**90.76** ±.10	**93.51** ±.06	**89.70** ±.08	**95.99** ±.03	7.7K
Asym	(3)	87.81 ±.19	86.97 ±.17	87.74 ±.24	90.16 ±.17	92.73 ±.16	89.08 ±.19	95.55 ±.12	1.5k
MT-Tri	(4)	87.92 ±.18	87.20 ±.23	87.73 ±.37	90.27 ±.10	92.96 ±.07	89.21 ±.19	95.50 ±.06	7.6k
FLORS		89.71	88.46	89.82	92.10	94.20	90.86	95.80	—

Table 3: Accuracy scores on dev set of target domain for POS tagging for 10% labeled data. Avg: average over the 5 SANCL domains. Hyperparameter *ep* (epochs) is tuned on Answers dev. μ_{pseudo}: average amount of added pseudo-labeled data. FLORS: results for Batch (u:big) from (Yin et al., 2015) (see §3).

	Target domains dev sets					Avg on	
Model	**Answers**	**Emails**	**Newsgroups**	**Reviews**	**Weblogs**	**targets**	**WSJ**
TnT*	88.55	88.14	88.66	90.40	93.33	89.82	95.75
Stanford*	88.92	88.68	89.11	91.43	94.15	90.46	96.83
Src	88.84 ±.15	88.24 ±.12	89.45 ±.23	91.24 ±.03	93.92 ±.17	90.34 ±.14	96.69 ±.08
Tri	89.34 ±.18	88.83 ±.07	89.32 ±.21	91.62 ±.06	94.40 ±.06	90.70 ±.12	96.84 ±.04
Tri-D	89.35 ±.16	88.66 ±.09	89.29 ±.12	91.58 ±.05	94.32 ±.05	90.62 ±.09	96.85 ±.06
Src (+glove)	89.35 ±.16	88.55 ±.14	**90.12** ±.31	91.48 ±.15	94.48 ±.07	90.80 ±.17	96.90 ±.04
Tri	**90.00** ±.03	**89.06** ±.16	90.04 ±.25	**91.98** ±.11	**94.74** ±.06	**91.16** ±.12	**96.99** ±.02
Tri-D	89.80 ±.19	88.85 ±.10	90.03 ±.22	**91.98** ±.09	94.70 ±.05	91.01 ±.13	96.95 ±.05
Asym	89.51 ±.15	88.47 ±.19	89.26 ±.16	91.60 ±.20	94.28 ±.15	90.62 ±.17	96.56 ±.01
MT-Tri	89.45 ±.05	88.65 ±.04	89.40 ±.22	91.63 ±.23	94.41 ±.05	90.71 ±.12	97.37 ±.07
FLORS*	90.30	89.44	90.86	92.95	94.71	91.66	96.59
	Target domains test sets					**Avg on**	
Model	**Answers**	**Emails**	**Newsgroups**	**Reviews**	**Weblogs**	**targets**	**WSJ**
TnT*	89.36	87.38	90.85	89.67	91.37	89.73	96.57
Stanford*	89.74	87.77	91.25	90.30	92.32	90.28	97.43
Src (+glove)	90.43 ±.13	87.95 ±.18	91.83 ±.20	90.04 ±.11	92.44 ±.14	90.54 ±.15	**97.50** ±.03
Tri	**91.21** ±.06	**88.30** ±.19	**92.18** ±.19	**90.06** ±.10	**92.85** ±.02	**90.92** ±.11	97.45 ±.03
Asym	90.62 ±.26	87.71 ±.07	91.40 ±.05	89.89 ±.22	92.37 ±.27	90.39 ±.17	97.19 ±.03
MT-Tri	90.53 ±.15	87.90 ±.07	91.45 ±.19	89.77 ±.26	92.35 ±.09	90.40 ±.15	97.37 ±.07
FLORS*	91.17	88.67	92.41	92.25	93.14	91.53	97.11

Table 4: Accuracy for POS tagging on the dev and test sets of the SANCL domains, models trained on full source data setup. Values for methods with * are from (Schnabel and Schütze, 2014).

in the results of Blitzer et al. (2007) and is corroborated in the results for asymmetric tri-training (Saito et al., 2017) and our method.

We note a weakness of this dataset is high variance. Existing approaches only report the mean, which makes an objective comparison difficult. For this reason, we believe it is essential to evaluate proposed approaches also on other tasks.

POS tagging Results for tagging in the low-data regime (10% of WSJ) are given in Table 3.

Self-training does not work for the sequence prediction task. We report only the best instantiation (throttling with n=800). Our results contribute to negative findings regarding self-training (Plank, 2011; Van Asch and Daelemans, 2016).

In the low-data setup, tri-training *with disagreement* works best, reaching an overall average accuracy of 89.70, closely followed by classic tri-training, and significantly outperforming the baseline on 4/5 domains. The exception is newsgroups, a difficult domain with high OOV rate where none of the approches beats the baseline (see §3.4). Our proposed MT-Tri is better than asymmetric tri-training, but falls below classic tri-training. It beats

	Ans	Email	Newsg	Rev	Webl
% unk tag	0.25	0.80	0.31	0.06	0.0
% OOV	8.53	10.56	10.34	6.84	8.45
% UWT	2.91	3.47	2.43	2.21	1.46
Accuracy on OOV tokens					
Src	54.26	57.48	**61.80**	59.26	**80.37**
Tri	**55.53**	**59.11**	61.36	**61.16**	79.32
Asym	52.86	56.78	56.58	59.59	76.84
MT-Tri	52.88	57.22	57.28	58.99	77.77
Accuracy on unknown word-tag (UWT) tokens					
Src	**17.68**	**11.14**	**17.88**	**17.31**	**24.79**
Tri	16.88	10.04	17.58	16.35	23.65
Asym	17.16	10.43	17.84	16.92	22.74
MT-Tri	16.43	11.08	17.29	16.72	23.13
FLORS*	17.19	15.13	21.97	21.06	21.65

Table 5: Accuracy scores on dev sets for OOV and unknown word-tag (UWT) tokens.

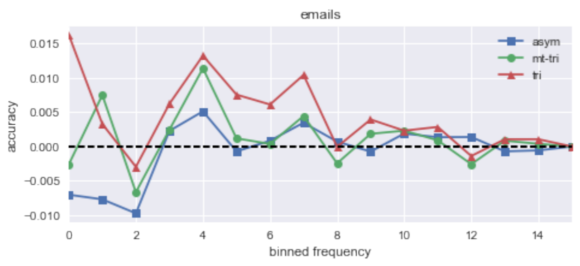


Figure 3: POS accuracy per binned log frequency.

the baseline significantly on only 2/5 domains (answers and emails). The FLORS tagger (Yin et al., 2015) fares better. Its contextual distributional features are particularly helpful on unknown word-tag combinations (see § 3.4), which is a limitation of the lexicalized generic bi-LSTM tagger.

For the high-data setup (Table 4) results are similar. Disagreement, however, is only favorable in the low-data setups; the effect of avoiding easy points no longer holds in the full data setup. Classic tri-training is the best method. In particular, traditional tri-training is complementary to word embedding initialization, pushing the non-pre-trained baseline to the level of SRC with Glove initalization. Tri-training pushes performance even further and results in the best model, significantly outperforming the baseline again in 4/5 cases, and reaching FLORS performance on weblogs. Multi-task tri-training is often slightly more effective than asymmetric tri-training (Saito et al., 2017); however, improvements for both are not robust across domains, sometimes performance even drops. The model likely is too simplistic for such a high-data POS setup, and exploring shared-private models might prove more fruitful (Liu et al., 2017). On the test sets, tri-training performs consistently the best.

POS analysis We analyze POS tagging accuracy with respect to word frequency[6] and unseen word-tag combinations (UWT) on the dev sets. Table 5 (top rows) provides percentage of unknown tags, OOVs and unknown word-tag (UWT) rate. The SANCL dataset is overall very challenging: OOV rates are high (6.8-11% compared to 2.3% in WSJ), so is the unknown word-tag (UWT) rate (answers and emails contain 2.91% and 3.47% UWT compared to 0.61% on WSJ) and almost all target domains even contain unknown tags (Schnabel and Schütze, 2014) (unknown tags: ADD,GW,NFP,XX), except for weblogs. Email is the domain with the highest OOV rate and highest unknown-tag-for-known-words rate. We plot accuracy with respect to word frequency on email in Figure 3, analyzing how the three methods fare in comparison to the baseline on this difficult domain.

Regarding OOVs, the results in Table 5 (second part) show that classic tri-training outperforms the source model (trained on only source data) on 3/5 domains in terms of OOV accuracy, except on two domains with high OOV rate (newsgroups and weblogs). In general, we note that tri-training works best on OOVs and on low-frequency tokens, which is also shown in Figure 3 (leftmost bins). Both other methods fall typically below the baseline in terms of OOV accuracy, but MT-Tri still outperforms Asym in 4/5 cases. Table 5 (last part) also shows that no bootstrapping method works well on unknown word-tag combinations. UWT tokens are very difficult to predict correctly using an unsupervised approach; the less lexicalized and more context-driven approach taken by FLORS is clearly superior for these cases, resulting in higher UWT accuracies for 4/5 domains.

4 Related work

Learning under Domain Shift There is a large body of work on domain adaptation. Studies on unsupervised domain adaptation include early work on *bootstrapping* (Steedman et al., 2003; McClosky et al., 2006a), *shared feature representations* (Blitzer et al., 2006, 2007) and *instance weighting* (Jiang and Zhai, 2007). Recent ap-

[6]The binned log frequency was calculated with base 2 (bin 0 are OOVs, bin 1 are singletons and rare words etc).

proaches include *adversarial learning* (Ganin et al., 2016) and *fine-tuning* (Sennrich et al., 2016). There is almost no work on bootstrapping approaches for recent neural NLP, in particular under domain shift. Tri-training is less studied, and only recently re-emerged in the vision community (Saito et al., 2017), albeit is not compared to classic tri-training.

Neural network ensembling Related work on self-ensembling approaches includes snapshot ensembling (Huang et al., 2017) or temporal ensembling (Laine and Aila, 2017). In general, the line between "explicit" and "implicit" ensembling (Huang et al., 2017), like dropout (Srivastava et al., 2014) or temporal ensembling (Saito et al., 2017), is more fuzzy. As we noted earlier our multi-task learning setup can be seen as a form of self-ensembling.

Multi-task learning in NLP Neural networks are particularly well-suited for MTL allowing for parameter sharing (Caruana, 1993). Recent NLP conferences witnessed a "tsunami" of deep learning papers (Manning, 2015), followed by what we call a multi-task learning "wave": MTL has been successfully applied to a wide range of NLP tasks (Cohn and Specia, 2013; Cheng et al., 2015; Luong et al., 2015; Plank et al., 2016; Fang and Cohn, 2016; Søgaard and Goldberg, 2016; Ruder et al., 2017; Augenstein et al., 2018). Related to it is the pioneering work on adversarial learning (DANN) (Ganin et al., 2016). For sentiment analysis we found tri-training and our MT-Tri model to outperform DANN. Our MT-Tri model lends itself well to shared-private models such as those proposed recently (Liu et al., 2017; Kim et al., 2017), which extend upon (Ganin et al., 2016) by having separate source and target-specific encoders.

5 Conclusions

We re-evaluate a range of traditional general-purpose bootstrapping algorithms in the context of neural network approaches to semi-supervised learning under domain shift. For the two examined NLP tasks classic tri-training works the best and even outperforms a recent state-of-the-art method. The drawback of tri-training it its time and space complexity. We therefore propose a more efficient multi-task tri-training model, which outperforms both traditional tri-training and recent alternatives in the case of sentiment analysis. For POS tagging, classic tri-training is superior, performing especially well on OOVs and low frequency tokens, which suggests it is less affected by error propagation. Overall we emphasize the importance of comparing neural approaches to strong baselines and reporting results across several runs.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289. Barbara is supported by NVIDIA corporation and thanks the Computing Center of the University of Groningen for HPC support.

References

Steven Abney. 2007. *Semisupervised learning for computational linguistics*. CRC Press.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces. In *Proceedings of NAACL-HLT 2018*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics*, 45(1):440.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06)*, pages 120–128.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. *NIPS*.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. 2006. *Semi-Supervised Learning*, volume 1. MIT press.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746. Association for Computational Linguistics.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 32–42, Sofia, Bulgaria. Association for Computational Linguistics.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. *arXiv preprint arXiv:1706.09733*.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR 2017*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *Proceedings of CoNLL-16*.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 587–593. Association for Computational Linguistics.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35.

Rob van der Goot, Barbara Plank, and Malvina Nissim. 2017. To normalize, or not to normalize: The impact of normalization on part-of-speech tagging. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 31–39, Copenhagen, Denmark. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. *Proceedings of ICML 2017*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, get M for free. In *Proceedings of ICLR 2017*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, Vancouver, Canada. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Samuli Laine and Timo Aila. 2017. Temporal Ensembling for Semi-Supervised Learning. In *Proceedings of ICLR 2017*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL-HLT 2016*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2015. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Christopher D Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and Self-Training for Parser Adaptation. *International Conference on Computational Linguistics (COLING) and Annual Meeting of the Association for Computational Linguistics (ACL)*, (July):337–344.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the State of the Art of Evaluation in Neural Language Models. In *arXiv preprint arXiv:1707.05589*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, 59.

Barbara Plank. 2011. *Domain adaptation for parsing*. University Library Groningen.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric Tri-training for Unsupervised Domain Adaptation. In *ICML 2017*.

Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *TACL*, 2:15–26.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mark Steedman, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Example selection for bootstrapping statistical parsers. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. pages 665–673.

Vincent Van Asch and Walter Daelemans. 2016. Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*.

Fangzhao Wu and Yongfeng Huang. 2016. Sentiment Domain Adaptation with Multiple Sources. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 301–310.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*.

Wenpeng Yin, Tobias Schnabel, and Hinrich Schütze. 2015. Online Updating of Word Representations for Part-of-Speech Tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, September, pages 1329–1334.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 322–332, Berlin, Germany. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Trans.Data Eng.*, 17(11):1529–1541.

Xiaojin Zhu. 2005. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

Fluency Boost Learning and Inference for Neural Grammatical Error Correction

Tao Ge **Furu Wei** **Ming Zhou**
Microsoft Research Asia, Beijing, China
{tage, fuwei, mingzhou}@microsoft.com

Abstract

Most of the neural sequence-to-sequence (seq2seq) models for grammatical error correction (GEC) have two limitations: (1) a seq2seq model may not be well generalized with only limited error-corrected data; (2) a seq2seq model may fail to completely correct a sentence with multiple errors through normal seq2seq inference. We attempt to address these limitations by proposing a fluency boost learning and inference mechanism. Fluency boosting learning generates fluency-boost sentence pairs during training, enabling the error correction model to learn how to improve a sentence's fluency from more instances, while fluency boosting inference allows the model to correct a sentence incrementally through multi-round seq2seq inference until the sentence's fluency stops increasing. Experiments show our approaches improve the performance of seq2seq models for GEC, achieving state-of-the-art results on both CoNLL-2014 and JFLEG benchmark datasets.

1 Introduction

Sequence-to-sequence (seq2seq) models (Cho et al., 2014; Sutskever et al., 2014) for grammatical error correction (GEC) have drawn growing attention (Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017; Schmaltz et al., 2017; Sakaguchi et al., 2017; Chollampatt and Ng, 2018) in recent years. However, most of the seq2seq models for GEC have two flaws. **First**, the seq2seq models are trained with only limited error-corrected sentence pairs like Figure 1(a). Limited by the size of training data, the models with millions of parameters may not be well generalized. Thus, it is common that the models fail to correct a sentence perfectly even if the sentence is slightly different from the training instance, as illustrated by Figure 1(b). **Second**, the seq2seq models usually cannot perfectly correct a sentence with many grammatical errors through single-round seq2seq inference, as shown in Figure 1(b) and 1(c), because some errors in a sentence may make the context strange, which confuses the models to correct other errors.

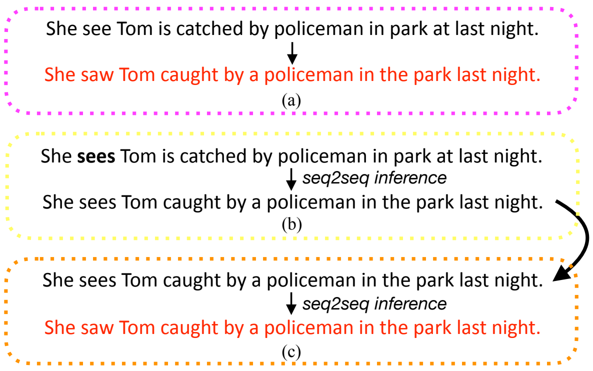

She see Tom is catched by policeman in park at last night.
She saw Tom caught by a policeman in the park last night.
(a)
She **sees** Tom is catched by policeman in park at last night.
seq2seq inference
She sees Tom caught by a policeman in the park last night.
(b)
She sees Tom caught by a policeman in the park last night.
seq2seq inference
She saw Tom caught by a policeman in the park last night.
(c)

Figure 1: **(a)** an error-corrected sentence pair; **(b)** if the sentence becomes slightly different, the model fails to correct it perfectly; **(c)** single-round seq2seq inference cannot perfectly correct the sentence, but multi-round inference can.

To address the above-mentioned limitations in model learning and inference, this paper proposes a novel fluency boost learning and inference mechanism, illustrated in Figure 2.

For fluency boosting learning, not only is a seq2seq model trained with original error-corrected sentence pairs, but also it generates less fluent sentences (e.g., from its n-best outputs) to establish new error-corrected sentence pairs by pairing them with their correct sentences during training, as long as the sentences' fluency[1] is be-

[1]A sentence's fluency score is defined to be inversely proportional to the sentence's cross entropy, as is in Eq (3).

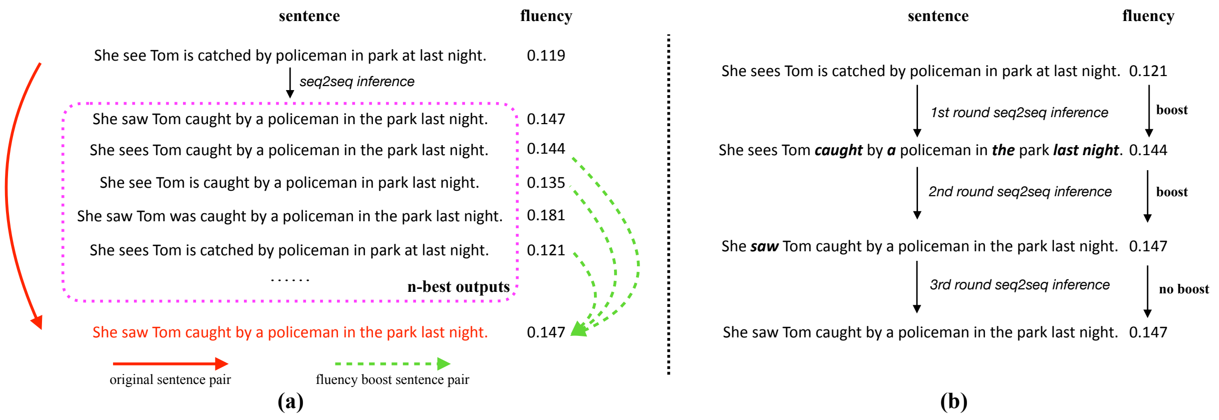

sentence
fluency
She see Tom is catched by policeman in park at last night. 0.119
seq2seq inference
She saw Tom caught by a policeman in the park last night. 0.147
She sees Tom caught by a policeman in the park last night. 0.144
She see Tom is caught by a policeman in park last night. 0.135
She saw Tom was caught by a policeman in the park last night. 0.181
She sees Tom is catched by policeman in park at last night. 0.121
n-best outputs
She saw Tom caught by a policeman in the park last night. 0.147
original sentence pair
fluency boost sentence pair
(a)
sentence
fluency
She sees Tom is catched by policeman in park at last night. 0.121
1st round seq2seq inference
boost
She sees Tom caught by a policeman in the park last night. 0.144
2nd round seq2seq inference
boost
She saw Tom caught by a policeman in the park last night. 0.147
3rd round seq2seq inference
no boost
She saw Tom caught by a policeman in the park last night. 0.147
(b)

Figure 2: Fluency boost learning and inference: **(a)** given a training instance (i.e., an error-corrected sentence pair), fluency boost learning establishes multiple fluency boost sentence pairs from the seq2seq's n-best outputs during training. The fluency boost sentence pairs will be used as training instances in subsequent training epochs, which helps expand the training set and accordingly benefits model learning; **(b)** fluency boost inference allows an error correction model to correct a sentence incrementally through multi-round seq2seq inference until its fluency score stops increasing.

low that of their correct sentences, as Figure 2(a) shows. Specifically, we call the generated error-corrected sentence pairs **fluency boost sentence pairs** because the sentence in the target side always improves fluency over that in the source side. The generated fluency boost sentence pairs during training will be used as additional training instances during subsequent training epochs, allowing the error correction model to see more grammatically incorrect sentences during training and accordingly improving its generalization ability.

For model inference, fluency boost inference mechanism allows the model to correct a sentence incrementally with multi-round inference as long as the proposed edits can boost the sentence's fluency, as Figure 2(b) shows. For a sentence with multiple grammatical errors, some of the errors will be corrected first. The corrected parts will make the context clearer, which may benefit the model to correct the remaining errors.

Experiments demonstrate fluency boost learning and inference enable neural seq2seq models to perform better for GEC and achieve state-of-the-art results on multiple GEC benchmarks.

Our contributions are summarized as follows:

- We present a novel learning and inference mechanism to address the limitations in previous seq2seq models for GEC.
- We propose and compare multiple novel fluency boost learning strategies, exploring the learning methodology for neural GEC.
- Our approaches are proven to be effective to improve neural seq2seq GEC models to achieve state-of-the-art results on CoNLL-2014 and JFLEG benchmark datasets.

2 Background: Neural grammatical error correction

As neural machine translation (NMT), a typical neural GEC approach uses a Recurrent Neural Network (RNN) based encoder-decoder seq2seq model (Sutskever et al., 2014; Cho et al., 2014) with attention mechanism (Bahdanau et al., 2014) to edit a raw sentence into the grammatically correct sentence it should be, as Figure 1(a) shows.

Given a raw sentence $\boldsymbol{x^r} = (x_1^r, \cdots, x_M^r)$ and its corrected sentence $\boldsymbol{x^c} = (x_1^c, \cdots, x_N^c)$ in which x_M^r and x_N^c are the M-th and N-th words of sentence $\boldsymbol{x^r}$ and $\boldsymbol{x^c}$ respectively, the error correction seq2seq model learns a probabilistic mapping $P(\boldsymbol{x^c}|\boldsymbol{x^r})$ from error-corrected sentence pairs through maximum likelihood estimation (MLE), which learns model parameters $\boldsymbol{\Theta_{crt}}$ to maximize the following equation:

$$\boldsymbol{\Theta}^*_{\boldsymbol{crt}} = \arg\max_{\boldsymbol{\Theta_{crt}}} \sum_{(\boldsymbol{x^r},\boldsymbol{x^c})\in\mathcal{S}^*} \log P(\boldsymbol{x^c}|\boldsymbol{x^r};\boldsymbol{\Theta_{crt}}) \qquad (1)$$

where $\mathcal{S}^*$ denotes the set of error-corrected sentence pairs.

For model inference, an output sequence $\boldsymbol{x^o} = (x_1^o, \cdots, x_i^o, \cdots, x_L^o)$ is selected through beam search, which maximizes the following equation:

$$P(\boldsymbol{x^o}|\boldsymbol{x^r}) = \prod_{i=1}^{L} P(x_i^o|\boldsymbol{x^r}, \boldsymbol{x^o}_{<i};\boldsymbol{\Theta_{crt}}) \qquad (2)$$

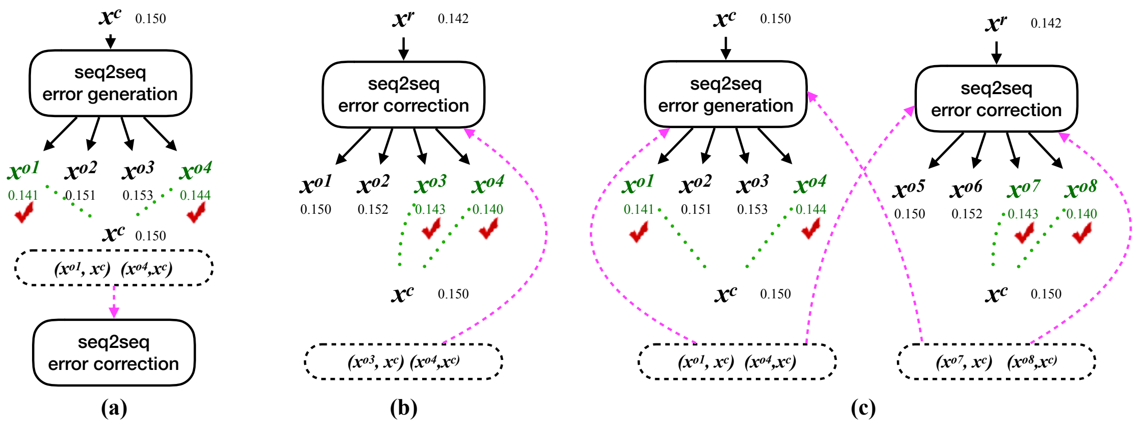


Figure 3: Three fluency boost learning strategies: **(a)** back-boost, **(b)** self-boost, **(c)** dual-boost; all of them generate fluency boost sentence pairs (the pairs in the dashed boxes) to help model learning during training. The numbers in this figure are fluency scores of their corresponding sentences.

3 Fluency boost learning

Conventional seq2seq models for GEC learns model parameters only from original error-corrected sentence pairs. However, such error-corrected sentence pairs are not sufficiently available. As a result, many neural GEC models are not very well generalized.

Fortunately, neural GEC is different from NMT. For neural GEC, its goal is improving a sentence's fluency[2] without changing its original meaning; thus, any sentence pair that satisfies this condition (we call it **fluency boost condition**) can be used as a training instance.

In this paper, we define $f(\boldsymbol{x})$ as the fluency score of a sentence $\boldsymbol{x}$:

$$f(\boldsymbol{x}) = \frac{1}{1 + H(\boldsymbol{x})} \tag{3}$$

$$H(\boldsymbol{x}) = -\frac{\sum_{i=1}^{|\boldsymbol{x}|} \log P(x_i|\boldsymbol{x}_{<i})}{|\boldsymbol{x}|} \tag{4}$$

where $P(x_i|\boldsymbol{x}_{<i})$ is the probability of x_i given context $\boldsymbol{x}_{<i}$, computed by a language model, and $|\boldsymbol{x}|$ is the length of sentence $\boldsymbol{x}$. $H(\boldsymbol{x})$ is actually the cross entropy of the sentence $\boldsymbol{x}$, whose range is $[0, +\infty)$. Accordingly, the range of $f(\boldsymbol{x})$ is $(0, 1]$.

The core idea of fluency boost learning is to generate fluency boost sentence pairs that satisfy the fluency boost condition during training, as Figure 2(a) illustrates, so that these pairs can further help model learning.

In this section, we present three fluency boost learning strategies: back-boost, self-boost, and dual-boost that generate fluency boost sentence pairs in different ways, as illustrated in Figure 3.

[2]Fluency of a sentence in this paper refers to how likely the sentence is written by a native speaker. In other words, if a sentence is very likely to be written by a native speaker, it should be regarded highly fluent.

3.1 Back-boost learning

Back-boost learning borrows the idea from back translation (Sennrich et al., 2016) in NMT, referring to training a backward model (we call it error generation model, as opposed to error correction model) that is used to convert a fluent sentence to a less fluent sentence with errors. Since the less fluent sentences are generated by the error generation seq2seq model trained with error-corrected data, they usually do not change the original sentence's meaning; thus, they can be paired with their correct sentences, establishing fluency boost sentence pairs that can be used as training instances for error correction models, as Figure 3(a) shows.

Specifically, we first train a seq2seq error generation model $\boldsymbol{\Theta_{gen}}$ with $\widetilde{\mathcal{S}^*}$ which is identical to $\mathcal{S}^*$ except that the source sentence and the target sentence are interchanged. Then, we use the model $\boldsymbol{\Theta_{gen}}$ to predict n-best outputs $\boldsymbol{x^{o_1}}, \cdots, \boldsymbol{x^{o_n}}$ given a correct sentence $\boldsymbol{x^c}$. Given the fluency boost condition, we compare the fluency of each output $\boldsymbol{x^{o_k}}$ (where $1 \leq k \leq n$) to that of its correct sentence $\boldsymbol{x^c}$. If an output sentence's fluency score is much lower than its correct sentence, we call it **a disfluency candidate** of $\boldsymbol{x^c}$.

To formalize this process, we first define $\mathcal{Y}_n(\boldsymbol{x}; \boldsymbol{\Theta})$ to denote the n-best outputs predicted by model $\boldsymbol{\Theta}$ given the input $\boldsymbol{x}$. Then, disfluency candidates of a correct sentence $\boldsymbol{x^c}$ can be derived:

$$\mathcal{D}_{back}(\boldsymbol{x^c}) = \{\boldsymbol{x^{o_k}} | \boldsymbol{x^{o_k}} \in \mathcal{Y}_n(\boldsymbol{x_c}; \boldsymbol{\Theta}_{gen}) \wedge \frac{f(\boldsymbol{x^c})}{f(\boldsymbol{x^{o_k}})} \geq \sigma\} \tag{5}$$

Algorithm 1 Back-boost learning

```
1: Train error generation model $\boldsymbol{\Theta_{gen}}$ with $\widetilde{\mathcal{S}^*}$;
2: for each sentence pair $(\boldsymbol{x^r}, \boldsymbol{x^c}) \in \mathcal{S}$ do
3:     Compute $\mathcal{D}_{back}(\boldsymbol{x^c})$ according to Eq (5);
4: end for
5: for each training epoch $t$ do
6:     $\mathcal{S}' \leftarrow \emptyset$;
7:     Derive a subset $\mathcal{S}_t$ by randomly sampling $|\mathcal{S}^*|$ elements from $\mathcal{S}$;
8:     for each $(\boldsymbol{x^r}, \boldsymbol{x^c}) \in \mathcal{S}_t$ do
9:         Establish a fluency boost pair $(\boldsymbol{x'}, \boldsymbol{x^c})$ by randomly sampling $\boldsymbol{x'} \in \mathcal{D}_{back}(\boldsymbol{x^c})$;
10:        $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(\boldsymbol{x'}, \boldsymbol{x^c})\}$;
11:    end for
12:    Update error correction model $\boldsymbol{\Theta_{crt}}$ with $\mathcal{S}^* \cup \mathcal{S}'$;
13: end for
```

Algorithm 2 Self-boost learning

```
1: for each sentence pair $(\boldsymbol{x^r}, \boldsymbol{x^c}) \in \mathcal{S}$ do
2:     $\mathcal{D}_{self}(\boldsymbol{x^c}) \leftarrow \emptyset$;
3: end for
4: $\mathcal{S}' \leftarrow \emptyset$
5: for each training epoch $t$ do
6:     Update error correction model $\boldsymbol{\Theta_{crt}}$ with $\mathcal{S}^* \cup \mathcal{S}'$;
7:     $\mathcal{S}' \leftarrow \emptyset$
8:     Derive a subset $\mathcal{S}_t$ by randomly sampling $|\mathcal{S}^*|$ elements from $\mathcal{S}$;
9:     for each $(\boldsymbol{x^r}, \boldsymbol{x^c}) \in \mathcal{S}_t$ do
10:        Update $\mathcal{D}_{self}(\boldsymbol{x^c})$ according to Eq (6);
11:        Establish a fluency boost pair $(\boldsymbol{x'}, \boldsymbol{x^c})$ by randomly sampling $\boldsymbol{x'} \in \mathcal{D}_{self}(\boldsymbol{x^c})$;
12:        $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(\boldsymbol{x'}, \boldsymbol{x^c})\}$;
13:    end for
14: end for
```

where $\mathcal{D}_{back}(\boldsymbol{x^c})$ denotes the disfluency candidate set for $\boldsymbol{x^c}$ in back-boost learning. σ is a threshold to determine if $\boldsymbol{x^{o_k}}$ is less fluent than $\boldsymbol{x^c}$ and it should be slightly larger[3] than 1.0, which helps filter out sentence pairs with unnecessary edits (e.g., I like this book. → I like the book.).

In the subsequent training epochs, the error correction model will not only learn from the original error-corrected sentence pairs ($\boldsymbol{x^r}$,$\boldsymbol{x^c}$), but also learn from fluency boost sentence pairs ($\boldsymbol{x^{o_k}}$,$\boldsymbol{x^c}$) where $\boldsymbol{x^{o_k}}$ is a sample of $\mathcal{D}_{back}(\boldsymbol{x^c})$.

We summarize this process in Algorithm 1 where $\mathcal{S}^*$ is the set of original error-corrected sentence pairs, and $\mathcal{S}$ can be tentatively considered identical to $\mathcal{S}^*$ when there is no additional native data to help model training (see Section 3.4). Note that we constrain the size of $\mathcal{S}_t$ not to exceed $|\mathcal{S}^*|$ (the 7th line in Algorithm 1) to avoid that too many fluency boost pairs overwhelm the effects of the original error-corrected pairs on model learning.

3.2 Self-boost learning

In contrast to back-boost learning whose core idea is originally from NMT, self-boost learning is original, which is specially devised for neural GEC. The idea of self-boost learning is illustrated by Figure 3(b) and was already briefly introduced in Section 1 and Figure 2(a). Unlike back-boost learning in which an error generation seq2seq model is trained to generate disfluency candidates, self-boost learning allows the error correction model to generate the candidates by itself. Since the disfluency candidates generated by the error correction seq2seq model trained with error-corrected data rarely change the input sentence's meaning; thus, they can be used to establish fluency boost sentence pairs.

For self-boost learning, given an error corrected pair $(\boldsymbol{x^r}, \boldsymbol{x^c})$, an error correction model $\boldsymbol{\Theta_{crt}}$ first predicts n-best outputs $\boldsymbol{x^{o_1}}, \cdots, \boldsymbol{x^{o_n}}$ for the raw sentence $\boldsymbol{x^r}$. Among the n-best outputs, any output that is not identical to $\boldsymbol{x^c}$ can be considered as an error prediction. Instead of treating the error predictions useless, self-boost learning fully exploits them. Specifically, if an error prediction $\boldsymbol{x^{o_k}}$ is much less fluent than that of its correct sentence $\boldsymbol{x^c}$, it will be added to $\boldsymbol{x^c}$'s disfluency candidate set $\mathcal{D}_{self}(\boldsymbol{x^c})$, as Eq (6) shows:

$$\begin{aligned}&\mathcal{D}_{self}(\boldsymbol{x^c}) = \mathcal{D}_{self}(\boldsymbol{x^c}) \cup \\ &\{\boldsymbol{x^{o_k}} | \boldsymbol{x^{o_k}} \in \mathcal{Y}_n(\boldsymbol{x_r}; \boldsymbol{\Theta}_{crt}) \wedge \frac{f(\boldsymbol{x^c})}{f(\boldsymbol{x^{o_k}})} \geq \sigma\}\end{aligned} \quad (6)$$

In contrast to back-boost learning, self-boost generates disfluency candidates from a different perspective – by editing the raw sentence $\boldsymbol{x^r}$ rather than the correct sentence $\boldsymbol{x^c}$. It is also noteworthy that $\mathcal{D}_{self}(\boldsymbol{x^c})$ is incrementally expanded because the error correction model $\boldsymbol{\Theta_{crt}}$ is dynamically updated, as shown in Algorithm 2.

3.3 Dual-boost learning

As introduced above, back- and self-boost learning generate disfluency candidates from different perspectives to create more fluency boost sentence pairs to benefit training the error correction model. Intuitively, the more diverse disfluency candidates generated, the more helpful for training an error correction model. Inspired by He et al. (2016) and Zhang et al. (2018), we propose a dual-boost learning strategy, combining both back- and self-boost's perspectives to generate disfluency candidates.

[3]In this paper, we set $\sigma = 1.05$ since the corrected sentence in our training data improves its corresponding raw sentence about 5% fluency on average.

Algorithm 3 Dual-boost learning

```
1: for each $(\boldsymbol{x}^r, \boldsymbol{x}^c) \in \mathcal{S}$ do
2:     $\mathcal{D}_{dual}(\boldsymbol{x}^c) \leftarrow \emptyset$;
3: end for
4: $\mathcal{S}' \leftarrow \emptyset$; $\mathcal{S}'' \leftarrow \emptyset$;
5: for each training epoch $t$ do
6:     Update error correction model $\boldsymbol{\Theta_{crt}}$ with $\mathcal{S}^* \cup \mathcal{S}'$;
7:     Update error generation model $\boldsymbol{\Theta_{gen}}$ with $\widetilde{\mathcal{S}^*} \cup \mathcal{S}''$;
8:     $\mathcal{S}' \leftarrow \emptyset$; $\mathcal{S}'' \leftarrow \emptyset$;
9:     Derive a subset $\mathcal{S}_t$ by randomly sampling $|\mathcal{S}^*|$ elements from $\mathcal{S}$;
10:    for each $(\boldsymbol{x}^r, \boldsymbol{x}^c) \in \mathcal{S}_t$ do
11:        Update $D_{dual}(\boldsymbol{x}^c)$ according to Eq (7);
12:        Establish a fluency boost pair $(\boldsymbol{x}', \boldsymbol{x}^c)$ by randomly sampling $\boldsymbol{x}' \in \mathcal{D}_{dual}(\boldsymbol{x}^c)$;
13:        $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(\boldsymbol{x}', \boldsymbol{x}^c)\}$;
14:        Establish a reversed fluency boost pair $(\boldsymbol{x}^c, \boldsymbol{x}'')$ by randomly sampling $\boldsymbol{x}'' \in \mathcal{D}_{dual}(\boldsymbol{x}^c)$;
15:        $\mathcal{S}'' \leftarrow \mathcal{S}'' \cup \{(\boldsymbol{x}^c, \boldsymbol{x}'')\}$;
16:    end for
17: end for
```

As Figure 3(c) shows, disfluency candidates in dual-boost learning are from both the error generation model and the error correction model :

$$\mathcal{D}_{dual}(\boldsymbol{x}^c) = \mathcal{D}_{dual}(\boldsymbol{x}^c) \cup$$
$$\{\boldsymbol{x}^{o_k} | \boldsymbol{x}^{o_k} \in \mathcal{Y}_n(\boldsymbol{x}_r; \boldsymbol{\Theta}_{crt}) \cup \mathcal{Y}_n(\boldsymbol{x}_c; \boldsymbol{\Theta}_{gen}) \wedge \frac{f(\boldsymbol{x}^c)}{f(\boldsymbol{x}^{o_k})} \geq \sigma\} \tag{7}$$

Moreover, the error correction model and the error generation model are dual and both of them are dynamically updated, which improves each other: the disfluency candidates produced by error generation model can benefit training the error correction model, while the disfluency candidates created by error correction model can be used as training data for the error generation model. We summarize this learning approach in Algorithm 3.

3.4 Fluency boost learning with large-scale native data

Our proposed fluency boost learning strategies can be easily extended to utilize the huge volume of native data which is proven to be useful for GEC.

As discussed in Section 3.1, when there is no additional native data, $\mathcal{S}$ in Algorithm 1–3 is identical to $\mathcal{S}^*$. In the case where additional native data is available to help model learning, $\mathcal{S}$ becomes:

$$\mathcal{S} = \mathcal{S}^* \cup \mathcal{C}$$

where $\mathcal{C} = \{(\boldsymbol{x}^c, \boldsymbol{x}^c)\}$ denotes the set of self-copied sentence pairs from native data.

4 Fluency boost inference

As we discuss in Section 1, some sentences with multiple grammatical errors usually cannot be perfectly corrected through normal seq2seq inference which does only single-round inference. Fortunately, neural GEC is different from NMT: its source and target language are the same. The characteristic allows us to edit a sentence more than once through multi-round model inference, which motivates our fluency boost inference. As Figure 2(b) shows, fluency boost inference allows a sentence to be incrementally edited through multi-round seq2seq inference as long as the sentence's fluency can be improved. Specifically, an error correction seq2seq model first takes a raw sentence $\boldsymbol{x}^r$ as an input and outputs a hypothesis $\boldsymbol{x}^{o_1}$. Instead of regarding $\boldsymbol{x}^{o_1}$ as the final prediction, fluency boost inference will then take $\boldsymbol{x}^{o_1}$ as the input to generate the next output $\boldsymbol{x}^{o_2}$. The process will not terminate unless $\boldsymbol{x}^{o_t}$ does not improve $\boldsymbol{x}^{o_{t-1}}$ in terms of fluency.

5 Experiments

5.1 Dataset and evaluation

Corpus	**#sent pair**
Lang-8	1,114,139
CLC	1,366,075
NUCLE	57,119
Total	2,537,333

Table 1: Error-corrected training data.

As previous studies (Ji et al., 2017), we use the public Lang-8 Corpus (Mizumoto et al., 2011; Tajiri et al., 2012), Cambridge Learner Corpus (CLC) (Nicholls, 2003) and NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) as our original error-corrected training data. Table 1 shows the stats of the datasets. In addition, we also collect 2,865,639 non-public error-corrected sentence pairs from Lang-8.com. The native data we use for fluency boost learning is English Wikipedia that contains 61,677,453 sentences.

We use CoNLL-2014 shared task dataset with original annotations (Ng et al., 2014), which contains 1,312 sentences, as our main test set for evaluation. We use MaxMatch (M^2) precision, recall and $F_{0.5}$ (Dahlmeier and Ng, 2012b) as our evaluation metrics. As previous studies, we use CoNLL-2013 test data as our development set.

5.2 Experimental setting

We set up experiments in order to answer the following questions:

Model	seq2seq			fluency boost			seq2seq (+LM)			fluency boost (+LM)		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
normal seq2seq	61.06	18.49	41.81	61.56	18.85	42.37	61.75	23.30	46.42	61.94	23.70	46.83
back-boost	61.66	19.54	43.09	61.43	19.61	43.07	61.47	24.74	47.40	61.24	25.01	47.48
self-boost	61.64	19.83	43.35	61.50	19.90	43.36	62.13	24.45	47.49	61.67	24.76	47.51
dual-boost	62.03	20.82	44.44	61.64	21.19	44.61	62.22	25.49	48.30	61.64	26.45	48.69
back-boost (+native)	63.93	22.03	46.31	63.95	22.12	46.40	62.04	27.43	49.54	61.98	27.70	49.68
self-boost (+native)	64.33	22.10	46.54	64.14	22.19	46.54	62.18	27.59	49.71	61.64	**28.37**	49.93
dual-boost (+native)	65.77	21.92	46.98	**65.82**	22.14	47.19	62.64	27.40	49.83	62.70	27.69	**50.04**
back-boost (+native)$\star$	**67.37**	24.31	49.75	67.25	24.35	49.73	64.61	28.44	51.51	64.46	28.78	51.66
self-boost (+native)$\star$	66.52	25.13	50.03	66.78	25.33	50.31	63.82	30.15	52.17	63.34	**31.63**	52.21
dual-boost (+native)$\star$	66.34	25.39	50.16	66.45	25.51	50.30	64.72	30.06	52.59	64.47	30.48	**52.72**

Table 2: Performance of seq2seq for GEC with different learning (row) and inference (column) methods on CoNLL-2014 dataset. (+LM) denotes decoding with the RNN language model through shallow fusion. The last 3 systems (with $\star$) use the additional non-public Lang-8 data for training.

- Whether is fluency boost learning mechanism helpful for training the error correction model, and which of the strategies (back-boost, self-boost, dual-boost) is the most effective?
- Whether does our fluency boost inference improve normal seq2seq inference for GEC?
- Whether can our approach improve neural GEC to achieve state-of-the-art results?

The training details for our seq2seq error correction model and error generation model are as follows: the encoder of the seq2seq models is a 2-layer bidirectional GRU RNN and the decoder is a 2-layer GRU RNN with the general attention mechanism (Luong et al., 2015). Both the dimensionality of word embeddings and the hidden size of GRU cells are 500. The vocabulary sizes of the encoder and decoder are 100,000 and 50,000 respectively. The models' parameters are uniformly initialized in [-0.1,0.1]. We train the models with an Adam optimizer with a learning rate of 0.0001 up to 40 epochs with batch size $= 128$. Dropout is applied to non-recurrent connections at a ratio of 0.15. For fluency boost learning, we generate disfluency candidates from 10-best outputs. During model inference, we set beam size to 5 and decode 1-best result with a 2-layer GRU RNN language model (Mikolov et al., 2010) through shallow fusion (Gülçehre et al., 2015) with weight $\beta = 0.15$. The RNN language model is trained from the native data mentioned in Section 5.1, which is also used for computing fluency score in Eq (3). UNK tokens are replaced with the source token with the highest attention weight.

We resolve spelling errors with a public spell checker[4] as preprocessing, as Xie et al. (2016) and Sakaguchi et al. (2017) do.

[4]https://azure.microsoft.com/en-us/services/cognitive-services/spell-check/

5.3 Experimental results

5.3.1 Effectiveness of fluency boost learning

Table 2 compares the performance of seq2seq error correction models with different learning and inference methods. By comparing by row, one can observe that our fluency boost learning approaches improve the performance over normal seq2seq learning, especially on the recall metric, since the fluency boost learning approaches generate a variety of grammatically incorrect sentences, allowing the error correction model to learn to correct much more sentences than the conventional learning strategy. Among the proposed three fluency boost learning strategies, dual-boost achieves the best result in most cases because it produces more diverse incorrect sentences (average $|\mathcal{D}_{dual}| \approx 9.43$) than either back-boost (avg $|\mathcal{D}_{back}| \approx 1.90$) or self-boost learning (avg $|\mathcal{D}_{self}| \approx 8.10$). With introducing large amounts of native text data, the performance of all the fluency boost learning approaches gets improved. One reason is that our learning approaches produce more error-corrected sentence pairs to let the model be better generalized. In addition, the huge volume of native data benefits the decoder to learn better to generate a fluent and error-free sentence.

We test the effect of hyper-parameter σ in Eq (5–7) on fluency boost learning and show the result in Table 3. When σ is slightly larger than 1.0 (e.g., $\sigma = 1.05$), the model achieves the best performance because it effectively avoids generating sentence pairs with unnecessary or undesirable edits that affect the performance, as we discussed in Section 3.1. When σ continues increasing, the disfluency candidate set $|\mathcal{D}_{dual}|$ drastically decreases, making the dual-boost learning gradually degrade to normal seq2seq learning.

Table 4 shows some examples of disfluency

σ	0	0.95	1.0	1.05	1.1	2.0
$\lvert\mathcal{D}_{dual}\rvert$	41.18	39.21	29.40	9.43	3.87	0.01
$F_{0.5}$	43.20	43.30	43.39	**44.44**	43.30	41.78

Table 3: The effect of σ on dual-boost learning with normal seq2seq inference. $\lvert\mathcal{D}_{dual}\rvert$ is the average size of dual-boost disfluency candidate sets.

Correct sentence	How autism occurs is not well understood.
Disfluency candidates	How autism occurs is not good understood.
	How autism occur is not well understood.
	What autism occurs is not well understood.
	How autism occurs is not well understand.
	How autism occurs does not well understood.

Table 4: Examples of disfluency candidates for a correct sentence in dual-boost learning.

candidates[5] generated in dual-boost learning given a correct sentence in the native data. It is clear that our approach can generate less fluent sentences with various grammatical errors and most of them are typical mistakes that a human learner tends to make. Therefore, they can be used to establish high-quality training data with their correct sentence, which will be helpful for increasing the size of training data to numbers of times, accounting for the improvement by fluency boost learning.

5.3.2 Effectiveness of fluency boost inference

The effectiveness of various inference approaches can be observed by comparing the results in Table 2 by column. Compared to the normal seq2seq inference and seq2seq (+LM) baselines, fluency boost inference brings about on average 0.14 and 0.18 gain on $F_{0.5}$ respectively, which is a significant[6] improvement, demonstrating multi-round edits by fluency boost inference is effective.

Take our best system (the last row in Table 2) as an example, among 1,312 sentences in the CoNLL-2014 dataset, seq2seq inference with shallow fusion LM edits 566 sentences. In contrast, fluency boost inference additionally edits 23 sentences during the second round inference, improving $F_{0.5}$ from 52.59 to 52.72.

5.3.3 Towards the state-of-the-art for GEC

Now, we answer the last question raised in Section 5.2 by testing if our approaches achieve the state-of-the-art result.

We first compare our best models – dual-boost learning (+native) with fluency boost inference and shallow fusion LM – to top-performing GEC systems evaluated on CoNLL-2014 dataset:

[5]We give more details about disfluency candidates, including error type proportion, in the supplementary notes.

[6]$p < 0.0005$ according to Wilcoxon Signed-Rank Test.

System	P	R	$F_{0.5}$
Spell check	53.01	8.16	25.25
CAMB14	39.71	30.10	37.33
CAMB16$_{\text{SMT}}$	45.39	21.82	37.33
CAMB16$_{\text{NMT}}$	-	-	39.90
CAMB17 (CAMB16$_{\text{SMT}}$ based)	51.09	25.30	42.44
CAMB17 (AMU16 based)	59.88	32.16	51.08
AMU14	41.62	21.40	35.01
AMU16	61.27	27.98	49.49
AMU16$\star$	63.52	30.49	52.21
CUUI	41.78	24.88	36.79
VT16$\star$	60.17	25.64	47.40
NUS14	53.55	19.14	39.39
NUS16	-	-	44.27
NUS17	62.74	32.96	53.14
Char-seq2seq	49.24	23.77	40.56
Nested-seq2seq	-	-	45.15
Adapt-seq2seq	-	-	41.37
dual-boost (single)	62.70	27.69	50.04
dual-boost (AMU16 based)	60.57	36.02	53.30
dual-boost (single)$\star$	**64.47**	30.48	52.72
dual-boost (AMU16 based)$\star$	61.24	**37.86**	**54.51**

Table 5: Performance of systems on CoNLL-2014 dataset. The system with bold fonts are based on seq2seq models. $\star$ denotes the system uses the non-public error-corrected data from Lang-8.com.

- CAMB14, CAMB16$_{\text{SMT}}$, CAMB16$_{\text{NMT}}$ and CAMB17: GEC systems (Felice et al., 2014; Yuan et al., 2016; Yuan and Briscoe, 2016; Yannakoudakis et al., 2017) developed by Cambridge University.
- AMU14 and AMU16: SMT-based GEC systems (Junczys-Dowmunt and Grundkiewicz, 2014, 2016) developed by AMU.
- CUUI and VT16: the former system (Rozovskaya et al., 2014) uses a classifier-based approach, which is improved by the latter system (Rozovskaya and Roth, 2016) through combining with an SMT-based approach.
- NUS14, NUS16 and NUS17: GEC systems (Susanto et al., 2014; Chollampatt et al., 2016a; Chollampatt and Ng, 2017) that combine SMT with other techniques (e.g., classifiers).
- Char-seq2seq: a character-level seq2seq model (Xie et al., 2016). It uses a rule-based method to synthesize errors for data augmentation.
- Nested-seq2seq: a nested attention neural hybrid seq2seq model (Ji et al., 2017).
- Adapt-seq2seq: a seq2seq model adapted to incorporate edit operations (Schmaltz et al., 2017).

Table 5 shows the evaluation results on the CoNLL-2014 dataset. Without using the non-public training data from Lang-8.com, our single model obtains 50.04 $F_{0.5}$, larlgely outperforming the other seq2seq models and only inferior to CAMB17 (AMU16 based) and NUS17. It should be noted, however, that the CAMB17 and NUS17 are actually re-rankers built on top of an SMT-based GEC system (AMU16's framework); thus, they are ensemble models. When we build our approach on top of AMU16 (i.e., we take AMU16's outputs as the input to our GEC system to edit on top of its outputs), we achieve 53.30 $F_{0.5}$ score. With introducing the non-public training data, our single and ensemble system obtain 52.72 and 54.51 $F_{0.5}$ score respectively, which is a state-of-the-art result[7] on CoNLL-2014 dataset.

Moreover, we evaluate our approach on JFLEG corpus (Napoles et al., 2017). JFLEG is the latest released dataset for GEC evaluation and it contains 1,501 sentences (754 in dev set and 747 in test set). To test our approach's generalization ability, we evaluate our single models used for CoNLL evaluation (in Table 5) on JFLEG without re-tuning.

Table 6 shows the JFLEG leaderboard. Instead of M^2 score, JFLEG uses GLEU (Napoles et al., 2015) as its evaluation metric, which is a fluency-oriented GEC metric based on a variant of BLEU (Papineni et al., 2002) and has several advantages over M^2 for GEC evaluation. It is observed that our single models consistently perform well on JFLEG, outperforming most of the CoNLL-2014 top-performing systems and yielding a state-of-the-art result[8] on this benchmark, demonstrating that our models are well generalized and perform stably on multiple datasets.

6 Related work

Most of advanced GEC systems are classifier-based (Chodorow et al., 2007; De Felice and Pulman, 2008; Han et al., 2010; Leacock et al., 2010; Tetreault et al., 2010a; Dale and Kilgarriff, 2011) or MT-based (Brockett et al., 2006; Dahlmeier and Ng, 2011, 2012a; Yoshimoto et al., 2013; Yuan and Felice, 2013; Behera and Bhattacharyya, 2013). For example, top-performing systems (Felice et al., 2014; Rozovskaya et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014) in CoNLL-2014 shared task (Ng et al., 2014) use either of the methods. Recently, many novel approaches (Susanto et al., 2014; Chollampatt et al., 2016b,a; Rozovskaya and Roth, 2016; Junczys-Dowmunt and Grundkiewicz, 2016; Mizumoto and Matsumoto, 2016; Yuan et al., 2016; Hoang et al., 2016; Yannakoudakis et al., 2017) have been proposed for GEC. Among them, seq2seq models (Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017; Sakaguchi et al., 2017; Schmaltz et al., 2017; Chollampatt and Ng, 2018) have caught much attention. Unlike the models trained only with original error-corrected data, we propose a novel fluency boost learning mechanism for dynamic data augmentation along with training for GEC, despite some previous studies that explore artificial error generation for GEC (Brockett et al., 2006; Foster and Andersen, 2009; Rozovskaya and Roth, 2010, 2011; Rozovskaya et al., 2012; Felice and Yuan, 2014; Xie et al., 2016; Rei et al., 2017). Moreover, we propose fluency boost inference which allows the model to repeatedly edit a sentence as long as the sentence's fluency can be improved. To the best of our knowledge, it is the first to conduct multi-round seq2seq inference for GEC, while similar ideas have been proposed for NMT (Xia et al., 2017).

In addition to the studies on GEC, there is also much research on grammatical error detection

System	JFLEG Dev GLEU	JFLEG Test GLEU
Source	38.21	40.54
CAMB14	42.81	46.04
CAMB16$_{SMT}$	46.10	-
CAMB16$_{NMT}$	47.20	52.05
CAMB17 (CAMB16$_{SMT}$ based)	47.72	-
CAMB17 (AMU16 based)	43.26	-
NUS16	46.27	50.13
NUS17	51.01	**56.78**
AMU16*	49.74	51.46
Nested-seq2seq	48.93	53.41
Sakaguchi et al. (2017)*	49.82	53.98
Ours	**51.35**	56.33
Ours (with non-public Lang-8 data)	**52.93**	**57.74**
Human	55.26	62.37

Table 6: JFLEG Leaderboard. Ours denote the single dual-boost models in Table 5. The systems with bold fonts are based on seq2seq models. * denotes the system is tuned on JFLEG.

[7]The state-of-the-art result on CoNLL-2014 dataset has been recently advanced by Chollampatt and Ng (2018) ($F_{0.5}$=54.79) and Grundkiewicz and Junczys-Dowmunt (2018) ($F_{0.5}$=56.25), which are contemporaneous to this paper. In contrast to the basic seq2seq model in this paper, they used advanced approaches for modeling (e.g., convolutional seq2seq with pre-trained word embedding, using edit operation features, ensemble decoding and advanced model combinations). It should be noted that their approaches are orthogonal to ours, making it possible to apply our fluency boost learning and inference mechanism to their models.

[8]The recently proposed SMT-NMT hybrid system (Grundkiewicz and Junczys-Dowmunt, 2018), which is tuned towards GLEU on JFLEG Dev set, reports a higher result (GLEU=61.50 on JFLEG test set).

(Leacock et al., 2010; Rei and Yannakoudakis, 2016; Kaneko et al., 2017) and GEC evaluation (Tetreault et al., 2010b; Madnani et al., 2011; Dahlmeier and Ng, 2012c; Napoles et al., 2015; Sakaguchi et al., 2016; Napoles et al., 2016; Bryant et al., 2017; Asano et al., 2017). We do not introduce them in detail because they are not much related to this paper's contributions.

7 Conclusion

We propose a novel fluency boost learning and inference mechanism to overcome the limitations of previous neural GEC models. Our proposed fluency boost learning fully exploits both error-corrected data and native data, largely improving the performance over normal seq2seq learning, while fluency boost inference utilizes the characteristic of GEC to incrementally improve a sentence's fluency through multi-round inference. The powerful learning and inference mechanism enables the seq2seq models to achieve state-of-the-art results on both CoNLL-2014 and JFLEG benchmark datasets.

Acknowledgments

We thank all the anonymous reviewers for their professional and constructive comments. We also thank Shujie Liu for his insightful discussions and suggestions.

References

Hiroki Asano, Tomoya Mizumoto, and Kentaro Inui. 2017. Reference-based metrics can be replaced with reference-less metrics in evaluating grammatical error correction systems. In *IJCNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bibek Behera and Pushpak Bhattacharyya. 2013. Automated grammar correction using hierarchical phrase-based statistical machine translation. In *IJCNLP*.

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting esl errors using phrasal smt techniques. In *COLING/ACL*.

Christopher Bryant, Mariano Felice, and E Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Martin Chodorow, Joel R Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *ACL-SIGSEM workshop on prepositions*.

Shamil Chollampatt, Duc Tam Hoang, and Hwee Tou Ng. 2016a. Adapting grammatical error correction based on the native language of writers with neural network joint models. In *EMNLP*.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Workshop on Innovative Use of NLP for Building Educational Applications*.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv preprint arXiv:1801.08831*.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016b. Neural network translation models for grammatical error correction. *arXiv preprint arXiv:1606.00189*.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with l1-induced paraphrases. In *EMNLP*.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A beam-search decoder for grammatical error correction. In *EMNLP/CoNLL*.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better evaluation for grammatical error correction. In *NAACL*.

Daniel Dahlmeier and Hwee Tou Ng. 2012c. Better evaluation for grammatical error correction. In *NAACL*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Workshop on innovative use of NLP for building educational applications*.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The hoo 2011 pilot shared task. In *European Workshop on Natural Language Generation*.

Rachele De Felice and Stephen G Pulman. 2008. A classifier-based approach to preposition and determiner error correction in l2 english. In *COLING*.

Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *Student Research Workshop at EACL*.

Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *CoNLL (Shared Task)*.

Jennifer Foster and Øistein E Andersen. 2009. Generrate: generating errors for use in grammatical error detection. In *Workshop on innovative use of nlp for building educational applications*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. *arXiv preprint arXiv:1804.05945*.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Na-Rae Han, Joel R Tetreault, Soo-Hwa Lee, and Jin-Young Ha. 2010. Using an error-annotated learner corpus to develop an esl/efl error correction system. In *LREC*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NIPS*.

Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting n-best hypotheses to improve an smt approach to grammatical error correction. In *IJCAI*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *ACL*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The amu system in the conll-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *CoNLL (Shared Task)*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353*.

Masahiro Kaneko, Yuya Sakaizawa, and Mamoru Komachi. 2017. Grammatical error detection using error-and grammaticality-specific word embeddings. In *IJCNLP*.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 3(1):1–134.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Nitin Madnani, Joel Tetreault, Martin Chodorow, and Alla Rozovskaya. 2011. They can help: Using crowdsourcing to improve the evaluation of grammatical error detection systems. In *ACL*.

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *IJCNLP*.

Tomoya Mizumoto and Yuji Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *NAACL*.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *ACL/IJCNLP*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *EMNLP*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *CoNLL (Shared Task)*.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Corpus Linguistics 2003 conference*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. *arXiv preprint arXiv:1707.05236*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *ACL*.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The illinois-columbia system in the conll-2014 shared task. In *CoNLL (Shared Task)*.

Alla Rozovskaya and Dan Roth. 2010. Training paradigms for correcting errors in grammar and usage. In *NAACL*.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for esl correction tasks. In *ACL*.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *ACL*.

Alla Rozovskaya, Mark Sammons, and Roth Dan. 2012. The ui system in the hoo 2012 shared task on error correction. In *Workshop on Building Educational Applications Using NLP*.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association of Computational Linguistics*, 4(1):169–182.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *IJCNLP*.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *EMNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL*.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *ACL*.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010a. Using parse features for preposition selection and error detection. In *ACL*.

Joel R Tetreault, Elena Filatova, and Martin Chodorow. 2010b. Rethinking grammatical error annotation and evaluation with the amazon mechanical turk. In *Workshop on Innovative Use of NLP for Building Educational Applications*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *NIPS*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Helen Yannakoudakis, Marek Rei, Øistein E Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *EMNLP*.

Ippei Yoshimoto, Tomoya Kose, Kensuke Mitsuzawa, Keisuke Sakaguchi, Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2013. Naist at 2013 conll grammatical error correction shared task. In *CoNLL (Shared Task)*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *NAACL*.

Zheng Yuan, Ted Briscoe, Mariano Felice, Zheng Yuan, Ted Briscoe, and Mariano Felice. 2016. Candidate re-ranking for smt-based grammatical error correction. In *Workshop on Innovative Use of NLP for Building Educational Applications*.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *CoNLL (Shared Task)*.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. *arXiv preprint arXiv:1803.00353*.

A Neural Architecture for Automated ICD Coding

Pengtao Xie[†*], Haoran Shi[§], Ming Zhang[§] and Eric P. Xing[†]
[†]Petuum Inc, USA
[*]Machine Learning Department, Carnegie Mellon University, USA
[§]School of Electronics Engineering and Computer Science, Peking University, China
{pengtao.xie,eric.xing}@petuum.com
shore@pku.edu.cn, mzhang@net.pku.edu.cn

Abstract

The International Classification of Diseases (ICD) provides a hierarchy of diagnostic codes for classifying diseases. Medical coding – which assigns a subset of ICD codes to a patient visit – is a mandatory process that is crucial for patient care and billing. Manual coding is time-consuming, expensive, and error-prone. In this paper, we build a neural architecture for automated coding. It takes the diagnosis descriptions (DDs) of a patient as inputs and selects the most relevant ICD codes. This architecture contains four major ingredients: (1) tree-of-sequences LSTM encoding of code descriptions (CDs), (2) adversarial learning for reconciling the different writing styles of DDs and CDs, (3) isotonic constraints for incorporating the importance order among the assigned codes, and (4) attentional matching for performing many-to-one and one-to-many mappings from DDs to CDs. We demonstrate the effectiveness of the proposed methods on a clinical datasets with 59K patient visits.

1 Introduction

The International Classification of Diseases (ICD) is a healthcare classification system maintained by the World Health Organization (Organization et al., 1978). It provides a hierarchy of diagnostic codes of diseases, disorders, injuries, signs, symptoms, etc. It is widely used for reporting diseases and health conditions, assisting in medical reimbursement decisions, collecting morbidity and mortality statistics, to name a few.

While ICD codes are important for making clinical and financial decisions, medical coding – which assigns proper ICD codes to a patient visit – is time-consuming, error-prone, and expensive. Medical coders review the diagnosis descriptions written by physicians in the form of textual phrases and sentences, and (if necessary) other information in the electronic health record of a clinical episode, then manually attribute the appropriate ICD codes by following the coding guidelines (O'malley et al., 2005). Several types of errors frequently occur. First, the ICD codes are organized in a hierarchical structure. For a node representing a disease C, the children of this node represent the subtypes of C. In many cases, the difference between disease subtypes is very subtle. It is common that human coders select incorrect subtypes. Second, when writing diagnosis descriptions, physicians often utilize abbreviations and synonyms, which causes ambiguity and imprecision when the coders are matching ICD codes to those descriptions (Sheppard et al., 2008). Third, in many cases, several diagnosis descriptions are closely related and should be mapped to a single ICD code. However, unexperienced coders may code each disease separately. Such errors are called *unbundling*. The cost incurred by coding errors and the financial investment spent on improving coding quality are estimated to be \$25 billion per year in the US (Lang, 2007; Farkas and Szarvas, 2008).

To reduce coding errors and cost, we aim at building an ICD coding model which automatically and accurately translates the free-text diagnosis descriptions into ICD codes. To achieve this goal, several technical challenges need to be addressed. First, there exists a hierarchical structure among the ICD codes. This hierarchy can be leveraged to improve coding accuracy. On one hand, if code A and B are both children of C, then it is unlikely to simultaneously assign A and B to a patient. On the other hand, if the distance between A and B in the code tree is smaller than that between A and C and we know A is the correct code, then B is more likely to be a correct code than C, since codes with smaller distance are more clinically relevant. How to explore this hierarchical structure for better coding is technically demanding. Second, the diagnosis descriptions and the textual descriptions of ICD codes are written in quite different styles even if they refer to the same disease. In particular, the textual description of an ICD code is formally and precisely worded, while diagnosis descriptions are usually written by physicians in an informal and ungrammatical way, with telegraphic phrases, abbreviations, and typos. Third, it is required that the assigned ICD codes are ranked according to their relevance to the patient. How to correctly determine this order is technically nontrivial. Fourth, as stated earlier, there does not necessarily exist an one-to-one mapping between diagnosis descriptions and ICD codes, and human coders should consider the overall health condition when assigning codes. In many cases, two closely related diagnosis descriptions need to be mapped onto a single combination ICD code. On the other hand, physicians may write two health conditions into one diagnosis description which should be mapped onto two ICD codes under such circumstances.

Contributions In this paper, we design a neural architecture to automatically perform ICD coding given the diagnosis descriptions. Specifically, we make the following contributions:

- We propose a tree-of-sequences LSTM architecture to simultaneously capture the hierarchical relationship among codes and the semantics of each code.
- We use an adversarial learning approach to reconcile the heterogeneous writing styles of diagnosis descriptions and ICD code descriptions.
- We use isotonic constraints to preserve the importance order among codes and develop an algorithm based on ADMM and isotonic projection to solve the constrained problem.
- We use an attentional matching mechanism to perform many-to-one and one-to-many mappings between diagnosis descriptions and codes.
- On a clinical datasets with 59K patient visits, we demonstrate the effectiveness of the proposed methods.

The rest of the paper is organized as follows. Section 2 introduces related works. Section 3 and 4 present the dataset and methods. Section 5 gives experimental results. Section 6 presents conclusions and discussions.

2 Related Works

Larkey and Croft (1996) studied the automatic assignment of ICD-9 codes to dictated inpatient discharge summaries, using a combination of three classifiers: k-nearest neighbors, relevance feedback, and Bayesian independence classifiers. This method assigns a single code to each patient visit. However, in clinical practice, each patient is usually assigned with multiple codes. Franz et al. (2000) investigated the automated coding of German-language free-text diagnosis phrases. This approach performs one-to-one mapping between diagnosis descriptions and ICD codes. This is not in accordance with the coding practice where one-to-many and many-to-one mappings widely exist (O'malley et al., 2005). Pestian et al. (2007) studied the assignment of ICD-9 codes to radiology reports. Kavuluru et al. (2013) proposed an unsupervised ensemble approach to automatically perform ICD-9 coding based on textual narratives in electronic health records (EHRs) Kavuluru et al. (2015) developed multi-label classification, feature selection, and learning to rank approaches for ICD-9 code assignment of in-patient visits based on EHRs. Koopman et al. (2015) explored the automatic ICD-10 classification of cancers from free-text death certificates. These methods did not consider the hierarchical relationship or importance order among codes.

The tree LSTM network was first proposed by (Tai et al., 2015) to model the constituent or dependency parse trees of sentences. Teng and Zhang (2016) extended the unidirectional tree LSTM to a bidirectional one. Xie and Xing (2017) proposed a sequence-of-trees LSTM network to model a passage. In this network, a sequential LSTM is used to compose a sequence of tree LSTMs. The tree LSTMs are built on the constituent parse trees of individual sentences and the sequential LSTM is built on the sequence of sentences. Our proposed tree-of-sequences LSTM network differs from the previous works in two-fold. First, it is applied to a code tree to capture the hierarchical relationship among codes. Second, it uses a tree LSTM to compose a hierarchy

Diagnosis Descriptions
1. Prematurity at 35 4/7 weeks gestation
2. Twin number two of twin gestation
3. Respiratory distress secondary to transient tachypnea of the newborn
4. Suspicion for sepsis ruled out
Assigned ICD Codes
1. V31.00 (Twin birth, mate liveborn, born in hospital, delivered without mention of cesarean section)
2. 765.18 (Other preterm infants, 2,000-2,499 grams)
3. 775.6 (Neonatal hypoglycemia)
4. 770.6 (Transitory tachypnea of newborn)
5. V29.0 (Observation for suspected infectious condition)
6. V05.3 (Need for prophylactic vaccination and inoculation against viral hepatitis)

Table 1: The diagnosis descriptions of a patient visit and the assigned ICD codes. Inside the parentheses are the descriptions of the codes. The codes are ranked according to descending importance.

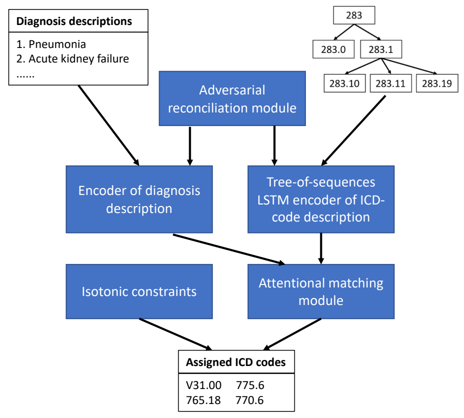


Figure 1: Architecture of the ICD Coding Model

of sequential LSTMs.

Adversarial learning (Goodfellow et al., 2014) has been widely applied to image generation (Goodfellow et al., 2014), domain adaption (Ganin and Lempitsky, 2015), feature learning (Donahue et al., 2016), text generation (Yu et al., 2017), to name a few. In this paper, we use adversarial learning for mitigating the discrepancy among the writing styles of a pair of sentences.

The attention mechanism was widely used in machine translation (Bahdanau et al., 2014), image captioning (Xu et al., 2015), reading comprehension (Seo et al., 2016), text classification (Yang et al., 2016), etc. In this work, we compute attention between sentences to perform many-to-one and one-to-many mappings.

3 Dataset and Preprocessing

We performed the study on the publicly available MIMIC-III dataset (Johnson et al., 2016), which contains de-identified electronic health records (EHRs) of 58,976 patient visits in the Beth Israel Deaconess Medical Center from 2001 to 2012. Each EHR has a clinical note called discharge summary, which contains multiple sections of information, such as 'discharge diagnosis', 'past medical history', etc. From the 'discharge diagnosis' and 'final diagnosis' sections, we extracted the diagnosis descriptions (DDs) written by physicians. Each DD is a short phrase or a sentence, articulating a certain disease or condition. Medical coders perform ICD coding mainly based on DDs. Following such a practice, in this paper, we set the inputs of the automated coding model to be the DDs while acknowledging that other information in the EHRs is also valuable and is referred to by coders for code assignment. For simplicity, we leave the incorporation of non-DD information to future study.

Each patient visit is assigned with a list of ICD codes, ranked in descending order of importance and relevance. For each visit, the number of codes is usually not equal to the number of diagnosis descriptions. These ground-truth codes serve as the labels to train our coding model. The entire dataset contains 6,984 unique codes, each of which has a textual description, describing a disease, symptom, or condition. The codes are organized into a hierarchy where the top-level codes correspond to general diseases while the bottom-level ones represent specific diseases. In the code tree, children of a node represent subtypes of a disease. Table 1 shows the DDs and codes of an exemplar patient.

4 Methods

In this section, we present a neural architecture for ICD coding.

4.1 Overview

Figure 1 shows the overview of our approach. The proposed ICD coding model consists of five modules. The model takes the ICD-code tree and diagnosis descriptions (DDs) of a patient as inputs and assigns a set of ICD codes to the patient. The encoder of DDs generates a latent representation vector for a DD. The encoder of ICD codes is a tree-of-sequences long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network. It takes the textual descriptions of the ICD codes and their hierarchical structure as in-

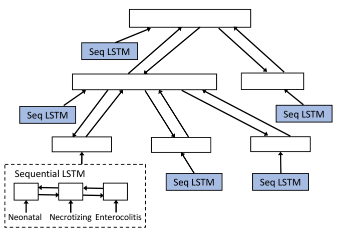


Figure 2: Tree-of-Sequences LSTM

puts and produces a latent representation for each code. The representation aims at simultaneously capturing the semantics of each code and the hierarchical relationship among codes. By incorporating the code hierarchy, the model can avoid selecting codes that are subtypes of the same disease and promote the selection of codes that are clinically correlated. The writing styles of DDs and code descriptions (CDs) are largely different, which makes the matching between a DD and a CD error-prone. To address this issue, we develop an adversarial learning approach to reconcile the writing styles. On top of the latent representation vectors of the descriptions, we build a discriminative network to distinguish which ones are DDs and which are CDs. The encoders of DDs and CDs try to make such a discrimination impossible. By doing this, the learned representations are independent of the writing styles and facilitate more accurate matching. The representations of DDs and CDs are fed into an attentional matching module to perform code assignment. This attentional mechanism allows multiple DDs to be matched to a single code and allows a single DD to be matched to multiple codes. During training, we incorporate the order of importance among codes as isotonic constraints. These constraints regulate the model's weight parameters so that codes with higher importance are given larger prediction scores.

4.2 Tree-of-Sequences LSTM Encoder

This section introduces the encoder of ICD codes. Each code has a description (a sequence of words) that tells the semantics of this code. We use a sequential LSTM (SLSTM) (Hochreiter and Schmidhuber, 1997) to encode this description. To capture the hierarchical relationship among codes, we build a tree LSTM (TLSTM) (Tai et al., 2015) along the code tree. At each TLSTM node, the input vector is the latent representation generated by the SLSTM. Combining these two types of LSTMs together, we obtain a tree-of-sequences LSTM network (Figure 2).

Sequential LSTM A sequential LSTM (SLSTM) (Hochreiter and Schmidhuber, 1997) network is a special type of recurrent neural network that (1) learns the latent representation (which usually reflects certain semantic information) of words, and (2) models the sequential structure among words. In the word sequence, each word t is allocated with an SLSTM unit, which consists of the following components: an input gate $\mathbf{i}_t$, a forget gate $\mathbf{f}_t$, an output gate $\mathbf{o}_t$, a memory cell $\mathbf{c}_t$, and a hidden state $\mathbf{s}_t$. These components (vectors) are computed as follows:

$$\begin{aligned}
&\mathbf{i}_t = \sigma(\mathbf{W}^{(i)}\mathbf{s}_{t-1} + \mathbf{U}^{(i)}\mathbf{x}_t + \mathbf{b}^{(i)})\\
&\mathbf{f}_t = \sigma(\mathbf{W}^{(f)}\mathbf{s}_{t-1} + \mathbf{U}^{(f)}\mathbf{x}_t + \mathbf{b}^{(f)})\\
&\mathbf{o}_t = \sigma(\mathbf{W}^{(o)}\mathbf{s}_{t-1} + \mathbf{U}^{(o)}\mathbf{x}_t + \mathbf{b}^{(o)})\\
&\mathbf{c}_t = \mathbf{i}_t \odot \tanh(\mathbf{W}^{(c)}\mathbf{s}_{t-1} + \mathbf{U}^{(c)}\mathbf{x}_t + \mathbf{b}^{(c)})\\
&\qquad + \mathbf{f}_t \odot \mathbf{c}_{t-1}\\
&\mathbf{s}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where $\mathbf{x}_t$ is the embedding vector of word t. $\mathbf{W}$, $\mathbf{U}$ are component-specific weight matrices and $\mathbf{b}$ are bias vectors.

Tree-of-sequences LSTM We use a bidirectional tree LSTM (TLSTM) (Tai et al., 2015; Xie and Xing, 2017) to capture the hierarchical relationships among codes. The inputs of this LSTM include the code hierarchy and hidden states of individual codes produced by the SLSTMs. It consists of a bottom-up TLSTM and a top-down TLSTM, which produce two hidden states $\mathbf{h}_\uparrow$ and $\mathbf{h}_\downarrow$ at each node in the tree.

In the bottom-up TLSTM, an internal node (representing a code C, having M children) is comprised of these components: an input gate $\mathbf{i}_\uparrow$, an output gate $\mathbf{o}_\uparrow$, a memory cell $\mathbf{c}_\uparrow$, a hidden state $\mathbf{h}_\uparrow$ and M child-specific forget gates $\{\mathbf{f}_\uparrow^{(m)}\}_{m=1}^{M}$ where $\mathbf{f}_\uparrow^{(m)}$ corresponds to the m-th child. The transition equations among components are:

$$\begin{aligned}
&\mathbf{i}_\uparrow = \sigma(\textstyle\sum_{m=1}^{M} \mathbf{W}_\uparrow^{(i,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(i)}\mathbf{s} + \mathbf{b}_\uparrow^{(i)})\\
&\forall m, \mathbf{f}_\uparrow^{(m)} = \sigma(\mathbf{W}_\uparrow^{(f,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(f,m)}\mathbf{s} + \mathbf{b}_\uparrow^{(f,m)})\\
&\mathbf{o}_\uparrow = \sigma(\textstyle\sum_{m=1}^{M} \mathbf{W}_\uparrow^{(o,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(o)}\mathbf{s} + \mathbf{b}_\uparrow^{(o)})\\
&\mathbf{u}_\uparrow = \tanh(\textstyle\sum_{m=1}^{M} \mathbf{W}_\uparrow^{(u,m)}\mathbf{h}_\uparrow^{(m)} + \mathbf{U}^{(u)}\mathbf{s} + \mathbf{b}_\uparrow^{(u)})\\
&\mathbf{c}_\uparrow = \mathbf{i}_\uparrow \odot \mathbf{u}_\uparrow + \textstyle\sum_{m=1}^{M} \mathbf{f}_\uparrow^{(m)} \odot \mathbf{c}_\uparrow^{(m)}\\
&\mathbf{h}_\uparrow = \mathbf{o}_\uparrow \odot \tanh(\mathbf{c}_\uparrow)
\end{aligned} \tag{2}$$

where $\mathbf{s}$ is the SLSTM hidden state that encodes the description of code C; $\{\mathbf{h}_{\uparrow}^{(m)}\}_{m=1}^{M}$ and $\{\mathbf{c}_{\uparrow}^{(m)}\}_{m=1}^{M}$ are the bottom-up TLSTM hidden states and memory cells of the children. $\mathbf{W}$, $\mathbf{U}$, $\mathbf{b}$ are component-specific weight matrices and bias vectors. For a leaf node having no children, its only input is the SLSTM hidden state $\mathbf{s}$ and no forget gates are needed.

In the top-down TLSTM, for a non-root node, it has such components: an input gate $\mathbf{i}_{\downarrow}$, a forget gate $\mathbf{f}_{\downarrow}$, an output gate $\mathbf{o}_{\downarrow}$, a memory cell $\mathbf{c}_{\downarrow}$ and a hidden state $\mathbf{h}_{\downarrow}$. The transition equations are:

$$\begin{aligned}
\mathbf{i}_{\downarrow} &= \sigma(\mathbf{W}_{\downarrow}^{(i)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(i)})\\
\mathbf{f}_{\downarrow} &= \sigma(\mathbf{W}_{\downarrow}^{(f)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(f)})\\
\mathbf{o}_{\downarrow} &= \sigma(\mathbf{W}_{\downarrow}^{(o)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(o)})\\
\mathbf{u}_{\downarrow} &= \tanh(\mathbf{W}_{\downarrow}^{(u)}\mathbf{h}_{\downarrow}^{(p)} + \mathbf{b}_{\downarrow}^{(u)})\\
\mathbf{c}_{\downarrow} &= \mathbf{i}_{\downarrow} \odot \mathbf{u}_{\downarrow} + \mathbf{f}_{\downarrow} \odot \mathbf{c}_{\downarrow}^{(p)}\\
\mathbf{h}_{\downarrow} &= \mathbf{o}_{\downarrow} \odot \tanh(\mathbf{c}_{\downarrow})
\end{aligned} \tag{3}$$

where $\mathbf{h}_{\downarrow}^{(p)}$ and $\mathbf{c}_{\downarrow}^{(p)}$ are the top-down TLSTM hidden state and memory cell of the parent of this node. For the root node which has no parent, $\mathbf{h}_{\downarrow}$ cannot be computed using the above equations. Instead, we set $\mathbf{h}_{\downarrow}$ to $\mathbf{h}_{\uparrow}$ (the bottom-up TLSTM hidden state generated at the root node). $\mathbf{h}_{\uparrow}$ captures the semantics of all codes in this hierarchy, which is then propagated downwards to each individual code via the top-down TLSTM dynamics.

We concatenate the hidden states of the two directions to obtain the bidirectional TLSTM encoding of each code $\mathbf{h} = [\mathbf{h}_{\uparrow}; \mathbf{h}_{\downarrow}]$. The bottom-up TLSTM composes the semantics of children (representing sub-diseases) and merge them into the current node, which hence captures child-to-parent relationship. The top-down TLSTM makes each node inherit the semantics of its parent, which captures parent-to-child relation. As a result, the hierarchical relationship among codes are encoded in the hidden states.

For the diagnosis descriptions of a patient, we use an SLSTM network to encode each description individually. The weight parameters of this SLSTM are tied with those of the SLSTM used for encoding code descriptions.

4.3 Attentional Matching

Next, we introduce how to map the DDs to codes. We denote the hidden representations of DDs and codes as $\{\mathbf{h}_m\}_{m=1}^{M}$ and $\{\mathbf{u}_n\}_{n=1}^{N}$ respectively, where M is the number of DDs of one patient and N is the total number of codes in the dataset. The mapping from DDs to codes is not one-to-one. In many cases, a code is assigned only when a certain combination of K ($1 < K \leq M$) diseases simultaneously appear within the M DDs and the value of K depends on this code. Among the K diseases, their importance of determining the assignment of this code is different. For the rest $M - K$ DDs, we can consider their importance score to be zero. We use a soft-attention mechanism (Bahdanau et al., 2014) to calculate these importance scores. For a code $\mathbf{u}_n$, the importance of a DD $\mathbf{h}_m$ to $\mathbf{u}_n$ is calculated as $a_{nm} = \mathbf{u}_n^{\top}\mathbf{h}_m$. We normalize the scores $\{a_{nm}\}_{m=1}^{M}$ of all DDs into a probabilistic simplex using the softmax operation: $\tilde{a}_{nm} = \exp(a_{nm})/\sum_{l=1}^{M}\exp(a_{nl})$. Given these normalized importance scores $\{\tilde{a}_{nm}\}_{m=1}^{M}$, we use them to weight the representations of DDs and get a single attentional vector of the M DDs: $\widehat{\mathbf{h}}_n = \sum_{m=1}^{M}\tilde{a}_{nm}\mathbf{h}_m$. Then we concatenate $\widehat{\mathbf{h}}_n$ and $\mathbf{u}_n$, and use a linear classifier to predict the probability that code n should be assigned: $p_n = \mathrm{sigmoid}(\mathbf{w}_n^{\top}[\widehat{\mathbf{h}}_n; \mathbf{u}_n] + b_n)$, where the coefficients $\mathbf{w}_n$ and bias b_n are specific to code n.

We train the weight parameters Θ of the proposed model using the data of L patient visits. Θ includes the sequential LSTM weights $\mathbf{W}_s$, tree LSTM weights $\mathbf{W}_t$ and weights $\mathbf{W}_p$ in the final prediction layer. Let $c^{(l)} \in \mathbb{R}^N$ be a binary vector where $c_n^{(l)} = 1$ if the n-th code is assigned to this patient and $c_n^{(l)} = 0$ if otherwise. Θ can be learned by minimizing the following prediction loss:

$$\min_{\Theta} \quad \mathcal{L}_{\text{pred}}(\Theta) = \sum_{l=1}^{L}\sum_{n=1}^{N}\mathrm{CE}(p_n^{(l)}, c_n^{(l)}) \tag{4}$$

where $p_n^{(l)}$ is the predicted probability that code n is assigned to patient visit l and $p_n^{(l)}$ is a function of Θ. $\mathrm{CE}(\cdot,\cdot)$ is the cross-entropy loss.

4.4 Adversarial Reconciliation of Writing Styles

We use an adversarial learning (Goodfellow et al., 2014) approach to reconcile the different writing styles of diagnosis descriptions (DDs) and code descriptions (CDs). The basic idea is: after encoded, if a description cannot be discerned to be a DD or a CD, then the difference in their writing styles is eliminated. We build a discriminative network which takes the encoding vector of a description as input and tries to identify it as a DD or CD. The encoders of DDs and CDs adjust their weight parameters so that such a discrimination is difficult to be achieved by the discriminative network. Consider all the descriptions $\{t_r, y_r\}_{r=1}^{R}$ where t_r is a description and y_r is a binary label. $y_r = 1$ if t_r is a DD and $y_r = 0$ if otherwise. Let $f(t_r; \mathbf{W}_s)$ denote the sequential LSTM (SLSTM) encoder parameterized by $\mathbf{W}_s$. This encoder is shared by the DDs and CDs. Note that for CDs, a tree LSTM is further applied on top of the encodings produced by the SLSTM. We use the SLSTM encoding vectors of CDs as the input of the discriminative network rather than using the TLSTM encodings since the latter are irrelevant to writing styles. Let $g(f(t_r; \mathbf{W}_s); \mathbf{W}_d)$ denote the discriminative network parameterized by $\mathbf{W}_d$. It takes the encoding vector $f(t_r; \mathbf{W}_s)$ as input and produces the probability that t_r is a DD. Adversarial learning is performed by solving this problem:

$$\max_{\mathbf{W}_s} \min_{\mathbf{W}_d} \mathcal{L}_{\text{adv}} = \sum_{r=1}^{R} \text{CE}(g(f(t_r; \mathbf{W}_s); \mathbf{W}_d), y_r) \tag{5}$$

The discriminative network tries to differentiate DDs from CDs by minimizing this classification loss while the encoder maximizes this loss so that DDs and CDs are not distinguishable.

4.5 Isotonic Constraints

Next, we incorporate the importance order among ICD codes. For the $D^{(l)}$ codes assigned to patient l, without loss of generality, we assume the order is $1 \succ 2 \cdots \succ D^{(l)}$ (the order is given by human coders as ground-truth in the MIMIC-III dataset). We use the predicted probability p_i $(1 \leq i \leq D^{(l)})$ defined in Section 4.3 to characterize the importance of code i. To incorporate the order, we impose an isotonic constraint on the probabilities: $p_1^{(l)} \succ p_2^{(l)} \cdots \succ p_{D^{(l)}}^{(l)}$, and solve the following problem:

$$\begin{array}{ll} \min_\Theta & \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d)) \\ s.t. & p_1^{(l)} \succ p_2^{(l)} \cdots \succ p_{D^{(l)}}^{(l)} \\ & \forall l = 1, \cdots, L \end{array} \tag{6}$$

where the probabilities $p_i^{(l)}$ are functions of Θ and λ is a tradeoff parameter.

We develop an algorithm based on the alternating direction method of multiplier (ADMM) (Boyd et al., 2011) to solve the problem defined in Eq.(6). Let $\mathbf{p}^{(l)}$ be a $|D^{(l)}|$-dimensional vector where the i-th element is $p_i^{(l)}$. We first write the problem into an equivalent form

$$\begin{array}{ll} \min_\Theta & \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d)) \\ s.t. & \mathbf{p}^{(l)} = \mathbf{q}^{(l)} \\ & q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)} \\ & \forall l = 1, \cdots, L \end{array} \tag{7}$$

Then we write down the augmented Lagrangian

$$\begin{array}{ll} \min\limits_{\Theta, \mathbf{q}, \mathbf{v}} & \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d)) \\ & + \langle \mathbf{p}^{(l)} - \mathbf{q}^{(l)}, \mathbf{v}^{(l)} \rangle + \frac{\rho}{2} \|\mathbf{p}^{(l)} - \mathbf{q}^{(l)}\|_2^2 \\ s.t. & q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)} \\ & \forall l = 1, \cdots, L \end{array} \tag{8}$$

We solve this problem by alternating between $\{\mathbf{p}^{(l)}\}_{l=1}^{L}$, $\{\mathbf{q}^{(l)}\}_{l=1}^{L}$ and $\{\mathbf{v}^{(l)}\}_{l=1}^{L}$ The sub-problem defined over $\mathbf{q}^{(l)}$ is

$$\begin{array}{ll} \min_{\mathbf{q}^{(l)}} & -\langle \mathbf{q}^{(l)}, \mathbf{v}^{(l)} \rangle + \frac{\rho}{2} \|\mathbf{p}^{(l)} - \mathbf{q}^{(l)}\|_2^2 \\ s.t. & q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)} \end{array} \tag{9}$$

which is an isotonic projection problem and can be solved via the algorithm proposed in (Yu and Xing, 2016). With $\{\mathbf{q}^{(l)}\}_{l=1}^{L}$ and $\{\mathbf{v}^{(l)}\}_{l=1}^{L}$ fixed, the sub-problem is $\min_\Theta$ $\mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d))$ which can be solved using stochastic gradient descent (SGD). The update of $\mathbf{v}^{(l)}$ is simple: $\mathbf{v}^{(l)} = \mathbf{v}^{(l)} + \rho(\mathbf{p}^{(l)} - \mathbf{q}^{(l)})$.

5 Experiments

In this section, we present experiment results.

5.1 Experimental Settings

Out of the 6,984 unique codes, we selected 2,833 codes that have the top frequencies to perform the study. We split the data into a train/validation/test dataset with 40k/7k/12k patient visits respectively. The hyperparameters were tuned on the validation set. The SLSTMs were bidirectional and dropout with 0.5 probability (Srivastava et al., 2014) was used. The size of hidden states in all LSTMs was set to 100. The word embeddings were trained on the fly and their dimension was set to 200. The tradeoff parameter λ was set to 0.1. The parameter ρ in the ADMM algorithm was set to 1. In the SGD algorithm for solving $\min_\Theta \mathcal{L}_{\text{pred}}(\Theta) + \max_{\mathbf{W}_d}(-\lambda \mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d))$, we used the ADAM (Kingma and Ba, 2014) optimizer with an initial learning rate 0.001 and a mini-batch size 20. Sensitivity (true positive rate) and specificity (true negative rate) were used to evaluate the code assignment performance. We calculated these two scores for each individual code on the test set, then took a weighted (proportional to codes' frequencies) average across all codes. To evaluate the ranking performance of codes, we used normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002).

5.2 Ablation Study

We perform ablation study to verify the effectiveness of each module in our model. To evaluate module X, we remove it from the model without changing other modules and denote such a baseline by No-X. The comparisons of No-X with the full model are given in Table 2.

	Sensitivity	Specificity
(Larkey and Croft, 1996)	0.15	0.17
(Franz et al., 2000)	0.19	0.21
(Pestian et al., 2007)	0.12	0.21
(Kavuluru et al., 2013)	0.09	0.11
(Kavuluru et al., 2015)	0.21	0.25
(Koopman et al., 2015)	0.18	0.20
LET	0.23	0.29
HierNet	0.26	0.30
HybridNet	0.25	0.31
BranchNet	0.25	0.29
No-TLSTM	0.23	0.28
Bottom-up TLSTM	0.27	0.31
No-AL	0.26	0.31
No-IC	0.24	0.29
No-AM	0.27	0.29
Our full model	**0.29**	**0.33**

Table 2: Sensitivity and Specificity on the Test Set

Tree-of-sequences LSTM To evaluate this module, we compared with the two configurations: (1) *No-TLSTM*, which removes the tree LSTM and directly uses the hidden states produced by the sequential LSTM as final representations of codes; (2) *Bottom-up TLSTM*, which removes the hidden states generated by the top-down TLSTM. In addition, we compared with four hierarchical classification baselines including (1) hierarchical network (HierNet) (Yan et al., 2015), (2) HybridNet (Hou et al., 2017), (3) branch network (BranchNet) (Zhu and Bain, 2017), (4) label embedding tree (LET) (Bengio et al., 2010), by using them to replace the bidirectional tree LSTM while keeping other modules untouched. Table 2 shows the average sensitivity and specificity scores achieved by these methods on the test set. We make the following observations. First, removing tree LSTM largely degrades performance: the sensitivity and specificity of No-TLSTM is 0.23 and 0.28 respectively while our full model (which uses bidirectional TLSTM) achieves 0.29 and 0.33 respectively. The reason is No-TLSTM ignores the hierarchical relationship among codes. Second, bottom-up tree LSTM alone performs less well than bidirectional tree LSTM. This demonstrates the necessity of the top-down TLSTM, which ensures every two codes are connected by directed paths and can more expressively capture code-relations in the hierarchy. Third, our method outperforms the four baselines. The possible reason is our method directly builds codes' hierarchical relationship into their representations while the baselines perform representation-learning and relationship-capturing separately.

Next, we present some qualitative results. For a patient (admission ID 147798) having a DD 'E Coli urinary tract infection', without using tree LSTM, two sibling codes 585.2 (chronic kidney disease, stage II (mild)) – which is the ground-truth – and 585.4 (chronic kidney disease, stage IV (severe)) are simultaneously assigned possibly because their textual descriptions are very similar (only differ in the level of severity). This is incorrect because 585.2 and 585.4 are the children of 585 (chronic kidney disease) and the severity level of this disease cannot simultaneously be mild and severe. After tree LSTM is added, the false prediction of 585.4 is eliminated, which demonstrates the effectiveness of tree LSTM in incorporating one constraint induced by the code hierarchy: among the nodes sharing the same parent, only one should be selected.

For patient 197205, No-TLSTM assigns the following codes: 462 (subacute sclerosing panencephalitis), 790.29 (other abnormal glucose), 799.9 (unspecified viral infection), and 285.21 (anemia in chronic kidney disease). Among these codes, the first three are ground-truth and the fourth one is incorrect (the ground-truth is 401.9 (unspecified essential hypertension)). Adding tree LSTM fixes this error. The average distance between 401.9 and the rest of ground-truth codes is 6.2. For the incorrectly assigned code 285.21, such a distance is 7.9. This demonstrates that tree LSTM is able to capture another constraint imposed by the hierarchy: codes with smaller tree-distance are more likely to be assigned together.

Position	2	4	6	8
No-IC	0.27	0.26	0.23	0.20
IC	0.32	0.29	0.27	0.23

Table 3: Comparison of NDCG Scores in the Ablation Study of Isotonic Constraints.

Adversarial learning To evaluate the efficacy of adversarial learning (AL), we remove it from the full model and refer to this baseline as No-AL. Specifically, in Eq.(6), the loss term $\max_{\mathbf{W}_d}(-\mathcal{L}_{\text{adv}}(\mathbf{W}_s, \mathbf{W}_d))$ is taken away. Table 2 shows the results, from which we observe that after AL is removed, the sensitivity and specificity are dropped from 0.29 and 0.33 to 0.26 and 0.31 respectively. No-AL does not reconcile different writing styles of diagnosis descriptions (DDs) and code descriptions (CDs). As a result, a DD and a CD that have similar semantics may be mismatched because their writing styles are different. For example, a patient (admission ID 147583) has a DD 'h/o DVT on anticoagulation', which contains abbreviation DVT (deep vein thrombosis). Due to the presence of this abbreviation, it is difficult to assign a proper code to this DD since the textual descriptions of codes do not contain abbreviations. With adversarial learning, our model can correctly map this DD to a ground-truth code: 443.9 (peripheral vascular disease, unspecified). Without AL, this code is not selected. As another example, a DD 'coronary artery disease, STEMI, s/p 2 stents placed in RCA' was given to patient 148532. This DD is written informally and ungrammatically, and contains too much detailed information, e.g., 's/p 2 stents placed in RCA'. Such a writing style is quite different from that of CDs. With AL, our model successfully matches this DD to a ground-truth code: 414.01 (coronary atherosclerosis of native coronary artery). On the contrary, No-AL fails to achieve this.

Isotonic constraint (IC) To evaluate this ingredient, we remove the ICs from Eq.(6) during training and denote this baseline as No-IC. We use NDCG to measure the ranking performance, which is calculated in the following way. Consider a testing patient-visit l where the ground-truth ICD codes are $\mathcal{M}^{(l)}$. For any code c, we define the relevance score of c to l as 0 if $c \notin \mathcal{M}^{(l)}$ and as $|\mathcal{M}^{(l)}| - r(c)$ if otherwise, where $r(c)$ is the ground-truth rank of c in $\mathcal{M}^{(l)}$. We rank codes in descending order of their corresponding prediction probabilities and obtain the predicted rank for each code. We calculate the NDCG scores at position 2, 4, 6, 8 based on the relevance scores and predicted ranks, which are shown in Table 3. As can be seen, using IC achieves much higher NDCG than No-IC, which demonstrates the effectiveness of IC in capturing the importance order among codes.

We also evaluate how IC affects the sensitivity and specificity of code assignment. As can be seen from Table 2, No-IC degrades the two scores from 0.29 and 0.33 to 0.24 and 0.29 respectively, which indicates that IC is helpful in training a model that can more correctly assign codes. This is because IC encourages codes that are highly relevant to the patients to be ranked at top positions, which prevents the selection of irrelevant codes.

Attentional matching (AM) In the evaluation of this module, we compare with a baseline – No-AM, which performs an unweighted average of the M DDs: $\widehat{\mathbf{h}}_n = \frac{1}{M}\sum_{m=1}^{M}\mathbf{h}_m$, concatenates $\widehat{\mathbf{h}}_n$ with $\mathbf{u}_n$ and feeds the concatenated vector into the final prediction layer. From Table 2, we can see our full model (with AM) outperforms No-AM, which demonstrates the effectiveness of attentional matching. In determining whether a code should be assigned, different DDs have different importance weights. No-AM ignores such weights, therefore performing less well.

AM can correctly perform many-to-one mapping from multiple DDs to a CD. For example, patient 190236 was given two DDs: 'renal insufficiency' and 'acute renal failure'. AM maps them to a combined ICD code: 403.91 (hypertensive chronic kidney disease, unspecified, with chronic kidney disease stage V or end stage renal disease), which is in the ground-truth provided by medical coders. On the contrary, No-AM fails to assign this code. On the other hand, AM is able to correctly map a DD to multiple CDs. For example, a DD 'congestive heart failure, diastolic' was given to patient 140851. AM successfully maps this DD to two codes: (1) 428.0 (congestive heart failure, unspecified); (2) 428.30 (diastolic heart failure, unspecified). Without AM, this DD is mapped only to 428.0.

5.3 Holistic Comparison with Other Baselines

In addition to evaluating the four modules individually, we also compared our full model with four other baselines proposed by (Larkey and Croft, 1996; Franz et al., 2000; Pestian et al., 2007; Kavuluru et al., 2013, 2015; Koopman et al., 2015) for ICD coding. Table 2 shows the results. As can be seen, our approach achieves much better sensitivity and specificity scores. The reason that our model works better is two-fold. First, our model is based on deep neural network, which has arguably better modeling power than linear methods used in the baselines. Second, our model is able to capture the hierarchical relationship and importance order among codes, can alleviate the discrepancy in writing styles and allows flexible many-to-one and one-to-many mappings from DDs to CDs. These merits are not possessed by the baselines.

6 Conclusions and Discussions

In this paper, we build a neural network model for automated ICD coding. Evaluations on the MIMIC-III dataset demonstrate the following. First, the tree-of-sequences LSTM network effectively discourages the co-selection of sibling codes and promotes the co-assignment of clinically-relevant codes. Adversarial learning improves the matching accuracy by alleviating the discrepancy among the writing styles of DDs and CDs. Third, isotonic constraints promote the correct ranking of codes. Fourth, the attentional matching mechanism is able to perform many-to-one and one-to-many mappings.

In the coding practice of human coders, in addition to the diagnosis descriptions, other information contained in nursing notes, lab values, and medical procedures are also leveraged for code assignment. We have initiated preliminary investigation along this line and added two new input sources: (1) the rest of discharge summary and (2) lab values. The sensitivity is improved from 0.29 to 0.32 and the specificity is improved from 0.33 to 0.35. A full study is ongoing.

At present, the major limitations of this work include: (1) it does not perform well on infrequent codes; (2) it is less capable of dealing with abbreviations. We will address these two issues in future by investigating diversity-promoting regularization (Xie et al., 2017) and leveraging an external knowledge base that maps medical abbreviations into their full names.

The proposed methods can be applied to other tasks in NLP. The tree-of-sequences model can be applied for ontology annotation. It takes the textual descriptions of concepts in the ontology and their hierarchical structure as inputs and produces a latent representation for each concept. The representations can simultaneously capture the semantics of codes and their relationships. The proposed adversarial reconciliation of writing styles and attentional matching can be applied for knowledge mapping or entity linking. For example, in tweets, we can use the method to map an informally written mention 'nbcbightlynews' to a canonical entity 'NBC Nightly News' in the knowledge base.

Acknowledgements

We would like to thank the anonymous reviewers for their very constructive and helpful comments and suggestions. Pengtao Xie and Eric P. Xing are supported by National Institutes of Health P30DA035778, Pennsylvania Department of Health BD4BH4100070287, and National Science Foundation IIS1617583.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Jason Weston, and David Grangier. 2010. Label embedding trees for large multi-class tasks. In *NIPS*.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Richárd Farkas and György Szarvas. 2008. Automatic construction of rule-based icd-9-cm coding systems. *BMC bioinformatics*, 9(3):S10.

Pius Franz, Albrecht Zaiss, Stefan Schulz, Udo Hahn, and Rüdiger Klar. 2000. Automated coding of diagnoses–three methods compared. In *Proceedings of the AMIA Symposium*, page 250. American Medical Informatics Association.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron

Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

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

Saihui Hou, Yushan Feng, and Zilei Wang. 2017. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 541–549.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Liwei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3.

Ramakanth Kavuluru, Sifei Han, and Daniel Harris. 2013. Unsupervised extraction of diagnosis codes from emrs using knowledge-based and extractive text summarization techniques. In *Canadian conference on artificial intelligence*, pages 77–88. Springer.

Ramakanth Kavuluru, Anthony Rios, and Yuan Lu. 2015. An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records. *Artificial intelligence in medicine*, 65(2):155–166.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Bevan Koopman, Guido Zuccon, Anthony Nguyen, Anton Bergheim, and Narelle Grayson. 2015. Automatic icd-10 classification of cancers from free-text death certificates. *International journal of medical informatics*, 84(11):956–965.

Dee Lang. 2007. Consultant report-natural language processing in the health care industry. *Cincinnati Children's Hospital Medical Center, Winter.*

Leah S Larkey and W Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 289–297. ACM.

Kimberly J O'malley, Karon F Cook, Matt D Price, Kimberly Raiford Wildes, John F Hurdle, and Carol M Ashton. 2005. Measuring diagnoses: Icd code accuracy. *Health services research*, 40(5p2):1620–1639.

World Health Organization et al. 1978. International classification of diseases:[9th] ninth revision, basic tabulation list with alphabetic index. *World Health Organization.*

John P Pestian, Christopher Brew, Paweł Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603.*

Joanna E Sheppard, Laura CE Weidner, Saher Zakai, Simon Fountain-Polley, and Judith Williams. 2008. Ambiguous abbreviations: an audit of abbreviations in paediatric note keeping. *Archives of disease in childhood*, 93(3):204–206.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075.*

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured lstm with head lexicalization. *arXiv preprint arXiv:1611.06788.*

Pengtao Xie, Aarti Singh, and Eric P. Xing. 2017. Uncorrelation and evenness: a new diversity-promoting regularizer. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3811–3820.

Pengtao Xie and Eric Xing. 2017. A constituent-centric neural architecture for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1405–1414.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention.

Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. 2015. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2740–2748.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Yao-Liang Yu and Eric P Xing. 2016. Exact algorithms for isotonic regression and related. In *Journal of Physics: Conference Series*, volume 699, page 012016. IOP Publishing.

Xinqi Zhu and Michael Bain. 2017. B-cnn: Branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890*.

Domain Adaptation with Adversarial Training and Graph Embeddings

Firoj Alam[*], **Shafiq Joty**[†], and **Muhammad Imran**[*]
Qatar Computing Research Institute, HBKU, Qatar[*]
School of Computer Science and Engineering[†]
Nanyang Technological University, Singapore[†]
{fialam, mimran}@hbku.edu.qa[*]
srjoty@ntu.edu.sg[†]

Abstract

The success of deep neural networks (DNNs) is heavily dependent on the availability of labeled data. However, obtaining labeled data is a big challenge in many real-world problems. In such scenarios, a DNN model can leverage labeled and unlabeled data from a related domain, but it has to deal with the shift in data distributions between the source and the target domains. In this paper, we study the problem of classifying social media posts during a crisis event (e.g., Earthquake). For that, we use labeled and unlabeled data from past similar events (e.g., Flood) and unlabeled data for the current event. We propose a novel model that performs adversarial learning based domain adaptation to deal with distribution drifts and graph based semi-supervised learning to leverage unlabeled data within a single unified deep learning framework. Our experiments with two real-world crisis datasets collected from Twitter demonstrate significant improvements over several baselines.

1 Introduction

The application that motivates our work is the time-critical analysis of social media (Twitter) data at the sudden-onset of an event like natural or man-made disasters (Imran et al., 2015). In such events, affected people post timely and useful information of various types such as reports of injured or dead people, infrastructure damage, urgent needs (e.g., food, shelter, medical assistance) on these social networks. Humanitarian organizations believe timely access to this important information from social networks can help significantly and reduce both human loss and economic damage (Varga et al., 2013; Vieweg et al., 2014; Power et al., 2013).

In this paper, we consider the basic task of classifying each incoming tweet during a crisis event (e.g., Earthquake) into one of the predefined classes of interest (e.g., *relevant* vs. *non-relevant*) in real-time. Recently, deep neural networks (DNNs) have shown great performance in classification tasks in NLP and data mining. However the success of DNNs on a task depends heavily on the availability of a large labeled dataset, which is not a feasible option in our setting (i.e., classifying tweets at the onset of an Earthquake). On the other hand, in most cases, we can have access to a good amount of labeled and abundant unlabeled data from past similar events (e.g., Floods) and possibly some unlabeled data for the current event. In such situations, we need methods that can leverage the labeled and unlabeled data in a past event (we refer to this as a *source* domain), and that can adapt to a new event (we refer to this as a *target* domain) without requiring any labeled data in the new event. In other words, we need models that can do *domain adaptation* to deal with the distribution drift between the domains and *semi-supervised* learning to leverage the unlabeled data in both domains.

Most recent approaches to semi-supervised learning (Yang et al., 2016) and domain adaptation (Ganin et al., 2016) use the automatic feature learning capability of DNN models. In this paper, we extend these methods by proposing a novel model that performs domain adaptation and semi-supervised learning within a single unified deep learning framework. In this framework, the basic task-solving network (a convolutional neural network in our case) is put together with two other networks – one for semi-supervised learning and the other for domain adaptation. The semi-supervised component learns internal representations (features) by predicting contextual nodes in a graph that encodes *similarity* between labeled and unlabeled training instances. The domain adaptation is achieved by training the feature extractor (or encoder) in *adversary* with respect to a domain discriminator, a binary classifier that tries to distinguish the domains. The overall idea is to learn high-level abstract representation that is discriminative for the main classification task, but is invariant across the domains. We propose a stochastic gradient descent (SGD) algorithm to train the components of our model simultaneously.

The evaluation of our proposed model is conducted using two Twitter datasets on scenarios where there is only unlabeled data in the target domain. Our results demonstrate the following.

1. When the network combines the semi-supervised component with the supervised component, depending on the amount of labeled data used, it gives 5% to 26% absolute gains in F1 compared to when it uses only the supervised component.
2. Domain adaptation with adversarial training improves over the adaptation baseline (i.e., a transfer model) by 1.8% to 4.1% absolute F1.
3. When the network combines domain adversarial training with semi-supervised learning, we get further gains ranging from 5% to 7% absolute in F1 across events.

Our source code is available on Github[1] and the data is available on CrisisNLP[2].

The rest of the paper is organized as follows. In Section 2, we present the proposed method, i.e., domain adaptation and semi-supervised graph embedding learning. In Section 3, we present the experimental setup and baselines. The results and analysis are presented in Section 4. In Section 5, we present the works relevant to this study. Finally, conclusions appear in Section 6.

2 The Model

We demonstrate our approach for domain adaptation with adversarial training and graph embedding on a tweet classification task to support crisis response efforts. Let $\mathcal{D}_S^l = \{\mathbf{t}_i, y_i\}_{i=1}^{L_s}$ and $\mathcal{D}_S^u = \{\mathbf{t}_i\}_{i=1}^{U_s}$ be the set of labeled and unlabeled tweets for a source crisis event S (e.g., Nepal earthquake), where $y_i \in \{1, \ldots, K\}$ is the class label for tweet $\mathbf{t}_i$, L_s and U_s are the number of labeled and unlabeled tweets for the source event, respectively. In addition, we have unlabeled tweets $\mathcal{D}_T^u = \{\mathbf{t}_i\}_{i=1}^{U_t}$ for a target event T (e.g., Queensland flood) with U_t being the number of unlabeled tweets in the target domain. Our ultimate goal is to train a cross-domain model $p(y|\mathbf{t}, \theta)$ with parameters θ that can classify any tweet in the target event T without having any information about class labels in T.

Figure 1 shows the overall architecture of our neural model. The input to the network is a tweet $\mathbf{t} = (w_1, \ldots, w_n)$ containing words that come from a finite vocabulary $\mathcal{V}$ defined from the training set. The first layer of the network maps each of these words into a distributed representation $\mathbb{R}^d$ by looking up a shared embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$. We initialize the embedding matrix E in our network with word embeddings that are pretrained on a large crisis dataset (Subsection 2.5). However, embedding matrix E can also be initialize randomly. The output of the look-up layer is a matrix $X \in \mathbb{R}^{n \times d}$, which is passed through a number of convolution and pooling layers to learn higher-level feature representations. A *convolution* operation applies a *filter* $\mathbf{u} \in \mathbb{R}^{k.d}$ to a window of k vectors to produce a new feature h_t as

$$h_t = f(\mathbf{u}.X_{t:t+k-1}) \tag{1}$$

where $X_{t:t+k-1}$ is the concatenation of k look-up vectors, and f is a nonlinear activation; we use rectified linear units or ReLU. We apply this filter to each possible k-length windows in X with stride size of 1 to generate a *feature map* $\mathbf{h}^j$ as:

$$\mathbf{h}^j = [h_1, \ldots, h_{n+k-1}] \tag{2}$$

We repeat this process N times with N different filters to get N different feature maps. We use a *wide* convolution (Kalchbrenner et al., 2014), which ensures that the filters reach the entire tweet, including the boundary words. This is done by performing *zero-padding*, where out-of-range (i.e., $t<1$ or $t>n$) vectors are assumed to be zero. With wide convolution, o zero-padding size and 1 stride size, each feature map contains $(n + 2o - k + 1)$ convoluted features. After the convolution, we apply a *max-pooling* operation to each of the feature maps,

$$\mathbf{m} = [\mu_p(\mathbf{h}^1), \cdots, \mu_p(\mathbf{h}^N)] \tag{3}$$

[1]https://github.com/firojalam/domain-adaptation
[2]http://crisisnlp.qcri.org

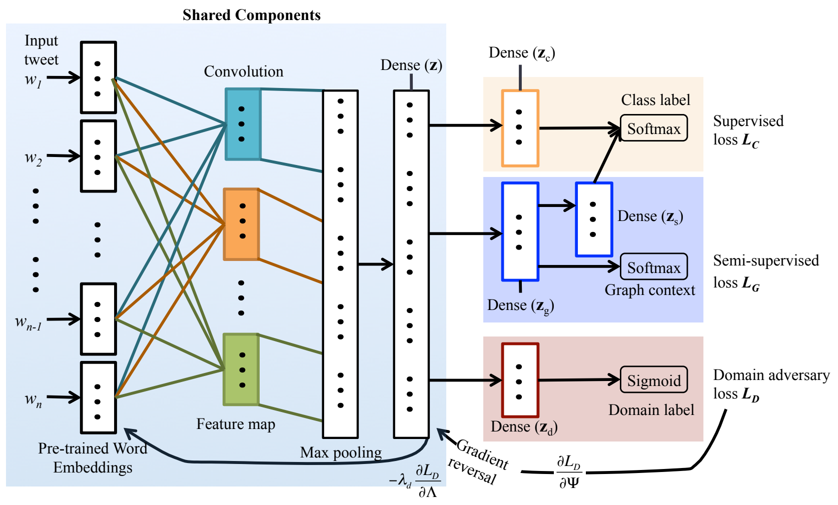


Figure 1: The system architecture of the domain adversarial network with graph-based semi-supervised learning. The shared components part is shared by supervised, semi-supervised and domain classifier.

where $\mu_p(\mathbf{h}^j)$ refers to the $\max$ operation applied to each window of p features with stride size of 1 in the feature map $\mathbf{h}^i$. Intuitively, the convolution operation composes local features into higher-level representations in the feature maps, and max-pooling extracts the most important aspects of each feature map while reducing the output dimensionality. Since each convolution-pooling operation is performed independently, the features extracted become invariant in order (i.e., where they occur in the tweet). To incorporate order information between the pooled features, we include a fully-connected (dense) layer

$$\mathbf{z} = f(V\mathbf{m}) \tag{4}$$

where V is the weight matrix. We choose a convolutional architecture for feature composition because it has shown impressive results on similar tasks in a supervised setting (Nguyen et al., 2017).

The network at this point splits into three branches (shaded with three different colors in Figure 1) each of which serves a different purpose and contributes a separate loss to the overall loss of the model as defined below:

$$\mathcal{L}(\Lambda, \Phi, \Omega, \Psi) = \mathcal{L}_C(\Lambda, \Phi) + \lambda_g \mathcal{L}_G(\Lambda, \Omega) + \lambda_d \mathcal{L}_D(\Lambda, \Psi) \tag{5}$$

where $\Lambda = \{U, V\}$ are the convolutional filters and dense layer weights that are shared across the three branches. The first component $\mathcal{L}_C(\Lambda, \Phi)$ is a supervised classification loss based on the labeled data in the source event. The second component $\mathcal{L}_G(\Lambda, \Omega)$ is a graph-based semi-supervised loss that utilizes both labeled and unlabeled data in the source and target events to induce structural similarity between training instances. The third component $\mathcal{L}_D(\Lambda, \Omega)$ is an adversary loss that again uses all available data in the source and target domains to induce domain invariance in the learned features. The tunable hyperparameters λ_g and λ_d control the relative strength of the components.

2.1 Supervised Component

The supervised component induces label information (e.g., *relevant* vs. *non-relevant*) directly in the network through the classification loss $\mathcal{L}_C(\Lambda, \Phi)$, which is computed on the labeled instances in the source event, $\mathcal{D}_S^l$. Specifically, this branch of the network, as shown at the top in Figure 1, takes the shared representations $\mathbf{z}$ as input and pass it through a task-specific dense layer

$$\mathbf{z}_c = f(V_c \mathbf{z}) \tag{6}$$

where V_c is the corresponding weight matrix. The activations $\mathbf{z}_c$ along with the activations from the semi-supervised branch $\mathbf{z}_s$ are used for classification. More formally, the classification layer defines a Softmax

$$p(y = k|\mathbf{t}, \theta) = \frac{\exp\left(W_k^T [\mathbf{z}_c; \mathbf{z}_s]\right)}{\sum_{k'} \exp\left(W_{k'}^T [\mathbf{z}_c; \mathbf{z}_s]\right)} \tag{7}$$

where $[.;.]$ denotes concatenation of two column vectors, W_k are the class weights, and $\theta = \{U, V, V_c, W\}$ defines the relevant parameters for this branch of the network with $\Lambda = \{U, V\}$ being the shared parameters and $\Phi = \{V_c, W\}$ being the parameters specific to this branch. Once learned, we use θ for prediction on test tweets. The classification loss $\mathcal{L}_C(\Lambda, \Phi)$ (or $\mathcal{L}_C(\theta)$) is defined as

$$\mathcal{L}_C(\Lambda, \Phi) = -\frac{1}{L_s}\sum_{i=1}^{L_s} \mathbb{I}(y_i = k) \log p(y_i = k|\mathbf{t}_i, \Lambda, \Phi) \quad (8)$$

where $\mathbb{I}(.)$ is an indicator function that returns 1 when the argument is true, otherwise it returns 0.

2.2 Semi-supervised Component

The semi-supervised branch (shown at the middle in Figure 1) induces structural similarity between training instances (labeled or unlabeled) in the source and target events. We adopt the recently proposed graph-based semi-supervised deep learning framework (Yang et al., 2016), which shows impressive gains over existing semi-supervised methods on multiple datasets. In this framework, a "similarity" graph G first encodes relations between training instances, which is then used by the network to learn internal representations (i.e., embeddings).

2.2.1 Learning Graph Embeddings

The semi-supervised branch takes the shared representation $\mathbf{z}$ as input and learns internal representations by predicting a node in the graph context of the input tweet. Following (Yang et al., 2016), we use *negative sampling* to compute the loss for predicting the context node, and we sample two types of contextual nodes: (*i*) one is based on the graph G to encode structural information, and (*ii*) the second is based on the labels in $\mathcal{D}_S^l$ to incorporate label information through this branch of the network. The ratio of positive and negative samples is controlled by a random variable $\rho_1 \in (0, 1)$, and the proportion of the two context types is controlled by another random variable $\rho_2 \in (0, 1)$; see Algorithm 1 of (Yang et al., 2016) for details on the sampling procedure.

Let (j, γ) is a tuple sampled from the distribution $p(j, \gamma|i, \mathcal{D}_S^l, G)$, where j is a context node of an input node i and $\gamma \in \{+1, -1\}$ denotes whether it is a positive or a negative sample; $\gamma = +1$ if $\mathbf{t}_i$ and $\mathbf{t}_j$ are neighbors in the graph (for graph-based context) or they both have same labels (for label-based context), otherwise $\gamma = -1$. The negative log loss for context prediction $\mathcal{L}_G(\Lambda, \Omega)$ can be written as

$$\mathcal{L}_G(\Lambda, \Omega) = -\frac{1}{L_s + U_s}\sum_{i=1}^{L_s+U_s} \mathbb{E}_{(j,\gamma)} \log \sigma\left(\gamma C_j^T \mathbf{z}_g(i)\right) \quad (9)$$

where $\mathbf{z}_g(i) = f(V_g\mathbf{z}(i))$ defines another dense layer (marked as *Dense* ($\mathbf{z}_g$) in Figure 1) having weights V_g, and C_j is the weight vector associated with the context node $\mathbf{t}_j$. Note that here $\Lambda = \{U, V\}$ defines the shared parameters and $\Omega = \{V_g, C\}$ defines the parameters specific to the semi-supervised branch of the network.

2.2.2 Graph Construction

Typically graphs are constructed based on a relational knowledge source, e.g., citation links in (Lu and Getoor, 2003), or distance between instances (Zhu, 2005). However, we do not have access to such a relational knowledge in our setting. On the other hand, computing distance between $n(n-1)/2$ pairs of instances to construct the graph is also very expensive (Muja and Lowe, 2014). Therefore, we choose to use k-nearest neighbor-based approach as it has been successfully used in other study (Steinbach et al., 2000).

The nearest neighbor graph consists of n vertices and for each vertex, there is an edge set consisting of a subset of n instances, i.e., tweets in our training set. The edge is defined by the distance measure $d(i, j)$ between tweets $\mathbf{t}_i$ and $\mathbf{t}_j$, where the value of d represents how similar the two tweets are. We used k-d tree data structure (Bentley, 1975) to efficiently find the nearest instances. To construct the graph, we first represent each tweet by averaging the word2vec vectors of its words, and then we measure $d(i, j)$ by computing the *Euclidean* distance between the vectors. The number of nearest neighbor k was set to 10. The reason of averaging the word vectors is that it is computationally simpler and it captures the relevant semantic information for our task in hand. Likewise, we choose to use Euclidean distance instead of cosine for computational efficiency.

2.3 Domain Adversarial Component

The network described so far can learn abstract features through convolutional and dense layers that are discriminative for the classification task (*relevant* vs. *non-relevant*). The supervised branch of the network uses labels in the source event to induce label information directly, whereas the semi-supervised branch induces similarity information between labeled and unlabeled instances. However, our goal is also to make these learned features invariant across domains or events (e.g., *Nepal Earthquake* vs. *Queensland Flood*). We achieve this by domain *adversarial* training of neural networks (Ganin et al., 2016).

We put a domain discriminator, another branch in the network (shown at the bottom in Figure 1) that takes the shared internal representation $\mathbf{z}$ as input, and tries to discriminate between the domains of the input — in our case, whether the input tweet is from $\mathcal{D}_S$ or from $\mathcal{D}_T$. The domain discriminator is defined by a sigmoid function:

$$\hat{\delta} = p(d = 1|\mathbf{t}, \Lambda, \Psi) = \text{sigm}(\mathbf{w}_d^T \mathbf{z}_d) \quad (10)$$

where $d \in \{0, 1\}$ denotes the domain of the input tweet $\mathbf{t}$, $\mathbf{w}_d$ are the final layer weights of the discriminator, and $\mathbf{z}_d = f(V_d \mathbf{z})$ defines the hidden layer of the discriminator with layer weights V_d. Here $\Lambda = \{U, V\}$ defines the shared parameters, and $\Psi = \{V_d, \mathbf{w}_d\}$ defines the parameters specific to the domain discriminator. We use the negative log-probability as the discrimination loss:

$$\mathcal{J}_i(\Lambda, \Psi) = -d_i \log \hat{\delta} - (1 - d_i) \log \left(1 - \hat{\delta}\right) \quad (11)$$

We can write the overall domain adversary loss over the source and target domains as

$$\mathcal{L}_D(\Lambda, \Psi) = -\frac{1}{L_s + U_s} \sum_{i=1}^{L_s+U_s} \mathcal{J}_i(\Lambda, \Psi) - \frac{1}{U_t} \sum_{i=1}^{U_t} \mathcal{J}_i(\Lambda, \Psi) \quad (12)$$

where $L_s + U_s$ and U_t are the number of training instances in the source and target domains, respectively. In adversarial training, we seek parameters (saddle point) such that

$$\theta^* = \underset{\Lambda, \Phi, \Omega}{\text{argmin}} \max_{\Psi} \mathcal{L}(\Lambda, \Phi, \Omega, \Psi) \quad (13)$$

which involves a maximization with respect to Ψ and a minimization with respect to $\{\Lambda, \Phi, \Omega\}$. In other words, the updates of the shared parameters $\Lambda = \{U, V\}$ for the discriminator work adversarially to the rest of the network, and vice versa. This is achieved by reversing the gradients of the discrimination loss $\mathcal{L}_D(\Lambda, \Psi)$, when they are back-propagated to the shared layers (see Figure 1).

2.4 Model Training

Algorithm 1 illustrates the training algorithm based on stochastic gradient descent (SGD). We first initialize the model parameters. The word embedding matrix E is initialized with pre-trained word2vec vectors (see Subsection 2.5) and is kept fixed during training.[3] Other parameters are initialized with small random numbers sampled from a uniform distribution (Bengio and Glorot, 2010). We use AdaDelta (Zeiler, 2012) adaptive update to update the parameters.

[3]Tuning E on our task by backpropagation increased the training time immensely (3 days compared to 5 hours on a Tesla GPU) without any significant performance gain.

Algorithm 1: Model Training with SGD

```
Input  : data $\mathcal{D}_S^l$, $\mathcal{D}_S^u$, $\mathcal{D}_T^u$; graph $G$
Output: learned parameters $\theta = \{\Lambda, \Phi\}$
1. Initialize model parameters $\{E, \Lambda, \Phi, \Omega, \Psi\}$;
2. repeat
      // Semi-supervised
      for each batch sampled from $p(j, \gamma|i, \mathcal{D}_S^l, G)$ do
          a) Compute loss $\mathcal{L}_G(\Lambda, \Omega)$
          b) Take a gradient step for $\mathcal{L}_G(\Lambda, \Omega)$;
      end
      // Supervised & domain adversary
      for each batch sampled from $\mathcal{D}_S^l$ do
          a) Compute $\mathcal{L}_C(\Lambda, \Phi)$ and $\mathcal{L}_D(\Lambda, \Psi)$
          b) Take gradient steps for $\mathcal{L}_C(\Lambda, \Phi)$ and
          $\mathcal{L}_D(\Lambda, \Psi)$;
      end
      // Domain adversary
      for each batch sampled from $\mathcal{D}_S^u$ and $\mathcal{D}_T^u$ do
          a) Compute $\mathcal{L}_D(\Lambda, \Psi)$
          b) Take a gradient step for $\mathcal{L}_D(\Lambda, \Psi)$;
      end
   until convergence;
```

In each iteration, we do three kinds of gradient updates to account for the three different loss components. First, we do an epoch over all the training instances updating the parameters for the semi-supervised loss, then we do an epoch over the labeled instances in the source domain, each time updating the parameters for the supervised and the domain adversary losses. Finally, we do an epoch over the unlabeled instances in the two domains to account for the domain adversary loss.

The main challenge in adversarial training is to balance the competing components of the network. If one component becomes smarter than the other, its loss to the shared layer becomes useless, and the training fails to converge (Arjovsky et al., 2017). Equivalently, if one component becomes weaker, its loss overwhelms that of the other, causing the training to fail. In our experiments, we observed the domain discriminator is weaker than the rest of the network. This could be due to the noisy nature of tweets, which makes the job for the domain discriminator harder. To balance the components, we would want the error signals from the discriminator to be fairly weak, also we would want the supervised loss to have more impact than the semi-supervised loss. In our experiments, the weight of the domain adversary loss λ_d was fixed to $1e - 8$, and the weight of the semi-supervised loss λ_g was fixed to $1e - 2$. Other sophisticated weighting schemes have been proposed recently

(Ganin et al., 2016; Arjovsky et al., 2017; Metz et al., 2016). It would be interesting to see how our model performs using these advanced tuning methods, which we leave as a future work.

2.5 Crisis Word Embedding

As mentioned, we used word embeddings that are pre-trained on a crisis dataset. To train the word-embedding model, we first pre-processed tweets collected using the AIDR system (Imran et al., 2014) during different events occurred between 2014 and 2016. In the preprocessing step, we lowercased the tweets and removed URLs, digit, time patterns, special characters, single character, username started with the @ symbol. After pre-processing, the resulting dataset contains about 364 million tweets and about 3 billion words.

There are several approaches to train word embeddings such as continuous bag-of-words (CBOW) and skip-gram models of wrod2vec (Mikolov et al., 2013), and Glove (Pennington et al., 2014). For our work, we trained the CBOW model from word2vec. While training CBOW, we filtered out words with a frequency less than or equal to 5, and we used a context window size of 5 and $k = 5$ negative samples. The resulting embedding model contains about 2 million words with vector dimensions of 300.

3 Experimental Settings

In this section, we describe our experimental settings – datasets used, settings of our models, compared baselines, and evaluation metrics.

3.1 Datasets

To conduct the experiment and evaluate our system, we used two real-world Twitter datasets collected during the *2015 Nepal earthquake* (NEQ) and the *2013 Queensland floods* (QFL). These datasets are comprised of millions of tweets collected through the Twitter streaming API[4] using event-specific keywords/hashtags.

To obtain the labeled examples for our task we employed paid workers from the Crowdflower[5] – a crowdsourcing platform. The annotation consists of two classes *relevant* and *non-relevant*. For the annotation, we randomly sampled 11,670 and 10,033 tweets from the Nepal earthquake and the Queensland floods datasets, respectively. Given a

Dataset	Relevant	Non-relevant	Train	Dev	Test
NEQ	5,527	6,141	7,000	1,167	3,503
QFL	5,414	4,619	6,019	1,003	3,011

Table 1: Distribution of labeled datasets for Nepal earthquake (NEQ) and Queensland flood (QFL).

tweet, we asked crowdsourcing workers to assign the *"relevant"* label if the tweet conveys/reports information useful for crisis response such as a report of injured or dead people, some kind of infrastructure damage, urgent needs of affected people, donations requests or offers, otherwise assign the *"non-relevant"* label. We split the labeled data into 60% as training, 30% as test and 10% as development. Table 1 shows the resulting datasets with class-wise distributions. Data preprocessing was performed by following the same steps used to train the word2vec model (Subsection 2.5). In all the experiments, the classification task consists of two classes: *relevant* and *non-relevant*.

3.2 Model Settings and Baselines

In order to demonstrate the effectiveness of our joint learning approach, we performed a series of experiments. To understand the contribution of different network components, we performed an ablation study showing how the model performs as a semi-supervised model alone and as a domain adaptation model alone, and then we compare them with the combined model that incorporates all the components.

3.2.1 Settings for Semi-supervised Learning

As a baseline for the semi-supervised experiments, we used the self-training approach (Scudder, 1965). For this purpose, we first trained a supervised model using the CNN architecture (i.e., shared components followed by the supervised part in Figure 1). The trained model was then used to automatically label the unlabeled data. Instances with a classifier confidence score ≥ 0.75 were then used to retrain a new model.

Next, we run experiments using our graph-based semi-supervised approach (i.e., shared components followed by the supervised and semi-supervised parts in Figure 1), which exploits unlabeled data. For reducing the computational cost, we randomly selected $50K$ unlabeled instances from the same domain. For our semi-supervised setting, one of the main goals was to understand how much labeled data is sufficient to obtain a

[4]https://dev.twitter.com/streaming/overview
[5]http://crowdflower.com

reasonable result. Therefore, we experimented our system by incrementally adding batches of instances, such as 100, 500, 2000, 5000, and all instances from the training set. Such an understanding can help us design the model at the onset of a crisis event with sufficient amount of labeled data. To demonstrate that the semi-supervised approach outperforms the supervised baseline, we run supervised experiments using the same number of labeled instances. In the supervised setting, only $\mathbf{z}_c$ activations in Figure 1 are used for classification.

3.2.2 Settings for Domain Adaptation

To set a baseline for the domain adaptation experiments, we train a CNN model (i.e., shared components followed by the supervised part in Figure 1) on one event (source) and test it on another event (target). We call this as *transfer baseline*.

To assess the performance of our domain adaptation technique alone, we exclude the semi-supervised component from the network. We train and evaluate models with this network configuration using different source and target domains.

Finally, we integrate all the components of the network as shown in Figure 1 and run domain adaptation experiments using different source and target domains. In all our domain adaptation experiments, we only use unlabeled instances from the target domain. In domain adaption literature, this is known as *unsupervised* adaptation.

3.2.3 Training Settings

We use 100, 150, and 200 filters each having the window size of 2, 3, and 4, respectively, and pooling length of 2, 3, and 4, respectively. We do not tune these hyperparameters in any experimental setting since the goal was to have an end-to-end comparison with the same hyperparameter setting and understand whether our approach can outperform the baselines or not. Furthermore, we do not filter out any vocabulary item in any settings.

As mentioned before in Subsection 2.4, we used AdaDelta (Zeiler, 2012) to update the model parameters in each SGD step. The learning rate was set to 0.1 when optimizing on the classification loss and to 0.001 when optimizing on the semi-supervised loss. The learning rate for domain adversarial training was set to 1.0. The maximum number of epochs was set to 200, and dropout rate of 0.02 was used to avoid overfitting (Srivastava et al., 2014). We used validation-based *early stopping* using the F-measure with a patience of 25,

Experiments	AUC	P	R	F1
NEPAL EARTHQUAKE				
Supervised	61.22	62.42	62.31	60.89
Semi-supervised (Self-training)	61.15	61.53	61.53	61.26
Semi-supervised (Graph-based)	64.81	64.58	64.63	65.11
QUEENSLAND FLOODS				
Supervised	80.14	80.08	80.16	80.16
Semi-supervised (Self-training)	81.04	80.78	80.84	81.08
Semi-supervised (Graph-based)	92.20	92.60	94.49	93.54

Table 2: Results using supervised, self-training, and graph-based semi-supervised approaches in terms of Weighted average AUC, precision (P), recall (R) and F-measure (F1).

i.e., we stop training if the score does not increase for 25 consecutive epochs.

3.2.4 Evaluation Metrics

To measure the performance of the trained models using different approaches described above, we use weighted average precision, recall, F-measure, and Area Under ROC-Curve (AUC), which are standard evaluation measures in the NLP and machine learning communities. The rationale behind choosing the weighted metric is that it takes into account the class imbalance problem.

4 Results and Discussion

In this section, we present the experimental results and discuss our main findings.

4.1 Semi-supervised Learning

In Table 2, we present the results obtained from the supervised, self-training based semi-supervised, and our graph-based semi-supervised experiments for the both datasets. It can be clearly observed that the graph-based semi-supervised approach outperforms the two baselines – supervised and self-training based semi-supervised. Specifically, the graph-based approach shows 4% to 13% absolute improvements in terms of F1 scores for the Nepal and Queensland datasets, respectively.

To determine how the semi-supervised approach performs in the early hours of an event when only fewer labeled instances are available, we mimic a batch-wise (not to be confused with minibatch in SGD) learning setting. In Table 3, we present the results using different batch sizes – 100, 500, 1,000, 2,000, and all labels.

From the results, we observe that models' performance improve as we include more labeled data

Exp.	100	500	1000	2000	All L
		NEPAL EARTHQUAKE			
L	43.63	52.89	56.37	60.11	60.89
L+50kU	52.32	59.95	61.89	64.05	65.11
		QUEENSLAND FLOOD			
L	48.97	76.62	80.62	79.16	80.16
L+∼21kU	75.08	85.54	89.08	91.54	93.54

Table 3: Weighted average F-measure for the graph-based semi-supervised settings using different batch sizes. *L* refers to labeled data, *U* refers to unlabeled data, *All L* refers to all labeled instances for that particular dataset.

— from 43.63 to 60.89 for NEQ and from 48.97 to 80.16 for QFL in the case of labeled only (***L***). When we compare supervised vs. semi-supervised (***L*** vs. ***L+U***), we observe significant improvements in F1 scores for the semi-supervised model for all batches over the two datasets. As we include unlabeled instances with labeled instances from the same event, performance significantly improves in each experimental setting giving 5% to 26% absolute improvements over the supervised models. These improvements demonstrate the effectiveness of our approach. We also notice that our semi-supervised approach can perform above 90% depending on the event. Specifically, major improvements are observed from batch size 100 to 1,000, however, after that the performance improvements are comparatively minor. The results obtained using batch sizes 500 and 1,000 are reasonably in the acceptable range when labeled and unlabeled instances are combined (i.e., *L+50kU* for Nepal and *L+∼21kU* for Queensland), which is also a reasonable number of training examples to obtain at the onset of an event.

4.2 Domain Adaptation

In Table 4, we present domain adaptation results. The first block shows event-specific (i.e., train and test on the same event) results for the supervised CNN model. These results set the upper bound for our domain adaptation methods. The *transfer* baselines are shown in the next block, where we train a CNN model in one domain and test it on a different domain. Then, the third block shows the results for the domain adversarial approach without the semi-supervised loss. These results show the importance of domain adversarial component. After that, the fourth block presents the performance of the model trained with graph embedding without domain adaptation to show the importance of semi-supervised learning. The final block present the results for the complete model that includes all the loss components.

Source	Target	AUC	P	R	F1
	IN-DOMAIN SUPERVISED MODEL				
Nepal	**Nepal**	61.22	62.42	62.31	60.89
Queensland	**Queensland**	80.14	80.08	80.16	80.16
	TRANSFER BASELINES				
Nepal	**Queensland**	58.99	59.62	60.03	59.10
Queensland	**Nepal**	54.86	56.00	56.21	53.63
	DOMAIN ADVERSARIAL				
Nepal	**Queensland**	60.15	60.62	60.71	60.94
Queensland	**Nepal**	57.63	58.05	58.05	57.79
	GRAPH EMBEDDING WITHOUT DOMAIN ADVERSARIAL				
Nepal	**Queensland**	60.38	60.86	60.22	60.54
Queensland	**Nepal**	54.60	54.58	55.00	54.79
	GRAPH EMBEDDING WITH DOMAIN ADVERSARIAL				
Nepal	**Queensland**	66.49	67.48	65.90	65.92
Queensland	**Nepal**	58.81	58.63	59.00	59.05

Table 4: Domain adaptation experimental results. Weighted average AUC, precision (P), recall (R) and F-measure (F1).

The results with domain adversarial training show improvements across both events – from 1.8% to 4.1% absolute gains in F1. These results attest that adversarial training is an effective approach to induce domain invariant features in the internal representation as shown previously by Ganin et al. (2016).

Finally, when we do both semi-supervised learning and unsupervised domain adaptation, we get further improvements in F1 scores ranging from 5% to 7% absolute gains. From these improvements, we can conclude that domain adaptation with adversarial training along with graph-based semi-supervised learning is an effective method to leverage unlabeled and labeled data from a different domain.

Note that for our domain adaptation methods, we only use unlabeled data from the target domain. Hence, we foresee future improvements of this approach by utilizing a small amount of target domain labeled data.

5 Related Work

Two lines of research are directly related to our work: (*i*) semi-supervised learning and (*ii*) domain adaptation. Several models have been proposed for semi-supervised learning. The earliest approach is self-training (Scudder, 1965), in which a trained model is first used to label unlabeled data instances followed by the model retraining with the most confident predicted labeled instances. The co-training (Mitchell, 1999) approach assumes that features can be split into two sets and each subset is then used to train a classifier with an assumption that the two sets are conditionally independent. Then each classifier classifies the unlabeled data, and then most confident data instances are used to re-train the other classifier, this process repeats multiple times.

In the graph-based semi-supervised approach, nodes in a graph represent labeled and unlabeled instances and edge weights represent the similarity between them. The structural information encoded in the graph is then used to regularize a model (Zhu, 2005). There are two paradigms in semi-supervised learning: 1) inductive – learning a function with which predictions can be made on unobserved instances, 2) transductive – no explicit function is learned and predictions can only be made on observed instances. As mentioned before, inductive semi-supervised learning is preferable over the transductive approach since it avoids building the graph each time it needs to infer the labels for the unlabeled instances.

In our work, we use a graph-based inductive deep learning approach proposed by Yang et al. (2016) to learn features in a deep learning model by predicting contextual (i.e., neighboring) nodes in the graph. However, our approach is different from Yang et al. (2016) in several ways. First, we construct the graph by computing the distance between tweets based on word embeddings. Second, instead of using count-based features, we use a convolutional neural network (CNN) to compose high-level features from the distributed representation of the words in a tweet. Finally, for context prediction, instead of performing a random walk, we select nodes based on their similarity in the graph. Similar similarity-based graph has shown impressive results in learning sentence representations (Saha et al., 2017).

In the literature, the proposed approaches for domain adaptation include supervised, semi-supervised and unsupervised. It also varies from linear kernelized approach (Blitzer et al., 2006) to non-linear deep neural network techniques (Glorot et al., 2011; Ganin et al., 2016). One direction of research is to focus on feature space distribution matching by reweighting the samples from the source domain (Gong et al., 2013) to map source into target. The overall idea is to learn a good feature representation that is invariant across domains. In the deep learning paradigm, Glorot et al. (Glorot et al., 2011) used Stacked Denoising Auto-Encoders (SDAs) for domain adaptation. SDAs learn a robust feature representation, which is artificially corrupted with small Gaussian noise. *Adversarial training* of neural networks has shown big impact recently, especially in areas such as computer vision, where generative unsupervised models have proved capable of synthesizing new images (Goodfellow et al., 2014; Radford et al., 2015; Makhzani et al., 2015). Ganin et al. (2016) proposed domain adversarial neural networks (DANN) to learn discriminative but at the same time domain-invariant representations, with domain adaptation as a target. We extend this work by combining with semi-supervised graph embedding for unsupervised domain adaptation.

In a recent work, Kipf and Welling (2016) present CNN applied directly on graph-structured datasets - citation networks and on a knowledge graph dataset. Their study demonstrate that graph convolution network for semi-supervised classification performs better compared to other graph based approaches.

6 Conclusions

In this paper, we presented a deep learning framework that performs domain adaptation with adversarial training and graph-based semi-supervised learning to leverage labeled and unlabeled data from related events. We use a convolutional neural network to compose high-level representation from the input, which is then passed to three components that perform supervised training, semi-supervised learning and domain adversarial training. For domain adaptation, we considered a scenario, where we have only unlabeled data in the target event. Our evaluation on two crisis-related tweet datasets demonstrates that by combining domain adversarial training with semi-supervised learning, our model gives significant improvements over their respective baselines. We have also presented results of batch-wise incremental training of the graph-based semi-supervised approach and show approximation regarding the number of labeled examples required to get an acceptable performance at the onset of an event.

References

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* abs/1701.07875. http://arxiv.org/abs/1701.07875.

Yoshua Bengio and Xavier Glorot. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the 13th Intl. Conference on Artificial Intelligence and Statistics*. Sardinia, Italy, AISTATS '10, pages 249–256.

Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9):509–517.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*. ACL, pages 120–128.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of MLR* 17(59):1–35.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

Boqing Gong, Kristen Grauman, and Fei Sha. 2013. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML (1)*. pages 222–230.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2672–2680.

Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2015. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)* 47(4):67.

Muhammad Imran, Carlos Castillo, Ji Lucas, Patrick Meier, and Sarah Vieweg. 2014. AIDR: Artificial intelligence for disaster response. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, pages 159–162.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* .

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proc. of ICML*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. 2015. Adversarial autoencoders. *CoRR* abs/1511.05644. http://arxiv.org/abs/1511.05644.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled generative adversarial networks. *CoRR* abs/1611.02163. http://arxiv.org/abs/1611.02163.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*. Available as arXiv preprint arXiv:1301.3781.

Tom Mitchell. 1999. The role of unlabeled data in supervised learning. In *Proceedings of the sixth international colloquium on cognitive science*. Citeseer, pages 2–11.

Marius Muja and David G Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(11):2227–2240.

Dat Nguyen, Kamela Ali Al Mannai, Shafiq Joty, Hassan Sajjad, Muhammad Imran, and Prasenjit Mitra. 2017. Robust classification of crisis-related data on social networks using convolutional neural networks. In *International AAAI Conference on Web and Social Media*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Robert Power, Bella Robinson, and David Ratcliffe. 2013. Finding fires with twitter. In *Proceedings of the Australasian Language Technology Association Workshop 2013 (ALTA 2013)*. pages 80–89.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR* abs/1511.06434. http://arxiv.org/abs/1511.06434.

Tanay Saha, Shafiq Joty, Naeemul Hassan, and Mohammad Hasan. 2017. Regularized and retrofitted models for learning sentence representation with context. In *CIKM*. ACM, Singapore, pages 547–556.

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory* 11(3):363–371.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Michael Steinbach, George Karypis, Vipin Kumar, et al. 2000. A comparison of document clustering techniques. In *KDD workshop on text mining*. Boston, volume 400, pages 525–526.

István Varga, Motoki Sano, Kentaro Torisawa, Chikara Hashimoto, Kiyonori Ohtake, Takao Kawai, Jong-Hoon Oh, and Stijn De Saeger. 2013. Aid is out there: Looking for help from tweets during a large scale disaster. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1619–1629. http://www.aclweb.org/anthology/P13-1159.

Sarah Vieweg, Carlos Castillo, and Muhammad Imran. 2014. Integrating social media communications into the rapid assessment of sudden onset disasters. In *International Conference on Social Informatics*. Springer, pages 444–461.

Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Xiaojin Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

TDNN: A Two-stage Deep Neural Network for Prompt-independent Automated Essay Scoring

Cancan Jin[1] **Ben He**[1,3] **Kai Hui**[2] **Le Sun**[3,4]
[1]School of Computer & Control Engineering,
University of Chinese Academy of Sciences, Beijing, China
[2] SAP SE, Berlin, Germany
[3] Institute of Software, Chinese Academy of Sciences, Beijing, China
[4] Beijing Advanced Innovation Center for Language Resources, Beijing, China
`jincancan15@mails.ucas.ac.cn`, `benhe@ucas.ac.cn`
`kai.hui@sap.com`, `sunle@iscas.ac.cn`

Abstract

Existing automated essay scoring (AES) models rely on rated essays for the target prompt as training data. Despite their successes in prompt-dependent AES, how to effectively predict essay ratings under a prompt-independent setting remains a challenge, where the rated essays for the target prompt are not available. To close this gap, a two-stage deep neural network (TDNN) is proposed. In particular, in the first stage, using the rated essays for non-target prompts as the training data, a shallow model is learned to select essays with an extreme quality for the target prompt, serving as pseudo training data; in the second stage, an end-to-end hybrid deep model is proposed to learn a prompt-dependent rating model consuming the pseudo training data from the first step. Evaluation of the proposed TDNN on the standard ASAP dataset demonstrates a promising improvement for the prompt-independent AES task.

1 Introduction

Automated essay scoring (AES) utilizes natural language processing and machine learning techniques to automatically rate essays written for a target prompt (Dikli, 2006). Currently, the AES systems have been widely used in large-scale English writing tests, e.g. Graduate Record Examination (GRE), to reduce the human efforts in the writing assessments (Attali and Burstein, 2006).

Existing AES approaches are prompt-dependent, where, given a target prompt, rated essays for this particular prompt are required for training (Dikli, 2006; Williamson, 2009; Foltz et al., 1999). While the established models are effective (Chen and He, 2013; Taghipour and Ng, 2016; Alikaniotis et al., 2016; Cummins et al., 2016; Dong et al., 2017), we argue that the models for prompt-independent AES are also desirable to allow for better feasibility and flexibility of AES systems especially when the rated essays for a target prompt are difficult to obtain or even unaccessible. For example, in a writing test within a small class, students are asked to write essays for a target prompt without any rated examples, where the prompt-dependent methods are unlikely to provide effective AES due to the lack of training data. Prompt-independent AES, however, has drawn little attention in the literature, where there only exists unrated essays written for the target prompt, as well as the rated essays for several non-target prompts.

We argue that it is not straightforward, if possible, to apply the established prompt-dependent AES methods for the mentioned prompt-independent scenario. On one hand, essays for different prompts may differ a lot in the uses of vocabulary, the structure, and the grammatic characteristics; on the other hand, however, established prompt-dependent AES models are designed to learn from these prompt-specific features, including the on/off-topic degree, the *tf-idf* weights of topical terms (Attali and Burstein, 2006; Dikli, 2006), and the n-gram features extracted from word semantic embeddings (Dong and Zhang, 2016; Alikaniotis et al., 2016). Consequently, the prompt-dependent models can hardly learn generalized rules from rated essays for non-target prompts, and are not suitable for the prompt-independent AES.

Being aware of this difficulty, to this end, a two-stage deep neural network, coined as *TDNN*, is proposed to tackle the prompt-independent AES problem. In particular, to mitigate the lack of the prompt-dependent labeled data, at the first stage, a shallow model is trained on a number of rated essays for several non-target prompts; given a target prompt and a set of essays to rate, the trained model is employed to generate pseudo training data by selecting essays with the extreme quality. At the second stage, a novel end-to-end hybrid deep neural network learns prompt-dependent features from these selected training data, by considering semantic, part-of-speech, and syntactic features.

The contributions in this paper are threefold: 1) a two-stage learning framework is proposed to bridge the gap between the target and non-target prompts, by only consuming rated essays for non-target prompts as training data; 2) a novel deep model is proposed to learn from pseudo labels by considering semantic, part-of-speech, and syntactic features; and most importantly, 3) to the best of our knowledge, the proposed TDNN is actually the first approach dedicated to addressing the prompt-independent AES. Evaluation on the standard ASAP dataset demonstrates the effectiveness of the proposed method.

The rest of this paper is organized as follows. In Section 2, we describe our novel TDNN model, including the two-stage framework and the proposed deep model. Following that, we describe the setup of our empirical study in Section 3, thereafter present the results and provide analyzes in Section 4. Section 5 recaps existing literature and put our work in context, before drawing final conclusions in Section 6.

2 Two-stage Deep Neural Network for AES

In this section, the proposed two-stage deep neural network (TDNN) for prompt-independent AES is described. To accurately rate an essay, on one hand, we need to consider its pertinence to the given prompt; on the other hand, the organization, the analyzes, as well as the uses of the vocabulary are all crucial for the assessment. Henceforth, both prompt-dependent and -independent factors should be considered, but the latter ones actually do not require prompt-dependent training data. Accordingly, in the proposed framework, a supervised ranking model is first trained to learn from prompt-independent data, hoping to roughly assess essays without considering the prompt; subsequently, given the test dataset, namely, a set of essays for a target prompt, a subset of essays are selected as positive and negative training data based on the prediction of the trained model from the first stage; ultimately, a novel deep model is proposed to learn both prompt-dependent and -independent factors on this selected subset. As indicated in Figure 1, the proposed framework includes two stages.

2.1 Overview

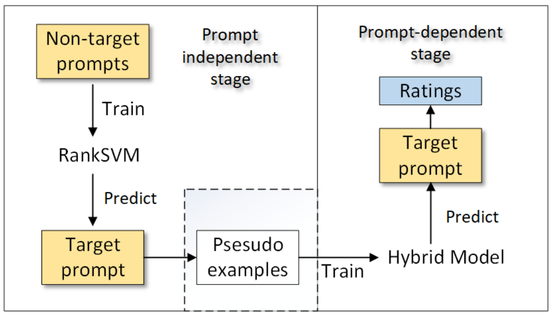


Figure 1: The architecture of the TDNN framework for prompt-independent AES.

Prompt-independent stage. Only the prompt-independent factors are considered to train a shallow model, aiming to recognize the essays with the extreme quality in the test dataset, where the rated essays for non-target prompts are used for training. Intuitively, one could recognize essays with the highest and the lowest scores correctly by solely examining their quality of writing, e.g., the number of typos, without even understanding them, and the prompt-independent features such as the number of grammatic and spelling errors should be sufficient to fulfill this screening procedure. Accordingly, a supervised model trained solely on prompt-independent features is employed to identify the essays with the highest and lowest scores in a given set of essays for the target prompt, which are used as the positive and negative training data in the follow-up prompt-dependent learning phase.

Prompt-dependent stage. Intuitively, most essays are with a quality in between the extremes, requiring a good understanding of their meaning to make an accurate assessment, e.g., whether the examples from the essay are convincing or whether the analyzes are insightful, making the consideration of prompt-dependent features crucial. To achieve that, a model is trained to learn from the comparison between essays with the highest and lowest scores for the target prompt according to the predictions from the first step. Akin to the settings in transductive transfer learning (Pan and

Yang, 2010), given essays for a particular prompt, quite a few confident essays at two extremes are selected and are used to train another model for a fine-grained content-based prompt-dependent assessment. To enable this, a powerful deep model is proposed to consider the content of the essays from different perspectives using semantic, part-of-speech (POS) and syntactic network. After being trained with the selected essays, the deep model is expected to memorize the properties of a good essay in response to the target prompt, thereafter accurately assessing all essays for it. In Section 2.2, building blocks for the selection of the training data and the proposed deep model are described in details.

2.2 Building Blocks

Select confident essays as training data. The identification of the extremes is relatively simple, where a RankSVM (Joachims, 2002) is trained on essays for different non-target prompts, avoiding the risks of over-fitting some particular prompts. A set of established prompt-independent features are employed, which are listed in Table 2. Given a prompt and a set of essays for evaluation, to begin with, the trained RankSVM is used to assign prediction scores to individual prompt-essay pairs, which are uniformly transformed into a 10-point scale. Thereafter, the essays with predicted scores in $[0, 4]$ and $[8, 10]$ are selected as negative and positive examples respectively, serving as the bad and good templates for training in the next stage. Intuitively, an essay with a score beyond eight out of a 10-point scale is considered good, while the one receiving less than or equal to four, is considered to be with a poor quality.

A hybrid deep model for fine-grained assessment. To enable a prompt-dependent assessment, a model is desired to comprehensively capture the ways in which a prompt is described or discussed in an essay. In this paper, semantic meaning, part-of-speech (POS), and the syntactic taggings of the token sequence from an essay are considered, grasping the quality of an essay for a target prompt. The model architecture is summarized in Figure 2. Intuitively, the model learns the semantic meaning of an essay by encoding it in terms of a sequence of word embeddings, denoted as $\overrightarrow{e}_{sem}$, hoping to understand what the essay is about; in addition, the part-of-speech information is encoded as a sequence of POS taggings, coined as $\overrightarrow{e}_{pos}$; ultimately, the structural connections between different components in an essay (e.g., terms or phrases) are further captured via syntactic network, leading to $\overrightarrow{e}_{synt}$, where the model learns the organization of the essay. Akin to (Li et al., 2015) and (Zhou and Xu, 2015), bi-LSTM is employed as a basic component to encode a sequence. Three features are separately captured using the stacked bi-LSTM layers as building blocks to encode different embeddings, whose outputs are subsequently concatenated and fed into several dense layers, generating the ultimate rating. In the following, the architecture of the model is described in details.

- *Semantic embedding.* Akin to the existing works (Alikaniotis et al., 2016; Taghipour and Ng, 2016), semantic word embeddings, namely, the pre-trained 50-dimension GloVe (Pennington et al., 2014), are employed. On top of the word embeddings, two bi-LSTM layers are stacked, namely, the essay layer is constructed on top of the sentence layer, ending up with the semantic representation of the whole essay, which is denoted as $\overrightarrow{e}_{sem}$ in Figure 2.

- *Part-Of-Speech (POS) embeddings* for individual terms are first generated by the Stanford Tagger (Toutanova et al., 2003), where 36 different POS tags present. Accordingly, individual words are embedded with 36-dimensional one-hot representation, and is transformed to a 50-dimensional vector through a lookup layer. After that, two bi-LSTM layers are stacked, leading to $\overrightarrow{e}_{pos}$. Take Figure 3 for example, given a sentence "Attention please, here is an example.", it is first converted into a POS sequence using the tagger, namely, VB, VBP, RB, VBZ, DT, NN; thereafter it is further mapped to vector space through one-hot embedding and a lookup layer.

- *Syntactic embedding* aims at encoding an essay in terms of the syntactic relationships among different syntactic components, by encoding an essay recursively. The Stanford Parser (Socher et al., 2013) is employed to label the syntactic structure of words and phrases in sentences, accounting for 59 different types in total. Similar to (Tai et al., 2015), we opt for three stacked bi-LSTM, aiming at encoding individual phrases, sentences, and ultimately the whole essay in sequence. In particular, according to the hierarchical structure from a parsing tree, the phrase-level bi-LSTM first encodes different phrases by consuming syntactic

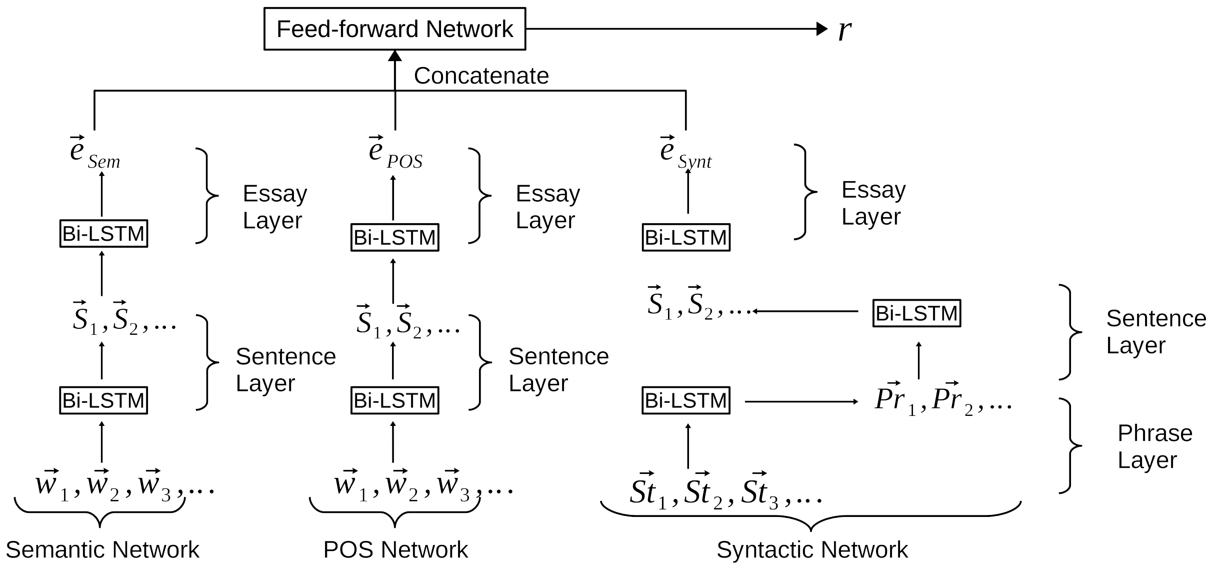


Figure 2: The model architecture of the proposed hybrid deep learning model.

embeddings ($\overrightarrow{St_i}$ in Figure 2) from a lookup table of individual syntactic units in the tree; thereafter, the encoded dense layers in individual sentences are further consumed by a sentence-level bi-LSTM, ending up with sentence-level syntactic representations, which are ultimately combined by the essay-level bi-LSTM, resulting in $\vec{e}_{synt}$. For example, the parsed tree for a sentence "Attention please, here is an example." is displayed in Figure 3. To start with, the sentence is parsed into ((NP VP)(NP VP NP)), and the dense embeddings are fetched from a lookup table for all tokens, namely, NP and VP; thereafter, the phrase-level bi-LSTM encodes (NP VP) and (NP VP NP) separately, which are further consumed by the sentence-level bi-LSTM. Afterward, essay-level bi-LSTM further combines the representations of different sentences into $\vec{e}_{synt}$.

```
(ROOT
  (S
    (S
      (NP (VB Attention))
      (VP (VBP please)))
    (, ,)
    (NP (RB here))
    (VP (VBZ is)
      (NP (DT an) (NN example)))
    (. .)))
```

Figure 3: An example of the context-free phrase structure grammar tree.

- *Combination.* A feed-forward network linearly transforms the concatenated representations of an essay from the mentioned three perspectives into a scalar, which is further normalized into $[0, 1]$ with a sigmoid function.

2.3 Objective and Training

Objective. Mean square error (MSE) is optimized, which is widely used as a loss function in regression tasks. Given N pairs of a target prompt p_i and an essay e_i, MSE measures the average value of square error between the normalized gold standard rating $r^*(p_i, e_i)$ and the predicted rating $r(p_i, e_i)$ assigned by the AES model, as summarized in Equation 1.

$$\frac{1}{N}\sum_{i=1}^{N}\left(r(p_i, e_i) - r^*(p_i, e_i)\right)^2 \tag{1}$$

Optimization. Adam (Kingma and Ba, 2014) is employed to minimize the loss over the training data. The initial learning rate η is set to 0.01 and the gradient is clipped between $[-10, 10]$ during training. In addition, dropout (Srivastava et al., 2014) is introduced for regularization with a dropout rate of 0.5, and 64 samples are used in each batch with batch normalization (Ioffe and Szegedy, 2015). 30% of the training data are reserved for validation. In addition, early stopping (Yao et al., 2007) is employed according to the validation loss, namely, the training is terminated if no decrease of the loss is observed for ten consecutive epochs. Once training is finished,

Prompt	#Essays	Avg Length	Score Range
1	1783	350	2-12
2	1800	350	1-6
3	1726	150	0-3
4	1772	150	0-3
5	1805	150	0-4
6	1800	150	0-4
7	1569	250	0-30
8	723	650	0-60

Table 1: Statistics for the ASAP dataset.

akin to (Dong et al., 2017), the model with the best quadratic weighted kappa on the validation set is selected.

3 Experimental Setup

Dataset. The Automated Student Assessment Prize (ASAP) dataset has been widely used for AES (Alikaniotis et al., 2016; Chen and He, 2013; Dong et al., 2017), and is also employed as the prime evaluation instrument herein. In total, ASAP consists of eight sets of essays, each of which associates to one prompt, and is originally written by students between Grade 7 and Grade 10. As summarized in Table 1, essays from different sets differ in their rating criteria, length, as well as the rating distribution[1].

Cross-validation. To fully employ the rated data, a prompt-wise eight-fold cross validation on the ASAP is used for evaluation. In each fold, essays corresponding to a prompt is reserved for testing, and the remaining essays are used as training data.

Evaluation metric. The model outputs are first uniformly re-scaled into $[0, 10]$, mirroring the range of ratings in practice. Thereafter, akin to (Yannakoudakis et al., 2011; Chen and He, 2013; Alikaniotis et al., 2016), we report our results primarily based on the quadratic weighted Kappa (QWK), examining the agreement between the predicted ratings and the ground truth. Pearson correlation coefficient (PCC) and Spearman rank-order correlation coefficient (SCC) are also reported. The correlations obtained from individual folds, as well as the average over all eight folds, are reported as the ultimate results.

Competing models. Since the prompt-independent AES is of interests in this work, the existing AES models are adapted for prompt-independent rating prediction, serving as baselines. This is due to the facts that the prompt-dependent and -independent models differ a lot in terms of problem settings and model designs, especially in their requirements for the training data, where the latter ones release the prompt-dependent requirements and thereby are accessible to more data.

No.	Feature
1	Mean & variance of word length in characters
2	Mean & variance of sentence length in words
3	Essay length in characters and words
4	Number of prepositions and commas
5	Number of unique words in an essay
6	Mean number of clauses per sentence
7	Mean length of clauses
8	Maximum number of clauses of a sentence in an essay
9	Number of spelling errors
10	Average depth of the parser tree of each sentence in an essay
11	Average depth of each leaf node in the parser tree of each sentence

Table 2: Handcrafted features used in learning the prompt-independent RankSVM.

- **RankSVM**, using handcrafted features for AES (Yannakoudakis et al., 2011; Chen et al., 2014), is trained on a set of pre-defined prompt-independent features as listed in Table 2, where the features are standardized beforehand to remove the mean and variance. The RankSVM is also used for the prompt-independent stage in our proposed TDNN model. In particular, the linear kernel RankSVM[2] is employed, where C is set to 5 according to our pilot experiments.

- **2L-LSTM.** Two-layer bi-LSTM with GloVe for AES (Alikaniotis et al., 2016) is employed as another baseline. Regularized word embeddings are dropped to avoid over-fitting the prompt-specific features.

- **CNN-LSTM.** This model (Taghipour and Ng, 2016) employs a convolutional (CNN) layer over one-hot representations of words, followed by an LSTM layer to encode word sequences in a given essay. A linear layer with sigmoid activation function is then employed to predict the essay rating.

- **CNN-LSTM-ATT.** This model (Dong et al., 2017) employs a CNN layer to encode word sequences into sentences, followed by an LSTM layer to generate the essay representation. An attention mechanism is added to model the influence of individual sentences on the final essay representation.

[1]Details of this dataset can be found at `https://www.kaggle.com/c/asap-aes`.

[2]`http://svmlight.joachims.org/`

For the proposed TDNN model, as introduced in Section 2.2, different variants of TDNN are examined by using one or multiple components out of the semantic, POS and the syntactic networks. The combinations being considered are listed in the following. In particular, the dimensions of POS tags and syntactic network are fixed to 50, whereas the sizes of the hidden units in LSTM, as well as the output units of the linear layers are tuned by grid search.

- **TDNN(Sem)** only includes the semantic building block, which is similar to the two-layer LSTM neural network from (Alikaniotis et al., 2016) but without regularizing the word embeddings;
- **TDNN(Sem+POS)** employs the semantic and the POS building blocks;
- **TDNN(Sem+Synt)** uses the semantic and the syntactic network building blocks;
- **TDNN(POS+Synt)** includes the POS and the syntactic network building blocks;
- **TDNN(ALL)** employs all three building blocks.

The use of POS or syntactic network alone is not presented for brevity given the facts that they perform no better than TDNN(POS+Synt) in our pilot experiments. Source code of the TDNN model is publicly available to enable further comparison[3].

4 Results and Analyzes

In this section, the evaluation results for different competing methods are compared and analyzed in terms of their agreements with the manual ratings using three correlation metrics, namely, QWK, PCC and SCC, where the best results for each prompt is highlighted in bold in Table 3.

It can be seen that, for seven out of all eight prompts, the proposed TDNN variants outperform the baselines by a margin in terms of QWK, and the TDNN variant with semantic and syntactic features, namely, TDNN(Sem+Synt), consistently performs the best among different competing methods. More precisely, as indicated in the bottom right corner in Table 3, on average, TDNN(Sem+Synt) outperforms the baselines by at least 25.52% under QWK, by 10.28% under PCC, and by 15.66% under SCC, demonstrating that the proposed model not only correlates better with the manual ratings in terms of QWK, but also linearly (PCC) and monotonically (SCC) correlates better with the manual ratings. As for the four baselines, note that, the relatively underperformed deep models suffer from larger variances of performance under different prompts, e.g., for prompts two and eight, 2L-LSTM's QWK is lower than 0.3. This actually confirms our choice of RankSVM for the first stage in TDNN, since a more complicated model (like 2L-LSTM) may end up with learning prompt-dependent signals, making it unsuitable for the prompt-independent rating prediction. As a comparison, RankSVM performs more stable among different prompts.

As for the different TDNN variants, it turns out that the joint uses of syntactic network with semantic or POS features can lead to better performances. This indicates that, when learning the prompt-dependent signals, apart from the widely-used semantic features, POS features and the sentence structure taggings (syntactic network) are also essential in learning the structure and the arrangement of an essay in response to a particular prompt, thereby being able to improve the results. It is also worth mentioning, however, when using all three features, the TDNN actually performs worse than when only using (any) two features. One possible explanation is that the uses of all three features result in a more complicated model, which over-fits the training data.

In addition, recall that the prompt-independent RankSVM model from the first stage enables the proposed TDNN in learning prompt-dependent information without manual ratings for the target prompt. Therefore, one would like to understand how good the trained RankSVM is in feeding training data for the model in the second stage. In particular, the precision, recall and F-score (P/R/F) of the essays selected by RanknSVM, namely, the negative ones rated between $[0, 4]$, and the positive ones rated between $[8, 10]$, are displayed in Figure 4. It can be seen that the P/R/F scores of both positive and negative classes differ a lot among different prompts. Moreover, it turns out that the P/R/F scores do not necessarily correlate with the performance of the TDNN model. Take TDNN(Sem+Synt), the best TDNN variant, as an example: as indicated in Table 4, the performance and the P/R/F scores of the pseudo examples are only weakly correlated in most cases.

To gain a better understanding in how the quality of pseudo examples affects the performance of TDNN, the sanctity of the selected essays are examined. In Figure 5, the relative precision of

[3]https://github.com/ucasir/TDNN4AES

Eval. Metric	QWK	PCC	SCC	QWK	PCC	SCC	QWK	PCC	SCC
Method	Prompt 1			Prompt 2			Prompt 3		
RankSVM	.7371	.6915	.6726	.4666	.4956	.4993	.4637	.5584	.5357
2L-LSTM	.4687	.6570	.4213	.2788	.6202	.6337	.5018	.6410	.6197
CNN-LSTM	.4320	.6933	.5108	.3230	.6513	.6395	.5454	**.6844**	.6541
CNN-LSTM-ATT	.6256	.7430	.6612	.4348	.7200	.6724	.4219	.5927	.6327
TDNN(Sem)	.7292	.7366	.7190	.6220	.7138	.7372	.6038	.6613	.6714
TDNN(Sem+POS)	.7305	.7413	.7209	.6551	.7276	.7469	.6112	.6706	.6809
TDNN(Sem+Synt)	**.7688**	**.7759**	**.7318**	**.6859**	**.7292**	**.7593**	**.6281**	.6759	**.7028**
TDNN(POS+Synt)	.7663	.7700	.7310	.6808	.7225	.7581	.6219	.6803	.6984
TDNN(All)	.7310	.7584	.7300	.6596	.7210	.7496	.6146	.6772	.6943
Method	Prompt 4			Prompt 5			Prompt 6		
RankSVM	.5112	.6250	.6325	.6690	.7103	.6651	.5285	.5443	.5239
2L-LSTM	.5754	.6527	.6354	.5128	.7375	.7360	.4951	.6528	.6669
CNN-LSTM	.7065	.7564	.7346	.6594	.6722	.6536	.5810	.6460	.6447
CNN-LSTM-ATT	.4665	.7224	.7383	.5348	.6531	.6505	.5149	.6291	.6637
TDNN(Sem)	.7398	.7412	.6934	.6874	.7585	.7323	.6278	.6524	.7205
TDNN(Sem+POS)	.7450	.7601	.7119	.6943	.7716	.7341	.6540	.6780	.7239
TDNN(Sem+Synt)	**.7578**	**.7616**	**.7492**	**.7366**	**.7993**	**.7960**	**.6752**	**.6903**	**.7434**
TDNN(POS+Synt)	.7561	.7591	.7440	.7332	.7983	.7866	.6593	.6759	.7354
TDNN(All)	.7527	.7609	.7251	.7302	.7974	.7794	.6557	.6874	.7350
Method	Prompt 7			Prompt 8			Average		
RankSVM	.5858	.6436	.6429	.4075	.5889	.6087	.5462	.6072	.5976
2L-LSTM	**.6690**	**.7637**	**.7607**	.2486	.5137	.4979	.4687	.6548	.6214
CNN-LSTM	.6609	.6849	.6865	.3812	.4666	.3872	.5362	.6569	.6139
CNN-LSTM-ATT	.6002	.6314	.6223	.4468	.5358	.4536	.5057	.6535	.6368
TDNN(Sem)	.5482	.6957	.6902	.5003	.6083	.6545	.5875	.6779	.6795
TDNN(Sem+POS)	.6239	.7111	.7243	.5519	.6219	.6614	.6582	.7103	.7130
TDNN(Sem+Synt)	.6587	.7201	.7380	**.5741**	**.6324**	**.6713**	**.6856**	**.7244**	**.7365**
TDNN(POS+Synt)	.6464	.7172	.7349	.5631	.6281	.6698	.6784	.7189	.7322
TDNN(All)	.6396	.7114	.7300	.5622	.6267	.6631	.6682	.7176	.7258

Table 3: Correlations between AES and manual ratings for different competing methods are reported for individual prompts. The average results among different prompts are summarized in the bottom right. The best results are highlighted in bold for individual prompts.

Neg/Pos	Metric	QWK	PCC	SCC
$[0, 4]$	Precision	+0.5151	+0.4286	+0.4471
	Recall	- 0.2362	- 0.1363	- 0.3491
	F-score	+0.4135	+0.4062	+0.1703
$[8, 10]$	Precision	+0.3526	+0.3224	+0.3885
	Recall	+0.0063	- 0.0415	- 0.2112
	F-score	+0.8339	+0.6905	+0.4221

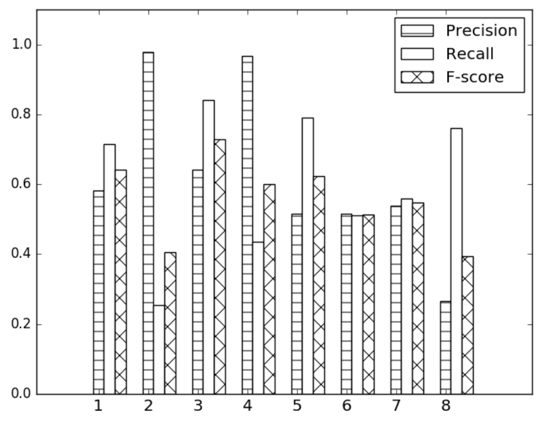
Table 4: Linear correlations between the performance of TDNN(Sem+Synt) and the precision, recall, and F-score of the selected pseudo examples.

Prpt	1	2	3	4	5	6	7	8
Neg	191	245	847	428	501	209	454	60
Pos	623	470	65	295	277	426	267	418

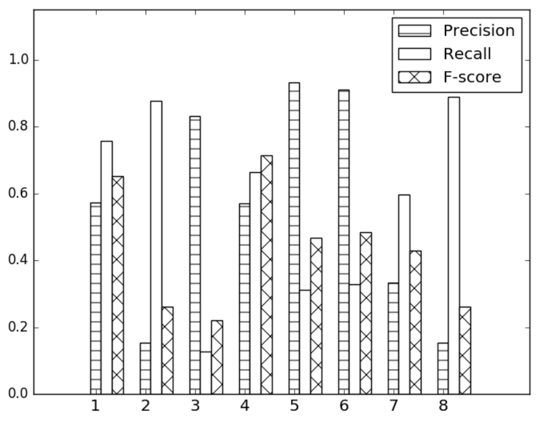
Table 5: The numbers of the selected positive and negative essays for each prompt.

the selected positive and negative training data by RankSVM are displayed for all eight prompts in terms of their concordance with the manual ratings, by computing the number of positive (negative) essays that are better (worse) than all negative (positive) essays. It can be seen that, such relative precision is at least 80% and mostly beyond 90% on different prompts, indicating that the overlap of the selected positive and negative essays are fairly small, guaranteeing that the deep model in the second stage at least learns from correct labels, which are crucial for the success of our TDNN model.

Beyond that, we further investigate the class balance of the selected training data from the first



(a) Negative



(b) Positive

Figure 4: The precision, recall and F-score of the pseudo negative or positive examples, which are rated within $[0, 4]$ or $[8, 10]$ by RankSVM.

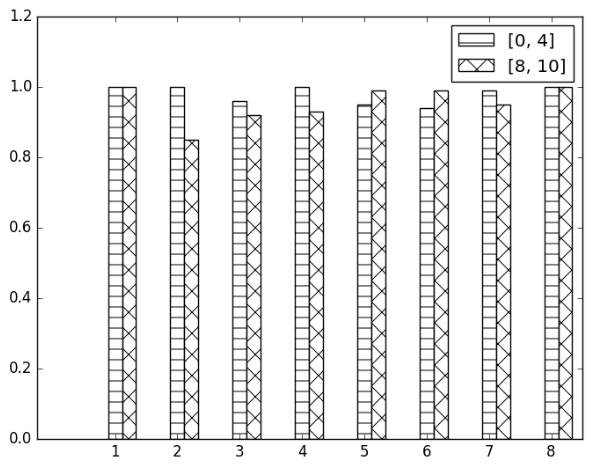


Figure 5: The sanctity of the selected positive and negative essays by RankSVM. The x-axis indicates different prompts and the y-axis is the relative precision.

stage, which could also influence the ultimate results. The number of selected positive and negative essays are reported in Table 5, where for prompts three and eight the training data suffers from serious imbalanced problem, which may explain their lower performance (namely, the two lowest QWKs among different prompts). On one hand, this is actually determined by real distribution of ratings for a particular prompt, e.g., how many essays are with an extreme quality for a given prompt in the target data. On the other hand, a fine-grained tuning of the RankSVM (e.g., tuning C_+ and C_- for positive and negative examples separately) may partially resolve the problem, which is left for the future work.

5 Related Work

Classical regression and classification algorithms are widely used for learning the rating model based on a variety of text features including lexical, syntactic, discourse and semantic features (Larkey, 1998; Rudner, 2002; Attali and Burstein, 2006; Mcnamara et al., 2015; Phandi et al., 2015). There are also approaches that see AES as a preference ranking problem by applying learning to ranking algorithms to learn the rating model. Results show improvement of learning to rank approaches over classical regression and classification algorithms (Chen et al., 2014; Yannakoudakis et al., 2011). In addition, Chen & He propose to incorporate the evaluation metric into the loss function of listwise learning to rank for AES (Chen and He, 2013).

Recently, there have been efforts in developing AES approaches based on deep neural networks (DNN), for which feature engineering is not required. Taghipour & Ng explore a variety of neural network model architectures based on recurrent neural networks which can effectively encode the information required for essay scoring and learn the complex connections in the data through the non-linear neural layers (Taghipour and Ng, 2016). Alikaniotis et al. introduce a neural network model to learn the extent to which specific words contribute to the text's score, which is embedded in the word representations. Then a two-layer bi-directional Long-Short Term Memory networks (bi-LSTM) is used to learn the meaning of texts, and finally the essay score is predicted through a mutli-layer feed-forward network (Alikaniotis et al., 2016). Dong & Zhang employ a hierarchical convolutional neural network (CNN) model, with a lower layer representing sentence structure and an upper layer representing essay structure based on sentence representations, to learn features automatically (Dong and Zhang, 2016). This model is later improved by employing attention layers. Specifically, the model learns text representation with LSTMs which can model the coherence and co-reference among sequences of words and sentences, and uses attention pooling to capture more relevant words and sentences that contribute to the final quality of essays (Dong et al., 2017). Song et al. propose a deep model for identifying discourse modes in an essay (Song et al., 2017).

While the literature has shown satisfactory performance of prompt-dependent AES, how to achieve effective essay scoring in a prompt-independent setting remains to be explored. Chen & He studied the usefulness of prompt-independent text features and achieved a human-machine rating agreement slightly lower than the use of all text features (Chen and He, 2013) for prompt-dependent essay scoring prediction. A constrained multi-task pairwise preference learning approach was proposed in (Cummins et al., 2016) to combine essays from multiple prompts for training. However, as shown by (Dong and Zhang, 2016; Zesch et al., 2015; Phandi et al., 2015), straightforward applications of existing AES methods for prompt-independent AES lead to a poor performance.

6 Conclusions & Future Work

This study aims at addressing the prompt-independent automated essay scoring (AES), where no rated essay for the target prompt is available. As demonstrated in the experiments, two kinds of established prompt-dependent AES models, namely, RankSVM for AES (Yannakoudakis et al., 2011; Chen et al., 2014) and the deep models for AES (Alikaniotis et al., 2016; Taghipour and Ng, 2016; Dong et al., 2017), fail to provide satisfactory performances, justifying our arguments in Section 1 that the application of established prompt-dependent AES models on prompt-independent AES is not straightforward. Therefore, a two-stage TDNN learning framework was proposed to utilize the prompt-independent features to generate pseudo training data for the target prompt, on which a hybrid deep neural network model is proposed to learn a rating model consuming semantic, part-of-speech, and syntactic signals. Through the experiments on the ASAP dataset, the proposed TDNN model outperforms the baselines, and leads to promising improvement in the human-machine agreement.

Given that our approach in this paper is similar to the methods for transductive transfer learning (Pan and Yang, 2010), we argue that the proposed TDNN could be further improved by migrating the non-target training data to the target prompt (Busto and Gall, 2017). Further study of the uses of transfer learning algorithms on prompt-independent AES needs to be undertaken.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (61472391), and the Project of Beijing Advanced Innovation Center for Language Resources (451122512).

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *ACL (1)*. The Association for Computer Linguistics.

Y. Attali and J. Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment* 4(3).

Pau Panareda Busto and Juergen Gall. 2017. Open set domain adaptation. In *ICCV*. IEEE Computer Society, pages 754–763.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *EMNLP*. ACL, pages 1741–1752.

Hongbo Chen, Jungang Xu, and Ben He. 2014. Automated essay scoring by capturing relative writing quality. *Comput. J.* 57(9):1318–1330.

Ronan Cummins, Meng Zhang, and Ted Briscoe. 2016. Constrained multi-task learning for automated essay scoring. In *ACL (1)*. The Association for Computer Linguistics.

S. Dikli. 2006. An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment* 5(1).

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring - an empirical study. In *EMNLP*. The Association for Computational Linguistics, pages 1072–1077.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *CoNLL*. Association for Computational Linguistics, pages 153–162.

Peter W Foltz, Darrell Laham, and Thomas K Landauer. 1999. Automated essay scoring: Applications to educational technology. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications*. volume 1999, pages 939–944.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*. JMLR.org, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*. ACM, pages 133–142.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Leah S. Larkey. 1998. Automatic essay grading using text categorization techniques. In *SIGIR*. ACM, pages 90–95.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? In *EMNLP*. The Association for Computational Linguistics, pages 2304–2314.

Danielle S. Mcnamara, Scott A. Crossley, Rod D. Roscoe, Laura K. Allen, and Jianmin Dai. 2015. A hierarchical classification approach to automated essay scoring. *Assessing Writing* 23:35–59.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. ACL, pages 1532–1543.

Peter Phandi, Kian Ming Adam Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *EMNLP*. The Association for Computational Linguistics, pages 431–439.

L. M Rudner. 2002. Automated essay scoring using bayes' theorem. *National Council on Measurement in Education New Orleans La* 1(2):3–21.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *ACL (1)*. The Association for Computer Linguistics, pages 455–465.

Wei Song, Dong Wang, Ruiji Fu, Lizhen Liu, Ting Liu, and Guoping Hu. 2017. Discourse mode identification in essays. In *ACL (1)*. Association for Computational Linguistics, pages 112–122.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *EMNLP*. The Association for Computational Linguistics, pages 1882–1891.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL (1)*. The Association for Computer Linguistics, pages 1556–1566.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*. The Association for Computational Linguistics.

D.M. Williamson. 2009. A framework for implementing automated scoring. In *Annual Meeting of the American Educational Research Association and the National Council on Measurement in Education, San Diego, CA*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *ACL*. The Association for Computer Linguistics, pages 180–189.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation* 26(2):289–315.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *BEA@NAACL-HLT*. The Association for Computer Linguistics, pages 224–232.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL (1)*. The Association for Computer Linguistics, pages 1127–1137.

Unsupervised Discrete Sentence Representation Learning for Interpretable Neural Dialog Generation

Tiancheng Zhao, Kyusong Lee and Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
{tianchez, kyusongl, max+}@cs.cmu.edu

Abstract

The encoder-decoder dialog model is one of the most prominent methods used to build dialog systems in complex domains. Yet it is limited because it cannot output interpretable actions as in traditional systems, which hinders humans from understanding its generation process. We present an unsupervised discrete sentence representation learning method that can integrate with any existing encoder-decoder dialog models for interpretable response generation. Building upon variational autoencoders (VAEs), we present two novel models, DI-VAE and DI-VST that improve VAEs and can discover interpretable semantics via either auto encoding or context predicting. Our methods have been validated on real-world dialog datasets to discover semantic representations and enhance encoder-decoder models with interpretable generation.[1]

1 Introduction

Classic dialog systems rely on developing a meaning representation to represent the utterances from both the machine and human users (Larsson and Traum, 2000; Bohus et al., 2007). The dialog manager of a conventional dialog system outputs the system's next action in a semantic frame that usually contains hand-crafted dialog acts and slot values (Williams and Young, 2007). Then a natural language generation module is used to generate the system's output in natural language based on the given semantic frame. This approach suffers from generalization to more complex domains because it soon become intractable to manually design a frame representation that covers all of the fine-grained system actions. The recently developed neural dialog system is one of the most prominent frameworks for developing dialog agents in complex domains. The basic model is based on encoder-decoder networks (Cho et al., 2014) and can learn to generate system responses without the need for hand-crafted meaning representations and other annotations.

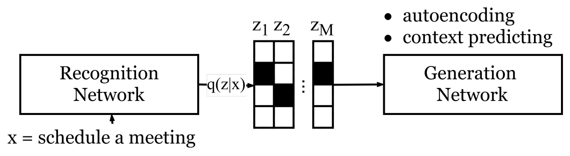


Figure 1: Our proposed models learn a set of discrete variables to represent sentences by either autoencoding or context prediction.

Although generative dialog models have advanced rapidly (Serban et al., 2016; Li et al., 2016; Zhao et al., 2017), they cannot provide interpretable system actions as in the conventional dialog systems. This inability limits the effectiveness of generative dialog models in several ways. First, having interpretable system actions enables human to understand the behavior of a dialog system and better interpret the system intentions. Also, modeling the high-level decision-making policy in dialogs enables useful generalization and data-efficient domain adaptation (Gašić et al., 2010). Therefore, the motivation of this paper is to develop an unsupervised neural recognition model that can discover interpretable meaning representations of utterances (denoted as *latent actions*) as a set of discrete latent variables from a large unlabelled corpus as shown in Figure 1. The discovered meaning representations will then be integrated with encoder decoder networks to achieve interpretable dialog generation while preserving

[1]Data and code are available at https://github.com/snakeztc/NeuralDialog-LAED.

all the merit of neural dialog systems.

We focus on learning discrete latent representations instead of dense continuous ones because discrete variables are easier to interpret (van den Oord et al., 2017) and can naturally correspond to categories in natural languages, e.g. topics, dialog acts and etc. Despite the difficulty of learning discrete latent variables in neural networks, the recently proposed Gumbel-Softmax offers a reliable way to back-propagate through discrete variables (Maddison et al., 2016; Jang et al., 2016). However, we found a simple combination of sentence variational autoencoders (VAEs) (Bowman et al., 2015) and Gumbel-Softmax fails to learn meaningful discrete representations. We then highlight the anti-information limitation of the evidence lowerbound objective (ELBO) in VAEs and improve it by proposing Discrete Information VAE (DI-VAE) that maximizes the mutual information between data and latent actions. We further enrich the learning signals beyond auto encoding by extending Skip Thought (Kiros et al., 2015) to Discrete Information Variational Skip Thought (DI-VST) that learns sentence-level distributional semantics. Finally, an integration mechanism is presented that combines the learned latent actions with encoder decoder models.

The proposed systems are tested on several real-world dialog datasets. Experiments show that the proposed methods significantly outperform the standard VAEs and can discover meaningful latent actions from these datasets. Also, experiments confirm the effectiveness of the proposed integration mechanism and show that the learned latent actions can control the sentence-level attributes of the generated responses and provide human-interpretable meaning representations.

2 Related Work

Our work is closely related to research in latent variable dialog models. The majority of models are based on Conditional Variational Autoencoders (CVAEs) (Serban et al., 2016; Cao and Clark, 2017) with continuous latent variables to better model the response distribution and encourage diverse responses. Zhao et al., (2017) further introduced dialog acts to guide the learning of the CVAEs. Discrete latent variables have also been used for task-oriented dialog systems (Wen et al., 2017), where the latent space is used to represent intention. The second line of related work is enriching the dialog context encoder with more fine-grained information than the dialog history. Li et al., (2016) captured speakers' characteristics by encoding background information and speaking style into the distributed embeddings. Xing et al., (2016) maintain topic encoding based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003) of the conversation to encourage the model to output more topic coherent responses.

The proposed method also relates to sentence representation learning using neural networks. Most work learns continuous distributed representations of sentences from various learning signals (Hill et al., 2016), e.g. the Skip Thought learns representations by predicting the previous and next sentences (Kiros et al., 2015). Another area of work focused on learning regularized continuous sentence representation, which enables sentence generation by sampling the latent space (Bowman et al., 2015; Kim et al., 2017). There is less work on discrete sentence representations due to the difficulty of passing gradients through discrete outputs. The recently developed Gumbel Softmax (Jang et al., 2016; Maddison et al., 2016) and vector quantization (van den Oord et al., 2017) enable us to train discrete variables. Notably, discrete variable models have been proposed to discover document topics (Miao et al., 2016) and semi-supervised sequence transaction (Zhou and Neubig, 2017)

Our work differs from these as follows: (1) we focus on learning interpretable variables; in prior research the semantics of latent variables are mostly ignored in the dialog generation setting. (2) we improve the learning objective for discrete VAEs and overcome the well-known posterior collapsing issue (Bowman et al., 2015; Chen et al., 2016). (3) we focus on unsupervised learning of salient features in dialog responses instead of hand-crafted features.

3 Proposed Methods

Our formulation contains three random variables: the dialog context $\mathbf{c}$, the response $\mathbf{x}$ and the latent action $\mathbf{z}$. The context often contains the discourse history in the format of a list of utterances. The response is an utterance that contains a list of word tokens. The latent action is a set of discrete variables that define high-level attributes of $\mathbf{x}$. Before introducing the proposed framework, we first identify two key properties that are essential in order for $\mathbf{z}$ to be *interpretable*:

1. $\mathbf{z}$ should capture salient sentence-level features about the response $\mathbf{x}$.
2. The meaning of latent symbols $\mathbf{z}$ should be independent of the context $\mathbf{c}$.

The first property is self-evident. The second can be explained: assume $\mathbf{z}$ contains a single discrete variable with K classes. Since the context $\mathbf{c}$ can be any dialog history, if the meaning of each class changes given a different context, then it is difficult to extract an intuitive interpretation by only looking at all responses with class $k \in [1, K]$. Therefore, the second property looks for latent actions that have *context-independent semantics* so that each assignment of $\mathbf{z}$ conveys the same meaning in all dialog contexts.

With the above definition of interpretable latent actions, we first introduce a recognition network $\mathcal{R} : q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$ and a generation network $\mathcal{G}$. The role of $\mathcal{R}$ is to map an sentence to the latent variable $\mathbf{z}$ and the generator $\mathcal{G}$ defines the learning signals that will be used to train $\mathbf{z}$'s representation. Notably, our recognition network $\mathcal{R}$ does not depend on the context $\mathbf{c}$ as has been the case in prior work (Serban et al., 2016). The motivation of this design is to encourage $\mathbf{z}$ to capture context-independent semantics, which are further elaborated in Section 3.4. With the $\mathbf{z}$ learned by $\mathcal{R}$ and $\mathcal{G}$, we then introduce an encoder decoder network $\mathcal{F} : p_{\mathcal{F}}(\mathbf{x}|\mathbf{z}, \mathbf{c})$ and and a policy network $\pi : p_{\pi}(\mathbf{z}|\mathbf{c})$. At test time, given a context $\mathbf{c}$, the policy network and encoder decoder will work together to generate the next response via $\tilde{\mathbf{x}} = p_{\mathcal{F}}(\mathbf{x}|\mathbf{z} \sim p_{\pi}(\mathbf{z}|\mathbf{c}), \mathbf{c})$. In short, $\mathcal{R}$, $\mathcal{G}$, $\mathcal{F}$ and π are the four components that comprise our proposed framework. The next section will first focus on developing $\mathcal{R}$ and $\mathcal{G}$ for learning interpretable $\mathbf{z}$ and then will move on to integrating $\mathcal{R}$ with $\mathcal{F}$ and π in Section 3.3.

3.1 Learning Sentence Representations from Auto-Encoding

Our baseline model is a sentence VAE with discrete latent space. We use an RNN as the recognition network to encode the response $\mathbf{x}$. Its last hidden state $h^{\mathcal{R}}_{|\mathbf{x}|}$ is used to represent $\mathbf{x}$. We define $\mathbf{z}$ to be a set of K-way categorical variables $\mathbf{z} = \{\mathbf{z}_1 ... \mathbf{z}_m ... \mathbf{z}_M\}$, where M is the number of variables. For each $\mathbf{z}_m$, its posterior distribution is defined as $q_{\mathcal{R}}(\mathbf{z}_m|\mathbf{x}) = \text{Softmax}(W_q h^{\mathcal{R}}_{|\mathbf{x}|} + b_q)$. During training, we use the Gumbel-Softmax trick to sample from this distribution and obtain low-variance gradients. To map the latent samples to the initial state of the decoder RNN, we define $\{e_1 ... e_m ... e_M\}$ where $e_m \in \mathbb{R}^{K \times D}$ and D is the generator cell size. Thus the initial state of the generator is: $h^{\mathcal{G}}_0 = \sum_{m=1}^{M} e_m(\mathbf{z}_m)$. Finally, the generator RNN is used to reconstruct the response given $h^{\mathcal{G}}_0$. VAEs is trained to maxmimize the evidence lowerbound objective (ELBO) (Kingma and Welling, 2013). For simplicity, later discussion drops the subscript m in $\mathbf{z}_m$ and assumes a single latent $\mathbf{z}$. Since each $\mathbf{z}_m$ is independent, we can easily extend the results below to multiple variables.

3.1.1 Anti-Information Limitation of ELBO

It is well-known that sentence VAEs are hard to train because of the posterior collapse issue. Many empirical solutions have been proposed: weakening the decoder, adding auxiliary loss etc. (Bowman et al., 2015; Chen et al., 2016; Zhao et al., 2017). We argue that the posterior collapse issue lies in ELBO and we offer a novel decomposition to understand its behavior. First, instead of writing ELBO for a single data point, we write it as an expectation over a dataset:

$$\begin{aligned}\mathcal{L}_{\text{VAE}} = \mathbb{E}_{\mathbf{x}}[&\mathbb{E}_{q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})}[\log p_{\mathcal{G}}(\mathbf{x}|\mathbf{z})] \\ &- \text{KL}(q_{\mathcal{R}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))]\end{aligned} \tag{1}$$

We can expand the KL term as Eq. 2 (derivations in Appendix A.1) and rewrite ELBO as:

$$\begin{aligned}\mathbb{E}_{\mathbf{x}}[\text{KL}(q_{\mathcal{R}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))] = \\ I(Z, X) + \text{KL}(q(\mathbf{z}) \| p(\mathbf{z}))\end{aligned} \tag{2}$$

$$\begin{aligned}\mathcal{L}_{\text{VAE}} = &\mathbb{E}_{q(\mathbf{z}|\mathbf{x})p(\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] \\ &- I(Z, X) - \text{KL}(q(\mathbf{z}) \| p(\mathbf{z}))\end{aligned} \tag{3}$$

where $q(\mathbf{z}) = \mathbb{E}_{\mathbf{x}}[q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})]$ and $I(Z, X)$ is the mutual information between Z and X. This expansion shows that the KL term in ELBO is trying to reduce the mutual information between latent variables and the input data, which explains why VAEs often ignore the latent variable, especially when equipped with powerful decoders.

3.1.2 VAE with Information Maximization and Batch Prior Regularization

A natural solution to correct the anti-information issue in Eq. 3 is to maximize both the data likelihood lowerbound and the mutual information between $\mathbf{z}$ and the input data:

$$\mathcal{L}_{\text{VAE}} + I(Z, X) = \mathbb{E}_{q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})p(\mathbf{x})}[\log p_{\mathcal{G}}(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z})\|p(\mathbf{z})) \tag{4}$$

Therefore, jointly optimizing ELBO and mutual information simply cancels out the information-discouraging term. Also, we can still sample from the prior distribution for generation because of $\text{KL}(q(\mathbf{z})\|p(\mathbf{z}))$. Eq. 4 is similar to the objectives used in adversarial autoencoders (Makhzani et al., 2015; Kim et al., 2017). Our derivation provides a theoretical justification to their superior performance. Notably, Eq. 4 arrives at the same loss function proposed in infoVAE (Zhao S et al., 2017). However, our derivation is different, offering a new way to understand ELBO behavior.

The remaining challenge is how to minimize $\text{KL}(q(\mathbf{z})\|p(\mathbf{z}))$, since $q(\mathbf{z})$ is an expectation over $q(\mathbf{z}|\mathbf{x})$. When $\mathbf{z}$ is continuous, prior work has used adversarial training (Makhzani et al., 2015; Kim et al., 2017) or Maximum Mean Discrepancy (MMD) (Zhao S et al., 2017) to regularize $q(\mathbf{z})$. It turns out that minimizing $\text{KL}(q(\mathbf{z})\|p(\mathbf{z}))$ for discrete $\mathbf{z}$ is much simpler than its continuous counterparts. Let $\mathbf{x}_n$ be a sample from a batch of N data points. Then we have:

$$q(\mathbf{z}) \approx \frac{1}{N}\sum_{n=1}^{N} q(\mathbf{z}|\mathbf{x}_n) = q'(\mathbf{z}) \tag{5}$$

where $q'(\mathbf{z})$ is a mixture of softmax from the posteriors $q(\mathbf{z}|\mathbf{x}_n)$ of each $\mathbf{x}_n$. We can approximate $\text{KL}(q(\mathbf{z})\|p(\mathbf{z}))$ by:

$$\text{KL}(q'(\mathbf{z})\|p(\mathbf{z})) = \sum_{k=1}^{K} q'(\mathbf{z}=k) \log \frac{q'(\mathbf{z}=k)}{p(\mathbf{z}=k)} \tag{6}$$

We refer to Eq. 6 as Batch Prior Regularization (BPR). When N approaches infinity, $q'(\mathbf{z})$ approaches the true marginal distribution of $q(\mathbf{z})$. In practice, we only need to use the data from each mini-batch assuming that the mini batches are randomized. Last, BPR is fundamentally different from multiplying a coefficient < 1 to anneal the KL term in VAE (Bowman et al., 2015). This is because BPR is a non-linear operation *log_sum_exp*. For later discussion, we denote our discrete infoVAE with BPR as DI-VAE.

3.2 Learning Sentence Representations from the Context

DI-VAE infers sentence representations by reconstruction of the input sentence. Past research in distributional semantics has suggested the meaning of language can be inferred from the adjacent context (Harris, 1954; Hill et al., 2016). The distributional hypothesis is especially applicable to dialog since the utterance meaning is highly contextual. For example, the dialog act is a well-known utterance feature and depends on dialog state (Austin, 1975; Stolcke et al., 2000). Thus, we introduce a second type of latent action based on sentence-level distributional semantics.

Skip thought (ST) is a powerful sentence representation that captures contextual information (Kiros et al., 2015). ST uses an RNN to encode a sentence, and then uses the resulting sentence representation to predict the previous and next sentences. Inspired by ST's robust performance across multiple tasks (Hill et al., 2016), we adapt our DI-VAE to Discrete Information Variational Skip Thought (DI-VST) to learn discrete latent actions that model distributional semantics of sentences. We use the same recognition network from DI-VAE to output $\mathbf{z}$'s posterior distribution $q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$. Given the samples from $q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$, two RNN generators are used to predict the previous sentence $\mathbf{x}_p$ and the next sentences $\mathbf{x}_n$. Finally, the learning objective is to maximize:

$$\mathcal{L}_{\text{DI-VST}} = \mathbb{E}_{q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})p(\mathbf{x}))}[\log(p^n_{\mathcal{G}}(\mathbf{x}_n|\mathbf{z})p^p_{\mathcal{G}}(\mathbf{x}_p|\mathbf{z}))] - \text{KL}(q(\mathbf{z})\|p(\mathbf{z})) \tag{7}$$

3.3 Integration with Encoder Decoders

We now describe how to integrate a given $q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$ with an encoder decoder and a policy network. Let the dialog context $\mathbf{c}$ be a sequence of utterances. Then a dialog context encoder network can encode the dialog context into a distributed representation $h^e = \mathcal{F}^e(\mathbf{c})$. The decoder $\mathcal{F}^d$ can generate the responses $\tilde{\mathbf{x}} = \mathcal{F}^d(h^e, \mathbf{z})$ using samples from $q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$. Meanwhile, we train π to predict the aggregated posterior $\mathbb{E}_{p(\mathbf{x}|\mathbf{c})}[q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})]$ from $\mathbf{c}$ via maximum likelihood training. This model is referred as Latent Action Encoder Decoder (LAED) with the following objective.

$$\mathcal{L}_{\text{LAED}}(\theta_{\mathcal{F}}, \theta_\pi) = \mathbb{E}_{q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})p(\mathbf{x},\mathbf{c})}[\log p_\pi(\mathbf{z}|\mathbf{c}) + \log p_{\mathcal{F}}(\mathbf{x}|\mathbf{z},\mathbf{c})] \tag{8}$$

Also simply augmenting the inputs of the decoders with latent action does not guarantee that the generated response exhibits the attributes of the give action. Thus we use the controllable text generation framework (Hu et al., 2017) by introducing $\mathcal{L}_{\text{Attr}}$, which reuses the same recognition network $q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$ as a fixed discriminator to penalize the decoder if its generated responses do not reflect the attributes in $\mathbf{z}$.

$$\mathcal{L}_{\text{Attr}}(\theta_{\mathcal{F}}) = \mathbb{E}_{q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})p(\mathbf{c},\mathbf{x})}[\log q_{\mathcal{R}}(\mathbf{z}|\mathcal{F}(\mathbf{c},\mathbf{z}))] \tag{9}$$

Since it is not possible to propagate gradients through the discrete outputs at $\mathcal{F}^d$ at each word step, we use a deterministic continuous relaxation (Hu et al., 2017) by replacing output of $\mathcal{F}^d$ with the probability of each word. Let $\mathbf{o}_t$ be the normalized probability at step $t \in [1, |\mathbf{x}|]$, the inputs to $q_{\mathcal{R}}$ at time t are then the sum of word embeddings weighted by $\mathbf{o}_t$, i.e. $h_t^{\mathcal{R}} = \text{RNN}(h_{t-1}^{\mathcal{R}}, \mathbf{E}\mathbf{o}_t)$ and $\mathbf{E}$ is the word embedding matrix. Finally this loss is combined with $\mathcal{L}_{\text{LAED}}$ and a hyperparameter λ to have Attribute Forcing LAED.

$$\mathcal{L}_{\text{attrLAED}} = \mathcal{L}_{\text{LAED}} + \lambda\mathcal{L}_{\text{Attr}} \tag{10}$$

3.4 Relationship with Conditional VAEs

It is not hard to see $\mathcal{L}_{\text{LAED}}$ is closely related to the objective of CVAEs for dialog generation (Serban et al., 2016; Zhao et al., 2017), which is:

$$\mathcal{L}_{\text{CVAE}} = \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z},\mathbf{c})] - \text{KL}(q(\mathbf{z}|\mathbf{x},\mathbf{c})\|p(\mathbf{z}|\mathbf{c})) \tag{11}$$

Despite their similarities, we highlight the key differences that prohibit CVAE from achieving interpretable dialog generation. First $\mathcal{L}_{\text{CVAE}}$ encourages $I(\mathbf{x},\mathbf{z}|\mathbf{c})$ (Agakov, 2005), which learns $\mathbf{z}$ that capture context-dependent semantics. More intuitively, $\mathbf{z}$ in CVAE is trained to generate $\mathbf{x}$ via $p(\mathbf{x}|\mathbf{z},\mathbf{c})$ so the meaning of learned $\mathbf{z}$ can only be interpreted along with its context $\mathbf{c}$. Therefore this violates our goal of learning context-independent semantics. Our methods learn $q_{\mathcal{R}}(\mathbf{z}|\mathbf{x})$ that only depends on $\mathbf{x}$ and trains $q_{\mathcal{R}}$ separately to ensure the semantics of $\mathbf{z}$ are interpretable standalone.

4 Experiments and Results

The proposed methods are evaluated on four datasets. The first corpus is Penn Treebank (PTB) (Marcus et al., 1993) used to evaluate sentence VAEs (Bowman et al., 2015). We used the version pre-processed by Mikolov (Mikolov et al., 2010). The second dataset is the Stanford Multi-Domain Dialog (SMD) dataset that contains 3,031 human-Woz, task-oriented dialogs collected from 3 different domains (navigation, weather and scheduling) (Eric and Manning, 2017). The other two datasets are chat-oriented data: Daily Dialog (DD) and Switchboard (SW) (Godfrey and Holliman, 1997), which are used to test whether our methods can generalize beyond task-oriented dialogs but also to to open-domain chatting. DD contains 13,118 multi-turn human-human dialogs annotated with dialog acts and emotions. (Li et al., 2017). SW has 2,400 human-human telephone conversations that are annotated with topics and dialog acts. SW is a more challenging dataset because it is transcribed from speech which contains complex spoken language phenomenon, e.g. hesitation, self-repair etc.

4.1 Comparing Discrete Sentence Representation Models

The first experiment used PTB and DD to evaluate the performance of the proposed methods in learning discrete sentence representations. We implemented DI-VAE and DI-VST using GRU-RNN (Chung et al., 2014) and trained them using Adam (Kingma and Ba, 2014). Besides the proposed methods, the following baselines are compared. **Unregularized models**: removing the $\text{KL}(q|p)$ term from DI-VAE and DI-VST leads to a simple discrete autoencoder (DAE) and discrete skip thought (DST) with stochastic discrete hidden units. **ELBO models**: the basic discrete sentence VAE (DVAE) or variational skip thought (DVST) that optimizes ELBO with regularization term $\text{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$. We found that standard training failed to learn informative latent actions for either DVAE or DVST because of the posterior collapse. Therefore, KL-annealing (Bowman et al., 2015) and bag-of-word loss (Zhao et al., 2017) are used to force these two models learn meaningful representations. We also include the results for VAE with continuous latent variables reported on the same PTB (Zhao et al., 2017). Additionally, we report the perplexity from a standard GRU-RNN language model (Zaremba et al., 2014).

The evaluation metrics include reconstruction perplexity (PPL), $\text{KL}(q(\mathbf{z})\|p(\mathbf{z}))$ and the mutual information between input data and latent variables $I(\mathbf{x}, \mathbf{z})$. Intuitively a good model should achieve low perplexity and KL distance, and simultaneously achieve high $I(\mathbf{x}, \mathbf{z})$. The discrete latent space for all models are M=20 and K=10. Mini-batch size is 30.

Dom	Model	PPL	KL$(q\|p)$	$I(\mathbf{x}, \mathbf{z})$
PTB	RNNLM	116.22	-	-
	VAE	73.49	15.94*	-
	DAE	66.49	2.20	0.349
	DVAE	70.84	0.315	0.286
	DI-VAE	**52.53**	**0.133**	**1.18**
DD	RNNLM	31.15	-	-
	DST	$\mathbf{x}_p$:28.23 $\mathbf{x}_n$:28.16	0.588	**1.359**
	DVST	$\mathbf{x}_p$:30.36 $\mathbf{x}_n$:30.71	**0.007**	0.081
	DI-VST	$\mathbf{x}_p$:**28.04** $\mathbf{x}_n$:**27.94**	0.088	1.028

Table 1: Results for various discrete sentence representations. The KL for VAE is KL$(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ instead of KL$(q(\mathbf{z})\|p(\mathbf{z}))$ (Zhao et al., 2017)

Table 1 shows that all models achieve better perplexity than an RNNLM, which shows they manage to learn meaningful $q(\mathbf{z}|\mathbf{x})$. First, for auto-encoding models, DI-VAE is able to achieve the best results in all metrics compared other methods. We found DAEs quickly learn to reconstruct the input but they are prone to overfitting during training, which leads to lower performance on the test data compared to DI-VAE. Also, since there is no regularization term in the latent space, $q(\mathbf{z})$ is very different from the $p(\mathbf{z})$ which prohibits us from generating sentences from the latent space. In fact, DI-VAE enjoys the same linear interpolation properties reported in (Bowman et al., 2015) (See Appendix A.2). As for DVAEs, it achieves zero $I(\mathbf{x}, \mathbf{z})$ in standard training and only manages to learn some information when training with KL-annealing and bag-of-word loss. On the other hand, our methods achieve robust performance without the need for additional processing. Similarly, the proposed DI-VST is able to achieve the lowest PPL and similar KL compared to the strongly regularized DVST. Interestingly, although DST is able to achieve the highest $I(\mathbf{x}, \mathbf{z})$, but PPL is not further improved. These results confirm the effectiveness of the proposed BPR in terms of regularizing $q(\mathbf{z})$ while learning meaningful posterior $q(\mathbf{z}|\mathbf{x})$.

In order to understand BPR's sensitivity to batch size N, a follow-up experiment varied the batch size from 2 to 60 (If N=1, DI-VAE is equivalent to DVAE). Figure 2 show that as N increases,

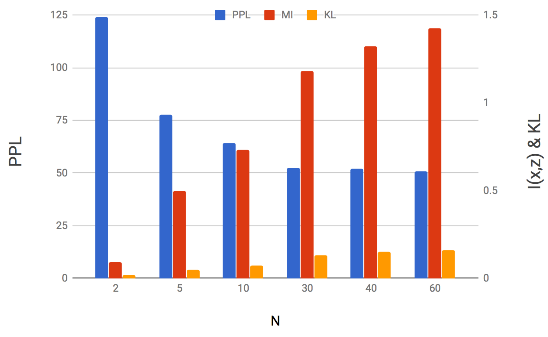


Figure 2: Perplexity and $I(\mathbf{x}, \mathbf{z})$ on PTB by varying batch size N. BPR works better for larger N.

perplexity, $I(\mathbf{x}, \mathbf{z})$ monotonically improves, while KL$(q\|p)$ only increases from 0 to 0.159. After $N > 30$, the performance plateaus. Therefore, using mini-batch is an efficient trade-off between $q(\mathbf{z})$ estimation and computation speed.

The last experiment in this section investigates the relation between representation learning and the dimension of the latent space. We set a fixed budget by restricting the maximum number of modes to be about 1000, i.e. $K^M \approx 1000$. We then vary the latent space size and report the same evaluation metrics. Table 2 shows that models with multiple small latent variables perform significantly better than those with large and few latent variables.

K, M	K^M	PPL	KL$(q\|p)$	$I(\mathbf{x}, \mathbf{z})$
1000, 1	1000	75.61	0.032	0.335
10, 3	1000	71.42	0.071	0.607
4, 5	1024	68.43	0.088	0.809

Table 2: DI-VAE on PTB with different latent dimensions under the same budget.

4.2 Interpreting Latent Actions

The next question is to interpret the meaning of the learned latent action symbols. To achieve this, the latent action of an utterance $\mathbf{x}_n$ is obtained from a greedy mapping: $a_n = \text{argmax}_k\, q_{\mathcal{R}}(\mathbf{z} = k|\mathbf{x}_n)$. We set M=3 and K=5, so that there are at most 125 different latent actions, and each $\mathbf{x}_n$ can now be represented by a_1-a_2-a_3, e.g. "How are you?" $\rightarrow$ 1-4-2. Assuming that we have access to manually clustered data according to certain classes (e.g. dialog acts), it is unfair to use classic cluster measures (Vinh et al., 2010) to evaluate the clusters from latent actions. This is because the uniform prior $p(\mathbf{z})$ evenly distribute the data to all possible latent actions, so that it is expected that frequent classes will be assigned to several latent actions. Thus we utilize the *homogeneity* metric (Rosenberg and Hirschberg, 2007) that measures if each latent action contains only members of a single class. We tested this on the SW and DD, which contain human annotated features and we report the latent actions' homogeneity w.r.t these features in Table 3. On DD, results show DI-VST works better than DI-VAE in terms of creating actions that are more coherent for emotion and dialog acts. The results are interesting on SW since DI-VST performs worse on dialog acts than DI-VAE. One reason is that the dialog acts in SW are more fine-grained (42 acts) than the ones in DD (5 acts) so that distinguishing utterances based on words in $\mathbf{x}$ is more important than the information in the neighbouring utterances.

	SW		**DD**	
	Act	Topic	Act	Emotion
DI-VAE	0.48	0.08	0.18	0.09
DI-VST	0.33	0.13	0.34	0.12

Table 3: Homogeneity results (bounded [0, 1]).

We then apply the proposed methods to SMD which has no manual annotation and contains task-oriented dialogs. Two experts are shown 5 randomly selected utterances from each latent action and are asked to give an action name that can describe as many of the utterances as possible. Then an Amazon Mechanical Turk study is conducted to evaluate whether other utterances from the same latent action match these titles. 5 workers see the action name and a different group of 5 utterances from that latent action. They are asked to select all utterances that belong to the given actions, which tests the homogeneity of the utterances falling in the same cluster. Negative samples are included to prevent random selection. Table 4 shows that both methods work well and DI-VST achieved better homogeneity than DI-VAE.

Since DI-VAE is trained to reconstruct its input and DI-VST is trained to model the context, they group utterances in different ways. For example, DI-VST would group "Can I get a restaurant", "I am looking for a restaurant" into one action where DI-VAE may denote two actions for them. Finally, Table 4.2 shows sample annotation results, which show cases of the different types of latent actions discovered by our models.

Model	Exp Agree	Worker κ	Match Rate
DI-VAE	85.6%	0.52	71.3%
DI-VST	93.3%	0.48	74.9%

Table 4: Human evaluation results on judging the homogeneity of latent actions in SMD.

Model	**Action**	**Sample utterance**
DI-VAE	scheduling	- sys: okay, scheduling a yoga activity with Tom for the 8th at 2pm. - sys: okay, scheduling a meeting for 6 pm on Tuesday with your boss to go over the quarterly report.
	requests	- usr: find out if it 's supposed to rain - usr: find nearest coffee shop
DI-VST	ask schedule info	- usr: when is my football activity and who is going with me? - usr: tell me when my dentist appointment is?
	requests	- usr: how about other coffee? - usr: 11 am please

Table 5: Example latent actions discovered in SMD using our methods.

4.3 Dialog Response Generation with Latent Actions

Finally we implement an LAED as follows. The encoder is a hierarchical recurrent encoder (Serban et al., 2016) with bi-directional GRU-RNNs as the utterance encoder and a second GRU-RNN as the discourse encoder. The discourse encoder output its last hidden state $h^e_{|\mathbf{x}|}$. The decoder is another GRU-RNN and its initial state of the decoder is obtained by $h^d_0 = h^e_{|\mathbf{x}|} + \sum_{m=1}^{M} e_m(\mathbf{z}_m)$, where $\mathbf{z}$ comes from the recognition network of the proposed methods. The policy network π is a 2-layer multi-layer perceptron (MLP) that models $p_\pi(\mathbf{z}|h^e_{|\mathbf{x}|})$. We use up to the previous 10 utterances as the dialog context and denote the LAED using DI-VAE latent actions as AE-ED and the one uses DI-VST as ST-ED.

First we need to confirm whether an LAED can generate responses that are consistent with the semantics of a given $\mathbf{z}$. To answer this, we use a pre-trained recognition network $\mathcal{R}$ to check if a generated response carries the attributes in the given action. We generate dialog responses on a test dataset via $\tilde{\mathbf{x}} = \mathcal{F}(\mathbf{z} \sim \pi(\mathbf{c}), \mathbf{c})$ with greedy RNN decoding. The generated responses are passed into the $\mathcal{R}$ and we measure *attribute accuracy* by counting $\tilde{\mathbf{x}}$ as correct if $\mathbf{z} = \text{argmax}_k\, q_{\mathcal{R}}(k|\tilde{\mathbf{x}})$. Table 4.3 shows our generated responses are highly consistent with the given latent actions. Also, latent actions from DI-VAE achieve higher attribute accuracy than the ones from DI-VST, because $\mathbf{z}$ from auto-encoding is explicitly trained for $\mathbf{x}$ reconstruction. Adding $\mathcal{L}_{attr}$ is effective in forcing the decoder to take $\mathbf{z}$ into account during its generation, which helps the most in more challenging open-domain chatting data, e.g. SW and DD. The accuracy of ST-ED on SW is worse than the other two datasets. The reason is that SW contains many short utterances that can be either a continuation of the same speaker or a new turn from the other speaker, whereas the responses in the other two domains are always followed by a different speaker. The more complex context pattern in SW may require special treatment. We leave it for future work.

Domain	AE-ED	$+L_{\text{attr}}$	ST-ED	$+L_{\text{attr}}$
SMD	93.5%	94.8%	91.9%	93.8%
DD	88.4%	93.6%	78.5%	86.1%
SW	84.7%	94.6%	57.3%	61.3%

Table 6: Results for attribute accuracy with and without attribute loss.

The second experiment checks if the policy network π is able to predict the right latent action given just the dialog context. We report both accuracy, i.e. $\text{argmax}_k\, q_{\mathcal{R}}(k|\mathbf{x}) = \text{argmax}_{k'}\, p_{\pi}(k'|\mathbf{c})$ and perplexity of $p_{\pi}(\mathbf{z}|\mathbf{c})$. The perplexity measure is more useful for open domain dialogs because decision-making in complex dialogs is often one-to-many given a similar context (Zhao et al., 2017). Table 7 shows the prediction scores on the three dialog datasets. These scores provide useful insights to understand the complexity of a dialog dataset. For example, accuracy on open-domain chatting is harder than the task-oriented SMD data. Also, it is intuitive that predicting system actions is easier than predicting user actions on SMD. Also, in general the prediction scores for ST-ED are higher the ones for AE-ED. The reason is related to our previous discussion about the granularity of the latent actions. Since latent actions from DI-VST mainly model the the type of utterances used in certain types of context, it is easier for the policy network to predict latent actions from DI-VST. Therefore, choosing the type of latent actions is a design choice and depends on the type of interpretability that is needed. We finish with an example generated from the two variants of LAED on SMD as shown in Table 8. Given a dialog context, our systems are able to output a probability distribution over different latent actions that have interpretable meaning along with their natural language realizations.

	SMD	
AE-ED	3.045 (51.5% sys 52.4% usr 50.5%)	
ST-ED	1.695 (75.5% sys 82.1% usr 69.2%)	
	DD	SW
AE-ED	4.47 (35.8%)	4.46 (31.68%)
ST-ED	3.89 (47.5%)	3.68 (33.2%)

Table 7: Performance of policy network. $\mathcal{L}_{attr}$ is included in training.

c	**usr**: Where does my friend live?	
Model	Action	Generated Responses
AE-ED	give loc info $p(\mathbf{z}\|\mathbf{c})$=0.34	- Tom is at 753 University Ave, and a road block. - Comfort Inn is at 7 miles away.
	give user info $p(\mathbf{z}\|\mathbf{c})$=0.22	- Your home address is 5671 barringer street. - Your home is at 10 ames street.
ST-ED	give loc info $p(\mathbf{z}\|\mathbf{c})$=0.93	- Jill's house is 8 miles away at 347 Alta Mesa Ave. - Jill lives at 347 Alta Mesa Ave.

Table 8: Interpretable dialog generation on SMD with top probable latent actions. AE-ED predicts more fine-grained but more error-prone actions.

5 Conclusion and Future Work

This paper presents a novel unsupervised framework that enables the discovery of discrete latent actions and interpretable dialog response generation. Our main contributions reside in the two sentence representation models DI-VAE and DI-VST, and their integration with the encoder decoder models. Experiments show the proposed methods outperform strong baselines in learning discrete latent variables and showcase the effectiveness of interpretable dialog response generation. Our findings also suggest promising future research directions, including learning better context-based latent actions and using reinforcement learning to adapt policy networks. We believe that this work is an important step forward towards creating generative dialog models that can not only generalize to large unlabelled datasets in complex domains but also be explainable to human users.

References

Felix Vsevolodovich Agakov. 2005. *Variational Information Maximization in Stochastic Environments*. Ph.D. thesis, University of Edinburgh.

John Langshaw Austin. 1975. *How to do things with words*. Oxford university press.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Dan Bohus, Antoine Raux, Thomas K Harris, Maxine Eskenazi, and Alexander I Rudnicky. 2007. Olympus: an open-source framework for conversational spoken language interface research. In *Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies*. Association for Computational Linguistics, pages 32–39.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. *arXiv preprint arXiv:1702.05962* .

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414* .

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 201–204.

John J Godfrey and Edward Holliman. 1997. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia* .

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483* .

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*. pages 1587–1596.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* .

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Staffan Larsson and David R Traum. 2000. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering* 6(3-4):323–340.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957* .

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* .

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* .

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*. pages 1727–1736.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069* .

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26(3):339–373.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*. pages 6309–6318.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11(Oct):2837–2854.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. *arXiv preprint arXiv:1705.10229* .

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language* 21(2):393–422.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340* .

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960* .

Shengjia Zhao S, Jiaming Song, and Stefano Ermon. 2017. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262* .

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. *arXiv preprint arXiv:1704.01691* .

Learning to Control the Specificity in Neural Response Generation

Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, Jun Xu and Xueqi Cheng
University of Chinese Academy of Sciences, Beijing, China
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
{zhangruqing,fanyixing}@software.ict.ac.cn
{guojiafeng,lanyanyan,junxu,cxq}@ict.ac.cn

Abstract

In conversation, a general response (e.g., "I don't know") could correspond to a large variety of input utterances. Previous generative conversational models usually employ a single model to learn the relationship between different utterance-response pairs, thus tend to favor general and trivial responses which appear frequently. To address this problem, we propose a novel controlled response generation mechanism to handle different utterance-response relationships in terms of specificity. Specifically, we introduce an explicit specificity control variable into a sequence-to-sequence model, which interacts with the usage representation of words through a Gaussian Kernel layer, to guide the model to generate responses at different specificity levels. We describe two ways to acquire distant labels for the specificity control variable in learning. Empirical studies show that our model can significantly outperform the state-of-the-art response generation models under both automatic and human evaluations.

1 Introduction

Human-computer conversation is a critical and challenging task in AI and NLP. There have been two major streams of research in this direction, namely task oriented dialog and general purpose dialog (i.e., chit-chat). Task oriented dialog aims to help people complete specific tasks such as buying tickets or shopping, while general purpose dialog attempts to produce natural and meaningful conversations with people regarding a wide range of topics in open domains (Perez-Marin, 2011; Sordoni et al.). In recent years, the latter has attracted much attention in both academia and industry as a way to explore the possibility in developing a general purpose AI system in language (e.g., chatbots).

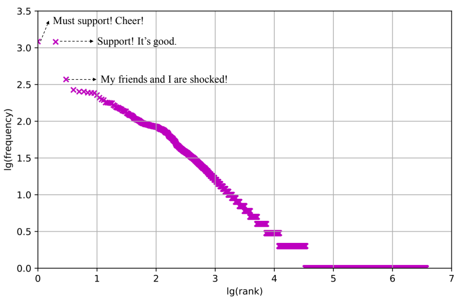


Figure 1: Rank-frequency distribution of the responses in the chit-chat corpus, with x and y axes being lg(rank order) and lg(frequency) respectively.

A widely adopted approach to general purpose dialog is learning a generative conversational model from large scale social conversation data. Most methods in this line are constructed within the statistical machine translation (SMT) framework, where a sequence-to-sequence (Seq2Seq) model is learned to "translate" an input utterance into a response. However, general purpose dialog is intrinsically different from machine translation. In machine translation, since every sentence and its translation are semantically equivalent, there exists a 1-to-1 relationship between them. However, in general purpose dialog, a general response (e.g., "I don't know") could correspond to a large variety of input utterances. For example, in the chit-chat corpus used in this study (as shown in Figure 1), the top three most frequently appeared responses are "Must support! Cheer!", "Support! It's good.", and "My friends and I are shocked!", where the response "Must support! Cheer!" is used for 1216 different input utterances. Previous Seq2Seq models, which treat all the utterance-response pairs uniformly and employ a single model to learn the relationship between them, will inevitably favor such general responses with high frequency. Although these responses are safe for replying different utterances, they are boring and trivial since they carry little information, and may quickly lead to an end of the conversation.

There have been a few efforts attempting to address this issue in literature. Li et al. (2016a) proposed to use the Maximum Mutual Information (MMI) as the objective to penalize general responses. It could be viewed as a post-processing approach which did not solve the generation of trivial responses fundamentally. Xing et al. (2017) pre-defined a set of topics from an external corpus to guide the generation of the Seq2Seq model. However, it is difficult to ensure that the topics learned from the external corpus are consistent with that in the conversation corpus, leading to the introduction of additional noises. Zhou et al. (2017) introduced latent responding factors to model multiple responding mechanisms. However, these latent factors are usually difficult in interpretation and it is hard to decide the number of the latent factors.

In our work, we propose a novel controlled response generation mechanism to handle different utterance-response relationships in terms of specificity. The key idea is inspired by our observation on everyday conversation between humans. In human-human conversation, people often actively control the specificity of responses depending on their own response purpose (which might be affected by a variety of underlying factors like their current mood, knowledge state and so on). For example, they may provide some interesting and specific responses if they like the conversation, or some general responses if they want to end it. They may provide very detailed responses if they are familiar with the topic, or just "I don't know" otherwise. Therefore, we propose to simulate the way people actively control the specificity of the response.

We employ a Seq2Seq framework and further introduce an explicit specificity control variable to represent the response purpose of the agent. Meanwhile, we assume that each word, beyond the semantic representation which relates to its meaning, also has another representation which relates to the usage preference under different response purpose. We name this representation as the usage representation of words. The specificity control variable then interacts with the usage representation of words through a Gaussian Kernel layer, and guides the Seq2Seq model to generate responses at different specificity levels. We refer to our model as Specificity Controlled Seq2Seq model (SC-Seq2Seq). Note that unlike the work by (Xing et al., 2017), we do not rely on any external corpus to learn our model. All the model parameters are learned on the same conversation corpus in an end-to-end way.

We employ distant supervision to train our SC-Seq2Seq model since the specificity control variable is unknown in the raw data. We describe two ways to acquire distant labels for the specificity control variable, namely Normalized Inverse Response Frequency (NIRF) and Normalized Inverse Word Frequency (NIWF). By using normalized values, we restrict the specificity control variable to be within a pre-defined continuous value range with each end has very clear meaning on the specificity. This is significantly different from the discrete latent factors in (Zhou et al., 2017) which are difficult in interpretation.

We conduct an empirical study on a large public dataset, and compare our model with several state-of-the-art response generation methods. Empirical results show that our model can generate either general or specific responses, and significantly outperform existing methods under both automatic and human evaluations.

2 Related Work

In this section, we briefly review the related work on conversational models and response specificity.

2.1 Conversational Models

Automatic conversation has attracted increasing attention over the past few years. At the very beginning, people started the research using hand-crafted rules and templates (Walker et al., 2001; Williams et al., 2013; Henderson et al., 2014). These approaches required little data for training but huge manual effort to build the model, which is very time-consuming. For now, conversational models fall into two major categories: retrieval-based and generation-based. Retrieval-based conversational models search the most suitable response from candidate responses using different schemas (Kearns, 2000; Wang et al., 2013; Yan et al., 2016). These methods rely on pre-existing responses, thus are difficult to be extended to open domains (Zhou et al., 2017). With the large amount of conversation data available on the Internet, generation-based conversational models developed within a SMT framework (Ritter et al., 2011; Cho et al., 2014; Bahdanau et al., 2015) show promising results. Shang et al. (2015) generated replies for short-text conversation by encoder-decoder-based neural network with local and global attentions. Serban et al. (2016) built an end-to-end dialogue system using generative hierarchical neural network. Gu et al. (2016) introduced copynet to simulate the repeating behavior of humans in conversation. Similarly, our model is also based on the encoder-decoder framework.

2.2 Response Specificity

Some recent studies began to focus on generating more specific or informative responses in conversation. It is also called a diversity problem since if each response is more specific, it would be more diverse between responses of different utterances. As an early work, Li et al. (2016a) used Maximum Mutual Information (MMI) as the objective to penalize general responses. Later, Li et al. (2017) proposed a data distillation method, which trains a series of generative models at different levels of specificity and uses a reinforcement learning model to choose the model best suited for decoding depending on the conversation context. These methods circumvented the general response issue by using either a post-processing approach or a data selection approach.

Besides, Li et al. (2016b) tried to build a personalized conversation engine by adding extra personal information. Xing et al. (2017) incorporated the topic information from an external corpus into the Seq2Seq framework to guide the generation. However, external dataset may not be always available or consistent with the conversation dataset in topics. Zhou et al. (2017) introduced latent responding factors to the Seq2Seq model to avoid generating safe responses. However, these latent factors are usually difficult in interpretation and hard to decide the number.

Moreover, Mou et al. (2016) proposed a content-introducing approach to generate a response based on a predicted keyword. Yao et al. (2016) attempted to improve the specificity with the reinforcement learning framework by using the averaged IDF score of the words in the response as a reward. Shen et al. (2017) presented a conditional variational framework for generating specific responses based on specific attributes. Unlike these existing methods, we introduce an explicit specificity control variable into a Seq2Seq model to handle different utterance-response relationships in terms of specificity.

3 Specificity Controlled Seq2Seq Model

In this section, we present the Specificity Controlled Seq2Seq model (SC-Seq2Seq), a novel Seq2Seq model designed for actively controlling the generated responses in terms of specificity.

3.1 Model Overview

The basic idea of a generative conversational model is to learn the mapping from an input utterance to its response, typically using an encoder-decoder framework. Formally, given an input utterance sequence $\mathbf{X} = (x_1, x_2, \dots, x_T)$ and a target response sequence $\mathbf{Y} = (y_1, y_2, \dots, y_{T'})$, a neural Seq2Seq model is employed to learn $p(\mathbf{Y}|\mathbf{X})$ based on the training corpus $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y})|\mathbf{Y} \text{ is the response of } \mathbf{X}\}$. By maximizing the likelihood of all the utterance-response pairs with a single mapping mechanism, the learned Seq2Seq model will inevitably favor those general responses that can correspond to a large variety of input utterances.

To address this issue, we assume that there are different mapping mechanisms between utterance-response pairs with respect to their specificity relation. Rather than involving some latent factors, we propose to introduce an explicit variable s into a Seq2Seq model to handle different utterance-response mappings in terms of specificity. By doing so, we hope that (1) s would have explicit meaning on specificity, and (2) s could not only interpret but also actively control the generation of the response $\mathbf{Y}$ given the input utterance $\mathbf{X}$. The goal of our model becomes to learn $p(\mathbf{Y}|\mathbf{X}, s)$ over the corpus $\mathcal{D}$, where we acquire distant labels for s from the same corpus for learning. The overall architecture of SC-Seq2Seq is depicted in Figure 2, and we will detail our model as follows.

3.1.1 Encoder

The encoder is to map the input utterance $\mathbf{X}$ into a compact vector that can capture its essential topics. Specifically, we use a bi-directional GRU (Cho et al., 2014) as the utterance encoder, and each word x_i is firstly represented by its semantic representation $\mathbf{e}_i$ mapped by semantic embedding

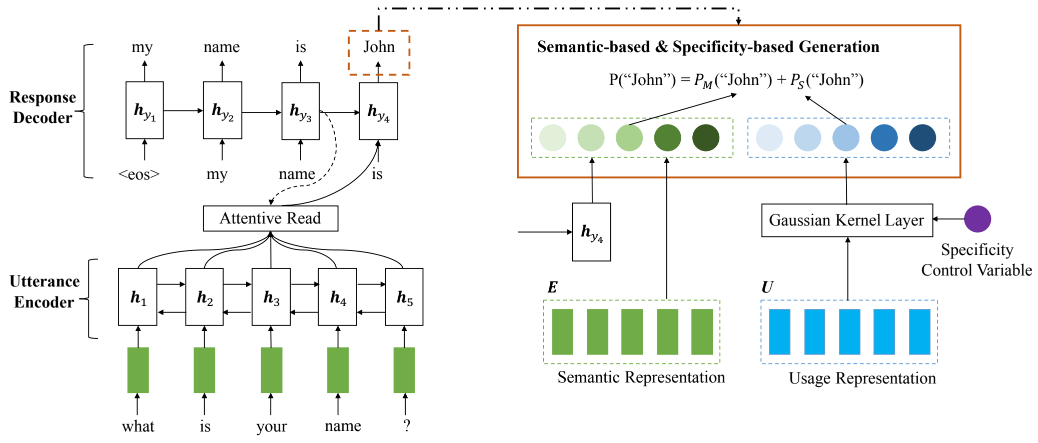


Figure 2: The overall architecture of SC-Seq2Seq model.

matrix $\mathbf{E}$ as the input of the encoder. Then, the encoder represents the utterance $\mathbf{X}$ as a series of hidden vectors $\{\mathbf{h}_t\}_{t=1}^{T}$ modeling the sequence from both forward and backward directions. Finally, we use the final backward hidden state as the initial hidden state of the decoder.

3.1.2 Decoder

The decoder is to generate a response $\mathbf{Y}$ given the hidden representations of the input utterance $\mathbf{X}$ under some specificity level denoted by the control variable s. Specifically, at step t, we define the probability of generating any target word y_t by a "mixture" of probabilities:

$$p(y_t) = \beta p_M(y_t) + \gamma p_S(y_t), \tag{1}$$

where $p_M(y_t)$ denotes the semantic-based generation probability, $p_S(y_t)$ denotes the specificity-based generation probability, β and γ are the coefficients.

Specifically, $p_M(y_t)$ is defined the same as that in traditional Seq2Seq model (Sutskever et al., 2014):

$$p_M(y_t = w) = \mathbf{w}^{\mathrm{T}}(\mathbf{W}_M^h \cdot \mathbf{h}_{y_t} + \mathbf{W}_M^e \cdot \mathbf{e}_{t-1} + \mathbf{b}_M), \tag{2}$$

where $\mathbf{w}$ is a one-hot indicator vector of the word w and $\mathbf{e}_{t-1}$ is the semantic representation of the $t-1$-th generated word in decoder. $\mathbf{W}_M^h$, $\mathbf{W}_M^e$ and $\mathbf{b}_M$ are parameters. $\mathbf{h}_{y_t}$ is the t-th hidden state in the decoder which is computed by:

$$\mathbf{h}_{y_t} = f(y_{t-1}, \mathbf{h}_{y_{t-1}}, \mathbf{c}_t), \tag{3}$$

where f is a GRU unit and $\mathbf{c}_t$ is the context vector to allow the decoder to pay different attention to different parts of input at different steps (Bahdanau et al., 2015).

$p_S(y_t)$ denotes the generation probability of the target word given the specificity control variable s. Here we introduce a Gaussian Kernel layer to define this probability. Specifically, we assume that each word, beyond its semantic representation $\mathbf{e}$, also has a usage representation $\mathbf{u}$ mapped by usage embedding matrix $\mathbf{U}$. The usage representation of a word denotes its usage preference under different specificity. The specificity control variable s then interacts with the usage representations through the Gaussian Kernel layer to produce the specificity-based generation probability $p_S(y_t)$:

$$p_S(y_t = w) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(\Psi_S(\mathbf{U},\mathbf{w}) - s)^2}{2\sigma^2}),$$
$$\Psi_S(\mathbf{U},\mathbf{w}) = \sigma(\mathbf{w}^{\mathrm{T}}(\mathbf{U} \cdot \mathbf{W}_U + \mathbf{b}_U)), \tag{4}$$

where σ^2 is the variance, and $\Psi_S(\cdot)$ maps the word usage representation into a real value with the specificity control variable s as the mean of the Gaussian distribution. $\mathbf{W}_U$ and $\mathbf{b}_U$ are parameters to be learned. Note here in general we can use any real-value function to define $\Psi_S(\mathbf{U},\mathbf{w})$. In this work, we use the sigmoid function $\sigma(\cdot)$ for $\Psi_S(\mathbf{U},\mathbf{w})$ since we want to define s within the range [0,1] so that each end has very clear meaning on the specificity, i.e., 0 denotes the most general response while 1 denotes the most specific response. In the next section, we will also keep this property when we define the distant label for the control variable.

3.2 Distant Supervision

We train our SC-Seq2Seq model by maximizing the log likelihood of generating responses over the training set $\mathcal{D}$:

$$\mathcal{L} = \sum_{(\mathbf{X},\mathbf{Y})\in\mathcal{D}} \log P(\mathbf{Y}|\mathbf{X}, s; \theta). \tag{5}$$

where θ denotes all the model parameters. Note here since s is an explicit control variable in our model, we need the triples $(\mathbf{X}, \mathbf{Y}, s)$ for training. However, s is not directly available in the raw conversation corpus, thus we acquire distant labels for s to learn our model. We introduce two ways of distant supervision on the specificity control variable s, namely Normalized Inverse Response Frequency (NIRF) and Normalized Inverse Word Frequency (NIWF).

3.2.1 Normalized Inverse Response Frequency

Normalized Inverse Response Frequency (NIRF) is based on the assumption that a response is more general if it corresponds to more input utterances in the corpus. Therefore, we use the inverse frequency of a response in a conversation corpus to indicate its specificity level. Specifically, we first build the response collection $\mathcal{R}$ by extracting all the responses from $\mathcal{D}$. For a response $\mathbf{Y} \in \mathcal{R}$, let $f_{\mathbf{Y}}$ denote its corpus frequency in $\mathcal{R}$, we compute its Inverse Response Frequency (IRF) as:

$$\mathrm{IRF}_{\mathbf{Y}} = \log(1 + |\mathcal{R}|)/f_{\mathbf{Y}}, \tag{6}$$

where $|\mathcal{R}|$ denotes the size of the response collection $\mathcal{R}$. Next, we use the min-max normalization method (Jain et al., 2005) to obtain the NIRF value. Namely,

$$\mathrm{NIRF}_{\mathbf{Y}} = \frac{\mathrm{IRF}_{\mathbf{Y}} - \min_{\mathbf{Y}' \in \mathcal{R}}(\mathrm{IRF}_{\mathbf{Y}'})}{\max_{\mathbf{Y}' \in \mathcal{R}}(\mathrm{IRF}_{\mathbf{Y}'}) - \min_{\mathbf{Y}' \in \mathcal{R}}(\mathrm{IRF}_{\mathbf{Y}'})}. \tag{7}$$

where $\max(\mathrm{IRF}_{\mathcal{R}})$ and $\min(\mathrm{IRF}_{\mathcal{R}})$ denotes the maximal and minimum IRF value in $\mathcal{R}$ respectively. The NIRF value is then used as the distant label of s in training. Note here by using normalized values, we aim to constrain the specificity control variable s to be within the pre-defined continuous value range [0,1].

3.2.2 Normalized Inverse Word Frequency

Normalized Inverse Word Frequency (NIWF) is based on the assumption that the specificity level of a response depends on the collection of words it contains, and the sentence is more specific if it contains more specific words. Hence, we can use the inverse corpus frequency of the words to indicate the specificity level of a response. Specifically, for a word y in the response $\mathbf{Y}$, we first obtain its Inverse Word Frequency (IWF) by:

$$\mathrm{IWF}_{y} = \log(1 + |\mathcal{R}|)/f_{y}, \tag{8}$$

where f_y denotes the number of responses in $\mathcal{R}$ containing the word y. Since a response usually contains a collection of words, there would be multiple ways to define the response-level IWF value, e.g., sum, average, minimum or maximum of the IWF values of all the words. In our work, we find that the best performance can be achieved by using the maximum of the IWF of all the words in $\mathbf{Y}$ to represent the response-level IWF by

$$\mathrm{IWF}_{\mathbf{Y}} = \max_{y \in \mathbf{Y}}(\mathrm{IWF}_{y}). \tag{9}$$

This is reasonable since a response is specific as long as it contains some specific words. We do not require all the words in a response to be specific, thus sum, average, and minimum would not be appropriate operators for computing the response-level IWF. Again, we use min-max normalization to obtain the NIWF value for the response $\mathbf{Y}$.

3.3 Specificity Controlled Response Generation

Given a new input utterance, we can employ the learned SC-Seq2Seq model to generate responses at different specificity levels by varying the control variable s. In this way, we can simulate human conversations where one can actively control the response specificity depending on his/her own mind. When we apply our model to a chatbot, there might be different ways to use the control variable for conversation in practice. If we want the agent to always generate informative responses, we can set s to 1 or some values close to 1. If we want the agent to be more dynamic, we can sample s within the range [0,1] to enrich the styles in the response. We may further employ some reinforcement learning technique to learn to adjust the control variable depending on users' feedbacks. This would make the agent even more vivid, and we leave this as our future work.

4 Experiment

In this section, we conduct experiments to verify the effectiveness of our proposed model.

4.1 Dataset Description

We conduct our experiments on the public Short Text Conversation (STC) dataset[1] released in NTCIR-13. STC maintains a large repository of post-comment pairs from the Sina Weibo which is one of the popular Chinese social sites.

[1]http://ntcirstc.noahlab.com.hk/STC2/stc-cn.htm

Utterance-response pairs	3,788,571
Utterance vocabulary #w	120,930
Response vocabulary #w	524,791
Utterance max #w	38
Utterance avg #w	13
Response max #w	74
Response avg #w	10

Table 1: Short Text Conversation (STC) data statistics: #w denotes the number of Chinese words.

STC dataset contains roughly 3.8 million post-comment pairs, which could be used to simulate the utterance-response pairs in conversation. We employ the Jieba Chinese word segmenter[2] to tokenize the utterances and responses into sequences of Chinese words, and the detailed dataset statistics are shown in Table 1. We randomly selected two subsets as the development and test dataset, each containing 10k pairs. The left pairs are used for training.

4.2 Baselines Methods

We compare our proposed SC-Seq2Seq model against several state-of-the-art baselines: (1) **Seq2Seq-att**: the standard Seq2Seq model with the attention mechanism (Bahdanau et al., 2015); (2) **MMI-bidi**: the Seq2Seq model using Maximum Mutual Information (MMI) as the objective function to reorder the generated responses (Li et al., 2016a); (3) **MARM**: the Seq2Seq model with a probabilistic framework to model the latent responding mechanisms (Zhou et al., 2017); (4) **Seq2Seq+IDF**: an extension of Seq2Seq-att by optimizing specificity under the reinforcement learning framework, where the reward is calculated as the sentence level IDF score of the generated response (Yao et al., 2016). We refer to our model trained using NIRF and NIWF as **SC-Seq2Seq**$_{\text{NIRF}}$ and **SC-Seq2Seq**$_{\text{NIWF}}$ respectively.

4.3 Implementation Details

As suggested in (Shang et al., 2015), we construct two separate vocabularies for utterances and responses by using 40,000 most frequent words on each side in the training data, covering 97.7% words in utterances and 96.1% words in responses respectively. All the remaining words are replaced by a special token <UNK> symbol.

We implemented our model in Tensorflow[3]. We tuned the hyper-parameters via the development set. Specifically, we use one layer of bi-directional GRU for encoder and another uni-directional GRU for decoder, with the GRU hidden unit size set as 300 in both the encoder and decoder. The dimension of semantic word embeddings in both utterances and responses is 300, while the dimension of usage word embeddings in responses is 50. We apply the Adam algorithm (Kingma and Ba, 2015) for optimization, where the parameters of Adam are set as in (Kingma and Ba, 2015). The variance σ^2 of the Gaussian Kernel layer is set as 1, and all other trainable parameters are randomly initialized by uniform distribution within [-0.08,0.08]. The mini-batch size for the update is set as 128. We clip the gradient when its norm exceeds 5.

Our model is trained on a Tesla K80 GPU card, and we run the training for up to 12 epochs, which takes approximately five days. We select the model that achieves the lowest perplexity on the development dataset, and we report results on the test dataset.

4.4 Evaluation Methodologies

For evaluation, we follow the existing work and employ both automatic and human evaluations: (1) **distinct-1 & distinct-2** (Li et al., 2016a): we count numbers of distinct unigrams and bigrams in the generated responses, and divide the numbers by total number of generated unigrams and bigrams. Distinct metrics (both the numbers and the ratios) can be used to evaluate the specificity/diversity of the responses. (2) **BLEU** (Papineni et al., 2002): BLEU has been proved strongly correlated with human evaluations. BLEU-n measures the average n-gram precision on a set of reference sentences. (3) **Average & Extrema** (Serban et al., 2017): Average and Extrema projects the generated response and the ground truth response into two separate vectors by taking the mean over the word embeddings or taking the extremum of each dimension respectively, and then computes the cosine similarity between them. (4) **Human evaluation**: Three labelers with rich Weibo experience were recruited to conduct evaluation. Responses from different models are randomly mixed for labeling. Labelers refer to 300 random sampled test utterances and score the quality of the responses with the following criteria: 1) **+2**: the response is not only semantically relevant and grammatical, but also informat-

[2]https://pypi.python.org/pypi/jieba

[3]https://www.tensorflow.org/

	Models	distinct-1	distinct-2	BLEU-1	BLEU-2	Average	Extrema
SC-Seq2Seq$_{\text{NIRF}}$	$s = 1$	5258/0.064	16195/0.269	15.109	7.023	0.578	0.380
	$s = 0.8$	5337/0.065	16105/0.271	15.112	7.003	0.578	0.381
	$s = 0.5$	5318/0.065	16183/0.269	15.054	7.001	0.578	0.380
	$s = 0.2$	5323/0.065	16087/0.270	15.168	7.032	0.580	0.380
	$s = 0$	5397/0.066	16319/0.271	15.093	7.011	0.577	0.380
SC-Seq2Seq$_{\text{NIWF}}$	$s = 1$	**11588/0.116**	**27144/0.347**	12.392	5.869	0.554	0.353
	$s = 0.8$	6006/0.051	17843/0.257	11.492	5.703	0.553	0.350
	$s = 0.5$	2835/0.050	9537/0.235	**16.122**	**7.674**	**0.609**	**0.399**
	$s = 0.2$	1534/0.048	5117/0.218	8.313	4.058	0.542	0.335
	$s = 0$	1038/0.046	3154/0.211	4.417	3.283	0.549	0.334

Table 2: Model analysis of our SC-Seq2Seq under the automatic evaluation.

Models	distinct-1	distinct-2	BLEU-1	BLEU-2	Average	Extrema
Seq2Seq-att	5048/0.060	15976/0.168	15.062	6.964	0.575	0.376
MMI-bidi	5074/0.082	12162/0.287	15.772	7.215	0.586	0.381
MARM	2566/0.096	3294/0.312	7.321	3.774	0.512	0.336
Seq2Seq+IDF	4722/0.052	15384/0.229	14.423	6.743	0.572	0.369
SC-Seq2Seq$_{\text{NIWF},s=1}$	**11588/0.116**	**27144/0.347**	12.392	5.869	0.554	0.353
SC-Seq2Seq$_{\text{NIWF},s=0.5}$	2835/0.050	9537/0.235	**16.122**	**7.674**	**0.609**	**0.399**

Table 3: Comparisons between our SC-Seq2Seq and the baselines under the automatic evaluation.

ive and interesting; 2) **+1**: the response is grammatical and can be used as a response to the utterance, but is too trivial (e.g., "I don't know"); 3) **+0**: the response is semantically irrelevant or ungrammatical (e.g., grammatical errors or UNK). Agreements to measure inter-rater consistency among three labelers are calculated with the Fleiss' kappa (Fleiss and Cohen, 1973).

4.5 Evaluation Results

Model Analysis: We first analyze our models trained with different distant supervision information. For each model, given a test utterance, we vary the control variable s by setting it to five different values (i.e., 0, 0.2, 0.5, 0.8, 1) to check whether the learned model can actually achieve different specificity levels. As shown in Table 2, we find that: (1) The SC-Seq2Seq model trained with NIRF cannot work well. The test performances are almost the same with different s value. This is surprising since the NIRF definition seems to be directly corresponding to the specificity of a response. By conducting further analysis, we find that even though the conversation dataset is large, it is still limited and a general response could appear very few times in this corpus. In other words, the inverse frequency of a response is very weakly correlated with its response specificity. (2) The SC-Seq2Seq model trained with NIWF can achieve our purpose. By varying the control variable s from 0 to 1, the generated responses turn from general to specific as measured by the distinct metrics. The results indicate that the max inverse word frequency in a response is a good distant label for the response specificity. (3) When we compare the generated responses against ground truth data, we find the SC-Seq2Seq$_{\text{NIWF}}$ model with the control variable s set to 0.5 can achieve the best performances. The results indicate that there are diverse responses in real data in terms of specificity, and it is necessary to take a balanced setting if we want to fit the ground truth.

Baseline Comparison: The performance comparisons between our model and the baselines are shown in Table 3. We have the following observations: (1) By using MMI as the objective, MMI-bidi can improve the specificity (in terms of distinct ratios) over the traditional Seq2Seq-att model. (2) MARM can achieve the best distinct ratios among the baseline methods, but the worst in terms of the distinct numbers. The results indicate that MARM tends to generate specific but very short responses. Meanwhile, its low BLEU scores also show that the responses generated by MARM deviate from the ground truth significantly. (3) By using the IDF information as the reward to train

	+2	+1	+0	kappa
Seq2Seq-att	29.32%	25.27%	45.41%	0.448
MMI-bidi	30.40%	24.85%	44.75%	0.471
MARM	20.11%	27.96%	51.93%	0.404
Seq2Seq+IDF	28.81%	23.87%	47.33%	0.418
SC-Seq2Seq$_{\text{NIWF,s=1}}$	42.47%	14.29%	43.24%	0.507
SC-Seq2Seq$_{\text{NIWF,s=0.5}}$	20.62%	40.16%	39.22%	0.451
SC-Seq2Seq$_{\text{NIWF,s=0}}$	14.34%	46.38%	39.28%	0.526

Table 4: Results on the human evaluation.

the Seq2Seq model, the Seq2Seq+IDF does not show much advantages, but only achieves comparable results as MMI-bidi. (4) By setting the control variable s to 1, our SC-Seq2Seq$_{\text{NIWF}}$ model can achieve the best specificity performance as evaluated by the distinct metrics. By setting the control variable s to 0.5, our SC-Seq2Seq$_{\text{NIWF}}$ model can best fit the ground truth data as evaluated by the BLEU scores, Average and Extrema. All the improvements over the baseline models are statistically significant (p-value < 0.01). These results demonstrate the effectiveness as well as the flexibility of our controlled generation model.

Table 4 shows the human evaluation results. We can observe that: (1) SC-Seq2Seq$_{\text{NIWF,s=1}}$ generates the most informative responses and interesting (labeled as "+2") and the least general responses than all the baseline models. Meanwhile, SC-Seq2Seq$_{\text{NIWF,s=0}}$ generates the most general responses (labeled as "+1"); (2) MARM generates the most bad responses (labeled as "+0"), which indicates the drawbacks of the unknown latent responding mechanisms; (3) The kappa values of our models are all larger than 0.4, considered as "moderate agreement" regarding quality of responses. The largest kappa value is achieved by SC-Seq2Seq$_{\text{NIWF,s=0}}$, which seems reasonable since it is easy to reach an agreement on general responses. Sign tests demonstrate the improvements of SC-Seq2Seq$_{\text{NIWF,s=1}}$ to the baseline models are statistically significant (p-value < 0.01). All the human judgement results again demonstrate the effectiveness of our controlled generation mechanism.

4.6 Case Study

To better understand how different models perform, we conduct some case studies. We randomly sample three utterances from the test dataset, and show the responses generated by different models.

Utterance	三室一厅的帐篷，你值得拥有！A tent with three rooms and one hall, you're worth it!
Seq2Seq-att	这个不错，很好！It looks good, very good!
MMI-bidi	这个不错，我喜欢。It looks good, I like it.
MARM	不错。Good.
Seq2Seq+IDF	帐篷不错。The tent is good.
s=1	我想知道这帐篷是什么牌子的？I want to know what the brand of this tent is?
s=0.8	我想知道这是在哪买的？I want to know where to buy it?
s=0.5	我也想买一个,可惜买不起！I want one too, but it's a pity that I can't afford it.
s=0.2	不错不错！Good, good.
s=0	真的吗？Really?
Utterance	哎哟! 被发现了! 汤唯有必要那么美吗? Oh! Got caught! Why is Tang Wei so beautiful?
Seq2Seq-att	我超喜欢她！I really like her!
MMI-bidi	好美。我超喜欢她！So beautiful. I really like her!
MARM	我不喜欢汤唯。I don't like Tang Wei.
Seq2Seq+IDF	我超喜欢她的，我喜欢。I really like her, I like.
s=1	我超喜欢她的眼妆，她的气质很美！I really like her eye make-up, and her temperament is so beautiful!
s=0.8	我超喜欢她的，她的眼睛很雪亮！I really like her, and her eyes are so bright!
s=0.5	我觉得她很美，很喜欢她！I think she is very beautiful, and I really like her!
s=0.2	我觉得她很美！I think she is very beautiful.
s=0	是的。Yes.
Utterance	将来也要养一只这么乖的的萨摩，让它陪着孩子一起长大。I will keep such a sweet samoyed, and let it accompany my children to grow up.
Seq2Seq-att	这不是萨摩吗? Is it a samoyed?
MMI-bidi	好可爱的萨摩。What a lovely samoyed.
MARM	好可爱啊。So lovely.
Seq2Seq+IDF	好可爱的狗狗。What a lovely dog.
s=1	萨摩耶好可爱，真想养一只! The samoyed is so lovely, and I really want to keep one!
s=0.8	萨摩好可爱，好想掐掐。The samoyed is so lovely, and I really want to pinch it.
s=0.5	好可爱的狗狗，好可爱的狗狗。What a lovely dog, what a lovely dog.
s=0.2	好可爱！好可爱！So lovely, so lovely!
s=0	好可爱！So lovely!

Table 5: Examples of response generation from the STC test data. s = 1, 0.8, 0.5, 0.2, 0 are the outputs of our SC-Seq2Seq$_{\text{NIWF}}$ with different s values.

As shown in Table 5, we can find that: (1) The responses generated by the four baselines are often quite general and short, which may quickly lead to an end of the conversation. (2) SC-Seq2Seq$_{\text{NIWF}}$ with large control variable values (i.e., $s > 0.5$) can generate very long and specific responses. In these responses, we can find many informative words. For example, in case 2 with s as 1 and 0.8, we can find words like "眼妆(eye make-up)", "气质(temperament)" and "雪亮(bright)" which are quite specific and strongly related to the conversation topic of "beauty". (3) When we decrease the control variable value, the generated responses become more and more general and shorter from our SC-Seq2Seq$_{\text{NIWF}}$ model.

爸爸(dad)		水果(fruits)		脂肪肝(fatty liver)		单反相机(DSLR)	
Usage	Semantic	Usage	Semantic	Usage	Semantic	Usage	Semantic
更好(better)	妈妈(mother)	尝试(attempt)	蔬菜(vegetables)	坐久(outsit)	胖(fat)	亚洲杯(Asian Cup)	照相机(camera)
睡觉(sleep)	哥哥(brother)	诱惑(tempt)	牛奶(milk)	素食主义(vegetarian)	减肥(diet)	读取(read)	摄影(photography)
快乐(happy)	老公(husband)	表现(express)	西瓜(watermelon)	散步(walk)	高血压(hypertension)	半球(hemispherical)	镜头(shot)
无聊(boring)	爷爷(grandfather)	拥有(own)	米饭(rice)	因果关系(causality)	亚健康(sub-health)	防辐射(anti-radiation)	影楼(studio)
电影(movie)	姑娘(girl)	梦想(dream)	巧克力(chocolate)	哑铃(dumbbell)	呕吐(emesis)	无人机(UAV)	写真(image)

Table 6: Target words and their top-5 similar words under usage and semantic representations respectively.

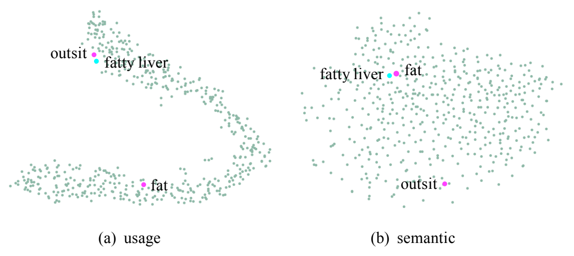


Figure 3: t-SNE embeddings of usage and semantic vectors.

4.7 Analysis on Usage Representations

We also conduct some analysis to understand the usage representations of words introduced in our model. We randomly sample 500 words from our SC-Seq2Seq$_{\text{NIWF}}$ and apply t-SNE (Maaten and Hinton, 2008) to visualize both usage and semantic embeddings. As shown in Figure 3, we can see that the two distributions are significantly different. In the usage space, words like "脂肪肝(fatty liver)" and "久坐(outsit)" lie closely which are both specific words, and both are far from the general words like "胖(fat)". On the contrary, in the semantic space, "脂肪肝(fatty liver)" is close to "胖(fat)" since they are semantically related, and both are far from the word "久坐(outsit)". Furthermore, given some sampled target words, we also show the top-5 similar words based on cosine similarity under both representations in Table 6. Again, we can see that the nearest neighbors of a same word are quite different under two representations. Neighbors based on semantic representations are semantically related, while neighbors based on usage representations are not so related but with similar specificity levels.

5 Conclusion

We propose a novel controlled response generation mechanism to handle different utterance-response relationships in terms of specificity. We introduce an explicit specificity control variable into the Seq2Seq model, which interacts with the usage representation of words to generate responses at different specificity levels. Empirical results showed that our model can generate either general or specific responses, and significantly outperform state-of-the-art generation methods.

6 Acknowledgments

This work was funded by the 973 Program of China under Grant No. 2014CB340401, the National Natural Science Foundation of China (NSFC) under Grants No. 61425016, 61472401, 61722211, and 20180290, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, and the National Key R&D Program of China under Grants No. 2016QY02D0405.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the conference on empirical methods in natural language processing*.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Anil Jain, Karthik Nandakumar, and Arun Ross. 2005. Score normalization in multimodal biometric systems. *Pattern recognition*, 38(12):2270–2285.

Michael Kearns. 2000. Cobot in lambdamoo: A social statistics agent.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Data distillation for controlling specificity in dialogue generation. *arXiv preprint arXiv:1702.06703*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Diana Perez-Marin. 2011. *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices*. IGI Global.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th annual meeting of the Association for Computational Linguistics*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HLT*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Marilyn A Walker, Rebecca Passonneau, and Julie E Boland. 2001. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 515–522. Association for Computational Linguistics.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the conference on empirical methods in natural language processing*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 55–64. ACM.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, and Kam-Fai Wong. 2016. An attentional neural conversation model with improved specificity. *arXiv preprint arXiv:1606.01292*.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *AAAI*, pages 3400–3407.

Multi-Turn Response Selection for Chatbots with Deep Attention Matching Network

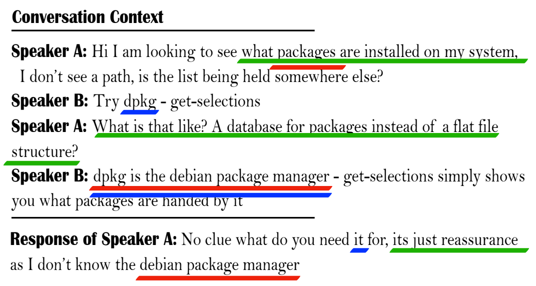
Xiangyang Zhou[*], **Lu Li**[*], **Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao**[†], **Dianhai Yu** and **Hua Wu**
Baidu Inc., Beijing, China
{zhouxiangyang, lilu12, dongdaxiang, liuyi05, chenying04, v_zhaoxin, yudianhai, wu_hua}@baidu.com

Abstract

Human generates responses relying on semantic and functional dependencies, including coreference relation, among dialogue elements and their context. In this paper, we investigate matching a response with its multi-turn context using dependency information based entirely on attention. Our solution is inspired by the recently proposed Transformer in machine translation (Vaswani et al., 2017) and we extend the attention mechanism in two ways. First, we construct representations of text segments at different granularities solely with stacked self-attention. Second, we try to extract the truly matched segment pairs with attention across the context and response. We jointly introduce those two kinds of attention in one uniform neural network. Experiments on two large-scale multi-turn response selection tasks show that our proposed model significantly outperforms the state-of-the-art models.

1 Introduction

Building a chatbot that can naturally and consistently converse with human-beings on open-domain topics draws increasing research interests in past years. One important task in chatbots is *response selection*, which aims to select the best-matched response from a set of candidates given the context of a conversation. Besides playing a critical role in *retrieval-based chatbots* (Ji et al., 2014), response selection models have been used in *automatic evaluation of dialogue generation* (Lowe et al., 2017) and the discriminator of *GAN*-based (Generative Adversarial Networks) neural dialogue generation (Li et al., 2017).



Conversation Context

Speaker A: Hi I am looking to see what packages are installed on my system, I don't see a path, is the list being held somewhere else?
Speaker B: Try dpkg - get-selections
Speaker A: What is that like? A database for packages instead of a flat file structure?
Speaker B: dpkg is the debian package manager - get-selections simply shows you what packages are handed by it

Response of Speaker A: No clue what do you need it for, its just reassurance as I don't know the debian package manager

Figure 1: Example of human conversation on Ubuntu system troubleshooting. Speaker A is seeking for a solution of package management in his/her system and speaker B recommend using, the debian package manager, dpkg. But speaker A does not know dpkg, and asks a *backchannel-question* (Stolcke et al., 2000), i.e., "no clue what do you need it for?", aiming to double-check if dpkg could solve his/her problem. Text segments with underlines in the same color across context and response can be seen as matched pairs.

Early studies on response selection only use the last utterance in context for matching a reply, which is referred to as *single-turn response selection* (Wang et al., 2013). Recent works show that the consideration of a multi-turn context can facilitate selecting the next utterance (Zhou et al., 2016; Wu et al., 2017). The reason why richer contextual information works is that human generated responses are heavily dependent on the previous dialogue segments at different granularities (words, phrases, sentences, etc), both semantically and functionally, over multiple turns rather than one turn (Lee et al., 2006; Traum and Heeman, 1996). Figure 1 illustrates semantic connectivities between segment pairs across context and response. As demonstrated, generally there are two kinds of matched segment pairs at different granularities across context and response: (1) surface text relevance, for example the lexical overlap of words "packages"-"package" and phrases "debian package manager"-"debian pack-

[*]Equally contributed.
[†] Work done as a visiting scholar at Baidu. Wayne Xin Zhao is an associate professor of Renmin University of China and can be reached at batmanfly@ruc.edu.cn.

age manager". (2) latent dependencies upon which segments are semantically/functionally related to each other. Such as the word "it" in the response, which refers to "dpkg" in the context, as well as the phrase "its just reassurance" in the response, which latently points to "what packages are installed on my system", the question that speaker A wants to double-check.

Previous studies show that capturing those matched segment pairs at different granularities across context and response is the key to multi-turn response selection (Wu et al., 2017). However, existing models only consider the textual relevance, which suffers from matching response that latently depends on previous turns. Moreover, Recurrent Neural Networks (RNN) are conveniently used for encoding texts, which is too costly to use for capturing multi-grained semantic representations (Lowe et al., 2015; Zhou et al., 2016; Wu et al., 2017). As an alternative, we propose to match a response with multi-turn context using dependency information based entirely on attention mechanism. Our solution is inspired by the recently proposed Transformer in machine translation (Vaswani et al., 2017), which addresses the issue of sequence-to-sequence generation only using attention, and we extend the key attention mechanism of Transformer in two ways:

self-attention By making a sentence attend to itself, we can capture its intra word-level dependencies. Phrases, such as "debian package manager", can be modeled with word-level self-attention over word-embeddings, and sentence-level representations can be constructed in a similar way with phrase-level self-attention. By hierarchically stacking self-attention from word embeddings, we can gradually construct semantic representations at different granularities.

cross-attention By making context and response attend to each other, we can generally capture dependencies between those latently matched segment pairs, which is able to provide complementary information to textual relevance for matching response with multi-turn context.

We jointly introduce self-attention and cross-attention in one uniform neural matching network, namely the Deep Attention Matching Network (DAM), for multi-turn response selection. In practice, DAM takes each single word of an utterance in context or response as the centric-meaning of an abstractive semantic segment, and hierarchically enriches its representation with stacked self-attention, gradually producing more and more sophisticated segment representations surrounding the centric-word. Each utterance in context and response are matched based on segment pairs at different granularities, considering both textual relevance and dependency information. In this way, DAM generally captures matching information between the context and the response from word-level to sentence-level, important matching features are then distilled with convolution & max-pooling operations, and finally fused into one single matching score via a single-layer perceptron.

We test DAM on two large-scale public multi-turn response selection datasets, the Ubuntu Corpus v1 and Douban Conversation Corpus. Experimental results show that our model significantly outperforms the state-of-the-art models, and the improvement to the best baseline model on $\mathbf{R}_{10}@1$ is over 4%. What is more, DAM is expected to be convenient to deploy in practice because most attention computation can be fully parallelized (Vaswani et al., 2017). Our contributions are two-folds: (1) we propose a new matching model for multi-turn response selection with self-attention and cross-attention. (2) empirical results show that our proposed model significantly outperforms the state-of-the-art baselines on public datasets, demonstrating the effectiveness of self-attention and cross-attention.

2 Related Work

2.1 Conversational System

To build an automatic conversational agent is a long cherished goal in Artificial Intelligence (AI) (Turing, 1950). Previous researches include *task-oriented dialogue system*, which focuses on completing tasks in vertical domain, and *chatbots*, which aims to consistently and naturally converse with human-beings on open-domain topics. Most modern chatbots are data-driven, either in a fashion of information-retrieval (Ji et al., 2014; Banchs and Li, 2012; Nio et al., 2014; Ameixa et al., 2014) or sequence-generation (Ritter et al., 2011). The retrieval-based systems enjoy the advantage of informative and fluent responses because it searches a large dialogue repository and selects

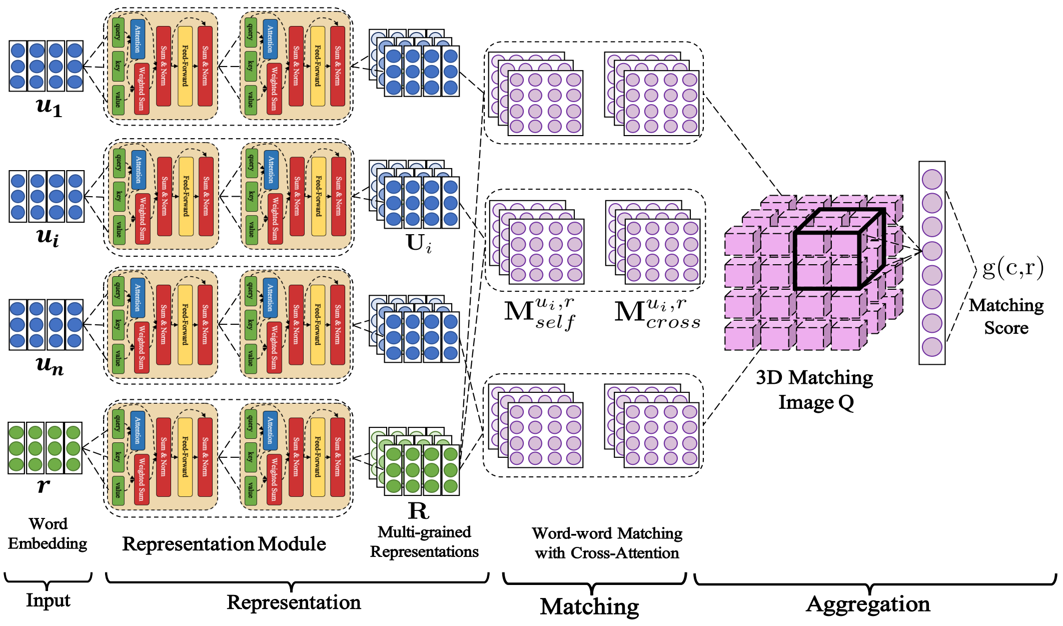


Figure 2: Overview of Deep Attention Matching Network.

candidate that best matches the current context. The generation-based models, on the other hand, learn patterns of responding from dialogues and can directly generalize new responses.

2.2 Response Selection

Researches on response selection can be generally categorized into *single-turn* and *multi-turn*. Most early studies are single-turn that only consider the last utterance for matching response (Wang et al., 2013, 2015). Recent works extend it to multi-turn conversation scenario, Lowe et al.,(2015) and Zhou et al.,(2016) use RNN to read context and response, use the last hidden states to represent context and response as two semantic vectors, and measure their relevance. Instead of only considering the last states of RNN, Wu et al.,(2017) take hidden state at each time step as a text segment representation, and measure the distance between context and response via segment-segment matching matrixes. Nevertheless, matching with dependency information is generally ignored in previous works.

2.3 Attention

Attention has been proven to be very effective in Natural Language Processing (NLP) (Bahdanau et al., 2015; Yin et al., 2016; Lin et al., 2017) and other research areas (Xu et al., 2015). Recently, Vaswani et al.,(2017) propose a novel sequence-to-sequence generation network, the Transformer, which is entirely based on attention. Not only Transformer can achieve better translation results than convenient RNN-based models, but also it is very fast in training/predicting as the computation of attention can be fully parallelized. Previous works on attention mechanism show the superior ability of attention to capture semantic dependencies, which inspires us to improve multi-turn response selection with attention mechanism.

3 Deep Attention Matching Network

3.1 Problem Formalization

Given a dialogue data set $\mathcal{D} = \{(c, r, y)_Z\}_{Z=1}^{N}$, where $c = \{u_0, ..., u_{n-1}\}$ represents a conversation context with $\{u_i\}_{i=0}^{n-1}$ as utterances and r as a response candidate. $y \in \{0, 1\}$ is a binary label, indicating whether r is a proper response for c. Our goal is to learn a matching model $g(c, r)$ with $\mathcal{D}$, which can measure the relevance between any context c and candidate response r.

3.2 Model Overview

Figure 2 gives an overview of DAM, which generally follows the **representation-matching-aggregation** framework to match response with multi-turn context. For each utterance $u_i = [w_{u_i,k}]_{k=0}^{n_{u_i}-1}$ in a context and its response candidate $r = [w_{r,t}]_{t=0}^{n_r-1}$, where n_{u_i} and n_r stand for the numbers of words, DAM first looks up a shared word embedding table and represents u_i and r as sequences of word embeddings, namely $\mathbf{U}_i^0 =$

$[e^0_{u_i,0}, ..., e^0_{u_i,n_{u_i}-1}]$ and $\mathbf{R}^0 = [e^0_{r,0}, ..., e^0_{r,n_r-1}]$ respectively, where $e \in \mathbb{R}^d$ denotes a d-dimension word embedding.

A representation module then starts to construct semantic representations at different granularities for u_i and r. Practically, L identical layers of self-attention are hierarchically stacked, each l^{th} self-attention layer takes the output of the $l-1^{th}$ layer as its input, and composites the input semantic vectors into more sophisticated representations based on self-attention. In this way, multi-grained representations of u_i and r are gradually constructed, denoted as $[\mathbf{U}^l_i]^L_{l=0}$ and $[\mathbf{R}^l]^L_{l=0}$ respectively.

Given $[\mathbf{U}^0_i, ..., \mathbf{U}^L_i]$ and $[\mathbf{R}^0, ..., \mathbf{R}^L]$, utterance u_i and response r are then matched with each other in a manner of segment-segment similarity matrix. Practically, for each granularity $l \in [0...L]$, two kinds of matching matrixes are constructed, i.e., the self-attention-match $\mathbf{M}^{u_i,r,l}_{self}$ and cross-attention-match $\mathbf{M}^{u_i,r,l}_{cross}$, measuring the relevance between utterance and response with textual information and dependency information respectively.

Those matching scores are finally merged into a 3D matching image $\mathbf{Q}$[1]. Each dimension of $\mathbf{Q}$ represents *each utterance in context*, *each word in utterance* and *each word in response* respectively. Important matching information between segment pairs across multi-turn context and candidate response is then extracted via convolution with max-pooling operations, and further fused into one matching score via a single-layer perceptron, representing the matching degree between the response candidate and the whole context.

Specifically, we use a shared component, the Attentive Module, to implement both self-attention in representation and cross-attention in matching. We will discuss in detail the implementation of Attentive Module and how we used it to implement both self-attention and cross-attention in following sections.

3.3 Attentive Module

Figure 3 shows the structure of Attentive Module, which is similar to that used in Transformer (Vaswani et al., 2017). Attentive Module has three input sentences: the query sentence, the key sentence and the value sentence, namely $\mathcal{Q} = [e_i]^{n_\mathcal{Q}-1}_{i=0}, \mathcal{K} = [e_i]^{n_\mathcal{K}-1}_{i=0}, \mathcal{V} = [e_i]^{n_\mathcal{V}-1}_{i=0}$ respectively, where $n_\mathcal{Q}$, $n_\mathcal{K}$ and $n_\mathcal{V}$ denote the number of words in each sentence and e_i stands for a d-dimension embedding, $n_\mathcal{K}$ is equal to $n_\mathcal{V}$. The Attentive Module first takes each word in the query sentence to attend to words in the key sentence via Scaled Dot-Product Attention (Vaswani et al., 2017), then applies those attention results upon the value sentence, which is defined as:

[1]We refer to it as **Q** because it is like a cube.

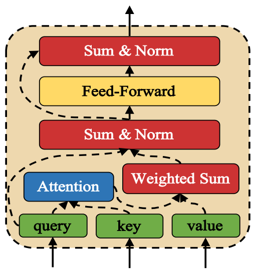


Figure 3: Attentive Module.

$$Att(\mathcal{Q}, \mathcal{K}) = \left[softmax\left(\frac{\mathcal{Q}[i] \cdot \mathcal{K}^T}{\sqrt{d}}\right)\right]^{n_\mathcal{Q}-1}_{i=0} \quad (1)$$

$$\mathcal{V}_{att} = Att(\mathcal{Q}, \mathcal{K}) \cdot \mathcal{V} \in \mathbb{R}^{n_\mathcal{Q} \times d} \quad (2)$$

where $\mathcal{Q}[i]$ is the i^{th} embedding in the query sentence $\mathcal{Q}$. Each row of $\mathcal{V}_{att}$, denoted as $\mathcal{V}_{att}[i]$, stores the fused semantic information of words in the value sentence that possibly have dependencies to the i^{th} word in query sentence. For each i, $\mathcal{V}_{att}[i]$ and $\mathcal{Q}[i]$ are then added up together, compositing them into a new representation that contains their joint meanings. A layer normalization operation (Ba et al., 2016) is then applied, which prevents vanishing or exploding of gradients. A feed-forward network **FFN** with RELU (LeCun et al., 2015) activation is then applied upon the normalization result, in order to further process the fused embeddings, defined as:

$$\mathbf{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (3)$$

where, x is a 2D-tensor in the same shape of query sentence $\mathcal{Q}$ and W_1, b_1, W_2, b_2 are learnt parameters. This kind of activation is empirically useful in other works, and we also adapt it in our model. The result $\mathbf{FFN}(x)$ is a 2D-tensor that has the same shape as x, $\mathbf{FFN}(x)$ is then residually added (He et al., 2016) to x, and the fusion result is then normalized as the final outputs. We refer to the whole Attentive Module as:

$$\mathbf{AttentiveModule}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) \quad (4)$$

As described, Attentive Module can capture dependencies across query sentence and key sentence, and further use the dependency information to composite elements in the query sentence and the value sentence into compositional representations. We exploit this property of the Attentive Module to construct multi-grained semantic representations as well as match with dependency information.

3.4 Representation

Given $\mathbf{U}_i^0$ or $\mathbf{R}^0$, the word-level embedding representations for utterance u_i or response r, DAM takes $\mathbf{U}_i^0$ ro $\mathbf{R}^0$ as inputs and hierarchically stacks the Attentive Module to construct multi-grained representations of u_i and r, which is formulated as:

$$\mathbf{U}_i^{l+1} = \textbf{AttentiveModule}(\mathbf{U}_i^l, \mathbf{U}_i^l, \mathbf{U}_i^l) \quad (5)$$

$$\mathbf{R}^{l+1} = \textbf{AttentiveModule}(\mathbf{R}^l, \mathbf{R}^l, \mathbf{R}^l) \quad (6)$$

where l ranges from 0 to $L-1$, denoting the different levels of granularity. By this means, words in each utterance or response repeatedly function together to composite more and more holistic representations, we refer to those multi-grained representations as $[\mathbf{U}_i^0, ..., \mathbf{U}_i^L]$ and $[\mathbf{R}^0, ..., \mathbf{R}^L]$ hereafter.

3.5 Utterance-Response Matching

Given $[\mathbf{U}_i^l]_{l=0}^L$ and $[\mathbf{R}^l]_{l=0}^L$, two kinds of segment-segment matching matrixes are constructed at each level of granularity l, i.e., the self-attention-match $\mathbf{M}_{self}^{u_i,r,l}$ and cross-attention-match $\mathbf{M}_{cross}^{u_i,r,l}$. $\mathbf{M}_{self}^{u_i,r,l}$ is defined as:

$$\mathbf{M}_{self}^{u_i,r,l} = \{\mathbf{U}_i^l[k]^T \cdot \mathbf{R}^l[t]\}_{n_{u_i} \times n_r} \quad (7)$$

in which, each element in the matrix is the dot-product of $\mathbf{U}_i^l[k]$ and $\mathbf{R}^l[t]$, the k^{th} embedding in $\mathbf{U}_i^l$ and the t^{th} embedding in $\mathbf{R}^l$, reflecting the textual relevance between the k^{th} segment in u_i and t^{th} segment in r at the l^{th} granularity. The cross-attention-match matrix is based on cross-attention, which is defined as:

$$\widetilde{\mathbf{U}}_i^l = \textbf{AttentiveModule}(\mathbf{U}_i^l, \mathbf{R}^l, \mathbf{R}^l) \quad (8)$$

$$\widetilde{\mathbf{R}}^l = \textbf{AttentiveModule}(\mathbf{R}^l, \mathbf{U}_i^l, \mathbf{U}_i^l) \quad (9)$$

$$\mathbf{M}_{cross}^{u_i,r,l} = \{\widetilde{\mathbf{U}}_i^l[k]^T \cdot \widetilde{\mathbf{R}}^l[t]\}_{n_{u_i} \times n_r} \quad (10)$$

where we use Attentive Module to make $\mathbf{U}_i^l$ and $\mathbf{R}^l$ crossly attend to each other, constructing two new representations for both of them, written as $\widetilde{\mathbf{U}}_i^l$ and $\widetilde{\mathbf{R}}^l$ respectively. Both $\widetilde{\mathbf{U}}_i^l$ and $\widetilde{\mathbf{R}}^l$ implicitly capture semantic structures that cross the utterance and response. In this way, those inter-dependent segment pairs are close to each other in representations, and dot-products between those latently inter-dependent pairs could get increased, providing dependency-aware matching information.

3.6 Aggregation

DAM finally aggregates all the segmental matching degrees across each utterance and response into a 3D matching image $\mathbf{Q}$, which is defined as:

$$\mathbf{Q} = \{\mathbb{Q}_{i,k,t}\}_{n \times n_{u_i} \times n_r} \quad (11)$$

where each pixel $\mathbb{Q}_{i,k,t}$ is formulated as:

$$\mathbb{Q}_{i,k,t} = [\mathbf{M}_{self}^{u_i,r,l}[k,t]]_{l=0}^L \oplus [\mathbf{M}_{cross}^{u_i,r,l}[k,t]]_{l=0}^L \quad (12)$$

$\oplus$ is concatenation operation, and each pixel has $2(L+1)$ channels, storing the matching degrees between one certain segment pair at different levels of granularity. DAM then leverages two-layered 3D convolution with max-pooling operations to distill important matching features from the whole image. The operation of 3D convolution with max-pooling is the extension of typical 2D convolution, whose filters and strides are 3D cubes[2]. We finally compute matching score $g(c,r)$ based on the extracted matching features $f_{match}(c,r)$ via a single-layer perceptron, which is formulated as:

$$g(c,r) = \sigma(W_3 f_{match}(c,r) + b_3) \quad (13)$$

where W_3 and b_3 are learnt parameters, and σ is sigmoid function that gives the probability if r is a proper candidate to c. The loss function of DAM is the negative log likelihood, defined as:

$$p(y|c,r) = g(c,r)y + (1-g(c,r))(1-y) \quad (14)$$

$$L(\cdot) = -\sum_{(c,r,y)\in\mathcal{D}} log(p(y|c,r)) \quad (15)$$

4 Experiment

[2]https://www.tensorflow.org/api_docs/python/tf/nn/conv3d

	Ubuntu Corpus				Douban Conversation Corpus					
	R$_2$@1	**R**$_{10}$@1	**R**$_{10}$@2	**R**$_{10}$@5	MAP	MRR	P@1	**R**$_{10}$@1	**R**$_{10}$@2	**R**$_{10}$@5
DualEncoder$_{lstm}$	0.901	0.638	0.784	0.949	0.485	0.527	0.320	0.187	0.343	0.720
DualEncoder$_{bilstm}$	0.895	0.630	0.780	0.944	0.479	0.514	0.313	0.184	0.330	0.716
MV-LSTM	0.906	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.904	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Multiview	0.908	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.899	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
SMN$_{dynamic}$	0.926	0.726	0.847	0.961	0.529	0.569	0.397	0.233	0.396	0.724
DAM	**0.938**	**0.767**	**0.874**	**0.969**	**0.550**	**0.601**	**0.427**	**0.254**	**0.410**	**0.757**
DAM$_{first}$	0.927	0.736	0.854	0.962	0.528	0.579	0.400	0.229	0.396	0.741
DAM$_{last}$	0.932	0.752	0.861	0.965	0.539	0.583	0.408	0.242	0.407	0.748
DAM$_{self}$	0.931	0.741	0.859	0.964	0.527	0.574	0.382	0.221	0.403	0.750
DAM$_{cross}$	0.932	0.749	0.863	0.966	0.535	0.585	0.400	0.234	0.411	0.733

Table 1: Experimental results of DAM and other comparison approaches on Ubuntu Corpus V1 and Douban Conversation Corpus.

4.1 Dataset

We test DAM on two public multi-turn response selection datasets, the Ubuntu Corpus V1 (Lowe et al., 2015) and the Douban Conversation Corpus (Wu et al., 2017). The former one contains multi-turn dialogues about Ubuntu system troubleshooting in English and the later one is crawled from a Chinese social networking on open-domain topics. The Ubuntu training set contains 0.5 million multi-turn contexts, and each context has one positive response that generated by human and one negative response which is randomly sampled. Both validation and testing sets of Ubuntu Corpus have 50k contexts, where each context is provided with one positive response and nine negative replies. The Douban corpus is constructed in a similar way to the Ubuntu Corpus, except that its validation set contains 50k instances with 1:1 positive-negative ratios and the testing set of Douban corpus is consisted of 10k instances, where each context has 10 candidate responses, collected via a tiny inverted-index system (Lucene[3]), and labels are manually annotated.

4.2 Evaluation Metric

We use the same evaluation metrics as in previous works (Wu et al., 2017). Each comparison model is asked to select k best-matched response from n available candidates for the given conversation context c, and we calculate the recall of the true positive replies among the k selected ones as the main evaluation metric, denoted as $\mathbf{R}_n@k = \frac{\sum_{i=1}^{k} y_i}{\sum_{i=1}^{n} y_i}$, where y_i is the binary label for each candidate. In addition to $\mathbf{R}_n@k$, we use MAP (Mean Average Precision) (Baeza-Yates et al., 1999), MRR (Mean Reciprocal Rank) (Voorhees et al., 1999), and Precision-at-one $P@1$ especially for Douban corpus, following the setting of previous works (Wu et al., 2017).

4.3 Comparison Methods

RNN-based models : Previous best performing models are based on RNNs, we choose representative models as baselines, including SMN$_{dynamic}$(Wu et al., 2017), Multiview(Zhou et al., 2016), DualEncoder$_{lstm}$ and DualEncoder$_{bilstm}$ (Lowe et al., 2015), DL2R (Yan et al., 2016), Match-LSTM (Wang and Jiang, 2017) and MV-LSTM (Pang et al., 2016), where SMN$_{dynamic}$ achieves the best scores against all the other published works, and we take it as our state-of-the-art baseline.

Ablation : To verify the effects of multi-grained representation, we setup two comparison models, i.e., DAM$_{first}$ and DAM$_{last}$, which dispense with the multi-grained representations in DAM, and use representation results from the 0th layer and Lth layer of self-attention instead. Moreover, we setup DAM$_{self}$ and DAM$_{cross}$, which only use self-attention-match or cross-attention-match respectively, in order to examine the effectiveness of both self-attention-match and cross-attention-match.

4.4 Model Training

We copy the reported evaluation results of all baselines for comparison. DAM is implemented in tensorflow[4], and the used vocabularies, word em-

[3]https://lucenent.apache.org/

[4]https://www.tensorflow.org. Our code and data will be available at https://github.com/baidu/Dialogue/DAM

bedding sizes for Ubuntu corpus and Douban corpus are all set as same as the SMN (Wu et al., 2017). We consider at most 9 turns and 50 words for each utterance (response) in our experiments, word embeddings are pre-trained using training sets via word2vec (Mikolov et al., 2013), similar to previous works. We use zero-pad to handle the variable-sized input and parameters in **FFN** are set to 200, same as word-embedding size. We test stacking 1-7 self-attention layers, and reported our results with 5 stacks of self-attention because it gains the best scores on validation set. The 1st convolution layer has 32 [3,3,3] filters with [1,1,1] stride, and its max-pooling size is [3,3,3] with [3,3,3] stride. The 2nd convolution layer has 16 [3,3,3] filters with [1,1,1] stride, and its max-pooling size is also [3,3,3] with [3,3,3] stride. We tune DAM and the other ablation models with adam optimizer (Le et al., 2011) to minimize loss function defined in Eq 15. Learning rate is initialized as 1e-3 and gradually decreased during training, and the batch-size is 256. We use validation sets to select the best models and report their performances on test sets.

4.5 Experiment Result

Table 1 shows the evaluation results of DAM as well as all comparison models. As demonstrated, DAM significantly outperforms other competitors on both Ubuntu Corpus and Douban Conversation Corpus, including SMN$_{dynamic}$, which is the state-of-the-art baseline, demonstrating the superior power of attention mechanism in matching response with multi-turn context. Besides, both the performances of DAM$_{first}$ and DAM$_{self}$ decrease a lot compared with DAM, which shows the effectiveness of self-attention and cross-attention. Both DAM$_{first}$ and DAM$_{last}$ underperform DAM, which demonstrates the benefits of using multi-grained representations. Also the absence of self-attention-match brings down the precision, as shown in DAM$_{cross}$, exhibiting the necessity of jointly considering textual relevance and dependency information in response selection.

One notable point is that, while DAM$_{first}$ is able to achieve close performance to SMN$_{dynamic}$, it is about 2.3 times faster than SMN$_{dynamic}$ in our implementation as it is very simple in computation. We believe that DAM$_{first}$ is more suitable to the scenario that has limitations in computation time or memories but requires high precise, such as industry application or working as an component in other neural networks like GANs.

5 Analysis

We use the Ubuntu Corpus for analyzing how self-attention and cross-attention work in DAM from both quantity analysis as well as visualization.

5.1 Quantity Analysis

We first study how DAM performs in different utterance number of context. The left part in Figure 4 shows the changes of $\mathbf{R}_{10}@1$ on Ubuntu Corpus across contexts with different number of utterance. As demonstrated, while being good at matching response with long context that has more than 4 utterances, DAM can still stably deal with short context that only has 2 turns.

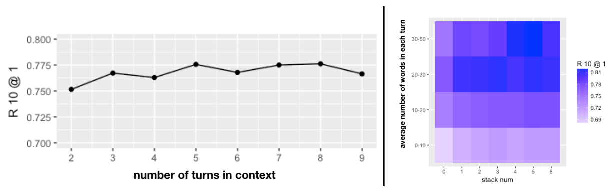


Figure 4: DAM's performance on Ubuntu Corpus across different contexts. The left part shows the performance in different utterance number of context. The right part shows performance in different average utterance text length of context as well as self-attention stack depth.

Moreover, the right part of Figure 4 gives the comparison of performance across different contexts with different average utterance text length and self-attention stack depth. As demonstrated, stacking self-attention can consistently improve matching performance for contexts having different average utterance text length, implying the stability advantage of using multi-grained semantic representations. The performance of matching short utterances, that have less than 10 words, is obviously lower than the other longer ones. This is because the shorter the utterance text is, the fewer information it contains, and the more difficult for selecting the next utterance, while stacking self-attention can still help in this case. However for long utterances like containing more than 30 words, stacking self-attention can significantly improve the matching performance, which means that the more information an utterance contains, the more stacked self-attention it needs to capture its intra semantic structures.

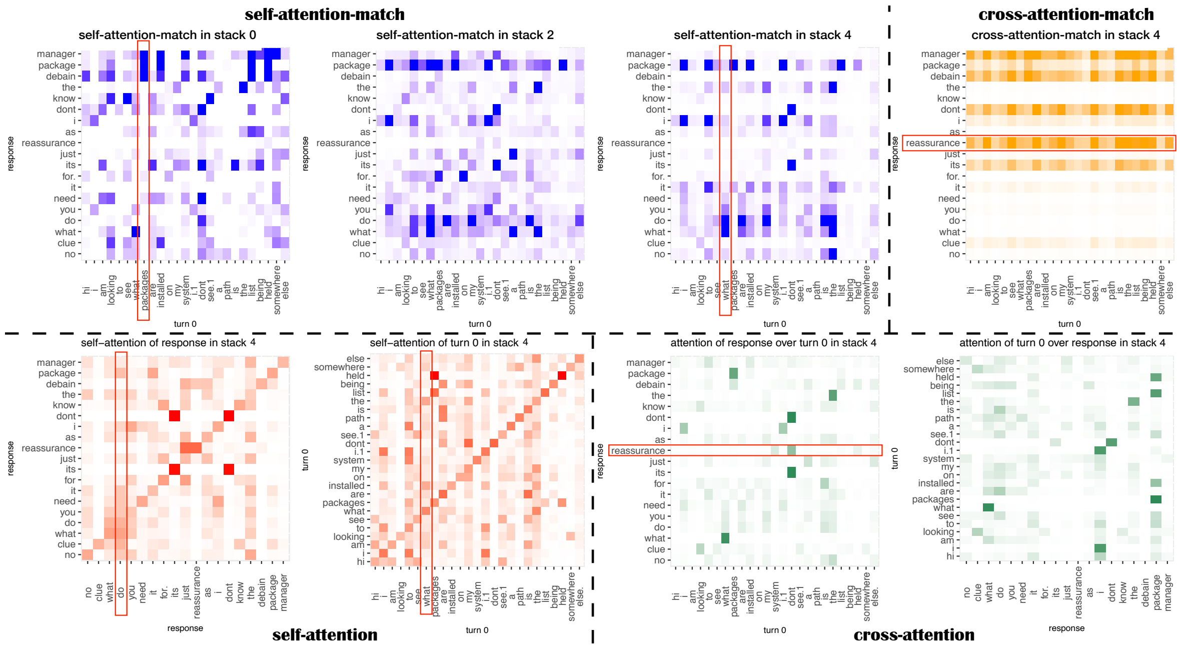


Figure 5: Visualization of self-attention-match, cross-attention-match as well as the distribution of self-attention and cross-attention in matching response with the first utterance in Figure 1. Each colored grid represents the matching degree or attention score between two words. The deeper the color is, the more important this grid is.

5.2 Visualization

We study the case in Figure 1 for analyzing in detail how self-attention and cross-attention work. Practically, we apply a softmax operation over self-attention-match and cross-attention-match, to examine the variance of dominating matching pairs during stacking self-attention or applying cross-attention. Figure 5 gives the visualization results of the 0th, 2nd and 4th self-attention-match matrixes, the 4th cross-attention-match matrix, as well as the distribution of self-attention and cross-attention in the 4th layer in matching response with the first utterance (turn 0) due to space limitation. As demonstrated, important matching pairs in *self-attention-match in stack 0* are nouns, verbs, like "package" and "packages", those are similar in topics. However matching scores between prepositions or pronouns pairs, such as "do" and "what", become more important in *self-attention-match in stack 4*. The visualization results of self-attention show the reason why matching between prepositions or pronouns matters, as demonstrated, self-attention generally capture the semantic structure of "no clue what do you need package manager" for "do" in response and "what packages are installed" for "what" in utterance, making segments surrounding "do" and "what" close to each other in representations, thus increases their dot-product results.

Also as shown in Figure 5, self-attention-match and cross-attention-match capture complementary information in matching utterance with response. Words like "reassurance" and "its" in response significantly get larger matching scores in cross-attention-match compared with self-attention-match. According to the visualization of cross-attention, "reassurance" generally depends on "system" "don't" and "held" in utterance, which makes it close to words like "list", "installed" or "held" of utterance. Scores of cross-attention-match trend to centralize on several segments, which probably means that those segments in response generally capture structure-semantic information across utterance and response, amplifying their matching scores against the others.

5.3 Error Analysis

To understand the limitations of DAM and where the future improvements might lie, we analyze 100 strong bad cases from test-set that fail in $\mathbf{R}_{10}$@5. We find two major kinds of bad cases: (1) **fuzzy-candidate**, where response candidates are basically proper for the conversation context, except for a few improper details. (2) **logical-error**, where response candidates are wrong due to logical mismatch, for example, given a conversation context **A:** "I just want to stay at home tomorrow.", **B:** "Why not go hiking? I can go with you.", response candidate like "Sure, I was planning to go out tomorrow." is logically wrong because it is contradictory to the first utterance of speaker A. We believe generating adversarial examples, rather than randomly sampling, during training procedure may be a good idea for addressing both **fuzzy-candidate** and **logical-error**, and to capture logic-level information hidden behind conversation text is also worthy to be studied in the future.

6 Conclusion

In this paper, we investigate matching a response with its multi-turn context using dependency information based entirely on attention. Our solution extends the attention mechanism of Transformer in two ways: (1) using stacked self-attention to harvest multi-grained semantic representations. (2) utilizing cross-attention to match with dependency information. Empirical results on two large-scale datasets demonstrate the effectiveness of self-attention and cross-attention in multi-turn response selection. We believe that both self-attention and cross-attention could benefit other research area, including spoken language understanding, dialogue state tracking or seq2seq dialogue generation. We would like to explore in depth how attention can help improve neural dialogue modeling for both chatbots and task-oriented dialogue systems in our future work.

Acknowledgement

We gratefully thank the anonymous reviewers for their insightful comments. This work is supported by the National Basic Research Program of China (973 program, No. 2014CB340505).

References

David Ameixa, Luisa Coheur, Pedro Fialho, and Paulo Quaresma. 2014. Luke, i am your father: dealing with out-of-domain requests by using movies subtitles. In *International Conference on Intelligent Virtual Agents*, pages 13–21. Springer.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *international conference on learning representations*.

Rafael E Banchs and Haizhou Li. 2012. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, pages 37–42. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. 2011. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 265–272.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Alan Lee, Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, and Bonnie Webber. 2006. Complexity of dependencies in discourse: Are dependencies in discourse more complex than in syntax. In *Proceedings of the 5th International Workshop on Treebanks and Linguistic Theories, Prague, Czech Republic*, page 12.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*, pages 372–381.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *international conference on learning representations*.

Ryan Lowe, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ACL*, pages 372–381.

Ryan Lowe, Nissan Pow, Iulian Vlad Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *annual meeting of the special interest group on discourse and dialogue*, pages 285–294.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lasguido Nio, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. 2014. Developing non-goal dialog system based on examples of drama television. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 355–361. Springer.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *AAAI*, pages 2793–2799.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *In Proc. EMNLP*, pages 583–593. Association for Computational Linguistics.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

David R Traum and Peter A Heeman. 1996. Utterance units in spoken dialogue. In *Workshop on Dialogue Processing in Spoken Language Systems*, pages 125–140.

Alan M Turing. 1950. Computing machinery and intelligence. *Mind*, 59(236):433–460.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, pages 77–82.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *EMNLP*, pages 935–945.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. *International Joint Conferences on Artificial Intelligence*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. *international conference on learning representations*.

Yu Wu, Wei Wu, Ming Zhou, and Zhoujun Li. 2017. Sequential match network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *ACL*, pages 372–381.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics*, 4(1):259–272.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *EMNLP*, pages 372–381.

MojiTalk: Generating Emotional Responses at Scale

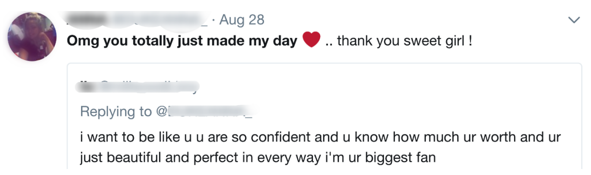
Xianda Zhou
Dept. of Computer Science and Technology
Tsinghua University
Beijing, 100084 China
zhou-xd13@mails.tsinghua.edu.cn

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
william@cs.ucsb.edu

Abstract

Generating emotional language is a key step towards building empathetic natural language processing agents. However, a major challenge for this line of research is the lack of large-scale labeled training data, and previous studies are limited to only small sets of human annotated sentiment labels. Additionally, explicitly controlling the emotion and sentiment of generated text is also difficult. In this paper, we take a more radical approach: we exploit the idea of leveraging Twitter data that are naturally labeled with emojis.

We collect a large corpus of Twitter conversations that include emojis in the response and assume the emojis convey the underlying emotions of the sentence. We investigate several conditional variational autoencoders training on these conversations, which allow us to use emojis to control the emotion of the generated text. Experimentally, we show in our quantitative and qualitative analyses that the proposed models can successfully generate high-quality abstractive conversation responses in accordance with designated emotions.



[redacted] · Aug 28
Omg you totally just made my day ❤️ .. thank you sweet girl !

[redacted]
Replying to @[redacted]
i want to be like u u are so confident and u know how much ur worth and ur just beautiful and perfect in every way i'm ur biggest fan

Figure 1: An example Twitter conversation with emoji in the response (top). We collected a large amount of these conversations, and trained a reinforced conditional variational autoencoder model to automatically generate abstractive emotional responses given any emoji.

1 Introduction

A critical research problem for artificial intelligence is to design intelligent agents that can perceive and generate human emotions. In the past decade, there has been significant progress in sentiment analysis (Pang et al., 2002, 2008; Liu, 2012) and natural language understanding—e.g., classifying the sentiment of online reviews. To build empathetic conversational agents, machines must also have the ability to learn to generate emotional sentences.

One of the major challenges is the lack of large-scale, manually labeled emotional text datasets. Due to the cost and complexity of manual annotation, most prior research studies primarily focus on small-sized labeled datasets (Pang et al., 2002; Maas et al., 2011; Socher et al., 2013), which are not ideal for training deep learning models with a large number of parameters.

In recent years, a handful of medium to large scale, emotional corpora in the area of emotion analysis (Go et al., 2016) and dialog (Li et al., 2017b) are proposed. However, all of them are limited to a traditional, small set of labels, for example, "happiness," "sadness," "anger," etc. or simply binary "positive" and "negative." Such coarse-grained classification labels make it difficult to capture the nuances of human emotion.

To avoid the cost of human annotation, we propose the use of naturally-occurring emoji-rich Twitter data. We construct a dataset using Twitter conversations with emojis in the response. The fine-grained emojis chosen by the users in the response can be seen as the natural label for the emotion of the response.

We assume that the emotions and nuances of emojis are established through the extensive usage by Twitter users. If we can create agents that are able to imitate Twitter users' language style when using those emojis, we claim that, to some extent, we have captured those emotions. Using a large collection of Twitter conversations, we then trained a conditional generative model to automatically generate the emotional responses. Figure 1 shows an example.

To generate emotional responses in dialogs, another technical challenge is to control the target emotion labels. In contrast to existing work (Huang et al., 2017) that uses information retrieval to generate emotional responses, the research question we are pursuing in this paper, is to design novel techniques that can generate abstractive responses of any given arbitrary emotions, without having human annotators to label a huge amount of training data.

To control the target emotion of the response, we investigate several encoder-decoder generation models, including a standard attention-based SEQ2SEQ model as the base model, and a more sophisticated CVAE model (Kingma and Welling, 2013; Sohn et al., 2015), as VAE is recently found convenient in dialog generation (Zhao et al., 2017).

To explicitly improve emotion expression, we then experiment with several extensions to the CVAE model, including a hybrid objective with policy gradient. The performance in emotion expression is automatically evaluated by a separate sentence-to-emoji classifier (Felbo et al., 2017). Additionally, we conducted a human evaluation to assess the quality of the generated emotional text.

Results suggest that our method is capable of generating state-of-the-art emotional text at scale. Our main contributions are three-fold:

- We provide a publicly available, large-scale dataset of Twitter conversation-pairs naturally labeled with fine-grained emojis.

- We are the first to use naturally labeled emojis for conducting large-scale emotional response generation for dialog.

- We apply several state-of-the-art generative models to train an emotional response generation system, and analysis confirms that our models deliver strong performance.

In the next section, we outline related work on sentiment analysis and emoji on Twitter data, as well as neural generative models. Then, we will introduce our new emotional research dataset and formalize the task. Next, we will describe the neural models we applied for the task. Finally, we will show automatic evaluation and human evaluation results, and some generated examples. Experiment details can be found in supplementary materials.

2 Related Work

In natural language processing, sentiment analysis (Pang et al., 2002) is an area that involves designing algorithms for understanding emotional text. Our work is aligned with some recent studies on using emoji-rich Twitter data for sentiment classification. Eisner et al. (2016) proposes a method for training emoji embedding EMOJI2VEC, and combined with word2vec (Mikolov et al., 2013), they apply the embeddings for sentiment classification. DeepMoji (Felbo et al., 2017) is closely related to our study: It makes use of a large, naturally labeled Twitter emoji dataset, and train an attentive bi-directional long short-term memory network (Hochreiter and Schmidhuber, 1997) model for sentiment analysis. Instead of building a sentiment classifier, our work focuses on generating emotional responses, given the context and the target emoji.

Our work is also in line with the recent progress of the application of Variational Autoencoder (VAE) (Kingma and Welling, 2013) in dialog generation. VAE (Kingma and Welling, 2013) encodes data in a probability distribution, and then samples from the distribution to generate examples. However, the original frameworks do not support end-to-end generation. Conditional VAE (CVAE) (Sohn et al., 2015; Larsen et al., 2015) was proposed to incorporate conditioning option in the generative process. Recent research in dialog generation shows that language generated by VAE models enjoy significantly greater diversity than traditional SEQ2SEQ models (Zhao et al., 2017), which is a preferable property toward building a true-to-life dialog agents.

In dialog research, our work aligns with recent advances in sequence-to-sequence models (Sutskever et al., 2014) using long short-term memory networks (Hochreiter and Schmidhuber, 1997). A slightly altered version of this model serves as our base model. Our modification enabled it to condition on single emojis. Li

😂	184,500	😎	9,505	😏	5,558	👊	2,771
😭	38,479	🙏	9,455	😋	5,114	😴	2,532
😊	30,447	😌	9,298	😐	5,026	😣	2,332
😉	25,018	😒	8,385	💯	4,738	😓	2,293
👍	19,832	😢	8,341	💙	4,623	🙊	1,698
😘	16,934	😔	8,293	😞	4,531	♥️	1,534
😩	17,009	💀	8,144	👏	4,287	😷	1,403
😁	15,563	💖	7,101	😑	4,205	😖	1,258
😍	15,046	😳	6,939	💪	4,066	😠	1,091
😅	14,121	😄	6,769	😫	3,973	🙅	698
💕	13,887	🙌	6,625	😡	3,841	✋	627
👀	13,741	🙈	6,558	😪	3,863	💔	423
❤️	13,147	💜	6,374	✌️	3,236	💟	250
☺️	10,927	😬	6,031	✨	3,072	🔫	243
👌	10,104	😕	5,849	💁	3,088	🎶	154
😜	9,546	😤	5,624	😈	2,969	🎧	130

Table 1: All 64 emoji labels, and number of conversations labeled by each emoji.

et al. (2016) use a reinforcement learning algorithm to improve the vanilla sequence-to-sequence model for non-task-oriented dialog systems, but their reinforced and its follow-up adversarial models (Li et al., 2017a) also do not model emotions or conditional labels. Zhao et al. (2017) recently introduced conditional VAE for dialog modeling, but neither did they model emotions in the conversations, nor explore reinforcement learning to improve results. Given a dialog history, Xie et. al.'s work recommends suitable emojis for current conversation. Xie et. al. (2016)compress the dialog history to vector representation through a hierarchical RNN and then map it to a emoji by a classifier, while in our model, the representation for original tweet, combined with the emoji embedding, is used to generate a response.

3 Dataset

We start by describing our dataset and approaches to collecting and processing the data. Social media is a natural source of conversations, and people use emojis extensively within their posts. However, not all emojis are used to express emotion and frequency of emojis are unevenly distributed. Inspired by DeepMoji (Felbo et al., 2017), we use 64 common emojis as labels (see Table 1), and collect a large corpus of Twitter conversations containing those emojis. Note that emojis with the difference only in skin tone are considered the same emoji.

Before: @amy ❤️ miss you soooo much!!! 😭 😭😭

After: ❤️ miss you soo much! 😭

Label: 😭

Figure 2: An artificial example illustrating preprocess procedure and choice of emoji label. Note that emoji occurrences in responses are counted before the deduplication process.

3.1 Data Collection

We crawled conversation pairs consisting of an original post and a response on Twitter from 12th to 14th of August, 2017. The response to a conversation must include at least one of the 64 emoji labels. Due to the limit of Twitter streaming API, tweets are filtered on the basis of words. In our case, a tweet can be reached only if at least one of the 64 emojis is used as a word, meaning it has to be a single character separated by blank space. However, this kind of tweets is arguably cleaner, as it is often the case that this emoji is used to wrap up the whole post and clusters of repeated emojis are less likely to appear in such tweets.

For both original tweets and responses, only English tweets without multimedia contents (such as URL, image or video) are allowed, since we assume that those contents are as important as the text itself for the machine to understand the conversation. If a tweet contains less than three alphabetical words, the conversation is not included in the dataset.

3.2 Emoji Labeling

Then we label responses with emojis. If there are multiple types of emoji in a response, we use the emoji with most occurrences inside the response. Among those emojis with same occurrences, we choose the least frequent one across the whole corpus, on the hypothesis that less frequent tokens better represent what the user wants to express. See Figure 2 for example.

3.3 Data Preprocessing

During preprocessing, all mentions and hashtags are removed, and punctuation[1] and emojis are separated if they are adjacent to words. Words with digits are all treated as the same special token.

In some cases, users use emojis and symbols in a cluster to express emotion extensively. To normalize the data, words with more than two repeated letters, symbol strings of more than one repeated punctuation symbols or emojis are shortened, for example, '!!!!' is shortened to '!', and 'yessss' to 'yess'. Note that we do not reduce duplicate letters completely and convert the word to the 'correct' spelling ('yes' in the example) since the length of repeated letters represents the intensity of emotion. By distinguishing 'yess' from 'yes', the emotional intensity is partially preserved in our dataset.

Then all symbols, emojis, and words are tokenized. Finally, we build a vocabulary of size 20K according to token frequency. Any tokens outside the vocabulary are replaced by the same special token.

We randomly split the corpus into 596,959 /32,600/32,600 conversation pairs for train /validation/test set[2]. Distribution of emoji labels within the corpus is presented in Table 1.

4 Generative Models

In this work, our goal is to generate emotional responses to tweets with the emotion specified by an emoji label. We assembled several generative models and trained them on our dataset.

4.1 Base: Attention-Based Sequence-to-Sequence Model

Traditional studies use deep recurrent architecture and encoder-decoder models to generate conversation responses, mapping original texts to target responses. Here we use a sequence-to-sequence (SEQ2SEQ) model (Sutskever et al., 2014) with global attention mechanism (Luong et al., 2015) as our base model (See Figure 3).

We use randomly initialized embedding vectors to represent each word. To specifically model the

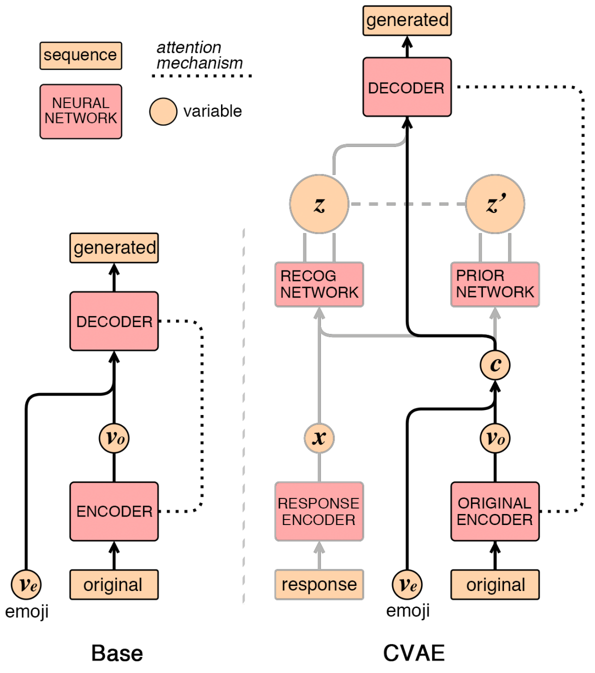


Figure 3: From bottom to top is a forward pass of data during training. **Left**: the base model encodes the original tweets in v_o, and generates responses by decoding from the concatenation of v_o and the embedded emoji, v_e. **Right**: In the CVAE model, all additional components (outlined in gray) can be added incrementally to the base model. A separate encoder encodes the responses in x. Recognition network inputs x and produces the latent variable z by reparameterization trick. During training, The latent variable z is concatenated with v_o and v_e and fed to the decoder.

emotion, we compute the embedding vector of the emoji label the same way as word embeddings. The emoji embedding is further reduced to smaller size vector v_e through a dense layer. We pass the embeddings of original tweets through a bidirectional RNN encoder of GRU cells (Schuster and Paliwal, 1997; Chung et al., 2014). The encoder outputs a vector v_o that represents the original tweet. Then v_o and v_e are concatenated and fed to a 1-layer RNN decoder of GRU cells. A response is then generated from the decoder.

4.2 Conditional Variational Autoencoder (CVAE)

Having similar encoder-decoder structures, SEQ2SEQ can be easily extended to a Conditional Variational Autoencoder (CVAE) (Sohn et al., 2015). Figure 3 illustrates the model: response encoder, recognition network, and prior network

[1]Emoticons (e.g. ':)', '(-:') are made of mostly punctuation marks. They are not examined in this paper. Common emoticons are treated as words during preprocessing.

[2]We will release the dataset with all tweets in its original form before preprocessing. To comply with Twitter's policy, we will include the tweet IDs in our release, and provide a script for downloading the tweets using the official API. No information of the tweet posters is collected.

are added on top of the SEQ2SEQ model. Response encoder has the same structure to original tweet encoder, but it has separate parameters. We use embeddings to represent Twitter responses and pass them through response encoder.

Mathematically, CVAE is trained by maximizing a variational lower bound on the conditional likelihood of x given c, according to:

$$p(x|c) = \int p(x|z,c)p(z|c)dz \tag{1}$$

z, c and x are random variables. z is the latent variable. In our case, the condition $c = [v_o; v_e]$, target x represents the response. Decoder is used to approximate $p(x|z,c)$, denoted as $p_D(x|z,c)$. Prior network is introduced to approximate $p(z|c)$, denoted as $p_P(z|c)$. Recognition network $q_R(z|x,c)$ is introduced to approximate true posterior $p(z|x,c)$ and will be absent during generation phase. By assuming that the latent variable has a multivariate Gaussian distribution with a diagonal covariance matrix, the lower bound to $\log p(x|c)$ can then be written by:

$$\begin{aligned}-\mathcal{L}(\theta_D,\theta_P,\theta_R;x,c) = \mathrm{KL}(q_R(z|x,c)||p_P(z|c))\\ -\mathbb{E}_{q_R(z|x,c)}(\log p_D(x|z,c))\end{aligned} \tag{2}$$

θ_D, θ_P, θ_R are parameters of those networks.

In recognition/prior network, we first pass the variables through an MLP to get the mean and log variance of z's distribution. Then we run a reparameterization trick (Kingma and Welling, 2013) to sample latent variables. During training, z by the recognition network is passed to the decoder and trained to approximate z' by the prior network. While during testing, the target response is absent, and z' by the prior network is passed to the decoder.

Our CVAE inherits the same attention mechanism from the base model connecting the original tweet encoder to the decoder, which makes our model deviate from previous works of CVAE on text data. Based on the attention memory as well as c and z, a response is finally generated from the decoder.

When handling text data, the VAE models that apply recurrent neural networks as the structure of their encoders/decoders may first learn to ignore the latent variable, and explain the data with the more easily optimized decoder. The latent variables lose its functionality, and the VAE deteriorates to a plain SEQ2SEQ model mathematically (Bowman et al., 2015). Some previous methods effectively alleviate this problem. Such methods are also important to keep a balance between the two items of the loss, namely KL loss and reconstruction loss. We use techniques of KL annealing, early stopping (Bowman et al., 2015) and bag-of-word loss (Zhao et al., 2017) in our models. The general loss with bag-of-word loss (see supplementary materials for details) is rewritten as:

$$\mathcal{L}' = \mathcal{L} + \mathcal{L}_{bow} \tag{3}$$

4.3 Reinforced CVAE

In order to further control the emotion of our generation more explicitly, we combine policy gradient techniques on top of the CVAE above and proposed Reinforced CVAE model for our task. We first train an emoji classifier on our dataset separately and fix its parameters thereafter. The classifier is used to produce reward for the policy training. It is a skip connected model of Bidirectional GRU-RNN layers (Felbo et al., 2017).

During the policy training, we first get the generated response x' by passing x and c through the CVAE, then feeding generation x' to classifier and get the probability of the emoji label as reward R. Let θ be parameters of our network, REINFORCE algorithm (Williams, 1992) is used to maximize the expected reward of generated responses:

$$\mathcal{J}(\theta) = \mathbb{E}_{p(x|c)}(R_\theta(x,c)) \tag{4}$$

The gradient of Equation 4 is approximated using the likelihood ratio trick (Glynn, 1990; Williams, 1992):

$$\nabla\mathcal{J}(\theta) = (R - r)\nabla\sum_{t}^{|x|}\log p(x_t|c, x_{1:t-1}) \tag{5}$$

r is the baseline value to keep estimate unbiased and reduce its variance. In our case, we directly pass x through emoji classifier and compute the probability of the emoji label as r. The model then encourages response generation that has $R > r$.

As REINFORCE objective is unrelated to response generation, it may make the generation model quickly deteriorate to some generic responses. To stabilize the training process, we propose two straightforward techniques to constrain the policy training:

1. Adjust rewards according to the position of the emoji label when all labels are ranked from high to low in order of the probability given by the emoji classifier. When the probability of the emoji label is of high rank among all possible emojis, we assume that the model has succeeded in emotion expression, thus there is no need to adjust parameters toward higher probability in this response. Modified policy gradient is written as:

$$\nabla \mathcal{J}'(\theta) = \alpha(R - r)\nabla \sum_{t}^{|x|} \log p(x_t|c, x_{1:t-1}) \tag{6}$$

   where $\alpha \in [0, 1]$ is a variant coefficient. The higher R ranks in all types of emoji label, the closer α is to 0.

2. Train Reinforced CVAE by a hybrid objective of REINFORCE and variational lower bound objective, learning towards both emotion accuracy and response appropriateness:

$$\mathbf{min}_{\theta}\mathcal{L}'' = \mathcal{L}' - \lambda\mathcal{J}' \tag{7}$$

   λ is a balancing coefficient, which is set to 1 in our experiments.

The algorithm outlining the training process of Reinforced CVAE can be found in the supplementary materials.

5 Experimental Results and Analyses

We conducted several experiments to finalize the hyper-parameters of our models (Table 2). During training, fully converged base SEQ2SEQ model is used to initialize its counterparts in CVAE models. Pretraining is vital to the success of our models since it is essentially hard for them to learn a latent variable space from total randomness. For more details, please refer to the supplementary materials.

In this section, we first report and analyze the general results of our models, including perplexity, loss and emotion accuracy. Then we take a closer look at the generation quality as well as our models' capability of expressing emotion.

5.1 General

To generally evaluate the performance of our models, we use generation perplexity and top-1/top-5 emoji accuracy on the test set. Perplexity indicates how much difficulty the model is having when generating responses. We also use top-5 emoji accuracy, since the meaning of different emojis may overlap with only a subtle difference. The machine may learn that similarity and give multiple possible labels as the answer.

Model	Perplexity	Emoji Accuracy Top1	Emoji Accuracy Top5
	Development		
Base	127.0	34.2%	57.6%
CVAE	37.1	40.7%	75.3%
Reinforced CVAE	38.1	42.2%	76.9%
	Test		
Base	130.6	33.9%	58.1%
CVAE	36.9	41.4%	75.1%
Reinforced CVAE	38.3	42.1%	77.3%

Table 2: Generation perplexity and emoji accuracy of the three models.

Note that we use the same emoji classifier for evaluation. Its accuracy (see supplementary materials) may not seem perfect, but it is the state-of-the-art emoji classifier given so many classes. Also, it's reasonable to use the same classifier in training for automated evaluation, as is in (Hu et al., 2017). We can obtain meaningful results as long as the classifier is able to capture the semantic relationship between emojis (Felbo et al., 2017).

As is shown in Table 2, CVAE significantly reduces the perplexity and increases the emoji accuracy over base model. Reinforced CVAE also adds to the emoji accuracy at the cost of a slight increase in perplexity. These results confirm that proposed methods are effective toward the generation of emotional responses.

When converged, the KL loss is 27.0/25.5 for the CVAE/Reinforced CVAE respectively, and reconstruction loss 42.2/40.0. The models achieved a balance between the two items of loss, confirming that they have successfully learned a meaningful latent variable.

5.2 Generation Diversity

SEQ2SEQ generates in a monotonous way, as several generic responses occur repeatedly, while the generation of CVAE models is of much more diversity. To showcase this disparity, we calculated the type-token ratios of unigrams/bigrams/trigrams in generated responses as

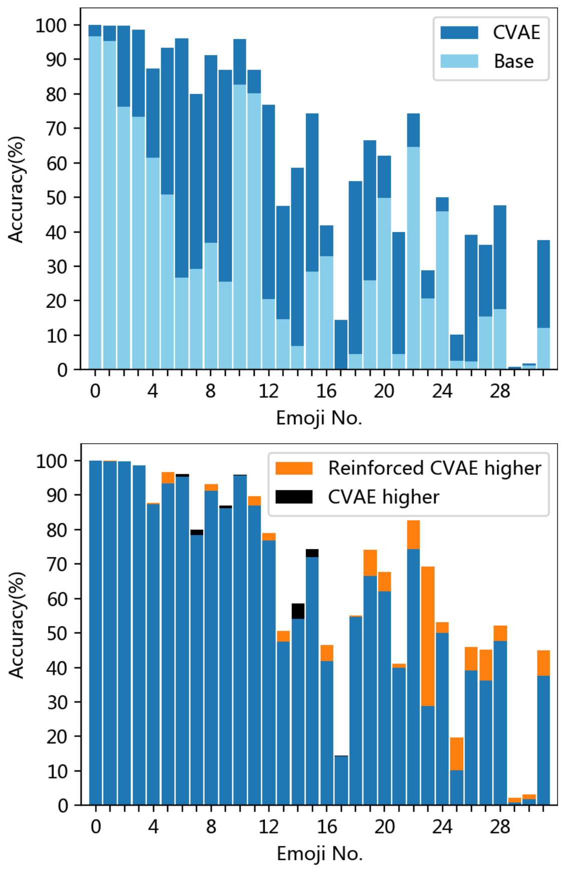


Figure 4: Top5 emoji accuracy of the first 32 emoji labels. Each bar represents an emoji and its length represents how many of all responses to the original tweets are top5 accurate. Different colors represent different models. Emojis are numbered in the order of frequencies in the dataset. No.0 is 😂, for instance, No.1 😭 and so on.
Top: CVAE v. Base.
Bottom: Reinforced CVAE v. CVAE. If Reinforced CVAE scores higher, the margin is marked in orange. If lower, in black.

the diversity score.

As shown in Table 3, results show that CVAE models beat the base models by a large margin. Diversity scores of Reinforced CVAE are reasonably compromised since it's generating more emotional responses.

5.3 Controllability of Emotions

There are potentially multiple types of emotion in reaction to an utterance. Our work makes it possible to generate a response to an arbitrary emotion by conditioning the generation on a specific type of emoji. In this section, we generate one response in reply to each original tweet in the dataset and condition on each emoji of the selected 64 emojis. We may have recorded some original tweets with different replies in the dataset, but an original tweet only need to be used once for each emoji, so we eliminate duplicate original tweets in the dataset. There are 30,299 unique original tweets in the test set.

Figure 4 shows the top-5 accuracy of each type of the first 32 emoji labels when the models generates responses from the test set conditioning on the same emoji. The results show that CVAE models increase the accuracy over every type of emoji label. Reinforced CVAE model sees a bigger increase on the less common emojis, confirming the effect of the emoji-specified policy training.

Model	Unigram	Bi-	Tri-
Base	0.0061	0.0199	0.0362
CVAE	0.0191	0.131	0.365
Reinforced CVAE	0.0160	0.118	0.337
Target responses	0.0353	0.370	0.757

Table 3: Type-token ratios of the generation by the three models. Scores of tokenized human-generated target responses are given for reference.

Setting	Model v. Base	Win	Lose	Tie
reply	CVAE	42.4%	43.0%	14.6%
reply	Reinforced CVAE	40.6%	39.6%	19.8%
emoji	CVAE	48.4%	26.2%	25.4%
emoji	Reinforced CVAE	50.0%	19.6%	30.4%

Table 4: Results of human evaluation. Tests are conducted pairwise between CVAE models and the base model.

5.4 Human Evaluation

We employed crowdsourced judges to evaluate a random sample of 100 items (Table 4), each being assigned to 5 judges on the Amazon Mechanical Turk. We present judges original tweets and generated responses. In the first setting of human evaluation, judges are asked to decide which one of the two generated responses better reply the original tweet. In the second setting, the emoji label is presented with the item discription, and judges are asked to pick one of the two generated responses that they decide better fits this emoji. (These two settings of evaluation are conducted separately so that it will not affect judges' verdicts.) Order of two generated responses under one item is permuted. Ties are permitted for an-

Content	sorry guys , was gunna stream tonight but i ’m still feeling like crap and my voice disappeared . i will make it up to you		
Target Emotion	😘	🙏	😢
Base	i ’m sorry you ’re going to be missed it	i ’m sorry for your loss	i ’m sorry you ’re going to be able to get it
CVAE	hope you are okay hun !	hi jason , i ’ll be praying for you	im sorry u better suck u off
Reinforced CVAE	hope you ’re feeling it	hope you had a speedy recovery man ! hope you feel better soon , please get well soon	dude i ’m so sorry for that i wanna hear it and i ’m sorry i can ’t go to canada with you but i wanna be away from canada
Content	add me in there my bro 🙏		
Target Emotion	😜	💪	💔
Base	i ’m not sure you ’ll be there	i ’m here for you	i ’m not ready for you
CVAE	you know , you need to tell me in your hometown !	you will be fine bro , i ’ll be in the gym for you	i can ’t wait 😔
Reinforced CVAE	you might have to get me hip hop off .	good luck bro ! this is about to be healthy	i ’m still undecided and i ’m still waiting
Content	don ’t tell me match of the day is delayed because of this shit		
Target Emotion	😂	👍	😞
Base	i ’m not even a fan of the game	i ’m not sure if you ever have any chance to talk to someone else	i ’m sorry i ’m not doubting you
CVAE	you can ’t do it bc you ’re in my mentions	see now a good point	hiya , unfortunately , it ’s not
Reinforced CVAE	oh my god i ’m saying this as long as i remember my twitter	fab mate , you ’ll enjoy the game and you ’ll get a win	it ’s the worst
Content	g i needed that laugh lmfaoo		
Target Emotion	😊	😒	😞
Base	i ’m glad you enjoyed it	i ’m not gonna lie	i ’m sorry i ’m not laughing
CVAE	good ! have a good time	i don ’t plan on that	me too . but it ’s a lot of me .
Reinforced CVAE	thank you for your tweet , you didn ’t know how much i guess	that ’s a bad idea , u gotta hit me up on my phone	i feel bad at this and i hope you can make a joke

Table 5: Some examples from our generated emotional responses. Context is the original tweet, and target emotion is specified by the emoji. Following are the responses generated by each of the three models based on the context and the target emotion.

swers. We batch five items as one assignment and insert an item with two identical outputs as the sanity check. Anyone who failed to choose “tie” for that item is considered as a careless judge and is therefore rejected from our test.

We then conducted a simplified Turing test. Each item we present judges an original tweet, its reply by a human, and its response generated from Reinforced CVAE model. We ask judges to decide which of the two given responses is written by a human. Other parts of the setting are similar to above-mentioned tests. It turned out 18% of the test subjects mistakenly chose machine-generated responses as human written, and 27% stated that they were not able to distinguish between the two responses.

In regard of the inter-rater agreement, there are four cases. The ideal situation is that all five judges choose the same answer for a item, and in the worst-case scenario, at most two judges choose the same answer. In light of this, we have counted that 32%/33%/31%/5% of all items have 5/4/3/2 judges in agreement, showing that our experiment has a reasonably reliable inter-rater agreement.

5.5 Case Study

We sampled some generated responses from all three models, and list them in Figure 5. Given an original tweet, we would like to generate responses with three different target emotions.

SEQ2SEQ only chooses to generate most frequent expressions, forming a predictable pattern for its generation (See how every sampled response by the base model starts with "I'm"). On the contrary, generation from the CVAE model is diverse, which is in line with previous quantitative analysis. However, the generated responses are sometimes too diversified and unlikely to reply to the original tweet.

Reinforced CVAE somtetimes tends to generate a lengthy response by stacking up sentences (See the responses to the first tweet when conditioning on the 'folded hands' emoji and the 'sad face' emoji). It learns to break the length limit of sequence generation during hybrid training, since the variational lower bound objective is competing with REINFORCE objective. The situation would be more serious is λ in Equation 7 is set higher. However, this phenomenon does not impair the fluency of generated sentences, as can be seen in Figure 5.

6 Conclusion and Future Work

In this paper, we investigate the possibility of using naturally annotated emoji-rich Twitter data for emotional response generation. More specifically, we collected more than half a million Twitter conversations with emoji in the response and assumed that the fine-grained emoji label chosen by the user expresses the emotion of the tweet. We applied several state-of-the-art neural models to learn a generation system that is capable of giving a response with an arbitrarily designated emotion. We performed automatic and human evaluations to understand the quality of generated responses. We trained a large scale emoji classifier and ran the classifier on the generated responses to evaluate the emotion accuracy of the generated response. We performed an Amazon Mechanical Turk experiment, by which we compared our models with a baseline sequence-to-sequence model on metrics of relevance and emotion. Experimentally, it is shown that our model is capable of generating high-quality emotional responses, without the need of laborious human annotations. Our work is a crucial step towards building intelligent dialog agents. We are also looking forward to transferring the idea of naturally-labeled emojis to task-oriented dialog and multi-turn dialog generation problems. Due to the nature of social media text, some emotions, such as fear and disgust, are underrepresented in the dataset, and the distribution of emojis is unbalanced to some extent. We will keep accumulating data and increase the ratio of underrepresented emojis, and advance toward more sophisticated abstractive generation methods.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CONLL*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop*.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *SocialNLP at EMNLP*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *EMNLP*.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Alec Go, Richa Bhayani, and Lei Huang. 2016. Sentiment140. `http://help.sentiment140.com/`.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Chieh-Yang Huang, Tristan Labetoulle, Ting-Hao Kenneth Huang, Yi-Pei Chen, Hung-Chen Chen, Vallari Srivastava, and Lun-Wei Ku. 2017. Moodswipe: A soft keyboard that suggests messages based on user-specified emotions. *EMNLP Demo*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ICLR*.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2015. Autoencoding beyond pixels using a learned similarity metric. *ICML*.

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *EMNLP*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. *EMNLP*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. *IJCNLP*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *EMNLP*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ruobing Xie, Zhiyuan Liu, Rui Yan, and Maosong Sun. 2016. Neural emoji recommendation in dialogue systems. *arXiv preprint arXiv:1612.04609*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.

Taylor's Law for Human Linguistic Sequences

Tatsuru Kobayashi[*]
Graduate School of
Information Science and Technology,
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656
Japan

Kumiko Tanaka-Ishii[†]
Research Center for
Advanced Science and Technology,
University of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo 153-8904
Japan

Abstract

Taylor's law describes the fluctuation characteristics underlying a system in which the variance of an event within a time span grows by a power law with respect to the mean. Although Taylor's law has been applied in many natural and social systems, its application for language has been scarce. This article describes a new quantification of Taylor's law in natural language and reports an analysis of over 1100 texts across 14 languages. The Taylor exponents of written natural language texts were found to exhibit almost the same value. The exponent was also compared for other language-related data, such as the child-directed speech, music, and programming language code. The results show how the Taylor exponent serves to quantify the fundamental structural complexity underlying linguistic time series. The article also shows the applicability of these findings in evaluating language models.

1 Introduction

Taylor's law characterizes how the variance of the number of events for a given time and space grows with respect to the mean, forming a power law. It is a quantification method for the clustering behavior of a system. Since the pioneering studies of this concept (Smith, 1938; Taylor, 1961), a substantial number of studies have been conducted across various domains, including ecology, life science, physics, finance, and human dynamics, as well summarized in (Eisler, Bartos, and Kertész, 2007).

[*]kobayashi@cl.rcast.u-tokyo.ac.jp
[†]kumiko@cl.rcast.u-tokyo.ac.jp

More recently, Cohen and Xu (2015) reported Taylor exponents for random sampling from various distributions, and Calif and Schmitt (2015) reported Taylor's law in wind energy data using a non-parametric regression. Those two papers also refer to research about Taylor's law in a wide range of fields.

Despite such diverse application across domains, there has been little analysis based on Taylor's law in studying natural language. The only such report, to the best of our knowledge, is Gerlach and Altmann (2014), but they measured the mean and variance by means of the vocabulary size within a document. This approach essentially differs from the original concept of Taylor analysis, which fundamentally counts the number of events, and thus the theoretical background of Taylor's law as presented in Eisler, Bartos, and Kertész (2007) cannot be applied to interpret the results.

For the work described in this article, we applied Taylor's law for texts, in a manner close to the original concept. We considered lexical fluctuation within texts, which involves the co-occurrence and burstiness of word alignment. The results can thus be interpreted according to the analytical results of Taylor's law, as described later. We found that the Taylor exponent is indeed a characteristic of texts and is universal across various kinds of texts and languages. These results are shown here for data including over 1100 single-author texts across 14 languages and large-scale newspaper data.

Moreover, we found that the Taylor exponents for other symbolic sequential data, including child-directed speech, programming language code, and music, differ from those for written natural language texts, thus distinguishing different kinds of data sources. The Taylor exponent in this sense could categorize and quantify the structural complexity of language. The Chomsky hierarchy (Chomsky, 1956) is, of course, the most important framework for such categorization. The Taylor exponent is another way to quantify the complexity of natural language: it allows for continuous quantification based on lexical fluctuation.

Since the Taylor exponent can quantify and characterize one aspect of natural language, our findings are applicable in computational linguistics to assess language models. At the end of this article, in §5, we report how the most basic character-based long short-term memory (LSTM) unit produces texts with a Taylor exponent of 0.50, equal to that of a sequence of independent and identically distributed random variables (an i.i.d. sequence). This shows how such models are limited in producing consistent co-occurrence among words, as compared with a real text. Taylor analysis thus provides a possible direction to reconsider the limitations of language models.

2 Related Work

This work can be situated as a study to quantify the complexity underlying texts. As summarized in (Tanaka-Ishii and Aihara, 2015), measures for this purpose include the entropy rate (Takahira, Tanaka-Ishii, and Lukasz, 2016; Bentz et al., 2017) and those related to the scaling behaviors of natural language. Regarding the latter, certain power laws are known to hold universally in linguistic data. The most famous among these are Zipf's law (Zipf, 1965) and Heaps' law (Heaps, 1978). Other, different kinds of power laws from Zipf's law are obtained through various methods of fluctuation analysis, but the question of how to quantify the fluctuation existing in language data has been controversial. Our work is situated as one such case of fluctuation analysis.

In real data, the occurrence timing of a particular event is often biased in a bursty, clustered manner, and fluctuation analysis quantifies the degree of this bias. Originally, this was motivated by a study of how floods of the Nile River occur in clusters (i.e., many floods coming after an initial flood) (Hurst, 1951). Such clustering phenomena have been widely reported in both natural and social domains (Eisler, Bartos, and Kertész, 2007).

Fluctuation analysis for language originates in (Ebeling and Pöeschel, 1994), which applied the approach to characters. That work corresponds to observing the average of the variances of each character's number of occurrences within a time span. Their method is strongly related to ours but different from two viewpoints: (1) Taylor analysis considers the variance with respect to the mean, rather than time; and (2) Taylor analysis does not average results over all elements. Because of these differences, the method in (Ebeling and Pöeschel, 1994) cannot distinguish real texts from an i.i.d. process when applied to word sequences (Takahashi and Tanaka-Ishii, 2018).

Event clustering phenomena cause a sequence to resemble itself in a self-similar manner. Therefore, studies of the fluctuation underlying a sequence can take another form of *long-range correlation analysis*, to consider the similarity between two subsequences underlying a time series. This approach requires a function to calculate the similarity of two sequences, and the autocorrelation function (ACF) is the main function considered. Since the ACF only applies to numerical data, both Altmann, Pierrehumbert, and Motter (2009) and Tanaka-Ishii and Bunde (2016) applied long-range correlation analysis by transforming text into intervals and showed how natural language texts are long-range correlated. Another recent work (Lin and Tegmark, 2016) proposed using mutual information instead of the ACF. Mutual information, however, cannot detect the long-range correlation underlying texts. All these works studied correlation phenomena via only a few texts and did not show any underlying universality with respect to data and language types. One reason is that analysis methods for long-range correlation are nontrivial to apply to texts.

Overall, the analysis based on Taylor's law in the present work belongs to the former approach of fluctuation analysis and shows the law's vast applicability and stability for written texts and even beyond, quantifying universal complexity underlying human linguistic sequences.

3 Measuring the Taylor Exponent

3.1 Proposed method

Given a set of elements W (words), let $X = X_1, X_2, \ldots, X_N$ be a discrete time series of length N, where $X_i \in W$ for all $i = 1, 2, \ldots, N$, i.e., each X_i represents a word. For a given segment length $\Delta t \in \mathbb{N}$ (a positive integer), a data sample X is segmented by the length Δt. The number of occurrences of a specific word $w_k \in W$ is counted for every segment, and the mean μ_k and standard deviation σ_k across segments are obtained. Doing this for all word kinds $w_1, \ldots, w_{|W|} \in W$ gives the distribution of σ with respect to μ. Following a previous work (Eisler, Bartos, and Kertész, 2007), in this article Taylor's law is defined to hold when μ and σ are correlated by a power law in the following way:

$$\sigma \propto \mu^{\alpha}. \tag{1}$$

Experimentally, the Taylor exponent α is known to take a value within the range of $0.5 \leq \alpha \leq 1.0$ across a wide variety of domains as reported in (Eisler, Bartos, and Kertész, 2007), including finance, meteorology, agriculture, and biology. Mathematically, it is analytically proven that $\alpha = 0.5$ for an i.i.d process, and the proof is included as Supplementary Material.

On the other hand, $\alpha = 1.0$ when all segments *always contain the same proportion of the elements of* W. For example, suppose that $W = \{a, b\}$. If b always occurs twice as often as a in all segments (e.g., three a and six b in one segment, two a and four b in another, etc.), then both the mean and standard deviation for b are twice those for a, so the exponent is 1.0.

In a real text, this cannot occur for all W, so $\alpha < 1.0$ for natural language text. Nevertheless, for a subset of words in W, this could happen, especially for a template-like sequence. For instance, consider a programming statement: `while (i < 1000) do i-`. Here, the words `while` and `do` always occur once, whereas `i` always occurs twice. This example shows that the exponent indicates how consistently words depend on each other in W, i.e., how words co-occur systematically in a coherent manner, thus indicating that the Taylor exponent is partly related to grammaticality.

To measure the Taylor exponent α, the mean and standard deviation are computed for every word kind[1] and then plotted in log-log coordinates. The number of points in this work was the number of different words. We fitted the points to a linear function in log-log coordinates by the least-squares method. We naturally took the logarithm of both $c\mu^{\alpha}$ and σ to estimate the exponent, because Taylor's law is a power law. The coefficient $\hat{c}$, and exponent $\hat{\alpha}$ are then estimated as the following:

$$\begin{aligned} \hat{c}, \hat{\alpha} &= \arg\min_{c,\alpha} \epsilon(c, \alpha), \\ \epsilon(c, \alpha) &= \sqrt{\frac{1}{|W|}\sum_{k=1}^{|W|}(\log \sigma_k - \log c\mu_k^{\alpha})^2}. \end{aligned}$$

This fit function could be a problem depending on the distribution of errors between the data points and the regression line. As seen later, the error distribution seems to differ with the kind of data: for a random source the error seems Gaussian, and so the above formula is relevant, whereas for real data, the distribution is biased. Changing the fit function according to the data source, however, would cause other essential problems for fair comparison. Here, because Cohen and Xu (2015) reported that most empirical works on Taylor's law used least-squares regression (including their own), this work also uses the above scheme[2], with the error defined as $\epsilon(\hat{c}, \hat{\alpha})$.

3.2 Data

Table 1 lists all the data used for this article. The data consisted of natural language texts, language-related sequences, and randomized data, listed as different blocks in the table. The natural language texts consisted of 1142 single-author long texts (first block, extracted from Project Gutenberg and Aozora Bunko across 14 languages[3], with the second block listing individual samples taken from Project Gutenberg together with the complete works of Shakespeare), and newspapers (third block, from the Gigaword corpus, available from the Linguistic Data Consortium in English, Chinese, and other major languages). Other sequences appear in the fourth block: the enwiki8 100-MB dump dataset (consisting of tag-annotated text from English Wikipedia), the 10 longest child-directed speech utterances in CHILDES data[4] (preprocessed by extracting only children's utterances), four program sources (in Lisp, Haskell, C++, and Python, crawled from

[1] In this work, words are not lemmatized, e.g. "say," "said," and "says" are all considered different words. This was chosen so in this work because the Taylor exponent considers systematic co-occurrence of words, and idiomatic phrases should thus be considered in their original forms.

[2] The code for estimating the exponent is available from https://github.com/Group-TanakaIshii/word_taylor.

[3] All texts above a size threshold (1 megabyte) were extracted from the two archives, resulting in 1142 texts.

[4] Child Language Data Exchange System (MacWhinney, 2000; Bol, 1995; Lieven, Salomo, and Tomasello, 2009; Rondal, 1985; Behrens, 2006; Gil and Tadmor, 2007; Oshima-Takane et al., 1995; Smoczynska, 1985; Anđelković, Ševa, and Moskovljević, 2001; Benedet et al., 2004; Plunkett and Strömqvist, 1992)

Table 1: Data we used in this article. For each dataset, length is the number of words, vocabulary is the number of different words. For detail of the data kind, see §3.2.

Texts	Language	$\hat{\alpha}$	Number	Length			Vocabulary		
		mean	of samples	Mean	Min	Max	Mean	Min	Max
Gutenberg	English	0.58	910	313127.4	185939	2488933	17237.7	7321	69812
	French	0.57	66	197519.3	169415	1528177	22098.3	14106	57193
	Finnish	0.55	33	197519.3	149488	396920	33597.1	26275	47263
	Chinese	0.61	32	629916.8	315099	4145117	15352.9	9153	60950
	Dutch	0.57	27	256859.2	198924	435683	19159.1	13880	31595
	German	0.59	20	236175.0	184321	331322	24242.3	11079	37228
	Italian	0.57	14	266809.0	196961	369326	29103.5	18641	45032
	Spanish	0.58	12	363837.2	219787	903051	26111.1	18111	36507
	Greek	0.58	10	159969.2	119196	243953	22805.7	15877	31386
	Latin	0.57	2	505743.5	205228	806259	59667.5	28739	90596
	Portuguese	0.56	1	261382.0	261382	261382	24719.0	24719	24719
	Hungarian	0.57	1	198303.0	198303	198303	38384.0	38384	38384
	Tagalog	0.59	1	208455.0	208455	208455	26335.0	26335	26335
Aozora	Japanese	0.59	13	616677.2	105343	2951320	19760.0	6620	49100
Moby Dick	English	0.58	1	254655.0	254655	254655	20473.0	20473	20473
Hong Lou Meng	Chinese	0.59	1	701256.0	701256	701256	18451.0	18451	18451
Les Miserables	French	0.57	1	691407.0	690417	690417	31956.0	31956	31956
Shakespeare (All)	English	0.59	1	1000238.0	1000238	1000238	40840.0	40840	40840
WSJ	English	0.56	1	22679513.0	22679513	22679513	137467.0	137467	137467
NYT	English	0.58	1	1528137194.0	1528137194	1528137194	3155495.0	3155495	3155495
People's Daily	Chinese	0.58	1	19420853.0	19420853	19420853	172140.0	172140	172140
Mainichi	Japanese	0.56	24 (yrs)	31321594.3	24483331	40270706	145534.5	127290	169270
enwiki8	tag-annotated	0.63	1	14647848.0	14647848	14647848	1430791.0	1430791	1430791
CHILDES	various	0.68	10	193434.0	48952	448772	9908.0	5619	17893
Programs	-	0.79	4	34161018.8	3697199	68622162	838907.8	127653	1545127
Music	-	0.79	12	135993.4	76629	215480	9187.9	907	27043
Moby Dick (shuffled)	-	0.50	10	254655.0	254655	254655	20473.0	20473	20473
Moby Dick (bigram)	-	0.50	10	300001.0	300001	300001	16963.8	16893	17056
3-layer stacked LSTM (character-based)	(English)	0.50	1	256425.0	256425	256425	50115.0	50115	50115
Neural MT	(English)	0.57	1	623235.0	623235	623235	27370.0	27370	27370

large representative archives, parsed, and stripped of natural language comments), and 12 pieces of musical data (long symphonies and so forth, transformed from MIDI into text with the software SMF2MML[5], with annotations removed).

As for the randomized data listed in the last block, we took the text of *Moby Dick* and generated 10 different shuffled samples and bigram-generated sequences. We also introduced LSTM-generated texts to consider the utility of our findings, as explained in §5.

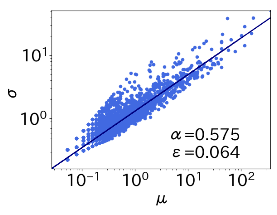
4 Taylor Exponents for Real Data

Figure 1 shows typical distributions for natural language texts, with two single-author texts ((a) and (b)) and two multiple-author texts (newspapers, (c) and (d)), in English and Chinese, respectively. The segment size was $\Delta t = 5620$ words[6], i.e., each segment had 5620 words and the horizontal axis indicates the averaged frequency of a specific word within a segment of 5620 words.

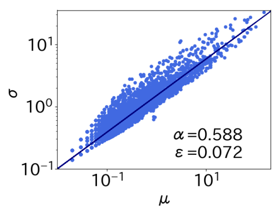
The points at the upper right represent the most frequent words, whereas those at the lower left represent the least frequent. Although the plots exhibited different distributions, they could globally be considered roughly aligned in a power-law

[5]`http://shaw.la.coocan.jp/smf2mml/`

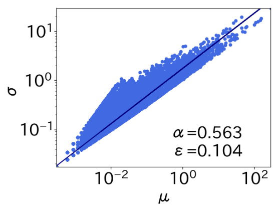
[6] In comparison, Figure 6 shows the effect on the exponent of varying Δt. As seen in that figure, larger Δt increased the differences in exponent among different data sets, making the differences more distinguishable. Thus, Δt had better be as large as possible while keeping μ and σ computable. For this article, we chose $\Delta t = 5620$, which was one of the Δt values used in Figure 6.



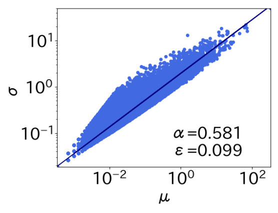
(a) Moby Dick



(b) Hong Lou Meng



(c) Wall Street Journal



(d) People's Daily

Figure 1: Examples of Taylor's law for natural language texts. *Moby Dick* and *Hong Lou Meng* are representative of single-author texts, and the two newspapers are representative of multiple-author texts, in English and Chinese, respectively. Each point represents a kind of word. The values of σ and μ for each word kind are plotted across texts within segments of size $\Delta t = 5620$. The Taylor exponents obtained by the least-squares method were all around 0.58.

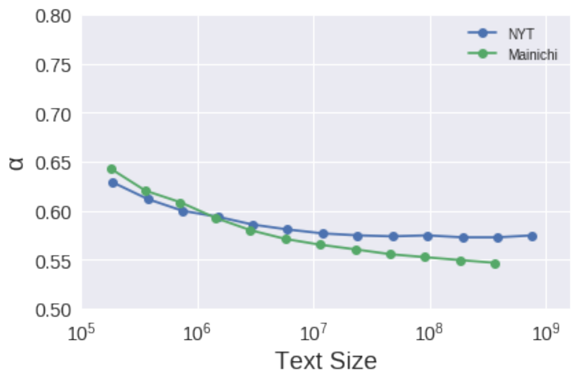


Figure 2: Taylor exponent $\hat{\alpha}$ (vertical axis) calculated for the two largest texts: The New York Times and The Mainichi newspapers. To evaluate the exponent's dependence on the text size, parts of each text were taken and the exponents were calculated for those parts, with points taken logarithmically. The window size was $\Delta t = 5620$. As the text size grew, the Taylor exponent slightly decreased.

manner. This finding is non-trivial, as seen in other analyses based on Taylor's law (Eisler, Bartos, and Kertész, 2007). The exponent α was almost the same even though English and Chinese are different languages using different kinds of script.

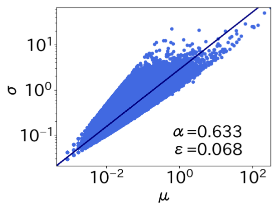
As explained in §3.1, the Taylor exponent indicates the degree of consistent co-occurrence among words. The value of 0.58 obtained here suggests that the words of natural language texts are not strongly or consistently coherent with respect to each other. Nevertheless, the value is well above 0.5, and for the real data listed in Table 1 (first to third blocks), not a single sample gave an exponent as low as 0.5.

Although the overall global tendencies in Figure 1 followed power laws, many points deviated significantly from the regression lines. The words with the greatest fluctuation were often keywords. For example, among words in *Moby Dick* with large μ, those with the largest σ included *whale*, *captain*, and *sailor*, whereas those with the smallest σ included functional words such as *to*, *that*, and *with*.

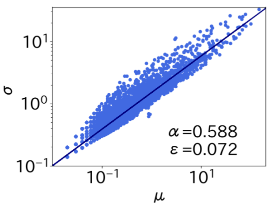
The Taylor exponent depended only slightly on the data size. Figure 2 shows this dependency for the two largest data sets used, The New York Times (NYT, 1.5 billion words) and The Mainichi (24 years) newspapers. When the data size was increased, the exponent exhibited a slight tendency to decrease. For the NYT, the decrease seemed to have a lower limit, as the figure shows that the exponent stabilized at around 10^7 words.

The reason for this decrease can be explained as follows. The Taylor exponent becomes larger when some words occur in a clustered manner. Making the text size larger increases the number of segments (since Δt was fixed in this experiment). If the number of clusters does not increase as fast as the increase in the number of segments, then the number of clusters per segment becomes smaller, leading to a smaller exponent. In other words, the influence of each consecutive co-occurrence of a particular word decays slightly as the overall text size grows.

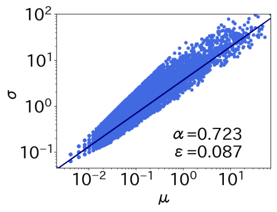
Analysis of different kinds of data showed how the Taylor exponent differed according to the data source. Figure 3 shows plots for samples from enwiki8 (tagged Wikipedia), the child-directed speech of Thomas (taken from CHILDES), programming language data sets, and music. The distributions appear different from those for the natural language texts, and the exponents were significantly larger. This means that these data sets contained expressions with fixed forms much more frequently than did the natural language texts.



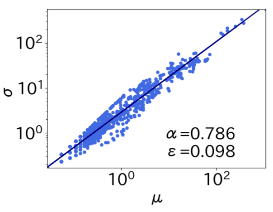
(a) enwiki8 (Wikipedia, tagged)



(b) Thomas (CHILDES)



(c) Lisp



(d) Bach's *St Matthew Passion*

Figure 3: Examples of Taylor's law for alternative data sets listed in Table 1: enwiki8 (tag-annotated Wikipedia), Thomas (longest in CHILDES), Lisp source code, and the music of Bach's *St Matthew Passion*. These examples exhibited larger Taylor exponents than did typical natural language texts.

Figure 4 summarizes the overall picture among the different data sources. The median and quantiles of the Taylor exponent were calculated for the different kinds of data listed in Table 1. The first two boxes show results with an exponent of 0.50. These results were each obtained from 10 random samples of the randomized sequences. We will return to these results in the next section.

The remaining boxes show results for real data. The exponents for texts from Project Gutenberg ranged from 0.53 to 0.68. Figure 5 shows a histogram of these texts with respect to the value of $\hat{\alpha}$. The number of texts decreased significantly at a value of 0.63, showing that the distribution of the Taylor exponent was rather tight. The kinds of texts at the upper limit of exponents for Project Gutenberg included structured texts of fixed style, such as dictionaries, lists of histories, and Bibles.

The majority of texts were in English, followed by French and then other languages, as listed in Table 1. Whether α distinguishes languages is a difficult question. The histogram suggests that Chinese texts exhibited larger values than did texts in Indo-European languages. We conducted a statistical test to evaluate whether this difference was significant as compared to English. Since the numbers of texts were very different, we used the non-parametric statistical test of the Brunner-Munzel method, among various possible methods, to test a null hypothesis of whether α was equal for the two distributions (Brunner and Munzel, 2000). The p-value for Chinese was $p = 1.24 \times 10^{-16}$, thus rejecting the null hypothesis at the significance level of 0.01. This confirms that α was generally larger for Chinese texts than for English texts. Similarly, the null hypothesis was rejected for Finnish and French, but it was accepted for German and Japanese at the 0.01 significance level. Since Japanese was accepted despite its large difference from English, we could not conclude whether the Taylor exponent distinguishes languages.

Turning to the last four columns of Figure 4, representing the enwiki8, child-directed speech (CHILDES), programming language, and music data, the Taylor exponents clearly differed from those of the natural language texts. Given the template-like nature of these four data sources, the results were somewhat expected. The kind of data thus might be distinguishable using the Taylor exponent. To confirm this, however, would require assembling a larger data set. Applying this approach with Twitter data and adult utterances would produce interesting results and remains for our future work.

The Taylor exponent also differed according to Δt, and Figure 6 shows the dependence of $\hat{\alpha}$ on Δt. For each kind of data shown in Figure 4, the mean exponent is plotted for various Δt. As reported in (Eisler, Bartos, and Kertész, 2007), the exponent is known to grow when the segment size gets larger. The reason is that words occur in a bursty, clustered manner at all length scales: no matter how large the segment size becomes, a segment will include either many or few instances of a given word, leading to larger variance growth. This phenomenon suggests how word co-occurrences in natural language are self-similar. The Taylor exponent is initially 0.5 when the segment size is very small. This can be analytically explained as follows (Eisler, Bartos, and Kertész, 2007). Consider the case of Δt=1. Let n be the frequency of a particular word in a segment. We have $\langle n \rangle \ll 1.0$, because the possibility of a specific word appearing in a segment becomes very small. Because $\langle n \rangle^2 \approx 0$, $\sigma^2 = \langle n^2 \rangle - \langle n \rangle^2 \approx \langle n^2 \rangle$. Because $n = 1$ or 0 (with Δt=1), $\langle n^2 \rangle = \langle n \rangle = \mu$. Thus, $\sigma^2 \approx \mu$.

Overall, the results show the possibility of ap-

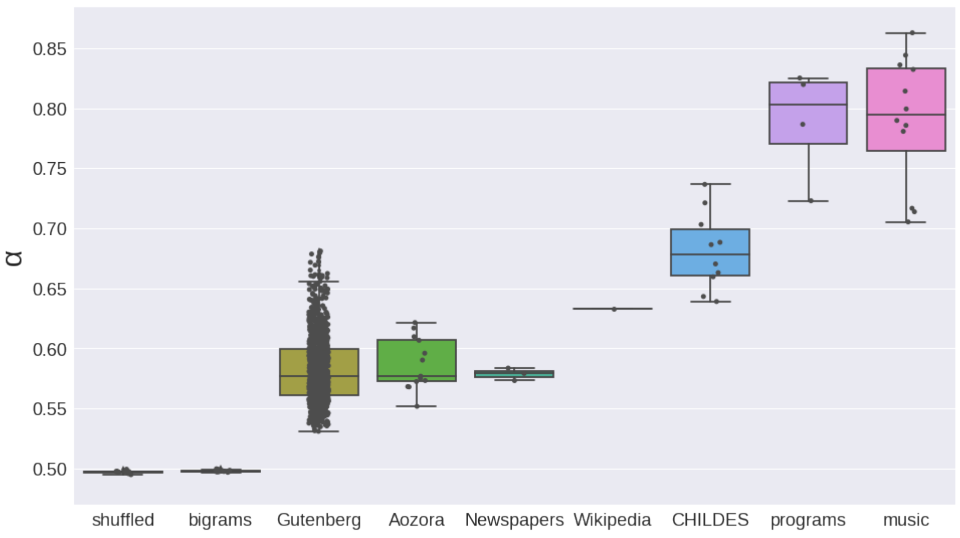


Figure 4: Box plots of the Taylor exponents for different kinds of data. Each point represents one sample, and samples from the same kind of data are contained in each box plot. The first two boxes are for the randomized data, while the remaining boxes are for real data, including both the natural language texts and language-related sequences. Each box ranges between the quantiles, with the middle line indicating the median, the whiskers showing the maximum and minimum, and some extreme values lying beyond.

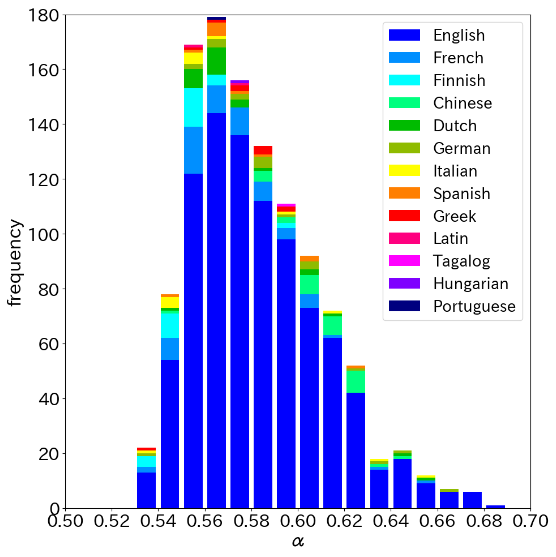


Figure 5: Histogram of Taylor exponents for long texts in Project Gutenberg (1129 texts). The legend indicates the languages, in frequency order. Each bar shows the number of texts with that value of $\hat{\alpha}$. Because of the skew of languages in the original conception of Project Gutenberg, the majority of the texts are in English, shown in blue, whereas texts in other languages are shown in other colors. The histogram shows how the Taylor exponent ranged fairly tightly around the mean, and natural language texts with an exponent larger than 0.63 were rare.

plying Taylor's exponent to quantify the complexity underlying coherence among words. Grammatical complexity was formalized by Chomsky via the Chomsky hierarchy (Chomsky, 1956), which describes grammar via rewriting rules. The constraints placed on the rules distinguish four different levels of grammar: regular, context-free, context-sensitive, and phrase structure. As indicated in (Badii and Politi, 1997), however, this does not quantify the complexity on a *continuous* scale. For example, we might want to quantify the complexity of child-directed speech as compared to that of adults, and this could be addressed in only a limited way through the Chomsky hierarchy. Another point is that the hierarchy is sentence-based and does not consider fluctuation in the kinds of words appearing.

5 Evaluation of Machine-Generated Text by the Taylor Exponent

The main contribution of this paper is the findings of Taylor's law behavior for real texts as presented thus far. This section explains the applicability of these findings, through results obtained with baseline language models.

As mentioned previously, i.i.d. mathematical processes have a Taylor exponent of 0.50. We show here that, even if a process is not trivially i.i.d., the exponent often takes a value of 0.50

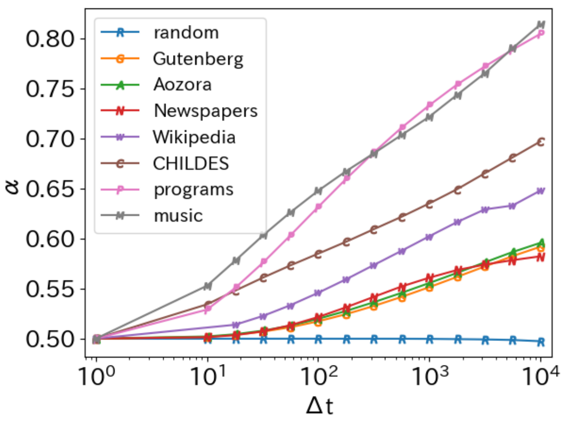
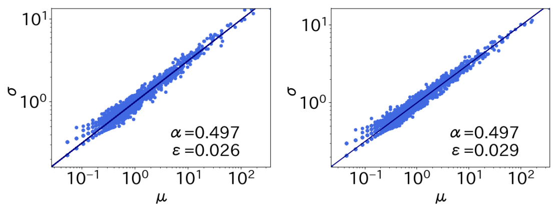


Figure 6: Growth of $\hat{\alpha}$ with respect to Δt, averaged across data sets within each data kind. The plot labeled "random" shows the average for the two datasets of randomized text from Moby Dick (shuffled and bigrams, as explained in §5). Since this analysis required a large amount of computation, for the large data sets (such as newspaper and programming language data), 4 million words were taken from each kind of data and used here. When Δt was small, the Taylor exponent was close to 0.5, as theoretically described in the main text. As Δt was increased, the value of $\hat{\alpha}$ grew. The maximum Δt was about 10,000, or about one-tenth of the length of one long literary text. For the kinds of data investigated here, $\hat{\alpha}$ grew almost linearly. The results show that, at a given Δt, the Taylor exponent has some capability to distinguish different kinds of text data.

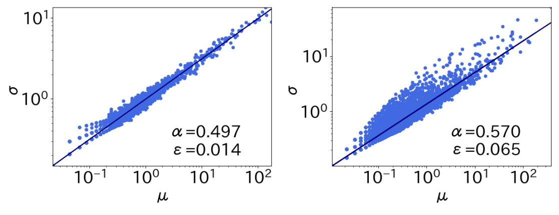


(a) Moby Dick (shuffled) (b) Moby Dick (bigram)

Figure 7: Taylor analysis of a shuffled text of *Moby Dick* and a randomized text generated by a bigram model. Both exhibited an exponent of 0.50.

for random processes, including texts produced by standard language models such as n-gram based models. A more complete work in this direction is reported in (Takahashi and Tanaka-Ishii, 2018).

Figure 7 shows samples from each of two simple random processes. Figure 7a shows the behavior of a shuffled text of *Moby Dick*. Obviously,



(a) Text produced by LSTM (3-layer stacked character-based)

(b) Machine-translated text using neural language model

Figure 8: Taylor analysis for two texts produced by standard neural language models: (a) a stacked LSTM model that learned the complete works of Shakespeare; and (b) a machine translation of *Les Misérables* (originally in French, translated into English), from a neural language model.

since the sequence was almost i.i.d. following Zipf distribution, the Taylor exponent was 0.50. Given that the Taylor exponent becomes larger for a sequence with words dependent on each other, as explained in §3, we would expect that a sequence generated by an n-gram model would exhibit an exponent larger than 0.50. The simplest such model is the bigram model, so a sequence of 300,000 words was probabilistically generated using a bigram model of *Moby Dick*. Figure 7b shows the Taylor analysis, revealing that the exponent remained 0.50.

This result does not depend much on the quality of the individual samples. The first and second box plots in Figure 4 show the distribution of exponents for 10 different samples for the shuffled and bigram-generated texts, respectively. The exponents were all around 0.50, with small variance.

State-of-the-art language models are based on neural models, and they are mainly evaluated by perplexity and in terms of the performance of individual applications. Since their architecture is complex, quality evaluation has become an issue. One possible improvement would be to use an evaluation method that qualitatively differs from judging application performance. One such method is to verify whether the properties underlying natural language hold for texts generated by language models. The Taylor exponent is one such possibility, among various properties of natural language texts.

As a step toward this approach, Figure 8 shows two results produced by neural language models. Figure 8a shows the result for a sample of 2 million characters produced by a standard (three-layer) stacked character-based LSTM unit that learned the complete works of Shakespeare. The model was optimized to minimize the cross-entropy with a stochastic gradient algorithm to predict the next character from the previous 128 characters. See (Takahashi and Tanaka-Ishii, 2017) for the details of the experimental settings. The Taylor exponent of the generated text was 0.50. This indicates that the character-level language model could not capture or reproduce the word-level clustering behavior in text. This analysis sheds light on the quality of the language model, separate from the prediction accuracy.

The application of Taylor's law for a wider range of language models appears in (Takahashi and Tanaka-Ishii, 2018). Briefly, state-of-the-art word-level language models can generate text whose Taylor exponent is larger than 0.50 but smaller than that of the dataset used for training. This indicates both the capability of modeling burstiness in text and the room for improvement. Also, the perplexity values correlate well with the Taylor exponents. Therefore, Taylor exponent can reasonably serve for evaluating machine-generated text.

In contrast to character-level neural language models, neural-network-based machine translation (NMT) models are, in fact, capable of maintaining the burstiness of the original text. Figure 8b shows the Taylor analysis for a machine-translated text of *Les Misérables* (from French to English), obtained from Google NMT (Wu et al., 2016). We split the text into 5000-character portions because of the API's limitation (See (Takahashi and Tanaka-Ishii, 2017) for the details). As is expected and desirable, the translated text retains the clustering behavior of the original text, as the Taylor exponent of 0.57 is equivalent to that of the original text.

6 Conclusion

We have proposed a method to analyze whether a natural language text follows Taylor's law, a scaling property quantifying the degree of consistent co-occurrence among words. In our method, a sequence of words is divided into given segments, and the mean and standard deviation of the frequency of every kind of word are measured. The law is considered to hold when the standard deviation varies with the mean according to a power law, thus giving the Taylor exponent.

Theoretically, an i.i.d. process has a Taylor exponent of 0.5, whereas larger exponents indicate sequences in which words co-occur systematically. Using over 1100 texts across 14 languages, we showed that written natural language texts follow Taylor's law, with the exponent distributed around 0.58. This value differed greatly from the exponents for other data sources: enwiki8 (tagged Wikipedia, 0.63), child-directed speech (CHILDES, around 0.68), and programming language and music data (around 0.79). These Taylor exponents imply that a written text is more complex than programming source code or music with regard to fluctuation of its components. None of the real data exhibited an exponent equal to 0.5. We conducted more detailed analysis varying the data size and the segment size.

Taylor's law and its exponent can also be applied to evaluate machine-generated text. We showed that a character-based LSTM language model generated text with a Taylor exponent of 0.5. This indicates one limitation of that model.

Our future work will include an analysis using other kinds of data, such as Twitter data and adult utterances, and a study of how Taylor's law relates to grammatical complexity for different sequences. Another direction will be to apply fluctuation analysis in formulating a statistical test to evaluate the structural complexity underlying a sequence.

Acknowledgments

This work was supported by JST Presto Grant Number JPMJPR14E5 and HITE funding. We thank Shuntaro Takahashi for offering his comments and providing the machine-generated data reported in §5.

References

Altmann, Eduardo G., Janet B. Pierrehumbert, and Adilson E. Motter. 2009. Beyond word frequency: Bursts, lulls, and scaling in the temporal distributions of words. *PLOS ONE*, 4(11):1–7.

Anđelković, Darinka, Nada Ševa, and Jasmina Moskovljević. 2001. *Serbian Corpus of Early Child Language*. Laboratory for Experimental Psychology, Faculty of Philosophy, and Department of General Linguistics, Faculty of Philology, University of Belgrade.

Badii, Remo and Antonio Politi. 1997. *Complexity: Hierarchical structures and scaling in physics*. Cambridge University Press.

Behrens, Heike. 2006. The input-output relationship in first language acquisition. *Language and Cognitive Processes*, 21:2–24.

Benedet, Maria, Celis Cruz, Maria Carrasco, and Catherine Snow. 2004. *Spanish BecaCESNo Corpus*. TalkBank.

Bentz, Christian, Dimitrios Alikaniotis, Michael Cysouw, and Ramon Ferrer i Cancho. 2017. The entropy of words—learnability and edxpressibvity across more than 1000 langauges. *Entropy*, (6).

Bol, Gerard W. 1995. *Implicational scaling in child language acquisition: the order of production of Dutch verb constructions*, Amsterdam Series in Child Language Development, chapter 3. Amsterdam: Institute for General Linguistics.

Brunner, Edgar and Ullrich Munzel. 2000. The nonparametric behrens-fisher problem: Asymtotic theory and a small-sample approximation. *Biometrical Journal*, 42:17–25.

Calif, Rudy and François G. Schmitt. 2015. Taylor law in wind energy data. *Resources*, 4(4):787–795.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124.

Cohen, Joel E. and Meng Xu. 2015. Random sampling of skewed distributions implies taylor's power law of fluctuation scaling. *Proceedings of the National Academy of Sciences*, 112(25):7749–7754.

Ebeling, Werner and Thorsten Pöeschel. 1994. Entropy and long-range correlations in literary english. *Europhys. Letters*, 26:241–246.

Eisler, Zoltán, Imre Bartos, and János Kertész. 2007. Fluctuation scaling in complex systems: Taylor's law and beyond. *Advances in Physics*, pages 89–142.

Gerlach, Martin and Eduardo G. Altmann. 2014. Scaling laws and fluctuations in the statistics of word frequencies. *New Journal of Physics*, 16(11):113010.

Gil, David and Uri Tadmor. 2007. *The MPI-EVA Jakarta Child Language Database*. A joint project of the Department of Linguistics, Max Planck Institute for Evolutionary Anthropology and the Center for Language and Culture Studies, Atma Jaya Catholic University.

Heaps, Harold S. 1978. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, Inc., Orlando, FL, USA.

Hurst, Harold E. 1951. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers*, 116:770–808.

Lieven, Elena, Dorothé Salomo, and Michael Tomasello. 2009. Two-year-old children's production of multiword utterances: A usage-based analysis. *Cognitive Linguistics*, 20(3):481–508.

Lin, Henry W. and Max Tegmark. 2016. Critical behavior from deep dynamics: A hidden dimension in natural language. *arXiv preprint*, abs/1606.06737.

MacWhinney, Brian. 2000. *The Childes Project*. New York: Psychology Press.

Montemurro, Marcelo A. and Pedro A. Pury. 2002. Long-range fractal correlations in literary corpora. *Fractals*, 10:451–461.

Oshima-Takane, Yuriko, Brian MacWhinney, Hidetosi Sirai, Susanne Miyata, and Norio Naka. 1995. *CHILDES manual for Japanese*. Montreal: McGill University.

Plunkett, Kim and Sven Strömqvist. 1992. The acquisition of scandinavian languages. In D. I. Slobin, editor, *The crosslinguistic study of language acquisition*, volume 3. Lawrence Erlbaum Associates, pages 457–556.

Rondal, Jean A. 1985. *Adult-child interaction and the process of language acquisition*. Praeger Publishers.

Smith, H. Fairfield. 1938. An empirical law describing hetero-geneity in the yields of agricultural crops. *Journal of Agriculture Science*, 28(1).

Smoczynska, Magdalena. 1985. The acquisition of polish. In D. I. Slobin, editor, *The crosslinguistic study of language acquisition*. Lawrence Erlbaum Associates, pages 595–686.

Takahashi, Shuntaro and Kumiko Tanaka-Ishii. 2017. Do neural nets learn statistical laws behind natural langauge? *PLoS One*. In press.

Takahashi, Shuntaro and Kumiko Tanaka-Ishii. 2018. Assesing language models with scaling properties. *arXiv preprint*, arXiv:1804.08881.

Takahira, Ryosuke, Kumiko Tanaka-Ishii, and Debowski Lukasz. 2016. Entropy rate estimates for natural language : A new extrapolation of compressed large-scale corpora. *Entropy*, 18(10).

Tanaka-Ishii, Kumiko and Shunsuke Aihara. 2015. Text constancy measures. *Computational Linguistics*, 41:481–502.

Tanaka-Ishii, Kumiko and Armin Bunde. 2016. Long-range memory in literary texts: On the universal clustering of the rare words. *PLOS One*. Online journal.

Taylor, L. Roy. 1961. Aggregation, variance and the mean. *Nature*, 732:189–190.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and et al. 2016. Google ' s neural machine translation system: Bridging the gap between human and machine translation. *arXiv*.

Zipf, George K. 1965. *Human behavior and the principle of least effort: An introduction to human ecology*. Hafner.

A Framework for Representing Language Acquisition in a Population Setting

Jordan Kodner
University of Pennsylvania
Department of Linguistics,
Dept. of Computer and Info. Science
jkodner@sas.upenn.edu

Christopher M. Cerezo Falco
University of Pennsylvania
Department. of Electrical
and Systems Engineering
ccerez@seas.upenn.edu

Abstract

Language variation and change are driven both by individuals' internal cognitive processes and by the social structures through which language propagates. A wide range of computational frameworks have been proposed to connect these drivers. We compare the strengths and weaknesses of existing approaches and propose a new analytic framework which combines previous network models' ability to capture realistic social structure with practically and more elegant computational properties. The framework privileges the process of language acquisition and embeds learners in a social network but is modular so that population structure can be combined with different acquisition models. We demonstrate two applications for the framework: a test of practical concerns that arise when modeling acquisition in a population setting and an application of the framework to recent work on phonological mergers in progress.

1 Introduction

The process of language change should be thought of as a two-step cycle in which 1) individuals acquire their native languages from their predecessors then 2) pass them on to their successors. Small changes accrue over time this way and create both small-scale interpersonal variation and large-scale typological differences. It is easy to draw a strong analogy here between linguistic evolution and biological evolution. Both feature classic descent with modification, except while phenotypes are transmitted through genes and acted on by natural selection, language is both transmitted through and constrained by the individual (Cavalli-Sforza and Feldman, 1981; Ritt, 2004, etc.).

But while evolution, linguistic or otherwise, is driven by forces acting on the individual, it unfolds on the level of populations (Cavalli-Sforza and Feldman, 1981). The influence of community-level social factors on the path of language change is a major focus of sociolinguistics (Labov, 2001; Milroy and Milroy, 1985; Rogers Everett, 1995). Ideally, one could observe population-level variation unfold in real time while testing out individual factors, but this is impossible because nobody can travel back in time or fit entire natural environments into a lab. Change that has already happened is out of reach, and change in progress is buried in a world of confounds. The classic sociolinguistic method instead approaches the problem by inferring causal factors from patterns discovered in field interviews and corpora (Labov, 1994; Labov et al., 2005, etc.). This is the primary source of empirical data in the field and the only way to look at language change in a naturalistic setting, but it is limited in that it cannot test cause and effect directly. More recently, controlled experimental studies have emerged as a complementary line of research which manipulate causal factors directly (Johnson et al., 1999; Campbell-Kibler, 2009, etc.), but are inherently removed natural time and scale. A third approach, the one we build upon here, relies on computational modeling to simulate how sociolinguistic factors might work together in larger populations (Klein, 1966; Blythe and Croft, 2012; Kauhanen, 2016, etc.).

It has long been argued that language acquisition is the primary cause of language change (Sweet, 1899; Lightfoot, 1979; Niyogi, 1998, etc.). In the last few decades, this connection has been modeled computationally (Gibson and Wexler, 1994; Kirby et al., 2000; Yang, 2000,

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1149–1159
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

etc.), leading to the strong conclusion that change is the inevitable consequence of mixed linguistic input or finite learning periods (Niyogi and Berwick, 1996), even if children are "perfect" learners. An important result connecting the learner and population emphasizes the need for this line of work: the space of paths of change available in populations is formally larger than the paths available to linear chains of iterated learners. Niyogi and Berwick (2009) prove formally that even *perfectly-mixed* (i.e., uniform and homogeneous social network) populations admit phase transitions in the path of change unavailable to chains of single learners commonly implemented in iterated learning (Kirby et al., 2000). This suggests that small-population experimental studies in sociolinguistics and in child language acquisition do not paint the full picture of language change.

We introduce a new framework for modeling language change in populations. It has an outer loop to represent generational progression, but it replaces the inner loop which calculates randomized interactions between agents with a single formula that is defined generally enough to allow the simulation of a wide range of scenarios. It builds upon the principled formalism described by Niyogi and Berwick (1996, et seq.), privileging the acquisition model and separating it from the population model. The resulting modular framework is described in the following sections. First, Section 1.1 presents a survey of previous simulation work followed by a description of the new population model in Section 2. Next, Section 3 addresses practical concerns relating population size to assumptions about language acquisition. Finally, Section 4 introduces a case study on phonological change which demonstrates the need for appropriate models both of acquisition and populations.

1.1 Related Work

Computational models for the propagation of linguistic variation have been employed with a variety of research goals in mind. Every paper implements its own framework with few exceptions, so comparison across studies is difficult. Additionally, since each model is essentially 'boutique,' it is always possible that models are designed consciously or unconsciously to achieve a specific outcome rather driven by underlying principles. We group these frameworks into three classes according to their implementation, *swarm*, *network*, and *algebraic*, and discusses their strengths and weaknesses.

The first class, called *swarm* here, models populations as collections of agents placed on a grid. They "swarm" around randomly according to some movement function, and "interact" when they occupy adjacent grid spaces (Satterfield, 2001; Harrison et al., 2002; Ke et al., 2008; Stanford and Kenny, 2013). This tends toward concrete interpretation, for example, more mobile populations are expressed directly by more mobile agents. They capture Bloomfield (1933)'s "principle of density" which describes the observation that geographically or socially close individuals interact more frequently than those farther away. On the other hand, they provide little control over network structure, relying on series of explicit movement constraints in order to direct their agents, and since each one moves randomly at each iteration, these models have potentially thousands of degrees of freedom. Such simulations should be run many times if any sort of statistically expected results are to be computed.

The second class, *network* frameworks, model speakers as nodes and interaction probabilities as weighted edges on network graphs (Minett and Wang, 2008; Baxter et al., 2009; Fagyal et al., 2010; Blythe and Croft, 2012; Kauhanen, 2016). These frameworks offer precise control over social network structure and can test specific community models from within sociolinguistics. However, implementations usually proceed by some kind of iterative probabilistic node-pair selection process, and in this way suffer from the same statistical pitfalls as swarm frameworks. In contrast to swarm models, interaction is rigidly restricted to immediately connected nodes, so to achieve gradient interaction probabilities, edges must be frequently updated or nearly fully-connected graphs with carefully assigned edge weights would need to be constructed and motivated.

The third class, *algebraic* frameworks, present analytic methods for determining the state of the network at the end of each iteration rather than relying on stochastic simulation of individual agents (Niyogi and Berwick, 1996, 1997; Yang, 2000; Baxter et al., 2006; Minett and Wang, 2008; Niyogi and Berwick, 2009). Removing that inner loop is a more mathematically elegant approach and avoids dealing unnecessarily with statistics behind random trials. Removing that loop speeds up calculation as well, making larger simulations more tractable than with network or swarm frameworks. But this power is achieved by sacrificing the social network. Up to this point, such models have, to our knowledge, only been defined over perfectly-mixed (i.e., no network effects) populations. That assumption is useful for reasoning about the mathematical theory behind language change, but it hinders such models' utility in empirical studies. For example, though Baxter et al. (2006) and Minett and Wang (2008) implement algebraic models for perfectly mixed populations, they fall back on network models to model network effects.

2 Framework for Transmission in Social Networks

Algebraic frameworks have their mathematical advantage, but network frameworks provide a richer model for representing real-world population structures and swarm models capture density effects by default. An ideal framework would combine the benefits of all three of these. Here we do just that. We introduce a framework that instantiates Niyogi and Berwick (1996)'s acquisition-driven formalism where change is handled explicitly as a two-step alternation between individual learners learning and populations interacting. It provides an analytic solution to the state of a network structure over which swarm-like behavior can be modeled.

We begin by conceptualizing the framework in terms of agents traveling probabilistically over a network structure as in Algo. 1 before introducing the analytic solution. There is an individual standing at every node in the graph, and at every iteration, each individual begins at some location and travels along the network's edges, at each step deciding to continue on or to stop and interact with the agent at that node. Any two agents with a non-zero weight path between them could potentially interact, so the overall probability of an interaction is a function of the shape of the network and the decay rate of the step probability. The shorter and higher weighted the path between two agents, the more likely they are to interact. This corresponds to the gradient interaction probabilities of swarm frameworks.

Algorithm 1: One iteration of the propagation model conceptualized on the level of an individual agent

```
for each individual node do
    Begin traveling;
    while traveling do
        Randomly select an outgoing edge by
          weight and follow it OR stop travel;
        increase chance of stopping next time;
    end
    Interact with the individual at the current
      node;
end
```

2.1 Representing the Network

Social networks are typically conceived of as graph structures with individuals as vertices and the social or geographical connections between individuals as edges, and this allows for a great deal of flexibility. If edges are undirected, then all interactions are equal and bidirectional, but if edges are directed, interactions may or may not be. Edges can be weighted to represent likelihood of interaction or some measure of social valuation, and this too can vary over time. Lastly, it is possible to add and remove nodes themselves to capture births, deaths, or migration.

The network structure is represented computationally here as an adjacency matrix $\mathbf{A}$. In a population of n individuals, this is $n \times n$ where each element a_{ij} is the weight of the connection from individual j to individual i. The matrix must be column stochastic (all columns sum to 1 and contain only positive elements) so that edge weights can be interpreted as probabilities. The special case where the matrix is symmetric (every $a_{ij} = a_{ji}$) models undirected edges, and more strongly, the model reduces to perfectly-mixed populations when each $a_{ij} = \frac{1}{n}$.

We define a notion of *communities* over the nodes of the network in order to add the option to categorize groups of individuals. Membership among c communities is identified with an $n \times c$ indicator matrix $\mathbf{C}$. Depending on the problem at hand, it is possible to calculate the average behavior of the learners within each community directly without having to calculate the behavior of each individual member.

2.2 Propagation in the Network

In a typical network model, the edge weights between nodes in $\mathbf{A}$ are interpreted directly as interaction probabilities, meaning that individuals only ever interact with their immediate graph neighbors. We take a different approach by allowing the agents to "travel" and potentially interact with any other agent whose node is connected by a path of non-zero edges. If the number of traveling steps were fixed at k, the probability of each pair interacting would be defined as $\mathbf{A}^k$. It is more complicated for us since the number of steps traveled is a random variable. The probability of j interacting with i ($p(ij)$) is the probability of them interacting after k steps times the probability of k for all values of k as in Eqn. 1. Combining this intuition with $\mathbf{A}$ yields the interaction probabilities for all i, j pairs.

$$p(ij) = \sum_k p(ij|k \text{ steps})\, p(k \text{ steps}) \tag{1}$$

The pattern of linguistic variants or grammars (in the formal sense where grammar g is the intensional equivalent of language L_g) within a network unfolds as a dynamical system over the course of many iterations, and learners' positions within the network mediate which ones they eventually acquire. In a system with g grammars and n individuals, a $n \times g$ row-stochastic matrix $\mathbf{G}$ specifies the probability with which each community expresses each grammar. Given this notion of interaction and the specification of grammars expressed within a network, it is possible to compute the distribution of grammars presented to each learner. This is the learners' linguistic environment and is represented by a matrix $\mathbf{E}$ in the same form as $\mathbf{G}^\top$.

An *environment function* $\mathcal{E}_n(\mathbf{G}_t, \mathbf{A}) = \mathbf{E}_{t+1}$ shown in Eqn. 2 calculates $\mathbf{E}$ by first calculating all the interaction probabilities in the network then multiplying those by the grammars which every agent expresses to get the environment $\mathbf{E}$. The α parameter from the geometric distribution[1] defines the travel decay rate. A lower α defines conceptually more mobile agents.

More generally, $\mathcal{E}_n$ is a special case of $\mathcal{E}(\mathbf{G}_t, \mathbf{C}_t, \mathbf{A}_t) = \mathbf{E}_{t+1}$ where the number of communities equals the number of individuals ($c = n$). $\mathbf{C}$ becomes the identity matrix without loss of generality, so the network's initial condition does not have to be defined explicitly. For any other community definition, an initial condition has to be defined as in Eqn. 3 which specifies the starting point in the network that each agent conceptually begins traveling from. The output of $\mathcal{E}$ is a $g \times c$ matrix giving the environment of the average agent in each community.[2]

$$\mathcal{E}_n(\mathbf{G}_t, \mathbf{A}) = \mathbf{G}_t^\top \alpha\, (\mathbf{I} - (1-\alpha)\mathbf{A})^{-1} \tag{2}$$
$$\mathcal{E}(\mathbf{G}_t, \mathbf{C}, \mathbf{A}) = \mathcal{E}_n(\mathbf{G}_t, \mathbf{A})\mathbf{C}(\mathbf{C}^\top\mathbf{C})^{-1} \tag{3}$$

The output of $\mathcal{E}$ must be broadcast to $g \times n$, which would result in the loss of some information unless the assumption can be made that each community is internally uniform. However, when that assumption can be made, the $n \times n$ adjacency matrix admits a $c \times c$ equitable partition $\mathbf{A}^\pi$ (Eqn. 4) (Schaub et al., 2016) which permits an alternate environment function $\mathcal{E}_{EP}(\mathbf{G}_t, \mathbf{C}, \mathbf{A})$ shown in Eqn. 5 that is equivalent to the lossless $\mathcal{E}_n$ if $\mathbf{A}$. If $n \gg c$, $\mathcal{E}_{EP}$ is much faster to calculate because it only inverts a small $c \times c$ matrix rather than a large $n \times n$. This makes it feasible to run much larger simulations than what has been done in the past.

$$\mathbf{A}^\pi = (\mathbf{C}^\top\mathbf{C})^{-1}\mathbf{C}^\top\mathbf{A}\mathbf{C} \tag{4}$$
$$\mathcal{E}_{EP} = \alpha\mathbf{G}^\top\mathbf{C}\,(\mathbf{I} - (1-\alpha)\mathbf{A}^\pi)^{-1}\,(\mathbf{C}^\top\mathbf{C})^{-1} \tag{5}$$

2.3 Learning in the Network

The environment function describes what inputs $\mathbf{E}_{t+1}$ are available to learners given the language expressed by the mature speakers of the previous age cohort with grammars $\mathbf{G_t}$. The second component of the framework describes the learning algorithm $\mathcal{A}(\mathbf{E}_{t+1}) = \mathbf{G}_{t+1}$, how individuals respond to their input environment. The resulting $\mathbf{G}_{t+1}$ describes which grammars those learners will eventually contribute to the subsequent generation's environment $\mathbf{E}_{t+2}$. This back-and-forth between adults' grammars $\mathbf{G}$ and childrens' environment $\mathbf{E}$ is the two-step cycle of language change (Fig. 1).

In *neutral change*, learners would acquire grammars at the rates that they are expressed in their environments, but there is good reason to believe

[1] In this paper, jump probabilities decay according to a geometric distribution, but other distributions including the Poisson have been implemented as well.

[2] $(\mathbf{I} - (1-\alpha)\mathbf{A})^{-1}$ and $\mathbf{C}(\mathbf{C}^\top\mathbf{C})^{-1}$ can be pre-computed if network structure does not change over time.

$$\ldots \mathbf{G_t} \rightarrow \mathbf{E_{t+1}} \rightarrow \mathbf{G_{t+1}} \ldots \mathbf{G_{t+i}} \rightarrow \mathbf{E_{t+i+1}} \ldots$$

Figure 1: Language change as an alternation between **G** and **E** matrices

that most language change involves differential fitness between competing variants, and most non-trivial learning algorithms yield some kind of fitness (Kroch, 1989; Yang, 2000; Blythe and Croft, 2012, etc.), so $\mathcal{A}$ is rarely neutral. A neutral and simple advantaged model are both considered in Section 3, and a more complex learning algorithm is described for Section 4.

3 Application: Testing Assumptions

The general nature of the framework described here renders it suitable for reproducing the results of previous works and evaluating their assumptions. To demonstrate this, we reproduce the major result from Kauhanen (2016), which tested the behavior of neutral change in networks of single-grammar learners, in order to dissect two of its primary assumptions. Implemented in a typical network framework, the original setup contains $n = 200$ individuals in probabilistically generated centralized networks in which individuals mature categorically to the single most frequent grammar in their input. The author found that categorical neutral change produced chaotic paths of change regardless of network shape and that periodically "rewiring" some of the network edges smoothed this out. Without commenting on rewiring, we find that the combination of n and choice of categorical learners conspire to create the chaotic results.

We create two communities, both centralized along the lines of the single cluster in Kauhanen (2016), initialize all members of cluster 1 with grammar g_1 and all members of cluster 2 with grammar g_2, and additional edges are added between members of clusters 1 and 2 to allow interaction. **G** is converted to an indicator matrix at the end of each learning iteration by rounding values to 0 and 1 in order to model categorical learners who only internalize the most common grammar in their inputs as in the original model.

In a pair of infinitely large clusters or two clusters where individuals are permitted to learn a probabilistic distribution of grammars, each cluster should homogenize to a 50/50 distribution of g_1 and g_2 after some number of iterations depending on the specifics of the network shape and setting for α creating the red curves in Fig. 2. At $n = 20000$, each of 10 trials roughly follows the path of the predicted curve, but when run at the original $n = 200$ for 10 trials, this produces the type of chaotic behavior which Kauhanen (2016) attempts to repair. The outcome appears to be the result of an assumption made out of convenience ($n = 200$) rather than a principled decision.

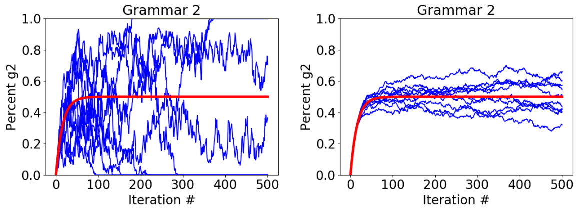


Figure 2: Predicted curve (red); neutral change at $n = 200$ (left; Kauhanen (2016)); neutral change at $n = 20000$ (right)

To further explore the impact of the population size assumption, we experiment on a model of advantaged change, which is typically contrasted with neutral change because of its tendency to produce "well-behaved" S-curve change (Blythe and Croft, 2012; Kauhanen, 2016). This time, only a single cluster is created, and the advantaged grammar is initially assigned to 1% of the population. As seen in Figure 3, results are chaotic for $n = 200$ once again and near predicted for $n = 20000$. This is important because at $n = 200$, advantaged change is chaotic, and most simulations both rise and fall. An experimenter who only studied advantaged change in small population might concluded that it is as ill-behaved as neutral change. While the conclusions that Kauhanen (2016) draws appear valid for $n = 200$, it is not clear to what extent they can be projected onto larger populations. This demonstrates the need for carefully choosing one's modeling assumptions and testing them out when possible.

4 Application: Mergers in Progress

The acquisition of phonological mergers in mixed input settings presents an interesting problem. It appears that mergers have an inherent advantage because they tend to spread at the expense of distinctions, and once they begin, they are rarely reversed (Labov, 1994). Yang (2009)'s acquisition model quantifies this advantage as the relatively

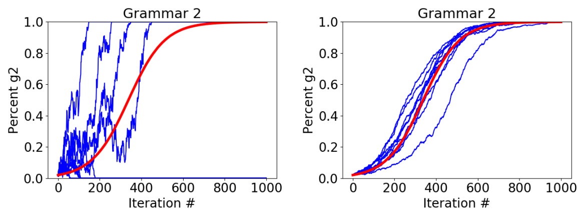


Figure 3: predicted curve (red); advantaged change at $n = 200$ (left; cf. Kauhanen (2016)); advantaged change at $n = 20000$ (right)

lower chance of misinterpretation if a listener assumes the merged grammar instead of the non-merged grammar once a sufficient proportion of the environment is merged. Applied to Johnson (2007)'s detailed population study of the frontier of the COT-CAUGHT merger in the small towns along the border between Rhode Island and Massachusetts, this accurately predicts the ratio of merged input for a child to acquire the merged grammar, however when applied to a perfectly mixed population of learners, it fails to model the spread of the merged grammar in the population. Yang's model is input-driven, so it is conducive to simulation with minimal assumptions past those drawn from the empirical data. We test the behavior of this learning model in a typical population network and demonstrate that it produces a reasonable path of change.

4.1 Background

The COT-CAUGHT *merger*, also called the *low back merger* describes the phenomenon present in varieties of North American English spoken in eastern New England, western Pennsylvania, the American West, and Canada among others where the vowel in words like *cot* and the vowel in words like *caught* have come to be pronounced the same (Labov et al., 2005, pp. 58-65). The geographical extent of the merger is currently expanding, which might be expected if the merger has a cognitive or social advantage associated with it. Johnson (2007)'s study of the merger's frontier on the border Rhode Island and Massachusetts uncovered an interesting social dynamic that illustrates the merger's speed: there are families where the parents and older siblings non-merged, but the younger siblings are. The merger has swept through in only a few years and passed between the siblings.

Yang (2009) seeks to understand why mergers have an advantage from a cognitive perspective, and his model treats the acquisition of mergers as an evolutionary process. Learners who receive both merged (M_+) and non-merged (M_-) input entertain both a merged (g_+) and non-merged (g_-) grammar and reward whichever grammar successfully parses the input. This kind of variational learner (Yang, 2000) is essentially an adaptation of the classic evolutionary Linear Reward Punishment model (Bush and Mosteller, 1953). The fitness of each grammar is the probability in the limit that it will fail to parse any given input, and since it is virtually always the case that this probability is different for both grammars, fitness is virtually always asymmetric. The variational learner is characterized as follows.

Given two grammars and an input token s, The learner parses s with g_1 with probability p and with g_2 with probability $q = 1 - p$. p is rewarded according to whether the choice of g successfully parses s ($g \to s$) or it fails to ($g \nrightarrow s$), where γ is some small constant.

$$p' = \begin{cases} p + \gamma q, & g \to s \\ (1-\gamma)p, & g \nrightarrow s \end{cases}$$

Given a specific problem, one can calculate a *penalty probability* C for each g, the proportion of input that would cause $g \nrightarrow s$. The grammar with the lower C has the advantage, so the other one will be driven down in the long run. C can be estimated from type frequencies in a corpus, and the model is non-parametric because these values do not depend on γ.

$$\lim_{t\to\infty} p_t = \frac{C_2}{C_1 + C_2} \qquad \lim_{t\to\infty} q_t = \frac{C_1}{C_1 + C_2}$$

To understand the COT-CAUGHT merger empirically, one must reason about what kind of input would trigger a penalty and then calculate the penalty probabilities of the merged grammar C_+ and non-merged grammar C_- from a corpus. This model considers parsing failure to be the rate of initial misinterpretation, and for a vowel merger, the only inputs that could create an initial misinterpretation are minimal pairs because they become homophones. Examples of COT-CAUGHT minimal pairs include *cot-caught*, *Don-Dawn*, *stock-stalk*, *odd-awed*, *collar-caller*, and so on.

The merged g_+ grammar collapses would-be minimal pairs into homophones, so the penalty rate C_+ comes down to lexical access. Under the observation that more frequent homophones are retrieved first regardless of syntactic context (Caramazza et al., 2001), g_+ listeners only suffer initial misinterpretation when the less frequent member of a pair is uttered regardless of the rate of M_+. If H is the sum token frequency of all minimal pairs and h^i_o, h^i_{oh} are the frequencies of the ith pair's members, then $C+$ is calculated by Eqn. 6.

In contrast, g_- listeners are sensitive to the phonemic distinction, so they misinterpret M_- input at the rate of mishearing one vowel for the other ϵ (Peterson and Barney, 1952) (second half of Eqn. 7). And given M_+ input, they misinterpret whenever they hear the phoneme which g_- does not expect (e.g., a merged speaker pronouncing *cot* with the CAUGHT vowel) times the probability of *not* mishearing that vowel (1-ϵ) plus ϵ times the probability of hearing the right vowel (i.e., the merged speaker pronounces *cot* with the COT vowel but it is misheard anyway) (first half of Eqn. 7). Since g_- misinterpretation rates are a function of the rate of M_+ (p) in the environment, there is a threshold of M_+ speakers above which the merged grammar has a fitness advantage over the non-merged one.

$$C_+ = \frac{1}{H}\sum_i \min(h^i_o, h^i_{oh}) \tag{6}$$

$$C_- = \frac{1}{H}\sum_i \big[p_0((1-\epsilon_{oh})h^i_o + \epsilon_{oh}h^i_{oh}) \tag{7}$$
$$+q_0(\epsilon_{oh}h^i_o + \epsilon_{oh}h^i_{oh})\big]$$

Calculating this threshold for the frequent minimal pairs that Yang extracts from the Wortschatz project (Biemann et al., 2004) corpus[3] and mishearing rates from Peterson and Barney (1952), the Yang model predicts that a learner exposed to at least $\sim$ 17% COT-CAUGHT-merged input will acquire the merger. This threshold represents a strong advantage for M_+ because it is well under the 50% threshold expected for neutral (non-advantaged) change and it is very close to what was found in Johnson (2007)'s sociolinguistic study. It predicts that younger children may have g_+ while their parents and even older siblings have g_- if the 17% threshold was crossed in **E** after the acquisition period of the older sibling but before that of the younger sibling.

4.2 Model Setup

All the mechanics behind the learning model reduce to a simple statement: *learners acquires g_+ iff $>$ 17% of their input is M_+ and they acquire g_- otherwise*. However, this kind of categorical learner in a perfectly-mixed population leads to immediate fixation at either g_- or g_+ in a single iteration, since the proportion of g_+ speakers in the population is equivalent to the proportion of M_+ input in every learner's environment. This is not realistic change. Clearly, social network structure is at least as important as the learning algorithm in modeling the spread of the merger.

We model the change in a non-uniform social network of 100 centralized clusters of 75 individuals each. 75 was chosen as half Dunbar's number, the maximum number of reliable social connections that an adult can maintain (Dunbar, 2010). There are two grammars, g_+ and g_-, and learners internalize one or the other according to the 17% threshold of M_+ in their input. One cluster represents the source of the merger and is initialized at 100% g_+, while the rest begin 100% g_-. Inter-cluster connections are chosen randomly so that some connections are between central members of the clusters and some are between peripheral members. The one merged cluster is connected to half the other clusters representing those at the frontier of the change, and each other cluster is connected to five randomly chosen ones.[4] This network structure echoes work in sociolinguistics, in particular, Milroy and Milroy (1985)'s notion of *strong* and *weak* connections in language change, where weak connections between social clusters are particularly important for propagation of a change.

Propagation of the merged grammar is calculated by $\mathcal{E}_n$ because we are interested in the behavior of individuals without loss of precision and because it cannot be assumed that each cluster is internally uniform.[5] Since the spread of the merger has been rapid enough to detect over a period of a few years, iterations are modeled as short age co-

[3]*Don* (1052) – *Dawn* (736); *collar* (403) – *caller* (23); *knotty* (25) – *naughty* (195); *odd* (830) – *awed* (80); *Otto* (67) – *auto* (260); *tot* (9) – *taught* (1327); *cot* (39) – *caught* (2444); *pond* (258) – *pawned* (31); *hock* (25) – *hawk* (127); *nod* (180) – *gnawed* (53); *sod* (30) – *sawed* (37)

[4]Originally, the clusters were set up as a "stepping-stone" chain with the merged community at one end, and that produced a similar S-curve. The structure presented here is more geographically plausible but not crucial for the results.

[5]$\alpha = 0.45$.

horts rather than full generations in the first experiments by updating only a randomly chosen 10% of nodes at each iteration because only a fraction of the population is learning at any given time. A model where every node is updated is investigated as well.

4.3 Results

The behavior of this simulation is shown graphically in Figure 4. The fine/colored lines indicate the rate of M_+ within each initially non-merged cluster, and the bold/black line shows the average rate across all initially non-merged. The merger spreads from cluster to cluster in succession over the "weak" inter-cluster connections and through each cluster over the 'strong' connections before moving on to the next ones.

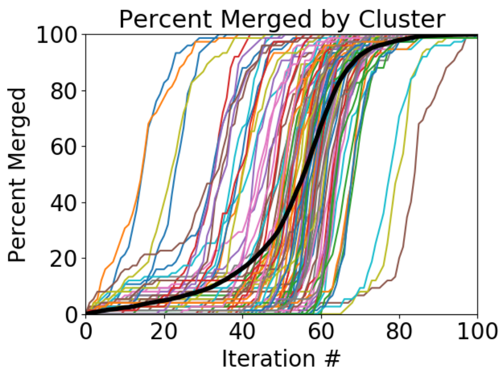


Figure 4: Spread of merger across communities (fine/colored) and population average (bold/black)

Most individual clusters exhibit a period of time in which only a few *early adopter* (Rogers Everett, 1995) members have the merger, a period of rapid diffusion of the merger, then some time where a few *laggards* resist the merger. As a result, most clusters exhibit an S-like shape. A few clusters change rapidly because of their especially well-connected positions in the network, and some lag behind the rest because they are poorly connected to the rest of the network. More interestingly, the population-wide average, the population-level data at the kind of granularity that is often studied, yields a smooth S-curve with a shallower slope than the individual clusters. The fact that it arises naturally here in a network that conforms with typical network shapes but was otherwise randomly generated is encouraging because the experiment was not set up so that it would produce such a curve, and the steep rate of change in individual clusters is what is expected for a change that is rapid enough to affect siblings differently.

In the above simulation, only a fraction of nodes were updated at each iteration in order to model a rapid change. In order to confirm that this choice is not affecting the results and to test a purer implementation of the framework presented here, we remove that constraint and update every node at each iteration. Figure 5 shows what happens over 20 iterations in a network that is otherwise identical but with 2/5 as many inter-cluster connections as the original. A qualitatively similar pattern arises, so the choice to update only a fraction of the population is not crucially affecting the results.

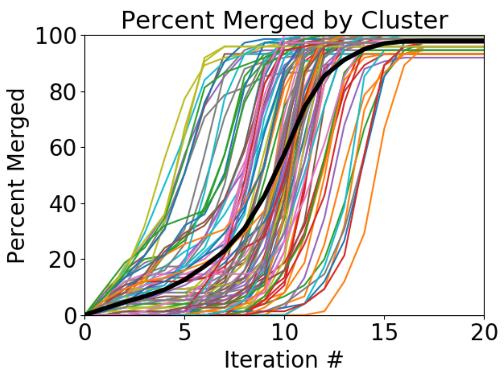


Figure 5: Spread of merger across communities (fine/colored) and population average (bold/black)

In all experiments so far, social connections were fixed at the first iteration even though connections in real populations tend to change over time. To investigate that modeling assumption, we perform another simulation in which connections are randomly updated both within and across clusters at each iteration akin to Kauhanen (2016)'s rewiring. The result as shown in Figure 6 is similar to before, with one major difference. The individual clusters transition more closely in time because no individual cluster remains poorly connected or especially well connected throughout the entire simulation.

Finally, we test our assumptions about population size by repeating the experiments on a smaller network of 40 clusters of 18 individuals. The results are qualitatively similar, but the S-curve appears to be more sensitive to probabilistic connections in the network. To explore this, we present the average network-wide rate of (M_+) across 10 trials, revealing that an S-like curve is formed each time but that its slope varies. A few trials never

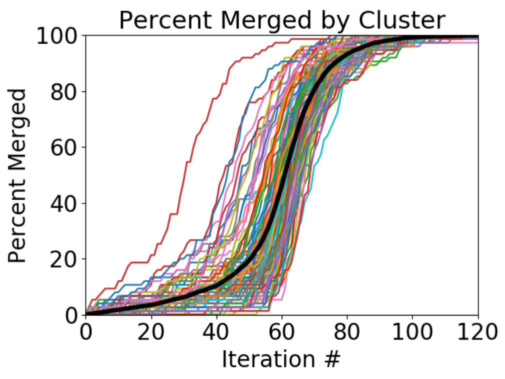


Figure 6: Spread of merger within communities (fine/colored) and as population average (bold/black). Network updated.

reach 100% because some of the clusters are not connected to the innovative one. The slope varies between trials, indicating that the rate of change is a function of both the population structure and the learning algorithm, but the network size does not substantially affect these results.

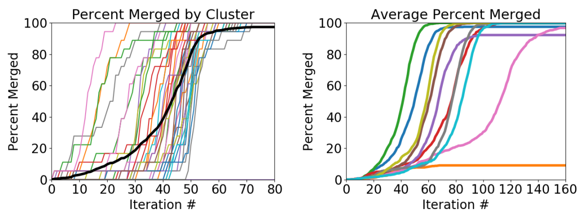


Figure 7: Single small network trial (left); average curves from 10 trials (right)

5 Discussion

The algebraic-network framework for modeling population-level language change presented here has substantial practical and theoretical advantages over previous ones. It is much simpler computationally than previous frameworks because it calculates the statistically expected behavior of each generation analytically and therefore removes the entire inner loop of calculating stochastic inter-agent interactions from the simulation. It follows the Niyogi and Berwick (1996) formalism for language change which presents a clean and modular way of reasoning about the problem and promotes the centrality of language acquisition.

In addition to the core algorithm, the framework offers enough flexibility to represent a wide variety of processes from the highly abstract (e.g., Kauhanen (2016)) to those grounded in sociolinguistic and acquisition research (e.g., Yang (2009)). In our investigation of Kauhanen's basic assumptions, we discover how seemingly innocuous decisions about population size and learning conspire to drive simulation results. If learners are conceived as categorical learners, population size becomes a deciding factor in the path of change. So while the original results are interesting and meaningful, they may only valid for small (on the order of 10^2) populations.

In our simulation of the spread of the COT-CAUGHT merger, we show how a cognitively-motivated model of acquisition requires a network model in order to represent population-level language change. The population is represented as a collection of individual clusters based on sociological work, but the clusters themselves are connected randomly. The fact that S-curves arise naturally from these networks underscores their centrality to language change.

One problem that this line of simulation work has always faced has been the lack of viable comparison between models because every study implements its own learning, network, and interaction models. The modular nature of our framework advances against this trend since it is now possible to hold the population model constant while slotting in various learning models to test them against one another and vice-versa. Finally, since this framework reduces to Niyogi & Berwick's models in perfectly-mixed populations, it can be used to reason about the formal dynamics of language change as well.

Without simulation, it would be difficult or impossible to undercover the interplay between acquisition and social structure on the propagation of language change. Neither factor alone can account for the theoretical or empirically observed patterns. Simulations of this kind which explicitly model both simultaneously is well equipped to provide insights that fieldwork and laboratory work cannot. As such, it is an invaluable complement to those more traditional methodologies.

Acknowledgments

We thank Charles Yang for is input and audiences at FWAV 4 and DiGS 19 for comments on earlier versions of this work. This research was funded by an NDSEG fellowship awarded to the first author by the US Dept. of Defense.

References

Gareth J Baxter, Richard A Blythe, William Croft, and Alan J McKane. 2006. Utterance selection model of language change. *Physical Review E*, 73(4):046118.

Gareth J Baxter, Richard A Blythe, William Croft, and Alan J McKane. 2009. Modeling language change: an evaluation of trudgill's theory of the emergence of new zealand english. *Language Variation and Change*, 21(02):257–296.

Christian Biemann, Stefan Bordag, Gerhard Heyer, Uwe Quasthoff, and Christian Wolff. 2004. Language-independent methods for compiling monolingual lexical data. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 217–228. Springer.

Leonard Bloomfield. 1933. *Language history: from Language (1933 ed.).* Holt, Rinehart and Winston.

Richard A Blythe and William Croft. 2012. S-curves and the mechanisms of propagation in language change. *Language*, 88(2):269–304.

Robert R Bush and Frederick Mosteller. 1953. A mathematical model for simple learning. In *Selected Papers of Frederick Mosteller*, pages 221–234. Springer.

Kathryn Campbell-Kibler. 2009. The nature of sociolinguistic perception. *Language Variation and Change*, 21(1):135–156.

Alfonso Caramazza, Albert Costa, Michele Miozzo, and Yanchao Bi. 2001. The specific-word frequency effect: implications for the representation of homophones in speech production. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(6):1430.

Luigi Luca Cavalli-Sforza and Marcus W Feldman. 1981. *Cultural transmission and evolution: a quantitative approach.* 16. Princeton University Press.

Robin Dunbar. 2010. *How many friends does one person need?: Dunbar's number and other evolutionary quirks.* Faber & Faber.

Zsuzsanna Fagyal, Samarth Swarup, Anna María Escobar, Les Gasser, and Kiran Lakkaraju. 2010. Centers and peripheries: Network roles in language change. *Lingua*, 120(8):2061–2079.

Edward Gibson and Kenneth Wexler. 1994. Triggers. *Linguistic inquiry*, 25(3):407–454.

K David Harrison, Mark Dras, Berk Kapicioglu, et al. 2002. Agent-based modeling of the evolution of vowel harmony. In *PROCEEDINGS-NELS*, 32; VOL 1, pages 217–236.

Daniel E Johnson. 2007. Stability and change along a dialect boundary: The low vowel mergers of southeastern new england. *University of Pennsylvania Working Papers in Linguistics*, 13(1):7.

Keith Johnson, Elizabeth A Strand, and Mariapaola D'Imperio. 1999. Auditory–visual integration of talker gender in vowel perception. *Journal of Phonetics*, 27(4):359–384.

Henri Kauhanen. 2016. Neutral change. *Journal of Linguistics*, pages 1–32.

Jinyun Ke, Tao Gong, and William SY Wang. 2008. Language change and social networks. *Communications in Computational Physics*, 3(4):935–949.

ed. Knight Chris Kirby, Simon, Michael Studdert-Kennedy, and James Hurford. 2000. *The evolutionary emergence of language: social function and the origins of linguistic form.* Cambridge University Press.

Sheldon Klein. 1966. Historical change in language using monte carlo techniques. *Mechanical Translation and Computational Linguistics*, 9(3):67–81.

Anthony S Kroch. 1989. Reflexes of grammar in patterns of language change. *Language variation and change*, 1(03):199–244.

William Labov. 1994. Principles of language change: Internal factors.

William Labov. 2001. Principles of language change: Social factors. *Malden, MA: Blackwell.*

William Labov, Sharon Ash, and Charles Boberg. 2005. *The atlas of North American English: Phonetics, phonology and sound change.* Walter de Gruyter.

David W Lightfoot. 1979. Principles of diachronic syntax. *Cambridge Studies in Linguistics London*, 23.

James Milroy and Lesley Milroy. 1985. Linguistic change, social network and speaker innovation. *Journal of linguistics*, 21(02):339–384.

James W Minett and William SY Wang. 2008. Modelling endangered languages: The effects of bilingualism and social structure. *Lingua*, 118(1):19–45.

Partha Niyogi. 1998. The logical problem of language change. In *The Informational Complexity of Learning*, pages 173–205. Springer.

Partha Niyogi and Robert C Berwick. 1996. A language learning model for finite parameter spaces. *Cognition*, 61(1):161–193.

Partha Niyogi and Robert C Berwick. 1997. A dynamical systems model for language change. *Complex Systems*, 11(3):161–204.

Partha Niyogi and Robert C Berwick. 2009. The proper treatment of language acquisition and change in a population setting. *Proceedings of the National Academy of Sciences*, 106(25):10124–10129.

Gordon E Peterson and Harold L Barney. 1952. Control methods used in a study of the vowels. *The Journal of the acoustical society of America*, 24(2):175–184.

Nikolaus Ritt. 2004. *Selfish sounds and linguistic evolution: A Darwinian approach to language change*. Cambridge University Press.

M Rogers Everett. 1995. Diffusion of innovations. *New York*, 12.

Teresa Satterfield. 2001. Toward a sociogenetic solution: Examining language formation processes through swarm modeling. *Social Science Computer Review*, 19(3):281–295.

Michael T Schaub, Neave O'Clery, Yazan N Billeh, Jean-Charles Delvenne, Renaud Lambiotte, and Mauricio Barahona. 2016. Graph partitions and cluster synchronization in networks of oscillators. *Chaos: An Interdisciplinary Journal of Nonlinear Science*.

James N Stanford and Laurence A Kenny. 2013. Revisiting transmission and diffusion: An agent-based model of vowel chain shifts across large communities. *Language Variation and Change*, 25(2):119.

Henry Sweet. 1899. *The practical study of languages*. London: Dent.

Charles Yang. 2009. Population structure and language change. *Ms., University of Pennsylvania*.

Charles D Yang. 2000. Internal and external forces in language change. *Language variation and change*, 12(03):231–250.

Prefix Lexicalization of Synchronous CFGs using Synchronous TAG

Logan Born and **Anoop Sarkar**
Simon Fraser University
School of Computing Science
{loborn,anoop}@sfu.ca

Abstract

We show that an ε-free, chain-free synchronous context-free grammar (SCFG) can be converted into a weakly equivalent synchronous tree-adjoining grammar (STAG) which is prefix lexicalized. This transformation at most doubles the grammar's rank and cubes its size, but we show that in practice the size increase is only quadratic. Our results extend Greibach normal form from CFGs to SCFGs and prove new formal properties about SCFG, a formalism with many applications in natural language processing.

1 Introduction

Greibach normal form (GNF; Greibach, 1965) is an important construction in formal language theory which allows every context-free grammar (CFG) to be rewritten so that the first character of each rule is a terminal symbol. A grammar in GNF is said to be prefix lexicalized, because the prefix of every production is a lexical item. GNF has a variety of theoretical and practical applications, including for example the proofs of the famous theorems due to Shamir and Chomsky-Schützenberger (Shamir, 1967; Chomsky and Schützenberger, 1963; Autebert et al., 1997). Other applications of prefix lexicalization include proving coverage of parsing algorithms (Gray and Harrison, 1972) and decidability of equivalence problems (Christensen et al., 1995).

By using prefix lexicalized synchronous context-free grammars (SCFGs), Watanabe et al. (2006) and Siahbani et al. (2013) obtain asymptotic and empirical speed improvements on a machine translation task. Using a prefix lexicalized grammar ensures that target sentences can be generated from left to right, which allows the use of beam search to constrain their decoder's search space as it performs a left-to-right traversal of translation hypotheses. To achieve these results, new grammars had to be heuristically constrained to include only prefix lexicalized productions, as there is at present no way to automatically convert an existing SCFG to a prefix lexicalized form.

This work investigates the formal properties of prefix lexicalized synchronous grammars as employed by Watanabe et al. (2006) and Siahbani et al. (2013), which have received little theoretical attention compared to non-synchronous prefix lexicalized grammars. To this end, we first prove that SCFG is not closed under prefix lexicalization. Our main result is that there is a method for prefix lexicalizing an SCFG by converting it to an equivalent grammar in a different formalism, namely synchronous tree-adjoining grammar (STAG) in regular form. Like the GNF transformation for CFGs, our method at most cubes the grammar size, but we show empirically that the size increase is only quadratic for grammars used in existing NLP tasks. The rank is at most doubled, and we maintain $O(n^{3k})$ parsing complexity for grammars of rank k. We conclude that although SCFG does not have a prefix lexicalized normal form like GNF, our conversion to prefix lexicalized STAG offers a practical alternative.

2 Background

2.1 SCFG

An SCFG is a tuple $G = (N, \Sigma, P, S)$ where N is a finite nonterminal alphabet, Σ is a finite terminal alphabet, $S \in N$ is a distinguished nonterminal called the start symbol, and P is a finite set of synchronous rules of the form

(1) $$\langle A_1 \to \alpha_1, A_2 \to \alpha_2 \rangle$$

for some $A_1, A_2 \in N$ and strings $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$.[1] Every nonterminal which appears in α_1

[1]A variant formalism exists which requires that $A_1 = A_2$; this is called syntax-directed transduction grammar (Lewis and Stearns, 1968) or syntax-directed translation schemata (Aho and Ullman, 1969). This variant is weakly equivalent to SCFG, but SCFG has greater strong generative capacity (Crescenzi et al., 2015).

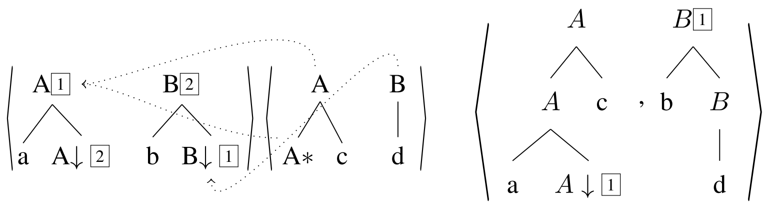


Figure 1: An example of synchronous rewriting in an STAG (left) and the resulting tree pair (right).

must be linked to exactly one nonterminal in α_2, and *vice versa*. We write these links using numerical annotations, as in (2).

(2) $\langle A \to A\boxed{1}B\boxed{2}, B \to B\boxed{2}A\boxed{1}\rangle$

An SCFG has rank k if no rule in the grammar contains more than k pairs of linked nodes.

In every step of an SCFG derivation, we rewrite one pair of linked nonterminals with a rule from P, in essentially the same way we would rewrite a single nonterminal in a non-synchronous CFG. For example, (3) shows linked A and B nodes being rewritten using (2):

(3)
$\langle X\boxed{1}A\boxed{2}, B\boxed{2}Y\boxed{1}\rangle \Rightarrow \langle X\boxed{1}A\boxed{2}B\boxed{3}, B\boxed{3}A\boxed{2}Y\boxed{1}\rangle$

Note how the $\boxed{1}$ and $\boxed{2}$ in rule (2) are renumbered to $\boxed{2}$ and $\boxed{3}$ during rewriting, to avoid an ambiguity with the $\boxed{1}$ already present in the derivation.

An SCFG derivation is complete when it contains no more nonterminals to rewrite. A completed derivation represents a string pair generated by the grammar.

2.2 STAG

An STAG (Shieber, 1994) is a tuple $G = (N, \Sigma, T, S)$ where N is a finite nonterminal alphabet, Σ is a finite terminal alphabet, $S \in N$ is a distinguished nonterminal called the start symbol, and T is a finite set of synchronous tree pairs of the form

(4) $\langle t_1, t_2\rangle$

where t_1 and t_2 are elementary trees as defined in Joshi et al. (1975). A *substitution site* is a leaf node marked by $\downarrow$ which may be rewritten by another tree; a *foot node* is a leaf marked by $*$ that may be used to rewrite a tree-internal node. Every substitution site in t_1 must be linked to exactly one nonterminal in t_2, and *vice versa*. As in SCFG, we write these links using numbered annotations; rank is defined for STAG the same way as for SCFG.

In every step of an STAG derivation, we rewrite one pair of linked nonterminals with a tree pair from T, using the same substitution and adjunction operations defined for non-synchronous TAG. For example, Figure 1 shows linked A and B nodes being rewritten and the tree pair resulting from this operation. See Joshi et al. (1975) for details about the underlying TAG formalism.

2.3 Terminology

We use *synchronous production* as a cover term for either a synchronous rule in an SCFG or a synchronous tree pair in an STAG.

Following Siahbani et al. (2013), we refer to the left half of a synchronous production as the source side, and the right half as the target side; this terminology captures the intuition that synchronous grammars model translational equivalence between a source phrase and its translation into a target language. Other authors refer to the two halves as the left and right components (Crescenzi et al., 2015) or, viewing the grammar as a transducer, the input and the output (Engelfriet et al., 2017).

We call a grammar ε-free if it contains no productions whose source or target side produces only the empty string ε.

2.4 Synchronous Prefix Lexicalization

Previous work (Watanabe et al., 2006; Siahbani et al., 2013) has shown that it is useful for the target side of a synchronous grammar to start with a terminal symbol. For this reason, we define a synchronous grammar to be prefix lexicalized when the leftmost character of the target side[2] of every synchronous production in that grammar is a terminal symbol.

Formally, this means that every synchronous rule in a prefix lexicalized SCFG (PL-SCFG) is

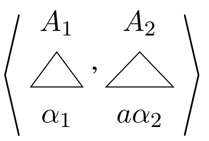
[2]All of the proofs in this work admit a symmetrical variant which prefix lexicalizes the source side instead of the target. We are not aware of any applications in NLP where source-side prefix lexicalization is useful, so we do not address this case.

of the form

(5) $$\langle A_1 \rightarrow \alpha_1, A_2 \rightarrow a\alpha_2 \rangle$$

where $A_1, A_2 \in N, \alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ and $a \in \Sigma$.

Every synchronous tree pair in a prefix lexicalized STAG (PL-STAG) is of the form

(6) 

where $A_1, A_2 \in N, \alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ and $a \in \Sigma$.

3 Closure under Prefix Lexicalization

We now prove that the class SCFG is not closed under prefix lexicalization.

Theorem 1. *There exists an SCFG which cannot be converted to an equivalent PL-SCFG.*

Proof. The SCFG in (7) generates the language $L = \{\langle a^i b^j c^i, b^j a^i \rangle \mid i \geq 0, j \geq 1\}$, but this language cannot be generated by any PL-SCFG:

(7)
$$\begin{array}{ll}
\langle S \rightarrow A\boxed{1}, & S \rightarrow A\boxed{1} \rangle \\
\langle A \rightarrow aA\boxed{1}c, & A \rightarrow A\boxed{1}a \rangle \\
\langle A \rightarrow bB\boxed{1}, & A \rightarrow bB\boxed{1} \rangle \\
\langle A \rightarrow b, & A \rightarrow b \rangle \\
\langle B \rightarrow bB\boxed{1}, & B \rightarrow bB\boxed{1} \rangle \\
\langle B \rightarrow b, & B \rightarrow b \rangle
\end{array}$$

Suppose, for the purpose of contradiction, that some PL-SCFG does generate L; call this grammar G. Then the following derivations must all be possible in G for some nontermials U, V, X, Y:

i) $\langle U\boxed{1}, V\boxed{1} \rangle \Rightarrow^* \langle b^k U\boxed{1} b^m, b^n V\boxed{1} b^p \rangle$, where $k + m = n + p$ and $n \geq 1$

ii) $\langle X\boxed{1}, Y\boxed{1} \rangle \Rightarrow^* \langle a^q X\boxed{1} c^q, a^r Y\boxed{1} a^s \rangle$, where $q = r + s$ and $r \geq 1$

iii) $\langle S\boxed{1}, S\boxed{1} \rangle \Rightarrow^* \langle \alpha_1 X\boxed{1} \alpha_2, b\alpha_3 Y\boxed{1} \alpha_4 \rangle$, where $\alpha_1, ..., \alpha_4 \in (N \cup \Sigma)^*$

iv) $\langle X\boxed{1}, Y\boxed{1} \rangle \Rightarrow^* \langle \alpha_5 U\boxed{1} \alpha_6, \alpha_7 V\boxed{1} \alpha_8 \rangle$, where $\alpha_5, \alpha_6, \alpha_8 \in (N \cup \Sigma)^*$, $\alpha_7 \in \Sigma(N \cup \Sigma)^*$

i and ii follow from the same arguments used in the pumping lemma for (non-synchronous) context free languages (Bar-Hillel et al., 1961): strings in L can contain arbitrarily many as, bs, and cs, so there must exist some pumpable cycles which generate these characters. In i, $k + m = n + p$ because the final derived strings must contain an equal number of bs, and $n \geq 1$ because G is prefix lexicalized; in ii the constraints on q, r and s follow likewise from L. iii follows from the fact that, in order to pump on the cycle in ii, this cycle must be reachable from the start symbol. iv follows from the fact that a context-free production cannot generate a discontinuous span. Once the cycle in i has generated a b, it is impossible for ii to generate an a on one side of the b and a c on the other. Therefore i must always be derived strictly later than ii, as shown in iv.

Now we obtain a contradiction. Given that G can derive all of i through iv, the following derivation is also possible:

(8)
$$\begin{array}{ll}
 & \langle S\boxed{1}, S\boxed{1} \rangle \\
\Rightarrow^* & \langle \alpha_1 X\boxed{1} \alpha_2, b\alpha_3 Y\boxed{1} \alpha_4 \rangle \\
\Rightarrow^* & \langle \alpha_1 a^q X\boxed{1} c^q \alpha_2, b\alpha_3 a^r Y\boxed{1} a^s \alpha_4 \rangle \\
\Rightarrow^* & \langle \alpha_1 a^q \alpha_5 U\boxed{1} \alpha_6 c^q \alpha_2, b\alpha_3 a^r \alpha_7 V\boxed{1} \alpha_8 a^s \alpha_4 \rangle \\
\Rightarrow^* & \langle \alpha_1 a^q \alpha_5 b^k U\boxed{1} b^m \alpha_6 c^q \alpha_2, \\
 & \quad b\alpha_3 a^r \alpha_7 b^n V\boxed{1} b^p \alpha_8 a^s \alpha_4 \rangle
\end{array}$$

But since $n, r \geq 1$, the target string derived this way contains an a before a b and does not belong to L.

This is a contradiction: if G is a PL-SCFG then it must generate i through iv, but if so then it also generates strings which do not belong to L. Thus no PL-SCFG can generate L, and SCFG must not be closed under prefix lexicalization. ■

There also exist grammars which cannot be prefix lexicalized because they contain cyclic chain rules. If an SCFG can derive something of the form $\langle X\boxed{1}, Y\boxed{1} \rangle \Rightarrow^* \langle xX\boxed{1}, Y\boxed{1} \rangle$, then it can generate arbitrarily many symbols in the source string without adding anything to the target string. Prefix lexicalizing the grammar would force it to generate some terminal symbol in the target string at each step of the derivation, making it unable to generate the original language where a source string may be unboundedly longer than its corresponding target. We call an SCFG chain-free if it does not contain a cycle of chain rules of this form. The remainder of this paper focuses on chain-free grammars, like (7), which cannot be converted to PL-SCFG despite containing no such cycles.

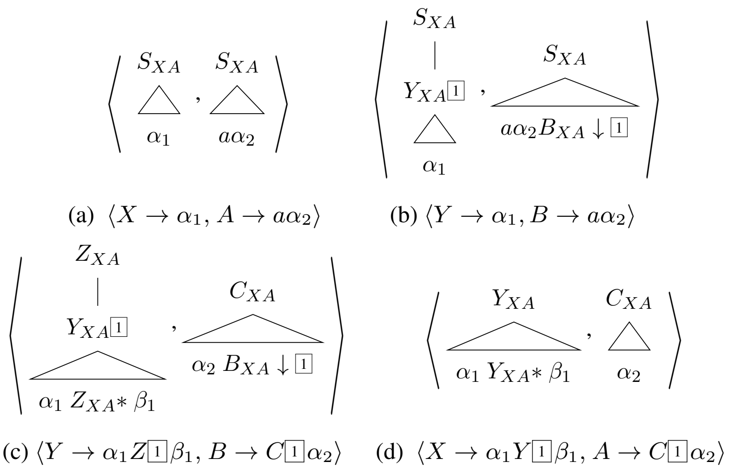
4 Prefix Lexicalization using STAG

We now present a method for prefix lexicalizing an SCFG by converting it to an STAG.

$$\langle X\boxed{1}, A\boxed{1}\rangle \Rightarrow \langle \alpha_1 Y_1\boxed{1}\beta_1, B_1\boxed{1}\gamma_1\rangle \Rightarrow \langle \alpha_1\alpha_2 Y_2\boxed{1}\beta_2\beta_1, B_2\boxed{1}\gamma_2\gamma_1\rangle$$

$$\Rightarrow^* \langle \alpha_1\cdots\alpha_t Y_t\boxed{1}\beta_t\cdots\beta_1, B_t\boxed{1}\gamma_t\cdots\gamma_1\rangle \Rightarrow \langle \alpha_1\cdots\alpha_t\alpha_{t+1}\beta_t\cdots\beta_1, a\gamma_{t+1}\gamma_t\cdots\gamma_1\rangle$$

Figure 2: A target-side terminal leftmost derivation. $a \in \Sigma$; $X, A, Y_i, B_i \in N$; and $\alpha_i, \beta_i, \gamma_i \in (N \cup \Sigma)^*$.



(a) $\langle X \to \alpha_1, A \to a\alpha_2\rangle$ (b) $\langle Y \to \alpha_1, B \to a\alpha_2\rangle$

(c) $\langle Y \to \alpha_1 Z\boxed{1}\beta_1, B \to C\boxed{1}\alpha_2\rangle$ (d) $\langle X \to \alpha_1 Y\boxed{1}\beta_1, A \to C\boxed{1}\alpha_2\rangle$

Figure 3: Tree-pairs in G_{XA} and the rules in G from which they derive.

Theorem 2. *Given a rank-k SCFG G which is ε-free and chain-free, an STAG H exists such that H is prefix lexicalized and $L(G) = L(H)$. The rank of H is at most 2k, and $|H| = O(|G|^3)$.*

Proof. Let $G = (N, \Sigma, P, S)$ be an ε-free, chain-free SCFG. We provide a constructive method for prefix lexicalizing the target side of G.

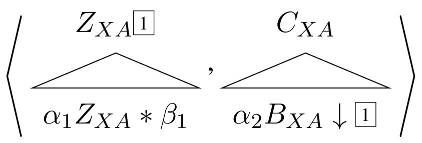
We begin by constructing an intermediate grammar G_{XA} for each pair of nonterminals $X, A \in N \setminus \{S\}$. For each pair $X, A \in N \setminus \{S\}$, G_{XA} will be constructed to generate the language of sentential forms derivable from $\langle X\boxed{1}, A\boxed{1}\rangle$ via a target-side terminal leftmost derivation (TTLD). A TTLD is a derivation of the form in Figure 2, where the leftmost nonterminal in the target string is expanded until it produces a terminal symbol as the first character. We write $\langle X\boxed{1}, A\boxed{1}\rangle \Rightarrow^*_{TTLD} \langle u, v\rangle$ to mean that $\langle X\boxed{1}, A\boxed{1}\rangle$ derives $\langle u, v\rangle$ by way of a TTLD; in this notation, $L_{XA} = \{\langle u, v\rangle | \langle X\boxed{1}, A\boxed{1}\rangle \Rightarrow^*_{TTLD} \langle u, v\rangle\}$ is the language of sentential forms derivable from $\langle X\boxed{1}, A\boxed{1}\rangle$ via a TTLD.

Given $X, A \in N \setminus \{S\}$ we formally define G_{XA} as an STAG over the terminal alphabet $\Sigma_{XA} = N \cup \Sigma$ and nonterminal alphabet $N_{XA} = \{Y_{XA} | Y \in N\}$, with start symbol S_{XA}. N_{XA} contains nonterminals indexed by XA to ensure that two intermediate grammars G_{XA} and G_{YB} do not interact as long as $\langle X, A\rangle \neq \langle Y, B\rangle$.

G_{XA} contains four kinds of tree pairs: [3]

- For each rule in G of the form $\langle X \to \alpha_1, A \to a\alpha_2\rangle$, $a \in \Sigma$, $\alpha_i \in (N\cup\Sigma)^*$, we add a tree pair of the form in Figure 3(a).
- For each rule in G of the form $\langle Y \to \alpha_1, B \to a\alpha_2\rangle$, $a \in \Sigma$, $\alpha_i \in (N\cup\Sigma)^*$, $Y, B \in N \setminus \{S\}$, we add a tree pair of the form in Figure 3(b).
- For each rule in G of the form $\langle Y \to \alpha_1 Z\boxed{1}\beta_1, B \to C\boxed{1}\alpha_2\rangle$, $Y, Z, B, C \in N \setminus \{S\}$, $\alpha_i, \beta_i \in (N \cup \Sigma)^*$, we add a tree pair of the form in Figure 3(c).

  As a special case, if $Y = Z$ we collapse the root node and adjunction site to produce a tree pair of the following form:

(9) 
$Z_{XA}\boxed{1}$
$\alpha_1 Z_{XA} * \beta_1$
C_{XA}
$\alpha_2 B_{XA}\downarrow\boxed{1}$

- For each rule in G of the form $\langle X \to \alpha_1 Y\boxed{1}\beta_1, A \to C\boxed{1}\alpha_2\rangle$, $Y, C \in N$, $\alpha_i, \beta_i \in (N \cup \Sigma)^*$, we add a tree pair of the form in Figure 3(d).

[3]In all cases, we assume that symbols in N (not N_{XA}) retain any links they bore in the original grammar, even though they belong to the terminal alphabet in G_{XA} and therefore do not participate in rewriting operations. In the final constructed grammar, these symbols will belong to the nonterminal alphabet again, and the links will function normally.

$$\langle A \to B\boxed{2}cA\boxed{1}, A \to A\boxed{1}cB\boxed{2}\rangle$$

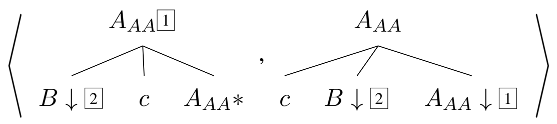


Figure 4: An SCFG rule and a tree pair based off that rule, taken from an intermediate grammar G_{AA}. The tree pair is formed according to the pattern illustrated in Figure 3(c). Observe that the B nodes retain the link they bore in the original rule. This link is not functional in the intermediate grammar (that is, it cannot be used for synchronous rewriting) because $B \notin N_{AA}$, but it will be functional when this tree pair is added to the final grammar H.

Figure 4 gives a concrete example of constructing an intermediate grammar tree pair on the basis of an SCFG rule.

Lemma 1. *G_{XA} generates the language L_{XA}.*

Proof. This can be shown by induction over derivations of increasing length. The proof is straightforward but very long, so we provide only a sketch; the complete proof is provided in the supplementary material.

As a base case, observe that a tree of the shape in Figure 3(a) corresponds straightforwardly to the derivation

(10) $$\langle X\boxed{1}, A\boxed{1}\rangle \Rightarrow \langle \alpha_1, a\alpha_2\rangle$$

which is a TTLD starting from $\langle X, A\rangle$. By construction, therefore, every TTLD of the shape in (10) corresponds to some tree in G_{XA} of shape 3(a); likewise every derivation in G_{XA} comprising a single tree of shape 3(a) corresponds to a TTLD of the shape in (10).

As a second base case, note that a tree of the shape in Figure 3(b) corresponds to the last step of a TTLD like (11):

(11) $$\langle X\boxed{1}, A\boxed{1}\rangle \Rightarrow^*_{TTLD} \langle uY\boxed{1}v, B\boxed{1}w\rangle$$
$$\Rightarrow \langle u\alpha_1 v, a\alpha_2 w\rangle$$

In the other direction, the last step of any TTLD of the shape in (11) will involve some rule of the shape $\langle Y \to \alpha_1, B \to a\alpha_2\rangle$; by construction G_{XA} must contain a corresponding tree pair of shape 3(b).

Together, these base cases establish a one-to-one correspondence between single-tree derivations in G_{XA} and the last step of a TTLD starting from $\langle X, A\rangle$.

Now, assume that the last n steps of every TTLD starting from $\langle X, A\rangle$ correspond to some derivation over n trees in G_{XA}, and *vice versa*. Then the last $n + 1$ steps of that TTLD will also correspond to some $n + 1$ tree derivation in G_{XA}, and *vice versa*.

To see this, consider the step $n + 1$ steps before the end of the TTLD. This step may be in the middle of the derivation, or it may be the first step of the derivation. If it is in the middle, then this step must involve a rule of the shape

(12) $$\langle Y \to \alpha_1 Z\boxed{1}\beta_1, B \to C\boxed{1}\alpha_2\rangle$$

The existence of such a rule in G implies the existence of a corresponding tree in G_{XA} of the shape in Figure 3(c). Adding this tree to the existing n-tree derivation yields a new $n + 1$ tree derivation corresponding to the last $n + 1$ steps of the TTLD.[4] In the other direction, if the $n + 1$th tree[5] of a derivation in G_{XA} is of the shape in Figure 3(c), then this implies the existence of a production in G of the shape in (12). By assumption the first n trees of the derivation in G_{XA} correspond to some TTLD in G; by prepending the rule from (12) to this TTLD we obtain a new TTLD of length $n + 1$ which corresponds to the entire $n + 1$ tree derivation in G_{XA}.

Finally, consider the case where the TTLD is only $n + 1$ steps long. The first step must involve a rule of the form

(13) $$\langle X \to \alpha_1 Y\boxed{1}\beta_1, A \to C\boxed{1}\alpha_2\rangle$$

The existence of such a rule implies the existence of a corresponding tree in G_{XA} of the shape in Figure 3(d). Adding this tree to the derivation which corresponds to the last n steps of the TTLD yields a new $n+1$ tree derivation corresponding to the entire $n + 1$ step TTLD. In the other direction, if the last tree of an $n + 1$ tree derivation in G_A is of the shape in Figure 3(d), then this implies the

[4]It is easy to verify by inspection of Figure 3 that whenever one rule from G can be applied to the output of another rule, then the tree pairs in G_{XA} which correspond to these rules can compose with one another. Thus we can add the new tree to the existing derivation and be assured that it will compose with one of the trees that is already present.

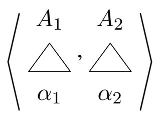
[5]Although trees in G_{XA} may contain symbols from the nonterminal alphabet of G, these symbols belong to the *terminal* alphabet in G_{XA}. Only nonterminals in N_{XA} will be involved in this derivation, and by construction there is at most one such nonterminal per tree. Thus a well-formed derivation structure in G_{XA} will never branch, and we can refer to the $n + 1$th tree pair as the one which is at depth n in the derivation structure.

existence of a production in G of the shape in (13). By assumption the first n trees of the derivation in G_{XA} correspond to some TTLD in G; by prepending the rule from (13) to this TTLD we obtain a new TTLD of length $n + 1$ which corresponds to the entire $n + 1$ tree derivation in G_{XA}.

Taken together, these cases establish a one-to-one correspondence between derivations in G_{XA} and TTLDs which start from $\langle X, A\rangle$; in turn they confirm that G_{XA} generates the desired language L_{XA}. □

Once we have constructed an intermediate grammar G_{XA} for each $X, A \in N \setminus \{S\}$, we obtain the final STAG H as follows:

1. Convert the input SCFG G to an equivalent STAG. For each rule $\langle A_1 \to \alpha_1, A_2 \to \alpha_2\rangle$, where $A_i \in N$, $\alpha_i \in (N \cup \Sigma)^*$, create a tree pair of the form

   (14)
   

   where each pair of linked nonterminals in the original rule become a pair of linked substitution sites in the tree pair. The terminal and nonterminal alphabets and start symbol are unchanged. Call the resulting STAG H.

2. For all $X, A \in N \setminus \{S\}$, add all of the tree pairs from the intermediate grammar G_{XA} to the new grammar H. Expand N to include the new nonterminal symbols in N_{XA}.

3. For every $X, A \in N$, in all tree pairs where the target tree's leftmost leaf is labeled with A and this node is linked to an X, replace this occurrence of A with S_{XA}. Also replace the linked node in the source tree.

4. For every $X, A \in N$, let R_{XA} be the set of all tree pairs rooted in S_{XA}, and let T_{XA} be the set of all tree pairs whose target tree's leftmost leaf is labeled with S_{XA}. For every $\langle s, t\rangle \in T_{XA}$ and every $\langle s', t'\rangle \in R_{XA}$, substitute or adjoin s' and t' into the linked S_{XA} nodes in s and t, respectively. Add the derived trees to H.

5. For all $X, A \in N$, let T_{XA} be defined as above. Remove all tree pairs in T_{XA} from H.

6. For all $X, A \in N$, let R_{XA} be defined as above. Remove all tree pairs in R_{XA} from H.

We now claim that H generates the same language as the original grammar G, and all of the target trees in H are prefix lexicalized.

The first claim follows directly from the construction. Step 1 merely rewrites the grammar in a new formalism. From Lemma 1 it is clear that steps 2–3 do not change the generated language: the set of string pairs generable from a pair of S_{XA} nodes is identical to the set generable from $\langle X, A\rangle$ in the original grammar. Step 4 replaces some nonterminals by all possible alternatives; steps 5–6 then remove the trees which were used in step 4, but since all possible combinations of these trees have already been added to the grammar, removing them will not alter the language.

The second claim follows from inspection of the tree pairs generated in Figure 3. Observe that, by construction, for all $X, A \in N$ every target tree rooted in S_{XA} is prefix lexicalized. Thus the trees created in step 4 are all prefix lexicalized variants of non-lexicalized tree pairs; steps 5–6 then remove the non-lexicalized trees from the grammar. ■

Figure 5 gives an example of this transformation applied to a small grammar. Note how A nodes at the left edge of the target trees end up rewritten as S_{AA} nodes, as per step 4 of the transformation.

5 Complexity & Formal Properties

Our conversion generates a subset of the class of prefix lexicalized STAGs in regular form, which we abbreviate to PL-RSTAG (regular form for TAG is defined in Rogers 1994). This section discusses some formal properties of PL-RSTAG.

Generative Capacity PL-RSTAG is weakly equivalent to the class of ε-free, chain-free SCFGs: this follows immediately from the proof that our transformation does not change the language generated by the input SCFG. Note that every TAG in regular form generates a context-free language (Rogers, 1994).

Alignments and Reordering PL-RSTAG generates the same set of reorderings (alignments) as SCFG. Observe that our transformation does not cause nonterminals which were linked in the original grammar to become unlinked, as noted for example in Figure 4. Thus subtrees which are gener-

$$\begin{array}{ll}\langle S \to B\boxed{2}cA\boxed{1}, & S \to A\boxed{1}cB\boxed{2}\rangle \\ \langle A \to B\boxed{2}cA\boxed{1}, & A \to A\boxed{1}cB\boxed{2}\rangle \\ \langle A \to a, & A \to a\rangle \\ \langle B \to b, & B \to b\rangle\end{array}$$

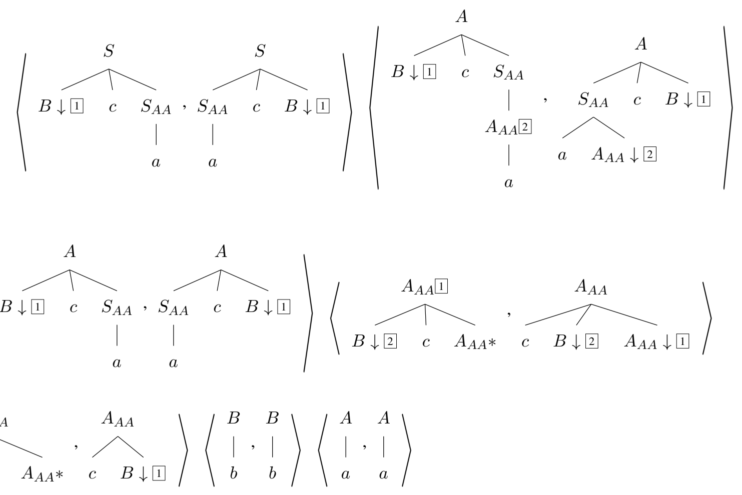


Figure 5: An SCFG and the STAG which prefix lexicalizes it. Non-productive trees have been omitted.

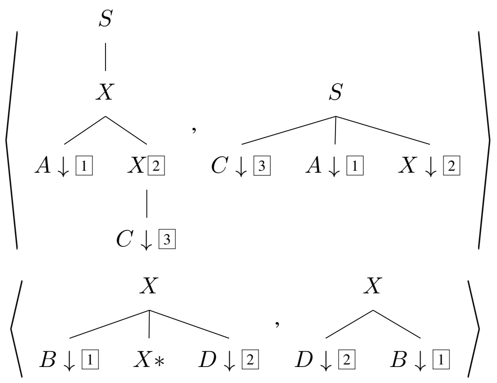
Grammar	$\lvert G\rvert$	$\lvert H\rvert$	% of G prefix lexicalized	$\log_{\lvert G\rvert}(\lvert H\rvert)$
Siahbani and Sarkar (2014a) (Zh-En)	18.5M	23.6T	63%	1.84
Example (7)	6	14	66%	1.47
ITG (10000 translation pairs)	10,003	170,000	99.97%	1.31

Table 1: Grammar sizes before and after prefix lexicalization, showing $O(n^2)$ size increase instead of the worst case $O(n^3)$. $|G|$ and $|H|$ give the grammar size before and after prefix lexicalization; $\log_{|G|}|H|$ is the increase as a power of the initial size. We also show the percentage of productions which are already prefix lexicalized in G.

ated by linked nonterminals in the original grammar will still be generated by linked nonterminals in the final grammar, so no reordering information is lost or added.[6] This result holds despite the fact that our transformation is only applicable to chain-free grammars: chain rules cannot introduce any reorderings, since by definition they involve only a single pair of linked nonterminals.

Grammar Rank If the input SCFG G has rank k, then the STAG H produced by our transformation has rank at most $2k$. To see this, observe that the construction of the intermediate grammars increases the rank by at most 1 (see Figure 3(b)). When a prefix lexicalized tree is substituted at the left edge of a non-lexicalized tree, the link on the substitution site will be consumed, but up to $k+1$ new links will be introduced by the substituting tree, so that the final tree will have rank at most $2k$.

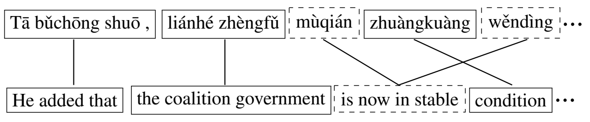
In the general case, rank-k STAG is more powerful than rank-k SCFG; for example, a rank-4 SCFG is required to generate the reordering in $\langle S \to A\boxed{1}B\boxed{2}C\boxed{3}D\boxed{4}, S \to C\boxed{3}A\boxed{1}D\boxed{4}B\boxed{2}\rangle$ (Wu, 1997), but this reordering is captured by the following rank-3 STAG:



For this reason, we speculate that it is possible to further transform the grammars produced by our lexicalization in order to reduce their rank, but the details of this transformation remain as future work.

This potentially poses a solution to an issue raised by Siahbani and Sarkar (2014b). On a Chinese-English translation task, they find that sentences like (15) involve reorderings which cannot be captured by a rank-2 prefix lexicalized SCFG:

(15)



If rank-k PL-RSTAG is more powerful than rank-k

[6]Although we consume one link whenever we substitute a prefix lexicalized tree at the left edge of an unlexicalized tree, that link can still be remembered and used to reconstruct the reorderings which occurred between the two sentences.

SCFG, using a PL-RSTAG here would permit capturing more reorderings without using grammars of higher rank.

Parse Complexity Because the grammar produced is in regular form, each side can be parsed in time $O(n^3)$ (Rogers, 1994), for an overall parse complexity of $O(n^{3k})$, where n is sentence length and k is the grammar rank.

Grammar Size and Experiments If H is the PL-RSTAG produced by applying our transformation to an SCFG G, then H contains $O(|G|^3)$ elementary tree pairs, where $|G|$ is the number of synchronous productions in G. When the set of nonterminals N is small compared to $|G|$, a tighter bound is given by $O(|G|^2|N|^2)$.

Table 1 shows the actual size increase on a variety of grammars: here $|G|$ is the size of the initial grammar, $|H|$ is the size after applying our transformation, and the increase is expressed as a power of the original grammar size. We apply our transformation to the grammar from Siahbani and Sarkar (2014a), which was created for a Chinese-English translation task known to involve complex reorderings that cannot be captured by PL-SCFG (Siahbani and Sarkar, 2014b). We also consider the grammar in (7) and an ITG (Wu, 1997) containing 10,000 translation pairs, which is a grammar of the sort that has previously been used for word alignment tasks (cf Zhang and Gildea 2005). We always observe an increase within $O(|G|^2)$ rather than the worst-case $O(|G|^3)$, because $|N|$ is small relative to $|G|$ in most grammars used for NLP tasks.

We also investigated how the proportion of prefix lexicalized rules in the original grammar affects the overall size increase. We sampled grammars with varying proportions of prefix lexicalized rules from the grammar in Siahbani and Sarkar (2014a); Table 2 shows the result of lexicalizing these samples. We find that the worst case size increase occurs when 50% of the original grammar is already prefix lexicalized. This is because the size increase depends on both the number of prefix lexicalized trees in the intermediate grammars (which grows with the proportion of lexicalized rules) and the number of productions which need to be lexicalized (which shrinks as the proportion of prefix lexicalized rules increases). At 50%, both factors contribute appreciably to the grammar size, analogous to how the function $f(x) = x(1 - x)$ takes its maximum at $x = 0.5$.

$\|G\|$	$\|H\|$	% of G prefix lexicalized	$\log_{\|G\|}(\|H\|)$
15k	42.4M	10%	1.83
15k	74.9M	20%	1.89
15k	97.8M	30%	1.91
15k	112M	40%	1.93
15k	118M	50%	1.93
15k	114M	60%	1.93
15k	102M	70%	1.92
15k	78.2M	80%	1.89
15k	43.6M	90%	1.83

Table 2: Effect of prefix lexicalized rules in G on final grammar size.

6 Applications

The LR decoding algorithm from Watanabe et al. (2006) relies on prefix lexicalized rules to generate a prefix of the target sentence during machine translation. At each step, a translation hypothesis is expanded by rewriting the leftmost nonterminal in its target string using some grammar rule; the prefix of this rule is appended to the existing translation and the remainder of the rule is pushed onto a stack, in reverse order, to be processed later. Translation hypotheses are stored in stacks according to the length of their translated prefix, and beam search is used to traverse these hypotheses and find a complete translation. During decoding, the source side is processed by an Earley-style parser, with the dot moving around to process nonterminals in the order they appear on the target side.

Since the trees on the target side of our transformed grammar are all of depth 1, and none of these trees can compose via the adjunction operation, they can be treated like context-free rules and used as-is in this decoding algorithm. The only change required to adapt LR decoding to use a PL-RSTAG is to make the source side use a TAG parser instead of a CFG parser; an Earley-style parser for TAG already exists (Joshi and Schabes, 1997), so this is a minor adjustment.

Combined with the transformation in Section 4, this suggests a method for using LR decoding without sacrificing translation quality. Previously, LR decoding required the use of heuristically generated PL-SCFGs, which cannot model some reorderings (Siahbani and Sarkar, 2014a). Now, an SCFG tailored for a translation task can be transformed directly to PL-RSTAG and used for decoding; unlike a heuristically induced PL-SCFG, the transformed PL-RSTAG will generate the same language as the original SCFG which is known to handle more reorderings.

Note that, since applying our transformation may double the rank of a grammar, this method may prove prohibitively slow. This highlights the need for future work to examine the generative power of rank-k PL-RSTAG relative to rank-k SCFG in the interest of reducing the rank of the transformed grammar.

7 Related Work

Our work continues the study of TAGs and lexicalization (e.g. Joshi et al. 1975; Schabes and Waters 1993). Schabes and Waters (1995) show that TAG can strongly lexicalize CFG, whereas CFG only weakly lexicalizes itself; we show a similar result for SCFGs. Kuhlmann and Satta (2012) show that TAG is not closed under strong lexicalization, and Maletti and Engelfriet (2012) show how to strongly lexicalize TAG using simple context-free tree grammars (CFTGs).

Other extensions of GNF to new grammar formalisms include Dymetman (1992) for definite clause grammars, Fernau and Stiebe (2002) for CF valence grammars, and Engelfriet et al. (2017) for multiple CFTGs. Although multiple CFTG subsumes SCFG (and STAG), Engelfriet et al.'s result appears to guarantee only that *some* side of every synchronous production will be lexicalized, whereas our result guarantees that it is always the target side that will be prefix lexicalized.

Lexicalization of synchronous grammars was addressed by Zhang and Gildea (2005), but they consider lexicalization rather than prefix lexicalization, and they only consider SCFGs of rank 2. They motivate their results using a word alignment task, which may be another possible application for our lexicalization.

Analogous to our closure result, Aho and Ullman (1969) prove that SCFG does not admit a normal form with bounded rank like Chomsky normal form.

Blum and Koch (1999) use intermediate grammars like our G_{XA}s to transform a CFG to GNF. Another GNF transformation (Rosenkrantz, 1967) is used by Schabes and Waters (1995) to define Tree Insertion Grammars (which are also weakly equivalent to CFG).

We rely on Rogers (1994) for the claim that our transformed grammars generate context-free languages despite allowing wrapping adjunction; an alternative proof could employ the results of Swanson et al. (2013), who develop their own context-free TAG variant known as osTAG.

Kaeshammer (2013) introduces the class of synchronous linear context-free rewriting systems to model reorderings which cannot be captured by a rank-2 SCFG. In the event that rank-k PL-RSTAG is more powerful than rank-k SCFG, our work can be seen as an alternative approach to the same problem.

Finally, Nesson et al. (2008) present an algorithm for reducing the rank of an STAG on-the-fly during parsing; this presents a promising avenue for proving a smaller upper bound on the rank increase caused by our transformation.

8 Conclusion and Future Work

We have demonstrated a method for prefix lexicalizing an SCFG by converting it to an equivalent STAG. This process is applicable to any SCFG which is ε- and chain-free. Like the original GNF transformation for CFGs our construction at most cubes the grammar size, though when applied to the kinds of synchronous grammars used in machine translation the size is merely squared. Our transformation preserves all of the alignments generated by SCFG, and retains properties such as $O(n^{3k})$ parsing complexity for grammars of rank k. We plan to verify whether rank-k PL-RSTAG is more powerful than rank-k SCFG in future work, and to reduce the rank of the transformed grammar if possible. We further plan to empirically evaluate our lexicalization on an alignment task and to offer a comparison against the lexicalization due to Zhang and Gildea (2005).

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. The research was also partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC RGPIN-2018-06437 and RGPAS-2018-522574) to the second author. We dedicate this paper to the memory of Prof. Aravind Joshi; a short hallway conversation with him at ACL 2014 was the seed for this paper.

References

Alfred V. Aho and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences* 3(1):37–56.

Jean-Michel Autebert, Jean Berstel, and Luc Boasson. 1997. Context-free languages and pushdown automata. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol. 1*, Springer-Verlag New York, Inc., New York, NY, USA, pages 111–174. http://dl.acm.org/citation.cfm?id=267846.267849.

Yehoshua Bar-Hillel, M. Perles, and Eliahu Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

Norbert Blum and Robert Koch. 1999. Greibach normal form transformation revisited. *Information and Computation* 150(1):112–118. https://doi.org/10.1006/inco.1998.2772.

Noam Chomsky and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, Elsevier, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pages 118–161.

Søren Christensen, Hans Hüttel, and Colin Stirling. 1995. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation* 121(2):143–148.

Pierluigi Crescenzi, Daniel Gildea, Andrea Marino, Gianluca Rossi, and Giorgio Satta. 2015. Synchronous context-free grammars and optimal linear parsing strategies. *Journal of Computer and System Sciences* 81(7):1333–1356. https://doi.org/10.1016/j.jcss.2015.04.003.

Marc Dymetman. 1992. A generalized greibach normal form for definite clause grammars. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '92, pages 366–372. https://doi.org/10.3115/992066.992126.

Joost Engelfriet, Andreas Maletti, and Sebastian Maneth. 2017. Multiple context-free tree grammars: Lexicalization and characterization. *arXiv preprint*. http://arxiv.org/abs/1707.03457.

Henning Fernau and Ralf Stiebe. 2002. Sequential grammars and automata with valences. *Theoretical Computer Science* 276(1):377–405. https://doi.org/10.1016/S0304-3975(01)00282-1.

James N. Gray and Michael A. Harrison. 1972. On the covering and reduction problems for context-free grammars. *Journal of the ACM* 19(4):675–698. https://doi.org/10.1145/321724.321732.

Sheila A. Greibach. 1965. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM* 12(1):42–52. https://doi.org/10.1145/321250.321254.

Aravind Joshi, Leon Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10(1):136–163. https://doi.org/10.1016/S0022-0000(75)80019-5.

Aravind Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 3: Beyond Words*, Springer-Verlag New York, Inc., New York, NY, USA, chapter 2, pages 69–124.

Miriam Kaeshammer. 2013. Synchronous linear context-free rewriting systems for machine translation. In M. Carpuat, L. Specia, and D. Wu, editors, *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation, SSST@NAACL-HLT 2013, Atlanta, GA, USA, 13 June 2013*. Association for Computational Linguistics, pages 68–77. http://aclweb.org/anthology/W/W13/W13-0808.pdf.

Marco Kuhlmann and Giorgio Satta. 2012. Tree-adjoining grammars are not closed under strong lexicalization. *Computational Linguistics* 38(3):617–629. https://doi.org/10.1162/COLI_a_00090.

Philip M. Lewis and Richard E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM* 15(3):465–488. https://doi.org/10.1145/321466.321477.

Andreas Maletti and Joost Engelfriet. 2012. Strong lexicalization of tree adjoining grammars. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*. The Association for Computational Linguistics, pages 506–515. http://www.aclweb.org/anthology/P12-1053.

Rebecca Nesson, Giorgio Satta, and Stuart M. Shieber. 2008. Optimal k-arization of synchronous tree-adjoining grammar. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 604–612. http://www.aclweb.org/anthology/P/P08/P08-1069.pdf.

James Rogers. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational*

Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '94, pages 155–162. https://doi.org/10.3115/981732.981754.

Daniel J. Rosenkrantz. 1967. Matrix equations and normal forms for context-free grammars. *Journal of the Association for Computing Machinery* 14(3):501–507.

Yves Schabes and Richard C. Waters. 1993. Lexicalized context-free grammars. In L. Schubert, editor, *31st Annual Meeting of the Association for Computational Linguistics, 22-26 June 1993, Ohio State University, Columbus, Ohio, USA, Proceedings.*. ACL, pages 121–129. http://aclweb.org/anthology/P/P93/P93-1017.pdf.

Yves Schabes and Richard C. Waters. 1995. Tree insertion grammar: Cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics* 21(4):479–513.

Eliahu Shamir. 1967. A representation theorem for algebraic and context-free power series in noncommuting variables. *Information and Control* 11(1/2):239–254. https://doi.org/10.1016/S0019-9958(67)90529-3.

Stuart M. Shieber. 1994. Restricting the weak-generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence* 10:371–385. https://doi.org/10.1111/j.1467-8640.1994.tb00003.x.

Maryam Siahbani, Baskaran Sankaran, and Anoop Sarkar. 2013. Efficient left-to-right hierarchical phrase-based translation with improved reordering. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1089–1099. http://www.aclweb.org/anthology/D13-1110.

Maryam Siahbani and Anoop Sarkar. 2014a. Expressive hierarchical rule extraction for left-to-right translation. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2014)., Vancouver, Canada*.

Maryam Siahbani and Anoop Sarkar. 2014b. Two improvements to left-to-right decoding for hierarchical phrase-based machine translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, pages 221–226. http://aclweb.org/anthology/D/D14/D14-1028.pdf.

Ben Swanson, Elif Yamangil, Eugene Charniak, and Stuart M. Shieber. 2013. A context free TAG variant. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*. The Association for Computational Linguistics, pages 302–310. http://aclweb.org/anthology/P/P13/P13-1030.pdf.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In N. Calzolari, C. Cardie, and P. Isabelle, editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computational Linguistics. http://aclweb.org/anthology/P06-1098.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics* 23(3):377–403.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In K. Knight, H. T. Ng, and K. Oflazer, editors, *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*. The Association for Computational Linguistics, pages 475–482. http://aclweb.org/anthology/P/P05/P05-1059.pdf.

Straight to the Tree: Constituency Parsing with Neural Syntactic Distance

Yikang Shen[*†]
MILA
University of Montréal

Zhouhan Lin[*†]
MILA
University of Montréal
AdeptMind Scholar

Athul Paul Jacob[†]
MILA
University of Waterloo

Alessandro Sordoni
Microsoft Research
Montréal, Canada

Aaron Courville and **Yoshua Bengio**
MILA
University of Montréal, CIFAR

Abstract

In this work, we propose a novel constituency parsing scheme. The model predicts a vector of real-valued scalars, named syntactic distances, for each split position in the input sentence. The syntactic distances specify the order in which the split points will be selected, recursively partitioning the input, in a top-down fashion. Compared to traditional shift-reduce parsing schemes, our approach is free from the potential problem of compounding errors, while being faster and easier to parallelize. Our model achieves competitive performance amongst single model, discriminative parsers in the PTB dataset and outperforms previous models in the CTB dataset.

1 Introduction

Devising fast and accurate constituency parsing algorithms is an important, long-standing problem in natural language processing. Parsing has been useful for incorporating linguistic prior in several related tasks, such as relation extraction, paraphrase detection (Callison-Burch, 2008), and more recently, natural language inference (Bowman et al., 2016) and machine translation (Eriguchi et al., 2017).

Neural network-based approaches relying on dense input representations have recently achieved competitive results for constituency parsing (Vinyals et al., 2015; Cross and Huang, 2016; Liu and Zhang, 2017b; Stern et al., 2017a). Generally speaking, either these approaches produce the parse tree sequentially, by governing the sequence of transitions in a transition-based parser (Nivre, 2004; Zhu et al., 2013; Chen and Manning, 2014; Cross and Huang, 2016), or use a chart-based approach by estimating non-linear potentials and performing exact structured inference by dynamic programming (Finkel et al., 2008; Durrett and Klein, 2015; Stern et al., 2017a).

[*] Equal contribution. Corresponding authors: yikang.shen@umontreal.ca, zhouhan.lin@umontreal.ca.
[†] Work done while at Microsoft Research, Montreal.

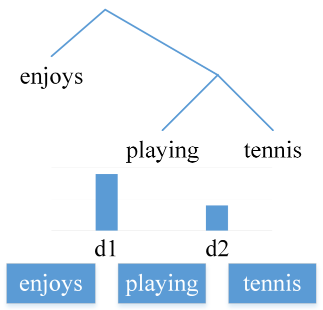


Figure 1: An example of how syntactic distances ($d1$ and $d2$) describe the structure of a parse tree: consecutive words with larger predicted distance are split earlier than those with smaller distances, in a process akin to divisive clustering.

Transition-based models decompose the structured prediction problem into a sequence of local decisions. This enables fast greedy decoding but also leads to compounding errors because the model is never exposed to its own mistakes during training (Daumé et al., 2009). Solutions to this problem usually complexify the training procedure by using structured training through beam-search (Weiss et al., 2015; Andor et al., 2016) and dynamic oracles (Goldberg and Nivre, 2012; Cross and Huang, 2016). On the other hand, chart-based models can incorporate structured loss functions during training and benefit from exact inference via the CYK algorithm but suffer from higher computational cost during decoding (Durrett and Klein, 2015; Stern et al., 2017a).

In this paper, we propose a novel, fully-parallel model for constituency parsing, based on the concept of "syntactic distance", recently introduced by (Shen et al., 2017) for language modeling. To construct a parse tree from a sentence, one can proceed in a top-down manner, recursively splitting larger constituents into smaller constituents, where the order of the splits defines the hierarchical structure. The syntactic distances are defined for each possible split point in the sentence. The order induced by the syntactic distances fully specifies the order in which the sentence needs to be recursively split into smaller constituents (Figure 1): in case of a binary tree, there exists a one-to-one correspondence between the ordering and the tree. Therefore, our model is trained to reproduce the ordering between split points induced by the ground-truth distances by means of a margin rank loss (Weston et al., 2011). Crucially, our model works *in parallel*: the estimated distance for each split point is produced independently from the others, which allows for an easy parallelization in modern parallel computing architectures for deep learning, such as GPUs. Along with the distances, we also train the model to produce the constituent labels, which are used to build the fully labeled tree.

Our model is fully parallel and thus does not require computationally expensive structured inference during training. Mapping from syntactic distances to a tree can be efficiently done in $\mathcal{O}(n \log n)$, which makes the decoding computationally attractive. Despite our strong conditional independence assumption on the output predictions, we achieve good performance for single model discriminative parsing in PTB (91.8 F1) and CTB (86.5 F1) matching, and sometimes outperforming, recent chart-based and transition-based parsing models.

2 Syntactic Distances of a Parse Tree

In this section, we start from the concept of syntactic distance introduced in Shen et al. (2017) for unsupervised parsing via language modeling and we extend it to the supervised setting. We propose two algorithms, one to convert a parse tree into a compact representation based on distances between consecutive words, and another to map the inferred representation back to a complete parse tree. The representation will later be used for supervised training. We formally define the syntactic distances of a parse tree as follows:

Algorithm 1 Binary Parse Tree to Distance
($\cup$ represents the concatenation operator of lists)

```
1: function DISTANCE(node)
2:     if node is leaf then
3:         $\mathbf{d} \leftarrow []$
4:         $\mathbf{c} \leftarrow []$
5:         $\mathbf{t} \leftarrow [\text{node.tag}]$
6:         $h \leftarrow 0$
7:     else
8:         $\text{child}_l, \text{child}_r \leftarrow$ children of node
9:         $\mathbf{d}_l, \mathbf{c}_l, \mathbf{t}_l, h_l \leftarrow \text{Distance}(\text{child}_l)$
10:        $\mathbf{d}_r, \mathbf{c}_r, \mathbf{t}_r, h_r \leftarrow \text{Distance}(\text{child}_r)$
11:        $h \leftarrow \max(h_l, h_r) + 1$
12:        $\mathbf{d} \leftarrow \mathbf{d}_l \cup [h] \cup \mathbf{d}_r$
13:        $\mathbf{c} \leftarrow \mathbf{c}_l \cup [\text{node.label}] \cup \mathbf{c}_r$
14:        $\mathbf{t} \leftarrow \mathbf{t}_l \cup \mathbf{t}_r$
15:    end if
16:    return $\mathbf{d}, \mathbf{c}, \mathbf{t}, h$
17: end function
```

Definition 2.1. Let $\mathbf{T}$ be a parse tree that contains a set of leaves $(w_0, ..., w_n)$. The height of the lowest common ancestor for two leaves (w_i, w_j) is noted as $\tilde{d}_j^i$. The syntactic distances of $\mathbf{T}$ can be any vector of scalars $\mathbf{d} = (d_1, ..., d_n)$ that satisfy:

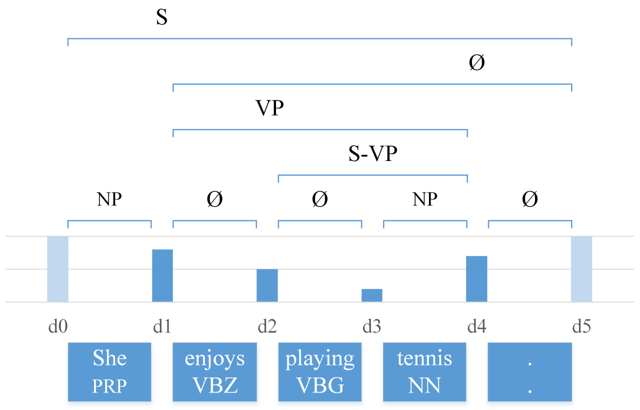
$$\text{sign}(d_i - d_j) = \text{sign}(\tilde{d}_i^{i-1} - \tilde{d}_j^{j-1}) \qquad (1)$$

In other words, $\mathbf{d}$ induces the same ranking order as the quantities $\tilde{d}_i^j$ computed between pairs of consecutive words in the sequence, i.e. $(\tilde{d}_1^0, ..., \tilde{d}_n^{n-1})$. Note that there are $n-1$ syntactic distances for a sentence of length n.

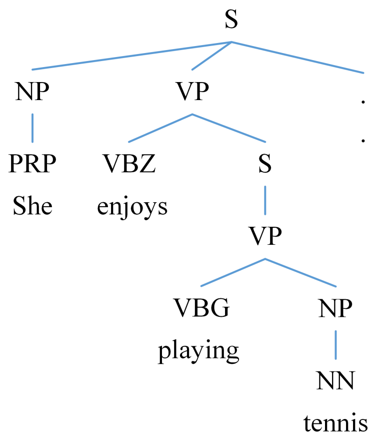
Example 2.1. Consider the tree in Fig. 1 for which $\tilde{d}_1^0 = 2$, $\tilde{d}_2^1 = 1$. An example of valid syntactic distances for this tree is any $\mathbf{d} = (d_1, d_2)$ such that $d_1 > d_2$.

Given this definition, the parsing model predicts a sequence of scalars, which is a more natural setting for models based on neural networks, rather than predicting a set of spans. For comparison, in most of the current neural parsing methods, the model needs to output a sequence of transitions (Cross and Huang, 2016; Chen and Manning, 2014).

Let us first consider the case of a binary parse tree. Algorithm 1 provides a way to convert it to a tuple $(\mathbf{d}, \mathbf{c}, \mathbf{t})$, where $\mathbf{d}$ contains the height of the inner nodes in the tree following a left-to-right (in order) traversal, $\mathbf{c}$ the constituent labels for each node in the same order and $\mathbf{t}$ the part-of-speech



(a) Boxes in the bottom are words and their corresponding POS tags predicted by an external tagger. The vertical bars in the middle are the syntactic distances, and the brackets on top of them are labels of constituents. The bottom brackets are the predicted unary label for each words, and the upper brackets are predicted labels for other constituent.



(b) The corresponding inferred grammar tree.

Figure 2: Inferring the parse tree with Algorithm 2 given distances, constituent labels, and POS tags. Starting with the full sentence, we pick split point 1 (as it is assigned to the larger distance) and assign label S to span (0,5). The left child span (0,1) is assigned with a tag PRP and a label NP, which produces an unary node and a terminal node. The right child span (1,5) is assigned the label ∅, coming from implicit binarization, which indicates that the span is not a real constituent and all of its children are instead direct children of its parent. For the span (1,5), the split point 4 is selected. The recursion of splitting and labeling continues until the process reaches a terminal node.

Algorithm 2 Distance to Binary Parse Tree

```
1:  function TREE($\mathbf{d}$,$\mathbf{c}$,$\mathbf{t}$)
2:      if $\mathbf{d} = []$ then
3:          node ← Leaf($\mathbf{t}$)
4:      else
5:          $i \leftarrow \arg\max_i(\mathbf{d})$
6:          child$_l$ ← Tree($\mathbf{d}_{<i}$, $\mathbf{c}_{<i}$, $\mathbf{t}_{<i}$)
7:          child$_r$ ← Tree($\mathbf{d}_{>i}$, $\mathbf{c}_{>i}$, $\mathbf{t}_{\geq i}$)
8:          node ← Node(child$_l$, child$_r$, $\mathbf{c}_i$)
9:      end if
10:     return node
11: end function
```

(POS) tags of each word in the left-to-right order. $\mathbf{d}$ is a valid vector of syntactic distances satisfying Definition 2.1.

Once a model has learned to predict these variables, Algorithm 2 can reconstruct a unique binary tree from the output of the model $(\hat{\mathbf{d}}, \hat{\mathbf{c}}, \hat{\mathbf{t}})$. The idea in Algorithm 2 is similar to the top-down parsing method proposed by Stern et al. (2017a), but differs in one key aspect: at each recursive call, there is no need to estimate the confidence for every split point. The algorithm simply chooses the split point i with the maximum $\hat{d}_i$, and assigns to the span the predicted label $\hat{c}_i$. This makes the running time of our algorithm to be in $\mathcal{O}(n \log n)$, compared to the $\mathcal{O}(n^2)$ of the greedy top-down algorithm by (Stern et al., 2017a). Figure 2 shows an example of the reconstruction of parse tree. Alternatively, the tree reconstruction process can also be done in a bottom-up manner, which requires the recursive composition of adjacent spans according to the ranking induced by their syntactic distance, a process akin to agglomerative clustering.

One potential issue is the existence of unary and n-ary nodes. We follow the method proposed by Stern et al. (2017a) and add a special empty label ∅ to spans that are not themselves full constituents but simply arise during the course of implicit binarization. For the unary nodes that contains one nonterminal node, we take the common approach of treating these as additional atomic labels alongside all elementary nonterminals (Stern et al., 2017a). For all terminal nodes, we determine whether it belongs to a unary chain or not by predicting an additional label. If it is predicted with a label different from the empty label, we conclude that it is a direct child of a unary constituent with that label. Otherwise if it is predicted to have an empty label, we conclude that it is a child of a bigger constituent which has other constituents or words as its siblings.

An n-ary node can arbitrarily be split into binary nodes. We choose to use the leftmost split point. The split point may also be chosen based on model prediction during training. Recovering an n-ary parse tree from the predicted binary tree simply requires removing the empty nodes and split combined labels corresponding to unary chains.

Algorithm 2 is a divide-and-conquer algorithm. The running time of this procedure is $\mathcal{O}(n \log n)$. However, the algorithm is naturally adapted for execution in a parallel environment, which can further reduce its running time to $\mathcal{O}(\log n)$.

3 Learning Syntactic Distances

We use neural networks to estimate the vector of syntactic distances for a given sentence. We use a modified hinge loss, where the target distances are generated by the tree-to-distance conversion given by Algorithm 1. Section 3.1 will describe in detail the model architecture, and Section 3.2 describes the loss we use in this setting.

3.1 Model Architecture

Given input words $\mathbf{w} = (w_0, w_1, ..., w_n)$, we predict the tuple $(\mathbf{d}, \mathbf{c}, \mathbf{t})$. The POS tags $\mathbf{t}$ are given by an external Part-Of-Speech (POS) tagger. The syntactic distances $\mathbf{d}$ and constituent labels $\mathbf{c}$ are predicted using a neural network architecture that stacks recurrent (LSTM (Hochreiter and Schmidhuber, 1997)) and convolutional layers.

Words and tags are first mapped to sequences of embeddings $\mathbf{e}_0^{\mathrm{w}}, ..., \mathbf{e}_n^{\mathrm{w}}$ and $\mathbf{e}_0^{\mathrm{t}}, ..., \mathbf{e}_n^{\mathrm{t}}$. Then the word embeddings and the tag embeddings are concatenated together as inputs for a stack of bidirectional LSTM layers:

$$\mathbf{h}_0^{\mathrm{w}}, ..., \mathbf{h}_n^{\mathrm{w}} = \mathrm{BiLSTM}_{\mathrm{w}}([\mathbf{e}_0^{\mathrm{w}}, \mathbf{e}_0^{\mathrm{t}}], ..., [\mathbf{e}_n^{\mathrm{w}}, \mathbf{e}_n^{\mathrm{t}}]) \tag{2}$$

where $\mathrm{BiLSTM}_{\mathrm{w}}(\cdot)$ is the word-level bidirectional layer, which gives the model enough capacity to capture long-term syntactical relations between words.

To predict the constituent labels for each word, we pass the hidden states representations $\mathbf{h}_0^{\mathrm{w}}, ..., \mathbf{h}_n^{\mathrm{w}}$ through a 2-layer network $\mathrm{FF}_c^{\mathrm{w}}$, with softmax output:

$$p(c_i^{\mathbf{w}}|\mathbf{w}) = \mathrm{softmax}(\mathrm{FF}_c^{\mathrm{w}}(\mathbf{h}_i^{\mathrm{w}})) \tag{3}$$

To compose the necessary information for inferring the syntactic distances and the constituency label information, we perform an additional convolution:

$$\mathbf{g}_1^{\mathrm{s}}, \ldots, \mathbf{g}_n^{\mathrm{s}} = \mathrm{CONV}(\mathbf{h}_0^{\mathrm{w}}, ..., \mathbf{h}_n^{\mathrm{w}}) \tag{4}$$

where $\mathbf{g}_i^{\mathrm{s}}$ can be seen as a draft representation for each split position in Algorithm 2. Note that the subscripts of g_i^ss start with 1, since we have $n-1$ positions as non-terminal constituents. Then, we stack a bidirectional LSTM layer on top of $\mathbf{g}_i^{\mathrm{s}}$:

$$\mathbf{h}_1^{\mathrm{s}}, ..., \mathbf{h}_n^{\mathrm{s}} = \mathrm{BiLSTM}_{\mathrm{s}}(\mathbf{g}_1^{\mathrm{s}}, \ldots, \mathbf{g}_n^{\mathrm{s}}) \tag{5}$$

where $\mathrm{BiLSTM}_{\mathrm{s}}$ fine-tunes the representation by conditioning on other split position representations. Interleaving between LSTM and convolution layers turned out empirically to be the best choice over multiple variations of the model, including using self-attention (Vaswani et al., 2017) instead of LSTM.

To calculate the syntactic distances for each position, the vectors $\mathbf{h}_1^{\mathrm{s}}, \ldots, \mathbf{h}_n^{\mathrm{s}}$ are transformed through a 2-layer feed-forward network FF_d with a single output unit (this can be done in parallel with 1x1 convolutions), with no activation function at the output layer:

$$\hat{d}_i = \mathrm{FF}_d(\mathbf{h}_i^s), \tag{6}$$

For predicting the constituent labels, we pass the same representations $\mathbf{h}_1^{\mathrm{s}}, \ldots, \mathbf{h}_n^{\mathrm{s}}$ through another 2-layer network $\mathrm{FF}_c^{\mathrm{s}}$, with softmax output.

$$p(c_i^{\mathrm{s}}|\mathbf{w}) = \mathrm{softmax}(\mathrm{FF}_c^{\mathrm{s}}(\mathbf{h}_i^{\mathrm{s}})) \tag{7}$$

The overall architecture is shown in Figure 2a. Since the output $(\mathbf{d}, \mathbf{c}, \mathbf{t})$ can be unambiguously transfered to a unique parse tree, the model implicitly makes all parsing decisions inside the recurrent and convolutional layers.

3.2 Objective

Given a set of training examples $\mathcal{D} = \{\langle \mathbf{d}_k, \mathbf{c}_k, \mathbf{t}_k, \mathbf{w}_k \rangle\}_{k=1}^K$, the training objective is the sum of the prediction losses of syntactic distances $\mathbf{d}_k$ and constituent labels $\mathbf{c}_k$.

Due to the categorical nature of variable $\mathbf{c}$, we use a standard softmax classifier with a cross-entropy loss L_{label} for constituent labels, using the estimated probabilities obtained in Eq. 3 and 7.

A naïve loss function for estimating syntactic distances is the mean-squared error (MSE):

$$L_{\text{dist}}^{\text{mse}} = \sum_i (d_i - \hat{d}_i)^2 \tag{8}$$

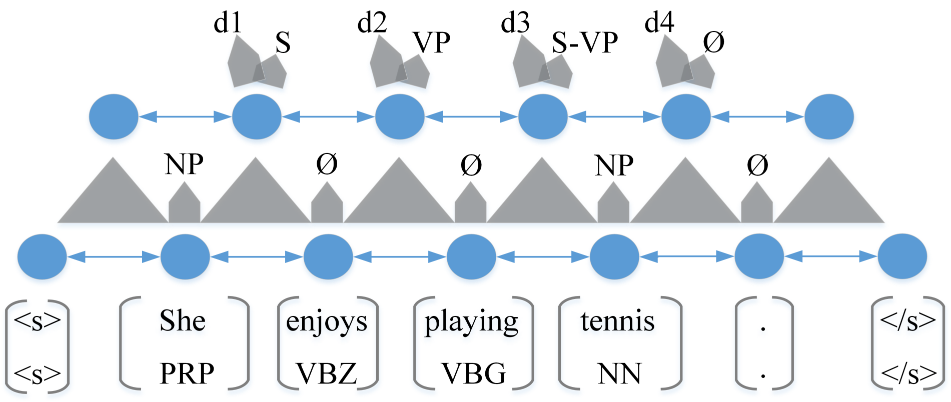


Figure 3: The overall visualization of our model. Circles represent hidden states, triangles represent convolution layers, block arrows represent feed-forward layers, arrows represent recurrent connections. The bottom part of the model predicts unary labels for each input word. The $\emptyset$ is treated as a special label together with other labels. The top part of the model predicts the syntactic distances and the constituent labels. The inputs of model are the word embeddings concatenated with the POS tag embeddings. The tags are given by an external Part-Of-Speech tagger.

The MSE loss forces the model to regress on the exact value of the true distances. Given that only the *ranking* induced by the ground-truth distances in $\mathbf{d}$ is important, as opposed to the absolute values themselves, using an MSE loss over-penalizes the model by ignoring ranking equivalence between different predictions.

Therefore, we propose to minimize a pair-wise learning-to-rank loss, similar to those proposed in (Burges et al., 2005). We define our loss as a variant of the hinge loss as:

$$L_{\text{dist}}^{\text{rank}} = \sum_{i,j>i} [1 - \text{sign}(d_i - d_j)(\hat{d}_i - \hat{d}_j)]^+, \quad (9)$$

where $[x]^+$ is defined as $\max(0, x)$. This loss encourages the model to reproduce the full ranking order induced by the ground-truth distances. The final loss for the overall model is just the sum of individual losses $L = L_{\text{label}} + L_{\text{dist}}^{\text{rank}}$.

4 Experiments

We evaluate our model described above on 2 different datasets, the standard Wall Street Journal (WSJ) part of the Penn Treebank (PTB) dataset, and the Chinese Treebank (CTB) dataset.

For evaluating the F1 score, we use the standard `evalb`[1] tool. We provide both labeled and unlabeled F1 score, where the former takes into consideration the constituent label for each predicted constituent, while the latter only considers the position of the constituents. In the tables below, we report the labeled F1 scores for comparison with previous work, as this is the standard metric usually reported in the relevant literature.

4.1 Penn Treebank

For the PTB experiments, we follow the standard train/valid/test separation and use sections 2-21 for training, section 22 for development and section 23 for test set. Following this split, the dataset has 45K training sentences and 1700, 2416 sentences for valid/test respectively. The placeholders with the `-NONE-` tag are stripped from the dataset during preprocessing. The POS tags are predicted with the Stanford Tagger (Toutanova et al., 2003).

We use a hidden size of 1200 for each direction on all LSTMs, with 0.3 dropout in all the feed-forward connections, and 0.2 recurrent connection dropout (Merity et al., 2017). The convolutional filter size is 2. The number of convolutional channels is 1200. As a common practice for neural network based NLP models, the embedding layer that maps word indexes to word embeddings is randomly initialized. The word embeddings are sized 400. Following (Merity et al., 2017), we randomly swap an input word embedding during training with the zero vector with probability of 0.1. We found this helped the model to generalize better. Training is conducted with Adam algorithm with l2 regularization decay 1×10^{-6}. We pick the result obtaining the highest labeled F1

[1] `http://nlp.cs.nyu.edu/evalb/`

Model	LP	LR	F1
Single Model			
Vinyals et al. (2015)	-	-	88.3
Zhu et al. (2013)	90.7	90.2	90.4
Dyer et al. (2016)	-	-	89.8
Watanabe and Sumita (2015)	-	-	90.7
Cross and Huang (2016)	92.1	90.5	91.3
Liu and Zhang (2017b)	92.1	91.3	91.7
Stern et al. (2017a)	93.2	90.3	91.8
Liu and Zhang (2017a)	-	-	91.8
Gaddy et al. (2018)	-	-	92.1
Stern et al. (2017b)	92.5	92.5	92.5
Our Model	92.0	91.7	91.8
Ensemble			
Shindo et al. (2012)	-	-	92.4
Vinyals et al. (2015)	-	-	90.5
Semi-supervised			
Zhu et al. (2013)	91.5	91.1	91.3
Vinyals et al. (2015)	-	-	92.8
Re-ranking			
Charniak and Johnson (2005)	91.8	91.2	91.5
Huang (2008)	91.2	92.2	91.7
Dyer et al. (2016)	-	-	93.3

Table 1: Results on the PTB dataset WSJ test set, Section 23. LP, LR represents labeled precision and recall respectively.

on the validation set, and report the corresponding test F1, together with other statistics. We report our results in Table 1. Our best model obtains a labeled F1 score of 91.8 on the test set (Table 1). Detailed dev/test set performances, including label accuracy is reported in Table 3.

Our model performs achieves good performance for single-model constituency parsing trained without external data. The best result from (Stern et al., 2017b) is obtained by a generative model. Very recently, we came to knowledge of Gaddy et al. (2018), which uses character-level LSTM features coupled with chart-based parsing to improve performance. Similar sub-word features can be also used in our model. We leave this investigation for future works. For comparison, other models obtaining better scores either use ensembles, benefit from semi-supervised learning, or recur to re-ranking of a set of candidates.

4.2 Chinese Treebank

We use the Chinese Treebank 5.1 dataset, with articles 001-270 and 440-1151 for training, articles

Model	LP	LR	F1
Single Model			
Charniak (2000)	82.1	79.6	80.8
Zhu et al. (2013)	84.3	82.1	83.2
Wang et al. (2015)	-	-	83.2
Watanabe and Sumita (2015)	-	-	84.3
Dyer et al. (2016)	-	-	84.6
Liu and Zhang (2017b)	85.9	85.2	85.5
Liu and Zhang (2017a)	-	-	86.1
Our Model	86.6	86.4	86.5
Semi-supervised			
Zhu et al. (2013)	86.8	84.4	85.6
Wang and Xue (2014)	-	-	86.3
Wang et al. (2015)	-	-	86.6
Re-ranking			
Charniak and Johnson (2005)	83.8	80.8	82.3
Dyer et al. (2016)	-	-	86.9

Table 2: Test set performance comparison on the CTB dataset

301-325 as development set, and articles 271-300 for test set. This is a standard split in the literature (Liu and Zhang, 2017b). The `-NONE-` tags are stripped as well. The hidden size for the LSTM networks is set to 1200. We use a dropout rate of 0.4 on the feed-forward connections, and 0.1 recurrent connection dropout. The convolutional layer has 1200 channels, with a filter size of 2. We use 400 dimensional word embeddings. During training, input word embeddings are randomly swapped with the zero vector with probability of 0.1. We also apply a l2 regularization weighted by 1×10^{-6} on the parameters of the network. Table 2 reports our results compared to other benchmarks. To the best of our knowledge, we set a new state-of-the-art for single-model parsing achieving 86.5 F1 on the test set. The detailed statistics are shown in Table 3.

4.3 Ablation Study

We perform an ablation study by removing components from a network trained with the best set of hyperparameters, and re-train the ablated version from scratch. This gives an idea of the relative contributions of each of the components in the model. Results are reported in Table 4. It seems that the top LSTM layer has a relatively big impact on performance. This may give additional capacity to the model for capturing long-term dependencies useful for label prediction. We also exper-

dev/test result		Prec.	Recall	F1	label accuracy
PTB	labeled	91.7/92.0	91.8/91.7	91.8/91.8	94.9/95.4%
	unlabeled	93.0/93.2	93.0/92.8	93.0/93.0	
CTB	labeled	89.4/86.6	89.4/86.4	89.4/86.5	92.2/91.1%
	unlabeled	91.1/88.9	91.1/88.6	91.1/88.8	

Table 3: Detailed experimental results on PTB and CTB datasets

Model	LP	LR	F1
Full model	92.0	91.7	91.8
w/o top LSTM	91.0	90.5	90.7
w. embedding	91.9	91.6	91.7
w. MSE loss	90.3	90.0	90.1

Table 4: Ablation test on the PTB dataset. "w/o top LSTM" is the full model without the top LSTM layer. "w. embedding" stands for the full model using the pretrained word embeddings. "w. MSE loss" stands for the full model trained with MSE loss.

Model	# sents/sec
Petrov and Klein (2007)	6.2
Zhu et al. (2013)	89.5
Liu and Zhang (2017b)	79.2
Stern et al. (2017a)	75.5
Our model	111.1
Our model w/o tree inference	351

Table 5: Parsing speed in sentences per second on the PTB dataset.

imented by using 300D GloVe (Pennington et al., 2014) embedding for the input layer but this didn't yield improvements over the model's best performance. Unsurprisingly, the model trained with MSE loss underperforms considerably a model trained with the rank loss.

4.4 Parsing Speed

The prediction of syntactic distances can be batched in modern GPU architectures. The distance to tree conversion is a $\mathcal{O}(n \log n)$ (n stand for the number of words in the input sentence) divide-and-conquer algorithm. We compare the parsing speed of our parser with other state-of-the-art neural parsers in Table 5. As the syntactic distance computation can be performed in parallel within a GPU, we first compute the distances in a batch, then we iteratively decode the tree with Algorithm 2. It is worth to note that this comparison may be unfair since some of the reported results may use very different hardware settings. We couldn't find the source code to re-run them on our hardware, to give a fair enough comparison. In our setting, we use an NVIDIA TITAN Xp graphics card for running the neural network part, and the distance to tree inference is run on an Intel Core i7-6850K CPU, with 3.60GHz clock speed.

5 Related Work

Parsing natural language with neural network models has recently received growing attention. These models have attained state-of-the-art results for dependency parsing (Chen and Manning, 2014) and constituency parsing (Dyer et al., 2016; Cross and Huang, 2016; Coavoux and Crabbé, 2016). Early work in neural network based parsing directly use a feed-forward neural network to predict parse trees (Chen and Manning, 2014). Vinyals et al. (2015) use a sequence-to-sequence framework where the decoder outputs a linearized version of the parse tree given an input sentence. Generally, in these models, the correctness of the output tree is not strictly ensured (although empirically observed).

Other parsing methods ensure structural consistency by operating in a transition-based setting (Chen and Manning, 2014) by parsing either in the top-down direction (Dyer et al., 2016; Liu and Zhang, 2017b), bottom-up (Zhu et al., 2013; Watanabe and Sumita, 2015; Cross and Huang, 2016) and recently in-order (Liu and Zhang, 2017a). Transition-based methods generally suffer from compounding errors due to exposure bias: during testing, the model is exposed to a very different regime (i.e. decisions sampled from the model itself) than what was encountered during training (i.e. the ground-truth decisions) (Daumé et al., 2009; Goldberg and Nivre, 2012). This can have catastrophic effects on test performance but can be mitigated to a certain extent by using beam-search instead of greedy decoding. (Stern et al., 2017b) proposes an effective inference method for generative parsing, which enables direct decoding in those models. More complex training methods have been devised in order to alleviate this problem (Goldberg and Nivre, 2012; Cross and Huang, 2016). Other efforts have been put into neural chart-based parsing (Durrett and Klein, 2015; Stern et al., 2017a) which ensure structural consistency and offer exact inference with CYK algorithm. (Gaddy et al., 2018) includes a simplified CYK-style inference, but the complexity still remains in $O(n^3)$.

In this work, our model learns to produce a particular representation of a tree in parallel. Representations can be computed in parallel, and the conversion from representation to a full tree can efficiently be done with a divide-and-conquer algorithm. As our model outputs decisions in parallel, our model doesn't suffer from the exposure bias. Interestingly, a series of recent works, both in machine translation (Gu et al., 2018) and speech synthesis (Oord et al., 2017), considered the sequence of output variables conditionally independent given the inputs.

6 Conclusion

We presented a novel constituency parsing scheme based on predicting real-valued scalars, named syntactic distances, whose ordering identify the sequence of top-down split decisions. We employ a neural network model that predicts the distances $\mathbf{d}$ and the constituent labels $\mathbf{c}$. Given the algorithms presented in Section 2, we can build an unambiguous mapping between each $(\mathbf{d}, \mathbf{c}, \mathbf{t})$ and a parse tree. One peculiar aspect of our model is that it predicts split decisions *in parallel*. Our experiments show that our model can achieve strong performance compare to previous models, while being significantly more efficient. Since the architecture of model is no more than a stack of standard recurrent and convolution layers, which are essential components in most academic and industrial deep learning frameworks, the deployment of this method would be straightforward.

Acknowledgement

The authors would like to thank Compute Canada for providing the computational resources. The authors would also like to thank Jackie Chi Kit Cheung for the helpful discussions. Zhouhan Lin would like to thank AdeptMind for generously supporting his research via scholarship.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2442–2452.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1466–1477.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*. pages 89–96.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 196–205.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics, pages 132–139.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 173–180.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 740–750.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 172–182.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–âĂŞ11.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning* 75(3):297–325.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 302–312.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 199âĂŞ–209.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 72–78.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*. Association for Computational Linguistics, pages 959–967.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. WhatâĂŹs going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*. pages 959–976.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proceedings of International Conference on Learning Representations*.

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

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 586–594.

Jiangming Liu and Yue Zhang. 2017a. In-order transition-based constituent parsing. *Transactions of the Association of Computational Linguistics* 5(1):413–424.

Jiangming Liu and Yue Zhang. 2017b. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics* 5:45–58.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182* .

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. 2017. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. pages 404–411.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. In *Proceedings of the International Conference on Learning Representations*.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 440–448.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017a. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 818–827.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017b. Effective inference for generative neural parsing. *arXiv preprint arXiv:1707.08976* .

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 173–180.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Zhiguo Wang, Haitao Mi, and Nianwen Xue. 2015. Feature optimization for constituent parsing via neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1138–1147.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 733–742.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Volume 1, Long Papers*. pages 1169–1179.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158* .

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. pages 2764–2770.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 434–443.

Gaussian Mixture Latent Vector Grammars

Yanpeng Zhao, Liwen Zhang, Kewei Tu
School of Information Science and Technology,
ShanghaiTech University, Shanghai, China
{zhaoyp1,zhanglw1,tukw}@shanghaitech.edu.cn

Abstract

We introduce Latent Vector Grammars (LVeGs), a new framework that extends latent variable grammars such that each nonterminal symbol is associated with a continuous vector space representing the set of (infinitely many) subtypes of the nonterminal. We show that previous models such as latent variable grammars and compositional vector grammars can be interpreted as special cases of LVeGs. We then present Gaussian Mixture LVeGs (GM-LVeGs), a new special case of LVeGs that uses Gaussian mixtures to formulate the weights of production rules over subtypes of nonterminals. A major advantage of using Gaussian mixtures is that the partition function and the expectations of subtype rules can be computed using an extension of the inside-outside algorithm, which enables efficient inference and learning. We apply GM-LVeGs to part-of-speech tagging and constituency parsing and show that GM-LVeGs can achieve competitive accuracies. Our code is available at https://github.com/zhaoyanpeng/lveg.

1 Introduction

In constituency parsing, refining coarse syntactic categories of treebank grammars (Charniak, 1996) into fine-grained subtypes has been proven effective in improving parsing results. Previous approaches to refining syntactic categories use tree annotations (Johnson, 1998), lexicalization (Charniak, 2000; Collins, 2003), or linguistically motivated category splitting (Klein and Manning, 2003). Matsuzaki et al. (2005) introduce latent variable grammars, in which each syntactic category (represented by a nonterminal) is split into a fixed number of subtypes and a discrete latent variable is used to indicate the subtype of the nonterminal when it appears in a specific parse tree. Since the latent variables are not observable in treebanks, the grammar is learned using expectation-maximization. Petrov et al. (2006) present a split-merge approach to learning latent variable grammars, which hierarchically splits each nonterminal and merges ineffective splits. Petrov and Klein (2008b) further allow a nonterminal to have different splits in different production rules, which results in a more compact grammar.

Recently, neural approaches become very popular in natural language processing (NLP). An important technique in neural approaches to NLP is to represent discrete symbols such as words and syntactic categories with continuous vectors or embeddings. Since the distances between such vector representations often reflect the similarity between the corresponding symbols, this technique facilitates more informed smoothing in learning functions of symbols (e.g., the probability of a production rule). In addition, what a symbol represents may subtly depend on its context, and a continuous vector representation has the potential of representing each instance of the symbol in a more precise manner. For constituency parsing, recursive neural networks (Socher et al., 2011) and their extensions such as compositional vector grammars (Socher et al., 2013) can be seen as representing nonterminals in a context-free grammar with continuous vectors. However, exact inference in these models is intractable.

In this paper, we introduce latent vector grammars (LVeGs), a novel framework of grammars with fine-grained nonterminal subtypes. A LVeG associates each nonterminal with a continuous vector space that represents the set of (infinitely many) subtypes of the nonterminal. For each instance of a nonterminal that appears in a parse tree, its subtype is represented by a latent vector. For each production rule over nonterminals, a non-negative continuous function specifies the weight of any fine-grained production rule over subtypes of the nonterminals. Compared with latent variable grammars which assume a small fixed number of subtypes for each nonterminal, LVeGs assume an unlimited number of subtypes and are potentially more expressive. By having weight functions of varying smoothness for different production rules, LVeGs can also control the level of subtype granularity for different productions, which has been shown to improve the parsing accuracy (Petrov and Klein, 2008b). In addition, similarity between subtypes of a nonterminal can be naturally modeled by the distance between the corresponding vectors, so by using continuous and smooth weight functions we can ensure that similar subtypes will have similar syntactic behaviors.

We further present Gaussian Mixture LVeGs (GM-LVeGs), a special case of LVeGs that uses mixtures of Gaussian distributions as the weight functions of fine-grained production rules. A major advantage of GM-LVeGs is that the partition function and the expectations of fine-grained production rules can be computed using an extension of the inside-outside algorithm. This makes it possible to efficiently compute the gradients during discriminative learning of GM-LVeGs. We evaluate GM-LVeGs on part-of-speech tagging and constituency parsing on a variety of languages and corpora and show that GM-LVeGs achieve competitive results.

It shall be noted that many modern state-of-the-art constituency parsers predict how likely a constituent is based on not only local information (such as the production rules used in composing the constituent), but also contextual information of the constituent. For example, the neural CRF parser (Durrett and Klein, 2015) looks at the words before and after the constituent; and RNNG (Dyer et al., 2016) looks at the constituents that are already predicted (in the stack) and the words that are not processed (in the buffer). In this paper, however, we choose to focus on the basic framework and algorithms of LVeGs and leave the incorporation of contextual information for future work. We believe that by laying a solid foundation for LVeGs, our work can pave the way for many interesting extensions of LVeGs in the future.

2 Latent Vector Grammars

A latent vector grammar (LVeG) considers subtypes of nonterminals as continuous vectors and associates each nonterminal with a latent vector space representing the set of its subtypes. For each production rule, the LVeG defines a weight function over the subtypes of the nonterminal involved in the production rule. In this way, it models the space of refinements of the production rule.

2.1 Model Definition

A latent vector grammar is defined as a 5-tuple $\mathcal{G} = (N, S, \Sigma, R, W)$, where N is a finite set of nonterminal symbols, $S \in N$ is the start symbol, Σ is a finite set of terminal symbols such that $N \cap \Sigma = \varnothing$, R is a set production rules of the form $X \to \gamma$ where $X \in N$ and $\gamma \in (N \cup \Sigma)^*$, W is a set of rule weight functions indexed by production rules in R (to be defined below). In the following discussion, we consider R in the Chomsky normal form (CNF) for clarity of presentation. However, it is straightforward to extend our formulation to the general case.

Unless otherwise specified, we always use capital letters $A, B, C, \dots$ for nonterminal symbols and use bold lowercase letters $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$ for their subtypes. Note that subtypes are represented by continuous vectors. For a production rule of the form $A \to BC$, its weight function is $W_{A \to BC}(\mathbf{a}, \mathbf{b}, \mathbf{c})$. For a production rule of the form $A \to w$ where $w \in \Sigma$, its weight function is $W_{A \to w}(\mathbf{a})$. The weight functions should be non-negative, continuous and smooth, and hence fine-grained production rules of similar subtypes of a nonterminal would have similar weight assignments. Rule weights can be normalized such that $\sum_{B,C} \int_{\mathbf{b},\mathbf{c}} W_{A \to BC}(\mathbf{a}, \mathbf{b}, \mathbf{c}) d\mathbf{b} d\mathbf{c} = 1$, which leads to a probabilistic context-free grammar (PCFG). Whether the weights are normalized or not leads to different model classes and accordingly different estimation methods. However, the two model classes are proven equivalent by Smith and Johnson (2007).

2.2 Relation to Other Models

Latent variable grammars (LVGs) (Matsuzaki et al., 2005; Petrov et al., 2006) associate each nonterminal with a discrete latent variable, which is used to indicate the subtype of the nonterminal when it appears in a parse tree. Through nonterminal-splitting and the expectation-maximization algorithm, fine-grained production rules can be automatically induced from a treebank.

We show that LVGs can be seen as a special case of LVeGs. Specifically, we can use one-hot vectors in LVeGs to represent latent variables in LVGs and define weight functions in LVeGs accordingly. Consider a production rule $r : A \to BC$. In a LVG, each nonterminal is split into a number of subtypes. Suppose A, B, and C are split into n_A, n_B, and n_C subtypes respectively. a_x is the x-th subtype of A, b_y is the y-th subtype of B, and c_z is the z-th subtype of C. $a_x \to b_y c_z$ is a fine-grained production rule of $A \to BC$, where $x = 1, \ldots, n_A$, $y = 1, \ldots, n_B$, and $z = 1, \ldots, n_C$. The probabilities of all the fine-grained production rules can be represented by a rank-3 tensor $\Theta_{A \to BC} \in \mathbb{R}^{n_A \times n_B \times n_C}$. To cast the LVG as a LVeG, we require that the latent vectors in the LVeG must be one-hot vectors. We achieve this by defining weight functions that output zero if any of the input vectors is not one-hot. Specifically, we define the weight function of the production rule $A \to BC$ as:

$$W_r(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \sum_{x,y,z} \Theta_{A \to BC} \mathbf{cba} \times (\delta(\mathbf{a} - \mathbf{a}_x) \times \delta(\mathbf{b} - \mathbf{b}_y) \times \delta(\mathbf{c} - \mathbf{c}_z)) \,, \qquad (1)$$

where $\delta(\cdot)$ is the Dirac delta function, $\mathbf{a}_x \in \mathbb{R}^{n_A}$, $\mathbf{b}_y \in \mathbb{R}^{n_B}$, $\mathbf{c}_z \in \mathbb{R}^{n_C}$ are one-hot vectors (which are zero everywhere with the exception of a single 1 at the x-th index of $\mathbf{a}_x$, the y-th index of $\mathbf{b}_y$, and the z-th index of $\mathbf{c}_z$) and $\Theta_{A \to BC}$ is multiplied sequentially by $\mathbf{c}$, $\mathbf{b}$, and $\mathbf{a}$.

Compared with LVGs, LVeGs have the following advantages. While a LVG contains a finite, typically small number of subtypes for each nonterminal, a LVeG uses a continuous space to represent an infinite number of subtypes. When equipped with weight functions of sufficient complexity, LVeGs can represent more fine-grained syntactic categories and production rules than LVGs. By controlling the complexity and smoothness of the weight functions, a LVeG is also capable of representing any level of subtype granularity. Importantly, this allows us to change the level of subtype granularity for the same nonterminal in different production rules, which is similar to multi-scale grammars (Petrov and Klein, 2008b). In addition, with a continuous space of subtypes in a LVeG, similarity between subtypes can be naturally modeled by their distance in the space and can be automatically learned from data. Consequently, with continuous and smooth weight functions, fine-grained production rules over similar subtypes would have similar weights in LVeGs, eliminating the need for the extra smoothing steps that are necessary in training LVGs.

Compositional vector grammars (CVGs) (Socher et al., 2013), an extension of recursive neural networks (RNNs) (Socher et al., 2011), can also be seen as a special case of LVeGs. For a production rule $r : A \to BC$, a CVG can be interpreted as specifying its weight function $W_r(\mathbf{a}, \mathbf{b}, \mathbf{c})$ in the following way. First, a neural network f indexed by B and C is used to compute a parent vector $\mathbf{p} = f_{BC}(\mathbf{b}, \mathbf{c})$. Next, the score of the parent vector is computed using a base PCFG and a vector $\mathbf{v}_{BC}$:

$$s(\mathbf{p}) = \mathbf{v}_{BC}^T \mathbf{p} + \log P(A \to BC) \,, \qquad (2)$$

where $P(A \to BC)$ is the rule probability from the base PCFG. Then, the weight function of the production rule $A \to BC$ is defined as:

$$W_r(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \exp\left(s(\mathbf{p})\right) \times \delta(\mathbf{a} - \mathbf{p}) \,. \qquad (3)$$

This form of weight functions in CVGs leads to point estimation of latent vectors in a parse tree, i.e., for each nonterminal in a given parse tree, only one subtype in the whole subtype space would lead to a non-zero weight of the parse. In addition, different parse trees of the same substring typically lead to different point estimations of the subtype vector at the root nonterminal. Consequently, CVGs cannot use dynamic programming for inference and hence have to resort to greedy search or beam search.

3 Gaussian Mixture LVeGs

A major challenge in applying LVeGs to parsing is that it is impossible to enumerate the infinite number of subtypes. Previous work such as CVGs resorts to point estimation and greedy search. In this section we present Gaussian Mixture LVeGs (GM-LVeGs), which use mixtures of Gaussian distributions as the weight functions in LVeGs. Because Gaussian mixtures have the nice property of being closed under product, summation, and marginalization, we can compute the partition function and the expectations of fine-grained production rules using dynamic programming. This in turn makes efficient learning and parsing possible.

3.1 Representation

In a GM-LVeG, the weight function of a production rule r is defined as a Gaussian mixture containing K_r mixture components:

$$W_r(\mathbf{r}) = \sum_{k=1}^{K_r} \rho_{r,k} \mathcal{N}(\mathbf{r}|\boldsymbol{\mu}_{r,k}, \boldsymbol{\Sigma}_{r,k}) \,, \tag{4}$$

where $\mathbf{r}$ is the concatenation of the latent vectors of the nonterminals in r, which denotes a fine-grained production rule of r. $\rho_{r,k} > 0$ is the k-th mixture weight (the mixture weights do not necessarily sum up to 1), $\mathcal{N}(\mathbf{r}|\boldsymbol{\mu}_{r,k}, \boldsymbol{\Sigma}_{r,k})$ is the k-th Gaussian distribution parameterized by mean $\boldsymbol{\mu}_{r,k}$ and covariance matrix $\boldsymbol{\Sigma}_{r,k}$, and K_r is the number of mixture components, which can be different for different production rules. Below we write $\mathcal{N}(\mathbf{r}|\boldsymbol{\mu}_{r,k}, \boldsymbol{\Sigma}_{r,k})$ as $\mathcal{N}_{r,k}(\mathbf{r})$ for brevity. Given a production rule of the form $A \to BC$, the GM-LVeG expects $\mathbf{r} = [\mathbf{a}; \mathbf{b}; \mathbf{c}]$ and $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$, where d is the dimension of the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$. We use the same dimension for all the subtype vectors.

For the sake of computational efficiency, we use diagonal or spherical Gaussian distributions, whose covariance matrices are diagonal, so that the inverse of covariance matrices in Equation 15–16 can be computed in linear time. A spherical Gaussian has a diagonal covariance matrix where all the diagonal elements are equal, so it has fewer free parameters than a diagonal Gaussian and results in faster learning and parsing. We empirically find that spherical Gaussians lead to slightly better balance between the efficiency and the parsing accuracy than diagonal Gaussians.

3.2 Parsing

The goal of parsing is to find the most probable parse tree T^* with unrefined nonterminals for a sentence $\mathbf{w}$ of n words $w_{1:n} = w_1 \ldots w_n$. This is formally defined as:

$$T^* = \operatorname*{argmax}_{T \in G(\mathbf{w})} P(T|\mathbf{w}) \,, \tag{5}$$

where $G(\mathbf{w})$ denotes the set of parse trees with unrefined nonterminals for $\mathbf{w}$. In a PCFG, T^* can be found using dynamic programming such as the CYK algorithm. However, parsing becomes intractable with LVeGs, and even with LVGs, the special case of LVeGs.

A common practice in parsing with LVGs is to use max-rule parsing (Petrov et al., 2006; Petrov and Klein, 2007). The basic idea of max-rule parsing is to decompose the posteriors over parses into the posteriors over production rules approximately. This requires calculating the expected counts of unrefined production rules in parsing the input sentence. Since Gaussian mixtures are closed under product, summation, and marginalization, in GM-LVeGs the expected counts can be calculated using the inside-outside algorithm in the following way. Given a sentence $w_{1:n}$, we first calculate the inside score $s_{\mathbf{I}}^A(\mathbf{a}, i, j)$ and outside score $s_{\mathbf{O}}^A(\mathbf{a}, i, j)$ for a nonterminal A over a span $w_{i:j}$ using Equation 6 and Equation 7 in Table 1 respectively. Note that both $s_{\mathbf{I}}^A(\mathbf{a}, i, j)$ and $s_{\mathbf{O}}^A(\mathbf{a}, i, j)$ are mixtures of Gaussian distributions of the subtype vector $\mathbf{a}$. Next, using Equation 8 in Table 1, we calculate the score $s(A \to BC, i, k, j)$ $(1 \le i \le k < j \le n)$, where $\langle A \to BC, i, k, j \rangle$ represents a production rule $A \to BC$ with nonterminals A, B, and C spanning words $w_{i:j}$, $w_{i,k}$, and $w_{k+1:j}$ respectively in the sentence $w_{1:n}$. Then the expected count (or posterior) of $\langle A \to BC, i, k, j \rangle$ is calculated as:

$$q(A \to BC, i, k, j) = \frac{s(A \to BC, i, k, j)}{s_{\mathbf{I}}(S, 1, n)} \,, \tag{9}$$

where $s_{\mathbf{I}}(S, 1, n)$ is the inside score for the start symbol S spanning the whole sentence $w_{1:n}$. After calculating all the expected counts, we can use the MAX-RULE-PRODUCT algorithm (Petrov and Klein, 2007) for parsing, which returns a parse with the highest probability that all the production rules are correct. Its objective function is given by

$$T_q^* = \operatorname*{argmax}_{T \in G(\mathbf{w})} \prod_{e \in T} q(e) \,, \tag{10}$$

where e ranges over all the 4-tuples $\langle A \to BC, i, k, j \rangle$ in the parse tree T. This objective function can be efficiently solved by dynamic programming such as the CYK algorithm.

Although the time complexity of the inside-outside algorithm with GM-LVeGs is polynomial in the sentence length and the nonterminal number, in practice the algorithm is still slow because the number of Gaussian components in the inside and outside scores increases dramatically with the recursion depth. To speed up the computation, we prune Gaussian components in the inside and outside scores using the following technique. Suppose we have a minimum pruning threshold k_{min}

$$s_{\mathbf{I}}^{A}(\mathbf{a},i,j) = \sum_{A\to BC\in R}\sum_{k=i,\cdots,j-1}\iint W_{A\to BC}(\mathbf{a},\mathbf{b},\mathbf{c})\times s_{\mathbf{I}}^{B}(\mathbf{b},i,k)\times s_{\mathbf{I}}^{C}(\mathbf{c},k+1,j)\,d\mathbf{b}d\mathbf{c}\,. \tag{6}$$

$$s_{\mathbf{O}}^{A}(\mathbf{a},i,j) = \sum_{B\to CA\in R}\sum_{k=1,\cdots,i-1}\iint W_{B\to CA}(\mathbf{b},\mathbf{c},\mathbf{a})\times s_{\mathbf{O}}^{B}(\mathbf{b},k,j)\times s_{\mathbf{I}}^{C}(\mathbf{c},k,i-1)\,d\mathbf{b}d\mathbf{c} + \sum_{B\to AC\in R}\sum_{k=j+1,\cdots,n}\iint W_{B\to AC}(\mathbf{b},\mathbf{a},\mathbf{c})\times s_{\mathbf{O}}^{B}(\mathbf{b},i,k)\times s_{\mathbf{I}}^{C}(\mathbf{c},j+1,k)\,d\mathbf{b}d\mathbf{c}\,. \tag{7}$$

$$s(A\to BC,i,k,j) = \iiint W_{A\to BC}(\mathbf{a},\mathbf{b},\mathbf{c})\times s_{\mathbf{O}}^{A}(\mathbf{a},i,j)\times s_{\mathbf{I}}^{B}(\mathbf{b},i,k)\times s_{\mathbf{I}}^{C}(\mathbf{c},k+1,j)\,d\mathbf{a}d\mathbf{b}d\mathbf{c}\,. \tag{8}$$

Table 1: Equation 6: $s_{\mathbf{I}}^{A}(\mathbf{a},i,j)$ is the inside score of a nonterminal A over a span $w_{i:j}$ in the sentence $w_{1:n}$, where $1 \leq i < j \leq n$. Equation 7: $s_{\mathbf{O}}^{A}(\mathbf{a},i,j)$ is the outside score of a nonterminal A over a span $w_{i:j}$ in the sentence $w_{1:n}$, where $1 \leq i \leq j \leq n$. Equation 8: $s(A \to BC, i, k, j)$ is the score of a production rule $A \to BC$ with nonterminals A, B, and C spanning words $w_{i:j}$, $w_{i,k}$, and $w_{k+1:j}$ respectively in the sentence $w_{1:n}$, where $1 \leq i \leq k < j \leq n$.

and a maximum pruning threshold k_{max}. Given an inside or outside score with k_c Gaussian components, if $k_c \leq k_{min}$, then we do not prune any Gaussian component; otherwise, we compute $k_{allow} = \min\{k_{min} + \text{floor}(k_c^{\vartheta}), k_{max}\}$ ($0 \leq \vartheta \leq 1$ is a constant) and keep only k_{allow} components with the largest mixture weights.

In addition to component pruning, we also employ two constituent pruning techniques to reduce the search space during parsing. The first technique is used by Petrov et al. (2006). Before parsing a sentence with a GM-LVeG, we run the inside-outside algorithm with the treebank grammar and calculate the posterior probability of every nonterminal spanning every substring. Then a nonterminal would be pruned from a span if its posterior probability is below a pre-specified threshold p_{min}. When parsing with GM-LVeGs, we only consider the unpruned nonterminals for each span.

The second constituent pruning technique is similar to the one used by Socher et al. (2013). Note that for a strong constituency parser such as the Berkeley parser (Petrov and Klein, 2007), the constituents in the top 200 best parses of a sentence can cover almost all the constituents in the gold parse tree. So we first use an existing constituency parser to run k-best parsing with $k = 200$ on the input sentence. Then we parse with a GM-LVeG and only consider the constituents that appear in the top 200 parses. Note that this method is different from the re-ranking technique because it may produce a parse different from the top 200 parses.

3.3 Learning

Given a training dataset $D = \{(T_i, \mathbf{w}_i) \,|\, i = 1, \ldots, m\}$ containing m samples, where T_i is the gold parse tree with unrefined nonterminals for the sentence $\mathbf{w}_i$, the objective of discriminative learning is to minimize the negative log conditional likelihood:

$$\mathcal{L}(\Theta) = -\log \prod_{i=1}^{m} P(T_i | \mathbf{w}_i; \Theta)\,, \tag{11}$$

where Θ represents the set of parameters of the GM-LVeG.

We optimize the objective function using the Adam (Kingma and Ba, 2014) optimization algorithm. The derivative with respect to Θ_r, the parameters of the weight function $W_r(\mathbf{r})$ of an unrefined production rule r, is calculated as follows (the derivation is in the supplementary material):

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta_r} = \sum_{i=1}^{m} \int \left(\frac{\partial W_r(\mathbf{r})}{\partial \Theta_r} \times \frac{\mathbb{E}_{P(t|\mathbf{w}_i)}[f_r(t)] - \mathbb{E}_{P(t|T_i)}[f_r(t)]}{W_r(\mathbf{r})} \right) d\mathbf{r}\,, \tag{12}$$

where t indicates a parse tree with nonterminal subtypes, and $f_r(t)$ is the number of occurrences of the unrefined rule r in the unrefined parse tree that is obtained by replacing all the subtypes in t with the corresponding nonterminals. The two expectations in Equation 12 can be efficiently computed using the inside-outside algorithm. Because the second expectation is conditioned on the parse tree T_i, in Equation 6 and Equation 7 we can skip all the summations and assign the values of B, C, and k according to T_i.

In GM-LVeGs, Θ_r is the set of parameters in a Gaussian mixture:

$$\Theta_r = \{(\rho_{r,k}, \boldsymbol{\mu}_{r,k}, \boldsymbol{\Sigma}_{r,k}) | k = 1, \dots, K_r\} . \quad (13)$$

According to Equation 12, we need to take the derivatives of $W_r(\mathbf{r})$ respect to $\rho_{r,k}$, $\boldsymbol{\mu}_{r,k}$, and $\boldsymbol{\Sigma}_{r,k}$ respectively:

$$\partial W_r(\mathbf{r}) / \partial \rho_{r,k} = \mathcal{N}_{r,k}(\mathbf{r}) , \quad (14)$$

$$\partial W_r(\mathbf{r}) / \partial \boldsymbol{\mu}_{r,k} = \rho_{r,k} \mathcal{N}_{r,k}(\mathbf{r}) \boldsymbol{\Sigma}_{r,k}^{-1} (\mathbf{r} - \boldsymbol{\mu}_{r,k}) , \quad (15)$$

$$\partial W_r(\mathbf{r}) / \partial \boldsymbol{\Sigma}_{r,k} = \rho_{r,k} \mathcal{N}_{r,k}(\mathbf{r}) \boldsymbol{\Sigma}_{r,k}^{-1} \frac{1}{2} \Big(- I + (\mathbf{r} - \boldsymbol{\mu}_{r,k})(\mathbf{r} - \boldsymbol{\mu}_{r,k})^T \boldsymbol{\Sigma}_{r,k}^{-1} \Big) . \quad (16)$$

Substituting Equation 14–16 into Equation 12, we have the full gradient formulations of all the parameters. In spite of the integral in Equation 12, we can derive a closed-form solution for the gradient of each parameter, which is shown in the supplementary material.

In order to keep each mixture weight $\rho_{r,k}$ positive, we do not directly optimize $\rho_{r,k}$; instead, we set $\rho_{r,k} = \exp(\theta_{\rho_{r,k}})$ and optimize $\theta_{\rho_{r,k}}$ by gradient descent. We use a similar trick to keep each covariance matrix $\boldsymbol{\Sigma}_{r,k}$ positive definite.

Since we use the inside-outside algorithm described in Section 3.2 to calculate the two expectations in Equation 12, we face the same efficiency problem that we encounter in parsing. To speed up the computation,we again use both component pruning and constituent pruning introduced in Section 3.2.

Because gradient descent is often sensitive to the initial values of the parameters, we employ the following informed initialization method. Mixture weights are initialized using the treebank grammar. Suppose in the treebank grammar $P(r)$ is the probability of a production rule r. We initialize the mixture weights in the weight function W_r by $\rho_{r,k} = \alpha \cdot P(r)$ where $\alpha > 1$ is a constant. We initialize all the covariance matrices to identity matrices and initialize each mean with a value uniformly sampled from $[-0.05, 0.05]$.

4 Experiment

We evaluate the GM-LVeG on part-of-speech (POS) tagging and constituency parsing and compare it against its special cases such as LVGs and CVGs. It shall be noted that in this paper we focus on the basic framework of LVeGs and aim to show its potential advantage over previous special cases. It is therefore not our goal to compete with the latest state-of-the-art approaches to tagging and parsing. In particular, we currently do not incorporate contextual information of words and constituents during tagging and parsing, while such information is critical in achieving state-of-the-art accuracy. We will discuss future improvements of LVeGs in Section 5.

4.1 Datasets

Parsing. We use the Wall Street Journal corpus from the Penn English Treebank (WSJ) (Marcus et al., 1994). Following the standard data splitting, we use sections 2 to 21 for training, section 23 for testing, and section 22 for development. We preprocess the treebank using a right-branching binarization procedure to obtain an unannotated X-bar grammar, so that there are only binary and unary production rules. To deal with the problem of unknown words in testing, we adopt the unknown word features used in the Berkeley parser and set the unknown word threshold to 1. Specifically, any word occurring less than two times is replaced by one of the 60 unknown word categories.

Tagging. (1) We use Wall Street Journal corpus from the Penn English Treebank (WSJ) (Marcus et al., 1994). Following the standard data splitting, we use sections 0 to 18 for training, sections 22 to 24 for testing, and sections 19 to 21 for development. (2) The Universal Dependencies treebank 1.4 (UD) (Nivre et al., 2016), in which English, French, German, Russian, Spanish, Indonesian, Finnish, and Italian treebanks are used. We use the original data splitting of these corpora for training and testing. For both WSJ and UD English treebanks, we deal with unknown words in the same way as we do in parsing. For the rest of the data, we use only one unknown word category and the unknown word threshold is also set to 1.

4.2 POS Tagging

POS tagging is the task of labeling each word in a sentence with the most probable part-of-speech tag. Here we focus on POS tagging with Hidden Markov Models (HMMs). Because HMMs are equivalent to probabilistic regular grammars, we can extend HMMs with both LVGs and LVeGs. Specifically, the hidden states in HMMs can be seen as nonterminals in regular grammars and therefore can be associated with latent variables or latent vectors.

We implement two training methods for LVGs. The first (LVG-G) is generative training using expectation-maximization that maximizes the joint probability of the sentence and the tags. The second (LVG-D) is discriminative training using gradient descent that maximizes the conditional probability of the tags given the sentence. In both cases, each nonterminal is split into a fixed number of subtypes. In our experiments we test 1, 2, 4, 8, and 16 subtypes of each nonterminal. Due to the limited space, we only report experimental results of LVG with 16 subtypes for each nonterminal. Full experimental results can be found in the supplementary material.

We experiment with two different GM-LVeGs: GM-LVeG-D with diagonal Gaussians and GM-LVeG-S with spherical Gaussians. In both cases, we fix the number of Gaussian components K_r to 4 and the dimension of the latent vectors d to 3. We do not use any pruning techniques in learning and inference because we find that our algorithm is fast enough with the current setting of K_r and d. We train the GM-LVeGs for 20 epoches and select the models with the best token accuracy on the development data for the final testing.

We report both token accuracy and sentence accuracy of POS tagging in Table 2. It can be seen that, on all the testing data, GM-LVeGs consistently surpass LVGs in terms of both token accuracy and sentence accuracy. GM-LVeG-D is slightly better than GM-LVeG-S in sentence accuracy, producing the best sentence accuracy on 5 of the 9 testing datasets. GM-LVeG-S performs slightly better than GM-LVeG-D in token accuracy on 5 of the 9 datasets. Overall, there is not significant difference between GM-LVeG-D and GM-LVeG-S. However, GM-LVeG-S admits more efficient learning than GM-LVeG-D in practice since it has fewer parameters.

4.3 Parsing

For efficiency, we train GM-LVeGs only on sentences with no more than 50 words (totally 39115 sentences). Since we have found that spherical Gaussians are better than diagonal Gaussians considering both model performance and learning efficiency, here we use spherical Gaussians in the weight functions. The dimension of latent vectors d is set to 3, and all the Gaussian mixtures have $K_r = 4$ components. We use $\alpha = 8$ in initializing mixture weights. We train the GM-LVeG for 15 epoches and select the model with the highest F1 score on the development data for the final testing. We use component pruning in both learning and parsing, with $k_{max} = 50$ and $\vartheta = 0.35$ in both learning and parsing, $k_{min} = 40$ in learning and $k_{min} = 20$ in parsing. During learning we use the first constituent pruning technique with the pruning threshold $p_{min} = 1e - 5$, and during parsing we use the second constituent pruning technique based on the Berkeley parser which produced 133 parses on average for each testing sentence. As can be seen, we use weaker pruning during training than during testing. This is because in training stronger pruning (even if accurate) results in worse estimation of the first expectation in Equation 12, which makes gradient computation less accurate.

We compare LVeGs with CVGs and several variants of LVGs: (1) LVG-G-16 and LVG-D-16, which are LVGs with 16 subtypes for each nonterminal with discriminative and generative training respectively (accuracies obtained from Petrov and Klein (2008a)); (2) Multi-scale grammars (Petrov and Klein, 2008b), trained without using the span features in order for a fair comparison; (3) Berkeley parser (Petrov and Klein, 2007) (accuracies obtained from Petrov and Klein (2008b) because Petrov and Klein (2007) do not report exact match scores). The experimental results are shown in Table 3. It can be seen that GM-LVeG-S produces the best F1 scores on both the development data and the testing data. It surpasses the Berkeley parser by 0.92% in F1 score on the testing data. Its exact match score on the testing data is only slightly lower than that of LVG-D-16.

We further investigate the influence of the latent vector dimension and the Gaussian component number on the efficiency and the parsing accuracy . We experiment on a small dataset (statistics of this dataset are in the supplemental material). We first fix the component number to 4 and experiment with the dimension 2, 3, 4, 5, 6, 7, 8, 9. Then we fix the dimension to 3 and experiment with the component number 2, 3, 4, 5, 6, 7, 8, 9. F1 scores on the development data are shown in the first row in Figure 1. Average time consumed per epoch in learning is shown in the second row in Figure 1. When $K_r = 4$, the best dimension is 5; when $d = 3$, the best Gaussian component number is 3. A higher dimension or a larger Gaussian component number hurts the model performance and requires much more time for learning. Thus

Model	WSJ		English		French		German		Russian		Spanish		Indonesian		Finnish		Italian	
	T	S	T	S	T	S	T	S	T	S	T	S	T	S	T	S	T	S
LVG-D-16	96.62	48.74	92.31	52.67	93.75	34.90	87.38	20.98	81.91	12.25	92.47	24.82	89.27	20.29	83.81	19.29	94.81	45.19
LVG-G-16	96.78	50.88	93.30	57.54	94.52	34.90	88.92	24.05	84.03	16.63	93.21	27.37	90.09	21.19	85.01	20.53	95.46	48.26
GM-LVeG-D	96.99	53.10	**93.66**	**59.46**	94.73	**39.60**	89.11	24.77	**84.21**	17.84	**93.76**	**32.48**	**90.24**	**21.72**	85.27	**23.30**	95.61	**50.72**
GM-LVeG-S	**97.00**	**53.11**	93.55	58.11	**94.74**	39.26	**89.14**	**25.58**	84.06	**18.44**	93.52	30.66	90.12	**21.72**	**85.35**	22.07	**95.62**	49.69

Table 2: Token accuracy (T) and sentence accuracy (S) for POS tagging on the testing data.

Model	dev (all)	test $\leq$ 40		test (all)	
	F1	F1	EX	F1	EX
LVG-G-16				88.70	35.80
LVG-D-16				89.30	**39.40**
Multi-Scale		89.70	39.60	89.20	37.20
Berkeley Parser		90.60	39.10	90.10	37.10
CVG (SU-RNN)	91.20	91.10		90.40	
GM-LVeG-S	**91.24**	**91.38**	**41.51**	**91.02**	39.24

Table 3: Parsing accuracy on the testing data of WSJ. EX indicates the exact match score.

our choice of $K_r = 4$ and $d = 3$ in GM-LVeGs for parsing is a good balance between the efficiency and the parsing accuracy.

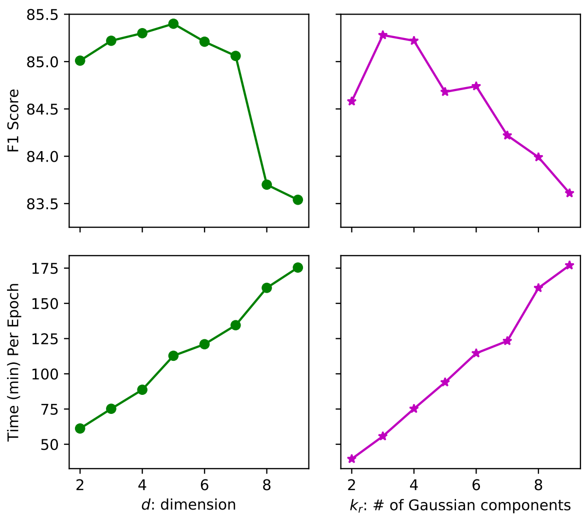


Figure 1: F1 score and average time (min) consumed per epoch in learning. **Left**: # of Gaussian components fixed to 4 with different dimensions; **Right**: dimension of Gaussians fixed to 3 with different # of Gaussian components.

5 Discussion

It shall be noted that in this paper we choose to focus on the basic framework and algorithms of LVeGs, and therefore we leave a few important extensions for future work. One extension is to incorporate contextual information of words and constituents. which is a crucial technique that can be found in most state-of-the-art approaches to parsing or POS tagging. One possible way to utilize contextual information in LVeGs is to allow the words in the context of an anchored production rule to influence the rule's weight function. For example, we may learn neural networks to predict the parameters of the Gaussian mixture weight functions in a GM-LVeG from the pre-trained embeddings of the words in the context.

In GM-LVeGs, we currently use the same number of Gaussian components for all the weight functions. A more desirable way would be automatically determining the number of Gaussian components for each production rule based on the ideal refinement granularity of the rule, e.g., we may need more Gaussian components for NP $\rightarrow$ DT NN than for NP $\rightarrow$ DT JJ, since the latter is rarely used. There are a few possible ways to learn the component numbers such as greedy addition and removal, the split-merge method, and sparsity priors over mixture weights.

An interesting extension beyond LVeGs is to have a single continuous space for subtypes of all the nonterminals. Ideally, subtypes of the same nonterminal or similar nonterminals are close to each other. The benefit is that similarity between nonterminals can now be modeled.

6 Conclusion

We present Latent Vector Grammars (LVeGs) that associate each nonterminal with a latent continuous vector space representing the set of subtypes of the nonterminal. For each production rule, a LVeG defines a continuous weight function over the subtypes of the nonterminals involved in the rule. We show that LVeGs can subsume latent variable grammars and compositional vector grammars as special cases. We then propose Gaussian mixture LVeGs (GM-LVeGs). which formulate weight functions of production rules by mixtures of Gaussian distributions. The partition function and the expectations of fine-grained production rules in GM-LVeGs can be efficiently computed using dynamic programming, which makes learning and inference with GM-LVeGs feasible.

We empirically show that GM-LVeGs can achieve competitive accuracies on POS tagging and constituency parsing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61503248), Major Program of Science and Technology Commission Shanghai Municipal (17JC1404102), and Program of Shanghai Subject Chief Scientist (A type) (No.15XD1502900). We would like to thank the anonymous reviewers for their careful reading and useful comments.

References

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the 30th National Conference on Artificial Intelligence*, volume 2, pages 1031–1036.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 302–312. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting on Association for Computational Linguistics*, pages 423–430. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 1659–1666.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 404–411. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2008a. Discriminative log-linear grammars with latent variables. In *Advances in Neural Information Processing Systems 20*, pages 1153–1160.

Slav Petrov and Dan Klein. 2008b. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 867–876. Association for Computational Linguistics.

Noah A Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

Richard Socher, John Bauer, Christopher D Manning, et al. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 455–465. Association for Computational Linguistics.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 129–136.

Extending a Parser to Distant Domains Using a Few Dozen Partially Annotated Examples

Vidur Joshi, Matthew Peters, Mark Hopkins
Allen Institute for AI, Seattle, WA
{vidurj, matthewp, markh}@allenai.org

Abstract

We revisit domain adaptation for parsers in the neural era. First we show that recent advances in word representations greatly diminish the need for domain adaptation when the target domain is syntactically similar to the source domain. As evidence, we train a parser on the Wall Street Journal alone that achieves over 90% F_1 on the Brown corpus. For more syntactically distant domains, we provide a simple way to adapt a parser using only dozens of partial annotations. For instance, we increase the percentage of error-free geometry-domain parses in a held-out set from 45% to 73% using approximately five dozen training examples. In the process, we demonstrate a new state-of-the-art single model result on the Wall Street Journal test set of 94.3%. This is an absolute increase of 1.7% over the previous state-of-the-art of 92.6%.

1 Introduction

Statistical parsers are often criticized for their performance outside of the domain they were trained on. The most straightforward remedy would be more training data in the target domain, but building treebanks (Marcus et al., 1993) is expensive.

In this paper, we revisit this issue in light of recent developments in neural natural language processing. Our paper rests on two observations:

Given [the circle [at the right] with [designated center, designated perpendicular, and radius 5]] .

In [the figure above] , [[AD = 4] , [AB = 3] and [CD = 9]] .

[Diameter AC] is perpendicular [to chord BD] [at E] .

Figure 1: An example of partial annotations. Annotators indicate that a span is a constituent by enclosing it in square brackets.

1. **It is trivial to train on partial annotations using a span-focused model.** Stern et al. (2017a) demonstrated that a parser with minimal dependence between the decisions that produce a parse can achieve state-of-the-art performance. We modify their parser, henceforth MSP, so that it trains directly on individual labeled spans instead of parse trees. This results in a parser that can be trained, with no adjustments to the training regime, from partial sentence bracketings.

2. **The use of contextualized word representations (Peters et al., 2017; McCann et al., 2017) greatly reduces the amount of data needed to train linguistic models.** Contextualized word representations, which encode tokens conditioned on their context in a sentence, have been shown to give significant boosts across a variety of NLP tasks, and also to reduce the amount of data needed by an order of magnitude in some tasks.

Taken together, this suggests a way to rapidly extend a newswire-trained parser to new domains. Specifically, we will show it is possible to achieve large out-of-domain performance improvements using only dozens of partially annotated sentences, like those shown in Figure 1. The resulting parser also does not suffer any degradation on the newswire domain.

Along the way, we provide several other notable contributions:

- We raise the state-of-the-art single-model F_1-score for constituency parsing from 92.6% to 94.3% on the Wall Street Journal (WSJ) test set. A trained model is publicly available.[1]
- We show that, even without domain-specific training data, our parser has much less out-of-domain degradation than previous parsers on "newswire-adjacent" domains like the Brown corpus.
- We provide a version of MSP which predicts its own POS tags (rather than requiring a third-party tagger).

2 The Reconciled Span Parser (RSP)

When we allow annotators to selectively annotate important phenomena, we make the process faster and simpler (Mielens et al., 2015). Unfortunately, this produces a disconnect between the model (which typically asserts the probability of a full parse tree) and the annotation task (which asserts the correctness of some subcomponent, like a constituent span or a dependency arc). There is a body of research (Hwa, 1999; Li et al., 2016) that discusses how to bridge this gap by modifying the training data, training algorithm, or the training objective.

Alternatively, we could just better align the model with the annotation task. Specifically, we could train a parser whose base model predicts exactly what we ask the annotator to annotate, e.g. whether a particular span is a constituent. This makes it trivial to train with partial or full annotations, because the training data reduces to a collection of span labels in either case.

Luckily, recent state-of-the-art results that model NLP tasks as independently classified spans (Stern et al., 2017a) suggest this strategy is currently viable. In this section, we present the Reconciled Span Parser (RSP), a modified version of the Minimal Span Parser (MSP) of Stern et al. (2017a). RSP differs from MSP in the following ways:

- **It is trained on a span classification task.** MSP trains on a maximum margin objective; that is, the loss function penalizes the violation of a margin between the scores of the gold parse and the next highest scoring parse decoded. This couples its training procedure with its decoding procedure, resulting in two versions, a top-down parser and a chart parser. To allow our model to be trained on partial annotations, we change the training task to be the span classification task described below.
- **It uses contextualized word representations instead of predicted part-of-speech tags.** Our model uses contextualized word representations as described in Peters et al. (2018). It does not take part-of-speech-tags as input, eliminating the dependence of the parser on a newswire-trained POS-tagger.

[1]http://allennlp.org/models

2.1 Overview

We will view a parse tree as a labeling of all the spans of a sentence such that:

- Every constituent span is labeled with the sequence of non-terminals assigned to it in the parse tree. For instance, span $(2, 4)$ in Figure 2b is labeled with the sequence $\langle S, \text{VP}\rangle$, as shown in Figure 2a.
- Every non-constituent is labeled with the empty sequence.

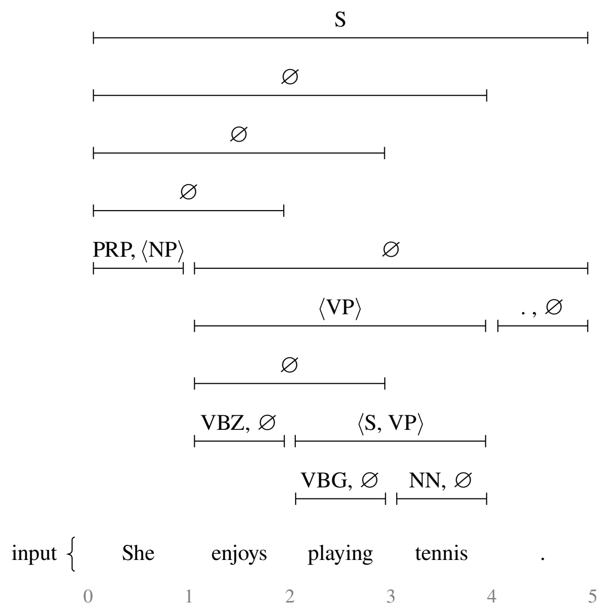
Given a sentence represented by a sequence of tokens x of length n, define $\mathsf{spans}(x) = \{(i, j) \mid 0 \le i < j \le n\}$. Define a *parse* for sentence x as a function $\pi : \mathsf{spans}(x) \mapsto \mathcal{L}$ where $\mathcal{L}$ is the set of all sequences of non-terminal tags, including the empty sequence.

We model the probability of a parse as the independent product of its span labels:

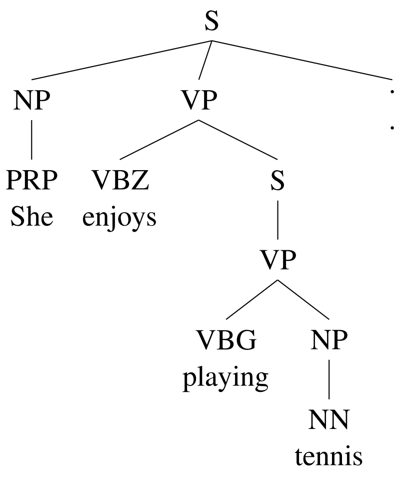
$$\begin{aligned} Pr(\pi|x) &= \prod_{s \in \mathsf{spans}(x)} Pr(\pi(s) \mid x, s) \\ \Rightarrow \log Pr(\pi|x) &= \sum_{s \in \mathsf{spans}(x)} \log Pr(\pi(s) \mid x, s) \end{aligned}$$

Hence, we will train a base model $\sigma(l \mid x, s)$ to estimate the log probability of label l for span s (given sentence x), and we will score the overall parse with:

$$\mathsf{score}(\pi|x) = \sum_{s \in \mathsf{spans}(x)} \sigma(\pi(s) \mid x, s)$$



(a) Spans classified by the parsing procedure. Note that leaves have their part-of-speech tags predicted in addition to their sequence of non-terminals.



(b) The resulting parse tree.

Figure 2: The correspondence between labeled spans and a parse tree. This diagram is adapted from figure 1 in (Stern et al., 2017a).

Note that this probability model accords mass to mis-structured trees (e.g. overlapping spans like (2, 5) and (3, 7) cannot both be constituents of a well-formed tree). We solve the following Integer Linear Program (ILP)[2] to find the highest scoring parse that admits a well-formed tree:

$$\max_{\delta} \sum_{(i,j)\in \mathsf{spans}(x)} v^{+}_{(i,j)}\delta_{(i,j)} + v^{-}_{(i,j)}(1-\delta_{(i,j)})$$

subject to:

$$\begin{aligned} i<k<j<m &\implies \delta_{(i,j)}+\delta_{(k,m)} \le 1 \\ (i,j)\in \mathsf{spans}(x) &\implies \delta_{(i,j)} \in \{0,1\} \end{aligned}$$

where:

$$\begin{aligned} v^{+}_{(i,j)} &= \max_{l \text{ s.t. } l\neq\varnothing} \sigma(l \mid x,(i,j)) \\ v^{-}_{(i,j)} &= \sigma(\varnothing \mid x,(i,j)) \end{aligned}$$

[2]There are a number of ways to reconcile the span conflicts, including an adaptation of the standard dynamic programming chart parsing algorithm to work with spans of an unbinarized tree. However it turns out that the classification model rarely produces span conflicts, so all methods we tried performed equivalently well.

2.2 Classification Model

For our span classification model $\sigma(l \mid x, s)$, we use the model from (Stern et al., 2017a), which leverages a method for encoding spans from (Wang and Chang, 2016; Cross and Huang, 2016). First, it creates a sentence encoding by running a two-layer bidirectional LSTM over the sentence to obtain forward and backward encodings for each position i, denoted by $\boldsymbol{f}_i$ and $\boldsymbol{b}_i$ respectively. Then, spans are encoded by the difference in LSTM states immediately before and after the span; that is, span (i, j) is encoded as the concatenation of the vector differences $\boldsymbol{f}_j - \boldsymbol{f}_{i-1}$ and $\boldsymbol{b}_i - \boldsymbol{b}_{j+1}$. A one-layer feedforward network maps each span representation to a distribution over labels.

Classification Model Parameters and Initializations

We preserve the settings used in Stern et al. (2017a) where possible. As a result, the size of the hidden dimensions of the LSTM and the feedforward network is 250. The dropout ratio for the LSTM is set to 0.4 . Unlike the model it is based on, our model uses word embeddings of length 1124. These result from concatenating a 100 dimension learned word embedding, with a 1024 di-

Parser	Rec	Prec	$\mathbf{F_1}$
RNNG (Dyer et al., 2016)	-	-	91.7
MSP (Stern et al., 2017a)	90.6	93.0	91.8
(Stern et al., 2017b)	92.6	92.6	92.6
RSP	93.8	94.8	94.3

Table 1: Parsing performance on WSJTEST, along with the results of other recent single-model parsers trained without external parse data.

	Recall	Precision	F1
all features	94.20	94.77	94.48
–ELMo	91.63	93.05	92.34

Table 2: Feature ablation on WSJDEV.

mension learned linear combination of the internal states of a bidirectional language model run on the input sentence as described in Peters et al. (2018). We refer to them below as ELMo (Embeddings from Language Models). For the learned embeddings, words with n occurrences in the training data are replaced by $\langle$UNK$\rangle$ with probability $\frac{1+\frac{n}{10}}{1+n}$. This does not affect the ELMo component of the word embeddings. As a result, even common words are replaced with probability at least $\frac{1}{10}$, making the model rely on the ELMo embeddings instead of the learned embeddings. To make the model self-contained, it does not take part-of-speech tags as input. Using a linear layer over the last hidden layer of the classification model, part-of-speech tags are predicted for spans containing single words.

3 Analysis of RSP

3.1 Performance on Newswire

On WSJTEST[3], RSP outperforms (see Table 1) all previous single models trained on WSJTRAIN by a significant margin, raising the state-of-the-art result from 92.6% to 94.3%. Additionally, our predicted part-of-speech tags achieve 97.72%[4] accuracy on WSJTEST.

[3]For all our experiments on the WSJ component of the Penn Treebank (Marcus et al., 1993), we use the standard split which is sections 2-21 for training, henceforth WSJTRAIN, section 22 for development, henceforth WSJDEV, and 23 for testing, henceforth WSJTEST.

[4]The split we used is not standard for part-of-speech tagging. As a result, we do not compare to part-of-speech taggers.

3.2 Beyond Newswire

The Brown Corpus

The Brown corpus (Marcus et al., 1993) is a standard benchmark used to assess WSJ-trained parsers outside of the newswire domain. When (Kummerfeld et al., 2012) parsed the various Brown verticals with the (then state-of-the-art) Charniak parser (Charniak, 2000; Charniak and Johnson, 2005; McClosky et al., 2006a), it achieved F_1 scores between 83% and 86%, even though its F_1 score on WSJTEST was 92.1%.

In Table 3, we discover that RSP does not suffer nearly as much degradation, with an average F_1-score of 90.3%. To determine whether this increased portability is because of the parser architecture or the use of ELMo vectors, we also run MSP on the Brown verticals. We used the Stanford tagger[5] (Toutanova et al., 2003) to tag WSJTRAIN and the Brown verticals so that MSP could be given these at train and test time. We learned that most of the improvement can be attributed to the ELMo word representations. In fact, even if we use MSP with gold POS tags, the average performance is 3.4% below RSP.

Question Bank and Genia

Despite being a standard benchmark for parsing domain adaptation, the Brown corpus has considerable commonality with newswire text. It is primarily composed of well-formed sentences with similar syntactic phenomena. Perhaps the main challenge with the Brown corpus is a difference in vocabulary, rather than a difference in syntax, which may explain the success of RSP, which leverages contextualized embeddings learned from a large corpus.

If we try to run RSP on a more syntactically divergent corpus like QuestionBank[6] (Judge et al., 2006), we find much more performance degradation. This is unsurprising, since WSJTRAIN does not contain many examples of question syntax. But how many examples do we need, to get good performance?

[5]We used the english-left3words-distsim.tagger model from the 2017-06-09 release of the Stanford POS tagger since it achieved the best accuracy on the Brown corpus.

[6]For all our experiments on QuestionBank, we use the following split: sentences 1-1000 and 2001-3000 for training, henceforth QBANKTRAIN, 1001-1500 and 3001-3500 for development, henceforth QBANKDEV, and 1501-2000 and 2501-4000 for testing, henceforth QBANKTEST. This split is described at https://nlp.stanford.edu/data/QuestionBank-Stanford.shtml.

Section	F$_1$			
	RSP	**MSP + Stanford POS tags**	**MSP + gold POS tags**	**Charniak**
F (popular)	91.42	87.01	87.84	85.91
G (biographies)	90.04	86.14	86.91	84.56
K (general)	90.08	85.53	86.46	84.09
L (mystery)	89.65	85.61	86.47	83.95
M (science)	90.52	86.91	87.52	84.65
N (adventure)	91.00	86.53	87.53	85.2
P (romance)	89.76	85.77	86.59	84.09
R (humor)	89.54	84.98	85.69	83.60
average	90.25	86.06	86.88	84.51

Table 3: Parsing performance on Brown verticals. MSP refers to the Minimal Span Parser (Stern et al., 2017a). Charniak refers to the Charniak parser with reranking and self-training (Charniak, 2000; Charniak and Johnson, 2005; McClosky et al., 2006a). MSP + Stanford POS tags refers to MSP trained and tested using part-of-speech tags predicted by the Stanford tagger (Toutanova et al., 2003).

Training Data		**Rec.**	**Prec.**	**F$_1$**
WSJ	QBANK			
40k	0	91.07	88.77	89.91
0	2k	94.44	96.23	95.32
40k	2k	95.84	97.02	96.43
40k	50	93.85	95.91	94.87
40k	100	95.08	96.06	95.57
40k	400	94.94	97.05	95.99

Table 4: Performance of RSP on QBANKDEV.

Training Data		**Rec**	**Prec**	**F$_1$**
WSJ	GENIA			
40k	0	72.51	88.84	79.85
0k	14k	88.04	92.30	90.12
40k	14k	88.24	92.33	90.24
40k	50	82.30	90.55	86.23
40k	100	83.94	89.97	86.85
40k	500	85.52	91.01	88.18

Table 5: Performance of RSP on GENIADEV.

For the experiments summarized in table 4 and table 5 involving 40k sentences from WSJTRAIN, we started with RSP trained on WSJTRAIN, and fine-tuned it on minibatches containing an equal number of target domain and WSJTRAIN sentences.

Surprisingly, with only 50 annotated questions (see Table 4), performance on QBANKDEV jumps 5 points, from 89.9% to 94.9%. This is only 1.5% below training with all of WSJTRAIN and QBANKTRAIN. The resulting system improves slightly on WSJTEST getting 94.38%.

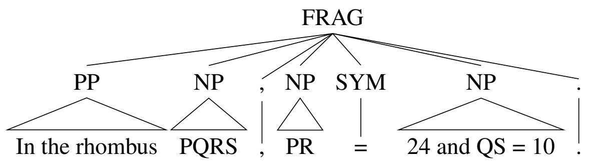
On the more difficult GENIA corpus of biomedical abstracts (Tateisi et al., 2005), we see a similar, if somewhat less dramatic, trend. See Table 5. With 50 annotated sentences, performance on GENIADEV jumps from 79.5% to 86.2%, outperforming all but one parser from David McClosky's thesis (McClosky, 2010) – the one that trains on all 14k sentences from GENIATRAIN and self-trains using 270k sentences from PubMed. That parser achieves 87.6%, which we outperform with just 500 sentences from GENIATRAIN.

These results suggest that it is currently feasible to extend a parser to a syntactically distant domain (for which no gold parses exist) with a couple hours of effort. We explore this possibility in the next section.

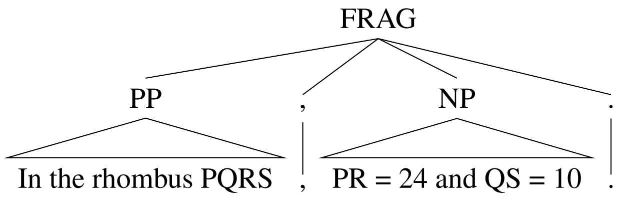
4 Rapid Parser Extension

To create a parser for their geometry question answering system, (Seo et al., 2015) did the following:

- Designed regular expressions to identify mathematical expressions.
- Replaced the identified expressions with dummy words.
- Parsed the resulting sentences.



(a) Training only on WSJTRAIN.



(b) Retraining on partial annotations.

Figure 3: The top-level split for the development sentence "In the rhombus PQRS, PR = 24 and QS = 10." before and after retraining RSP on 63 partially annotated geometry statements.

- Substituted the regex-analyzed expressions for the dummy words in the parses.

It is clear why this was necessary. Figure 3 (top) shows how RSP (trained only on WSJTRAIN) parses the sentence "In the rhombus PQRS, PR = 24 and QS = 10." The result is completely wrong, and useless to a downstream application.

Still, beyond just the inconvenience of building additional infrastructure, there are downsides to the "regex-and-replace" strategy:

1. **It assumes that each expression always maps to the same constituent label.** Consider "$2x = 3y$". This is a verb phrase in the sentence "In the above figure, x is prime and $2x = 3y$." However, it is a noun phrase in the sentence "The equation $2x = 3y$ has 2 solutions." If we replace both instances with the same dummy word, the parser will almost certainly become confused in one of the two instances.

2. **It assumes that each expression is always a constituent.** Suppose that we replace the expression "$AB < 30$" with a dummy word. This means we cannot properly parse a sentence like "When angle $AB < 30$, the lines are parallel," because the constituent "angle AB" no longer exists in the resulting sentence.

3. **It does not handle other syntactic variation.** As we will see in the next section, the geometry domain has a propensity for using right-attaching participial adjective phrases, like "labeled x" in the phrase "the segment labeled x." Encouraging a parser to recognize this syntactic construct is out-of-scope for the "regex-and-replace" strategy.

Instead, we propose directly extending the parser by providing a few domain-specific examples like those in Figure 1. Because RSP's model directly predicts span constituency, we can simply mark up a sentence with the "tricky" domain-specific constituents that the model will not already have learned from WSJTRAIN. For instance, we mark up NOUN-LABEL constructs like "chord BD", and equations like "AD = 4".

From these marked-up sentences, we can extract training instances declaring the constituency of certain spans (like "to chord BD" in the third example) and the implied non-constituency of certain spans (like "perpendicular to chord" in the third example). We also allow annotators to explicitly declare the non-constituency of a span via an alternative markup (not shown).

We do not require annotators to provide span labels (although they can if desired). If a training instance merely declares a span to be a constituent (but does not provide a particular label), then the loss function only records loss when that span is classified as a non-constituent (i.e. any label is ok).

5 Experiments

5.1 Geometry Questions

We took the publicly available training data from (Seo et al., 2015), split the data into sentences, and then annotated each sentence as in Figure 1. Next, we randomly split these sentences into GEOTRAIN and GEODEV[7]. After removing duplicate sentences spanning both sets, we ended up with 63 annotated sentences in GEOTRAIN and 62 in GEODEV. In GEOTRAIN, we made an average of 2.8 constituent declarations and 0.3 (explicit) non-constituent declarations per sentence.

After preparing the data, we started with RSP trained on WSJTRAIN, and fine-tuned it on minibatches containing 50 randomly selected WSJTRAIN sentences, plus all of GEOTRAIN. The results are in table 6. After fine-tuning, the model

[7]GEOTRAIN and GEODEV are available at https://github.com/vidurj/parser-adaptation/tree/master/data.

Training Data	GeoDev		WsjTest
	correct constituents %	error-free %	F_1
WsjTrain	71.9	45.2	94.28
WsjTrain + GeoTrain	87.0	72.6	94.30

Table 6: RSP performance on GeoDev.

Training Data	BioChemDev		WsjTest
	correct constituents %	error-free %	F_1
WsjTrain	70.1	27.0	94.28
WsjTrain + BioChemTrain	79.5	46.7	94.23

Table 7: RSP performance on BioChemDev.

- **Given [a circle with [the tangent shown]] .**
- **Find the hypotenuse of [the triangle labeled t] .**
- **Examine [the following diagram with [the square highlighted]] .**

Figure 4: Three partial annotations targeting right-attaching participial adjectives.

gets 87% of the 185 annotations on GeoDev correct, compared with 71.9% before fine-tuning[8]. Moreover, the fraction of sentences with no errors increases from 45.2% to 72.6%. With only a few dozen partially-annotated training examples, not only do we see a large increase in domain performance, but there is also no degradation in the parser's performance on newswire. Some GeoDev parses have enormous qualitative differences, like the example shown in Figure 3.

For the GeoDev sentences on which we get errors after retraining, the errors fall predominantly into three categories. First, approximately 44% have some mishandled math syntax, like failing to recognize "dimensions 16 by 8" as a constituent, or providing a flat structuring of the equation "BAC = 1/4 * ACB" (instead of recognizing "1/4 * ACB" as a subconstituent). Second, approximately 19% have PP-attachment errors. Third, another 19% fail to correctly analyze right-attaching participial adjectives like "labeled x" in the noun phrase "the segment labeled x" or "indicated" in the noun phrase "the center indicated." This phenomenon is unusually frequent in geometry but was insufficiently marked-up in our training examples. For instance, while we have a training instance "Find [the measure of [the angle designated by x]]," it does not explicitly highlight the constituency of "designated by x". This suggests that in practice, this domain adaptation method could benefit from an iterative cycle in which a user assesses the parser's errors on their target domain, creates some partial annotations that address these issues, retrains the parser, and then repeats the process until satisfied. As a proof-of-concept, we invented 3 additional sentences with right-attaching participial adjectives (shown in Figure 4), added them to GeoTrain, and then retrained. Indeed, the handling of participial adjectives in GeoDev improved, increasing the overall percentage of correctly identified constituents to 88.6% and the percentage of error-free sentences to 75.8%.

5.2 Biomedicine and Chemistry

We ran a similar experiment using biomedical and chemistry text, taken from the unannotated data provided by (Nivre et al., 2007). We partially annotated 134 sentences and randomly split them into BioChemTrain (72 sentences) and BioChemDev (62 sentences)[9]. In BioChemTrain, we made an average of 4.2 constituent declarations per sentence. We made no non-constituent declarations.

Again, we started with RSP trained on WsjTrain, and fine-tuned it on minibatches containing annotations from 50 randomly selected Wsj-

[8]This improvement has a p-value of 10^{-4} under the one-sided, two-sample difference between proportions test.

[9]BioChemTrain and BioChemDev are available at https://github.com/vidurj/parser-adaptation/tree/master/data.

TRAIN sentences, plus all of BIOCHEMTRAIN. Table 7 shows the improvement in the percentage of correctly-identified annotated constituents and the percentage of test sentences for which the parse agrees with every annotation. As with the geometry domain, we get significant improvements using only dozens of partially annotated training sentences.

6 Related Work

The two major themes of this paper, domain adaptation and learning from partial annotation, each have a long tradition in natural language processing.

6.1 Domain Adaptation

Domain adaptation has been recognized as a major NLP problem for over a decade (Ben-David et al., 2006; Blitzer et al., 2006; Daumé, 2007; Finkel and Manning, 2009). In particular, domain adaptation for parsers (Plank, 2011; Ma and Xia, 2013) has received considerable attention. Much of this work (McClosky et al., 2006b; Reichart and Rappoport, 2007; Sagae and Tsujii, 2007; Kawahara and Uchimoto, 2008; McClosky et al., 2010; Sagae, 2010; Baucom et al., 2013; Yu et al., 2015) has focused on how to best use co-training (Blum and Mitchell, 1998) or self-training to augment a small domain corpus, or how to best combine models to perform well on a particular domain.

In this work, we focus on the direct impact that just a few dozen partially annotated out-of-domain examples can have, when using a particular neural model with contextualized word representations. Co-training, self-training, and model combination are orthogonal to our approach. Our work is a spiritual successor to (Garrette and Baldridge, 2013), which shows how to train a part-of-speech tagger with a minimal amount of annotation effort.

6.2 Learning from Partial Annotation

Most literature on training parsers from partial annotations (Sassano and Kurohashi, 2010; Spreyer et al., 2010; Flannery et al., 2011; Flannery and Mori, 2015; Mielens et al., 2015) focuses on dependency parsing. (Li et al., 2016) provides a good overview. Here we highlight three important high-level strategies.

The first is "complete-then-train" (Mirroshandel and Nasr, 2011; Majidi and Crane, 2013), which "completes" every partially annotated dependency parse by finding the most likely parse (according to an already trained parser model) that respects the constraints of the partial annotations. These "completed" parses are then used to train a new parser.

The second strategy (Nivre et al., 2014; Li et al., 2016) is similar to "complete-then-train," but integrates parse completion into the training process. At each iteration, new "complete" parses are created using the parser model from the most recent training iteration.

The third strategy (Li et al., 2014, 2016) transforms each partial annotation into a forest of parses that encodes all fully-specified parses permitted by the partial annotation. Then, the training objective is modified to support optimization over these forests.

Our work differs from these in two respects. First, since we are training a constituency parser, our partial annotations are constituent bracketings rather than dependency arcs. Second, and more importantly, we can use the partial annotations for training without modifying either the training algorithm or the training data.

While the bulk of the literature on training from partial annotations focuses on dependency parsing, the earliest papers (Pereira and Schabes, 1992; Hwa, 1999) focus on constituency parsing. These leverage an adapted version of the inside-outside algorithm for estimating the parameters of a probabilistic context-free grammar (PCFG). Our work is not tied to PCFG parsing, nor does it require a specialized training algorithm when going from full annotations to partial annotations.

7 Conclusion

Recent developments in neural natural language processing have made it very easy to build custom parsers. Not only do contextualized word representations help parsers learn the syntax of new domains with very few examples, but they also work extremely well with parsing models that correspond directly with a granular and intuitive annotation task (like identifying whether a span is a constituent). This allows you to train with either full or partial annotations without any change to the training process.

This work provides a convenient path forward for the researcher who requires a parser for their domain, but laments that "parsers don't work outside of newswire." With a couple hours of effort (and a layman's understanding of syntactic building blocks), they can get significant performance improvements. We envision an iterative use case in which a user assesses a parser's errors on their target domain, creates some partial annotations to teach the parser how to fix these errors, then retrains the parser, repeating the process until they are satisfied.

References

Eric Baucom, Levi King, and Sandra Kübler. 2013. Domain adaptation for parsing. In *RANLP*.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *NIPS*.

John Blitzer, Ryan T. McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, pages 92–100.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *ANLP*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 173–180.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *EMNLP*.

Hal Daumé. 2007. Frustratingly easy domain adaptation. *CoRR* abs/0907.1815.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. *CoRR* abs/1602.07776. http://arxiv.org/abs/1602.07776.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *HLT-NAACL*.

Daniel Flannery, Yusuke Miyao, Graham Neubig, and Shinsuke Mori. 2011. Training dependency parsers from partially annotated corpora. In *IJCNLP*.

Daniel Flannery and Shinsuke Mori. 2015. Combining active learning and partial annotation for domain adaptation of a japanese dependency parser. In *IWPT*.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 138–147.

Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. *CoRR* cs.CL/9905001.

John Judge, Aoife Cahill, and Josef Van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 497–504.

Daisuke Kawahara and Kiyotaka Uchimoto. 2008. Learning reliability of parses for domain adaptation of dependency parsing. In *IJCNLP*.

Jonathan K. Kummerfeld, David Leo Wright Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *EMNLP-CoNLL*.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Soft cross-lingual syntax projection for dependency parsing. In *COLING*.

Zhenghua Li, Yue Zhang, Jiayuan Chao, and Min Zhang. 2016. Training dependency parsers with partial annotation. *CoRR* abs/1609.09247.

Xuezhe Ma and Fei Xia. 2013. Dependency parser adaptation with subtrees from auto-parsed target domain data. In *ACL*.

Saeed Majidi and Gregory R. Crane. 2013. Committee-based active learning for dependency parsing. In *TPDL*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

David McClosky. 2010. Any domain parsing: automatic domain adaptation for natural language parsing .

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *HLT-NAACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *ACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *HLT-NAACL*.

Jason Mielens, Liang Sun, and Jason Baldridge. 2015. Parse imputation for dependency annotations. In *ACL*.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2011. Active learning for dependency parsing using partially annotated sentences. In *IWPT*.

Joakim Nivre, Yoav Goldberg, and Ryan T. McDonald. 2014. Constrained arc-eager dependency parsing. *Computational Linguistics* 40:249–527.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. *ArXiv e-prints* .

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Barbara Plank. 2011. *Domain adaptation for parsing*. Citeseer.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL*.

Kenji Sagae. 2010. Self-training without reranking for parser domain adaptation and its impact on semantic role labeling.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *EMNLP-CoNLL*.

Manabu Sassano and Sadao Kurohashi. 2010. Using smaller constituents rather than sentences in active learning for japanese dependency parsing. In *ACL*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*.

Kathrin Spreyer, Lilja Ovrelid, and Jonas Kuhn. 2010. Training parsers on partial trees: A cross-language comparison. In *LREC*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017a. A minimal span-based neural constituency parser. *CoRR* abs/1705.03919. http://arxiv.org/abs/1705.03919.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017b. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 1695–1700. https://aclanthology.info/papers/D17-1178/d17-1178.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax annotation for the genia corpus. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. http://aclweb.org/anthology/N/N03/N03-1033.pdf.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 2306–2315.

Juntao Yu, Mohab Elkaref, and Bernd Bohnet. 2015. Domain adaptation for dependency parsing via self-training. In *IWPT*.

Paraphrase to Explicate: Revealing Implicit Noun-Compound Relations

Vered Shwartz **Ido Dagan**
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
vered1986@gmail.com dagan@cs.biu.ac.il

Abstract

Revealing the implicit semantic relation between the constituents of a noun-compound is important for many NLP applications. It has been addressed in the literature either as a classification task to a set of pre-defined relations or by producing free text paraphrases explicating the relations. Most existing paraphrasing methods lack the ability to generalize, and have a hard time interpreting infrequent or new noun-compounds. We propose a neural model that generalizes better by representing paraphrases in a continuous space, generalizing for both unseen noun-compounds and rare paraphrases. Our model helps improving performance on both the noun-compound paraphrasing and classification tasks.

1 Introduction

Noun-compounds hold an implicit semantic relation between their constituents. For example, a 'birthday cake' is a cake *eaten on* a birthday, while 'apple cake' is a cake *made of* apples. Interpreting noun-compounds by explicating the relationship is beneficial for many natural language understanding tasks, especially given the prevalence of noun-compounds in English (Nakov, 2013).

The interpretation of noun-compounds has been addressed in the literature either by classifying them to a fixed inventory of ontological relationships (e.g. Nastase and Szpakowicz, 2003) or by generating various free text paraphrases that describe the relation in a more expressive manner (e.g. Hendrickx et al., 2013).

Methods dedicated to paraphrasing noun-compounds usually rely on corpus co-occurrences of the compound's constituents as a source of explicit relation paraphrases (e.g. Wubben, 2010; Versley, 2013). Such methods are unable to generalize for unseen noun-compounds. Yet, most noun-compounds are very infrequent in text (Kim and Baldwin, 2007), and humans easily interpret the meaning of a new noun-compound by generalizing existing knowledge. For example, consider interpreting *parsley cake* as a cake made of parsley vs. *resignation cake* as a cake eaten to celebrate quitting an unpleasant job.

We follow the paraphrasing approach and propose a semi-supervised model for paraphrasing noun-compounds. Differently from previous methods, we train the model to predict either a paraphrase expressing the semantic relation of a noun-compound (predicting '[w_2] made of [w_1]' given 'apple cake'), or a missing constituent given a combination of paraphrase and noun-compound (predicting 'apple' given 'cake made of [w_1]'). Constituents and paraphrase templates are represented as continuous vectors, and semantically-similar paraphrase templates are embedded in proximity, enabling better generalization. Interpreting 'parsley cake' effectively reduces to identifying paraphrase templates whose "selectional preferences" (Pantel et al., 2007) on each constituent fit 'parsley' and 'cake'.

A qualitative analysis of the model shows that the top ranked paraphrases retrieved for each noun-compound are plausible even when the constituents never co-occur (Section 4). We evaluate our model on both the paraphrasing and the classification tasks (Section 5). On both tasks, the model's ability to generalize leads to improved performance in challenging evaluation settings.[1]

[1]The code is available at github.com/vered1986/panic

2 Background

2.1 Noun-compound Classification

Noun-compound classification is the task concerned with automatically determining the semantic relation that holds between the constituents of a noun-compound, taken from a set of pre-defined relations.

Early work on the task leveraged information derived from lexical resources and corpora (e.g. Girju, 2007; Ó Séaghdha and Copestake, 2009; Tratz and Hovy, 2010). More recent work broke the task into two steps: in the first step, a noun-compound representation is learned from the distributional representation of the constituent words (e.g. Mitchell and Lapata, 2010; Zanzotto et al., 2010; Socher et al., 2012). In the second step, the noun-compound representations are used as feature vectors for classification (e.g. Dima and Hinrichs, 2015; Dima, 2016).

The datasets for this task differ in size, number of relations and granularity level (e.g. Nastase and Szpakowicz, 2003; Kim and Baldwin, 2007; Tratz and Hovy, 2010). The decision on the relation inventory is somewhat arbitrary, and subsequently, the inter-annotator agreement is relatively low (Kim and Baldwin, 2007). Specifically, a noun-compound may fit into more than one relation: for instance, in Tratz (2011), *business zone* is labeled as CONTAINED (zone contains business), although it could also be labeled as PURPOSE (zone whose purpose is business).

2.2 Noun-compound Paraphrasing

As an alternative to the strict classification to pre-defined relation classes, Nakov and Hearst (2006) suggested that the semantics of a noun-compound could be expressed with multiple prepositional and verbal paraphrases. For example, *apple cake* is a cake *from*, *made of*, or which *contains* apples.

The suggestion was embraced and resulted in two SemEval tasks. SemEval 2010 task 9 (Butnariu et al., 2009) provided a list of plausible human-written paraphrases for each noun-compound, and systems had to rank them with the goal of high correlation with human judgments. In SemEval 2013 task 4 (Hendrickx et al., 2013), systems were expected to provide a ranked list of paraphrases extracted from free text.

Various approaches were proposed for this task. Most approaches start with a pre-processing step of extracting joint occurrences of the constituents from a corpus to generate a list of candidate paraphrases. Unsupervised methods apply information extraction techniques to find and rank the most meaningful paraphrases (Kim and Nakov, 2011; Xavier and Lima, 2014; Pasca, 2015; Pavlick and Pasca, 2017), while supervised approaches learn to rank paraphrases using various features such as co-occurrence counts (Wubben, 2010; Li et al., 2010; Surtani et al., 2013; Versley, 2013) or the distributional representations of the noun-compounds (Van de Cruys et al., 2013).

One of the challenges of this approach is the ability to generalize. If one assumes that sufficient paraphrases for all noun-compounds appear in the corpus, the problem reduces to ranking the existing paraphrases. It is more likely, however, that some noun-compounds do not have any paraphrases in the corpus or have just a few. The approach of Van de Cruys et al. (2013) somewhat generalizes for unseen noun-compounds. They represented each noun-compound using a compositional distributional vector (Mitchell and Lapata, 2010) and used it to predict paraphrases from the corpus. Similar noun-compounds are expected to have similar distributional representations and therefore yield the same paraphrases. For example, if the corpus does not contain paraphrases for *plastic spoon*, the model may predict the paraphrases of a similar compound such as *steel knife*.

In terms of sharing information between semantically-similar paraphrases, Nulty and Costello (2010) and Surtani et al. (2013) learned "is-a" relations between paraphrases from the co-occurrences of various paraphrases with each other. For example, the specific '[w_2] extracted from [w_1]' template (e.g. in the context of *olive oil*) generalizes to '[w_2] made from [w_1]'. One of the drawbacks of these systems is that they favor more frequent paraphrases, which may co-occur with a wide variety of more specific paraphrases.

2.3 Noun-compounds in other Tasks

Noun-compound paraphrasing may be considered as a subtask of the general paraphrasing task, whose goal is to generate, given a text fragment, additional texts with the same meaning. However, general paraphrasing methods do not guarantee to explicate implicit information conveyed in the original text. Moreover, the most notable source for extracting paraphrases is multiple translations of the same text (Barzilay and McKeown,

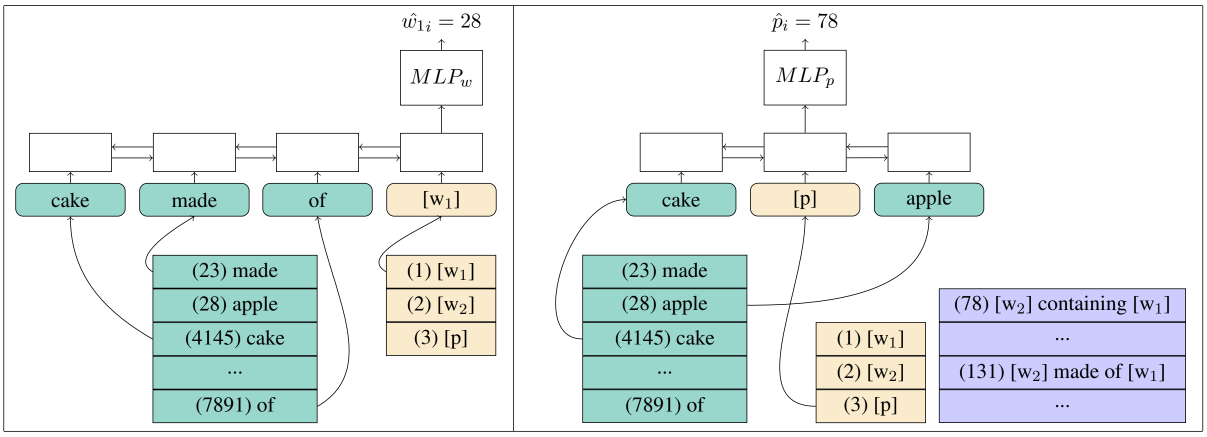


Figure 1: An illustration of the model predictions for w_1 and p given the triplet *(cake, made of, apple)*. The model predicts each component given the encoding of the other two components, successfully predicting '*apple*' given '*cake made of* [w$_1$]', while predicting '[w$_2$] *containing* [w$_1$]' for '*cake* [p] *apple*'.

2001; Ganitkevitch et al., 2013; Mallinson et al., 2017). If a certain concept can be described by an English noun-compound, it is unlikely that a translator chose to translate its foreign language equivalent to an explicit paraphrase instead.

Another related task is Open Information Extraction (Etzioni et al., 2008), whose goal is to extract relational tuples from text. Most system focus on extracting verb-mediated relations, and the few exceptions that addressed noun-compounds provided partial solutions. Pal and Mausam (2016) focused on segmenting multi-word noun-compounds and assumed an is-a relation between the parts, as extracting *(Francis Collins, is, NIH director)* from "NIH director Francis Collins". Xavier and Lima (2014) enriched the corpus with compound definitions from online dictionaries, for example, interpreting *oil industry* as *(industry, produces and delivers, oil)* based on the WordNet definition "industry that produces and delivers oil". This method is very limited as it can only interpret noun-compounds with dictionary entries, while the majority of English noun-compounds don't have them (Nakov, 2013).

3 Paraphrasing Model

As opposed to previous approaches, that focus on predicting a paraphrase template for a given noun-compound, we reformulate the task as a multi-task learning problem (Section 3.1), and train the model to also predict a missing constituent given the paraphrase template and the other constituent. Our model is semi-supervised, and it expects as input a set of noun-compounds and a set of constrained part-of-speech tag-based templates that make valid prepositional and verbal paraphrases. Section 3.2 details the creation of training data, and Section 3.3 describes the model.

3.1 Multi-task Reformulation

Each training example consists of two constituents and a paraphrase (w_2, p, w_1), and we train the model on 3 subtasks: (1) predict p given w_1 and w_2, (2) predict w_1 given p and w_2, and (3) predict w_2 given p and w_1. Figure 1 demonstrates the predictions for subtasks (1) (right) and (2) (left) for the training example (*cake, made of, apple*). Effectively, the model is trained to answer questions such as "what can cake be made of?", "what can be made of apple?", and "what are the possible relationships between cake and apple?".

The multi-task reformulation helps learning better representations for paraphrase templates, by embedding semantically-similar paraphrases in proximity. Similarity between paraphrases stems either from lexical similarity and overlap between the paraphrases (e.g. 'is made of' and 'made of'), or from shared constituents, e.g. '[w$_2$] involved in [w$_1$]' and '[w$_2$] in [w$_1$] industry' can share [w$_1$] = *insurance* and [w$_2$] = *company*. This allows the model to predict a correct paraphrase for a given noun-compound, even when the constituents *do not occur with that paraphrase in the corpus*.

3.2 Training Data

We collect a training set of (w_2, p, w_1, s) examples, where w_1 and w_2 are constituents of a noun-compound $w_1 w_2$, p is a templated paraphrase, and s is the score assigned to the training instance.[2]

[2]We refer to "paraphrases" and "paraphrase templates" interchangeably. In the extracted templates, [w$_2$] always precedes [w$_1$], probably because w$_2$ is normally the head noun.

We use the 19,491 noun-compounds found in the SemEval tasks datasets (Butnariu et al., 2009; Hendrickx et al., 2013) and in Tratz (2011). To extract patterns of part-of-speech tags that can form noun-compound paraphrases, such as '[w$_2$] VERB PREP [w$_1$]', we use the SemEval task training data, but we do not use the lexical information in the gold paraphrases.

Corpus. Similarly to previous noun-compound paraphrasing approaches, we use the Google N-gram corpus (Brants and Franz, 2006) as a source of paraphrases (Wubben, 2010; Li et al., 2010; Surtani et al., 2013; Versley, 2013). The corpus consists of sequences of n terms (for $n \in \{3, 4, 5\}$) that occur more than 40 times on the web. We search for n-grams following the extracted patterns and containing w_1 and w_2's lemmas for some noun-compound in the set. We remove punctuation, adjectives, adverbs and some determiners to unite similar paraphrases. For example, from the 5-gram '*cake made of sweet apples*' we extract the training example *(cake, made of, apple)*. We keep only paraphrases that occurred at least 5 times, resulting in 136,609 instances.

Weighting. Each n-gram in the corpus is accompanied with its frequency, which we use to assign scores to the different paraphrases. For instance, '*cake of apples*' may also appear in the corpus, although with lower frequency than '*cake from apples*'. As also noted by Surtani et al. (2013), the shortcoming of such a weighting mechanism is that it prefers shorter paraphrases, which are much more common in the corpus (e.g. count('*cake made of apples*') $\ll$ count('*cake of apples*')). We overcome this by normalizing the frequencies *for each paraphrase length*, creating a distribution of paraphrases in a given length.

Negative Samples. We add 1% of negative samples by selecting random corpus words w_1 and w_2 that do not co-occur, and adding an example (w_2, [w$_2$] is unrelated to [w$_1$], w_1, s_n), for some predefined negative samples score s_n. Similarly, for a word w_i that did not occur in a paraphrase p we add (w_i, p, UNK, s_n) or (UNK, p, w_i, s_n), where UNK is the unknown word. This may help the model deal with non-compositional noun-compounds, where w_1 and w_2 are unrelated, rather than forcibly predicting some relation between them.

3.3 Model

For a training instance (w_2, p, w_1, s), we predict each item given the encoding of the other two.

Encoding. We use the 100-dimensional pre-trained GloVe embeddings (Pennington et al., 2014), which are fixed during training. In addition, we learn embeddings for the special words [w$_1$], [w$_2$], and [p], which are used to represent a missing component, as in "cake made of [w$_1$]", "[w$_2$] made of apple", and "cake [p] apple".

For a missing component $x \in \{$[p], [w$_1$], [w$_2$]$\}$ surrounded by the sequences of words $v_{1:i-1}$ and $v_{i+1:n}$, we encode the sequence using a bidirectional long-short term memory (bi-LSTM) network (Graves and Schmidhuber, 2005), and take the ith output vector as representing the missing component: $bLS(v_{1:i}, x, v_{i+1:n})_i$.

In bi-LSTMs, each output vector is a concatenation of the outputs of the forward and backward LSTMs, so the output vector is expected to contain information on valid substitutions both with respect to the previous words $v_{1:i-1}$ and the subsequent words $v_{i+1:n}$.

Prediction. We predict a distribution of the vocabulary of the missing component, i.e. to predict w_1 correctly we need to predict its index in the word vocabulary V_w, while the prediction of p is from the vocabulary of paraphrases in the training set, V_p. We predict the following distributions:

$$\begin{aligned}
\hat{p} &= \text{softmax}(W_p \cdot bLS(\vec{w_2}, [p], \vec{w_1})_2) \\
\hat{w_1} &= \text{softmax}(W_w \cdot bLS(\vec{w_2}, \vec{p}_{1:n}, [w_1])_{n+1}) \\
\hat{w_2} &= \text{softmax}(W_w \cdot bLS([w_2], \vec{p}_{1:n}, \vec{w_1})_1)
\end{aligned} \quad (1)$$

where $W_w \in \mathcal{R}^{|V_w| \times 2d}$, $W_p \in \mathcal{R}^{|V_p| \times 2d}$, and d is the embeddings dimension.

During training, we compute cross-entropy loss for each subtask using the gold item and the prediction, sum up the losses, and weight them by the instance score. During inference, we predict the missing components by picking the best scoring index in each distribution:[3]

$$\begin{aligned}
\hat{p}_i &= \text{argmax}(\hat{p}) \\
\hat{w_1}_i &= \text{argmax}(\hat{w_1}) \\
\hat{w_2}_i &= \text{argmax}(\hat{w_2})
\end{aligned} \quad (2)$$

The subtasks share the pre-trained word embeddings, the special embeddings, and the biLSTM parameters. Subtasks (2) and (3) also share W_w, the MLP that predicts the index of a word.

[3]In practice, we pick the k best scoring indices in each distribution for some predefined k, as we discuss in Section 5.

$[w_1]$	$[w_2]$	**Predicted Paraphrases**	$[w_2]$	**Paraphrase**	**Predicted** $[w_1]$	**Paraphrase**	$[w_1]$	**Predicted** $[w_2]$
cataract	surgery	$[w_2]$ of $[w_1]$	surgery	$[w_2]$ to treat $[w_1]$	heart	$[w_2]$ to treat $[w_1]$	cataract	surgery
		$[w_2]$ on $[w_1]$			brain			drug
		$[w_2]$ to remove $[w_1]$			back			patient
		$[w_2]$ in patients with $[w_1]$			knee			transplant
software	company	$[w_2]$ of $[w_1]$	company	$[w_2]$ engaged in $[w_1]$	management	$[w_2]$ engaged in $[w_1]$	software	company
		$[w_2]$ to develop $[w_1]$			production			firm
		$[w_2]$ in $[w_1]$ industry			computer			engineer
		$[w_2]$ involved in $[w_1]$			business			industry
stone	wall	$[w_2]$ is of $[w_1]$	meeting	$[w_2]$ held in $[w_1]$	spring	$[w_2]$ held in $[w_1]$	morning	party
		$[w_2]$ of $[w_1]$			afternoon			meeting
		$[w_2]$ is made of $[w_1]$			hour			rally
		$[w_2]$ made of $[w_1]$			day			session

Table 1: Examples of top ranked predicted components using the model: predicting the paraphrase given w_1 and w_2 (left), w_1 given w_2 and the paraphrase (middle), and w_2 given w_1 and the paraphrase (right).

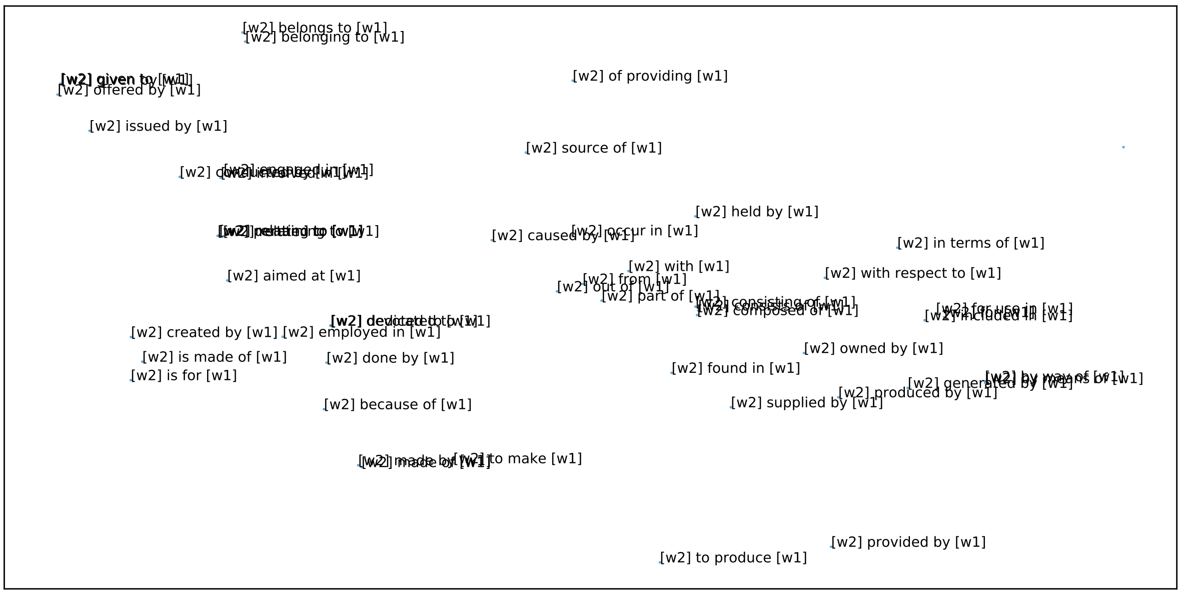


Figure 2: A t-SNE map of a sample of paraphrases, using the paraphrase vectors encoded by the biLSTM, for example $bLS([w_2]$ made of $[w_1])$.

Implementation Details. The model is implemented in DyNet (Neubig et al., 2017). We dedicate a small number of noun-compounds from the corpus for validation. We train for up to 10 epochs, stopping early if the validation loss has not improved in 3 epochs. We use Momentum SGD (Nesterov, 1983), and set the batch size to 10 and the other hyper-parameters to their default values.

4 Qualitative Analysis

To estimate the quality of the proposed model, we first provide a qualitative analysis of the model outputs. Table 1 displays examples of the model outputs for each possible usage: predicting the paraphrase given the constituent words, and predicting each constituent word given the paraphrase and the other word.

The examples in the table are from among the top 10 ranked predictions for each component-pair. We note that most of the (w_2, paraphrase, w_1) triplets in the table do not occur in the training data, but are rather generalized from similar examples. For example, there is no training instance for "company in the software industry" but there is a "firm in the software industry" and a company in many other industries.

While the frequent prepositional paraphrases are often ranked at the top of the list, the model also retrieves more specified verbal paraphrases. The list often contains multiple semantically-similar paraphrases, such as '$[w_2]$ involved in $[w_1]$' and '$[w_2]$ in $[w_1]$ industry'. This is a result of the model training objective (Section 3) which positions the vectors of semantically-similar paraphrases close to each other in the embedding space, based on similar constituents.

To illustrate paraphrase similarity we compute a t-SNE projection (Van Der Maaten, 2014) of the embeddings of all the paraphrases, and draw a sample of 50 paraphrases in Figure 2. The projection positions semantically-similar but lexically-divergent paraphrases in proximity, likely due to many shared constituents. For instance, 'with', 'from', and 'out of' can all describe the relation between food words and their ingredients.

5 Evaluation: Noun-Compound Interpretation Tasks

For quantitative evaluation we employ our model for two noun-compound interpretation tasks. The main evaluation is on retrieving and ranking paraphrases (§5.1). For the sake of completeness, we also evaluate the model on classification to a fixed inventory of relations (§5.2), although it wasn't designed for this task.

5.1 Paraphrasing

Task Definition. The general goal of this task is to interpret each noun-compound to multiple prepositional and verbal paraphrases. In SemEval 2013 Task 4,[4] the participating systems were asked to retrieve a ranked list of paraphrases for each noun-compound, which was automatically evaluated against a similarly ranked list of paraphrases proposed by human annotators.

Model. For a given noun-compound $w_1 w_2$, we first predict the $k = 250$ most likely paraphrases: $\hat{p}_1, ..., \hat{p}_k = \text{argmax}_k \, \hat{p}$, where $\hat{p}$ is the distribution of paraphrases defined in Equation 1.

While the model also provides a score for each paraphrase (Equation 1), the scores have not been optimized to correlate with human judgments. We therefore developed a re-ranking model that receives a list of paraphrases and re-ranks the list to better fit the human judgments.

We follow Herbrich (2000) and learn a pairwise ranking model. The model determines which of two paraphrases of the same noun-compound should be ranked higher, and it is implemented as an SVM classifier using scikit-learn (Pedregosa et al., 2011). For training, we use the available training data with gold paraphrases and ranks provided by the SemEval task organizers. We extract the following features for a paraphrase p:

1. The part-of-speech tags contained in p
2. The prepositions contained in p
3. The number of words in p
4. Whether p ends with the special [w$_1$] symbol
5. $cosine(bLS([w_2], p, [w_1])_2, \vec{V_p}^{\hat{p}_i}) \cdot \hat{p}^{\hat{p}_i}$

where $\vec{V_p}^{\hat{p}_i}$ is the biLSTM encoding of the predicted paraphrase computed in Equation 1 and $\hat{p}^{\hat{p}_i}$ is its confidence score. The last feature incorporates the original model score into the decision, as to not let other considerations such as preposition frequency in the training set take over.

During inference, the model sorts the list of paraphrases retrieved for each noun-compound according to the pairwise ranking. It then scores each paraphrase by multiplying its rank with its original model score, and prunes paraphrases with final score < 0.025. The values for k and the threshold were tuned on the training set.

Evaluation Settings. The SemEval 2013 task provided a scorer that compares words and n-grams from the gold paraphrases against those in the predicted paraphrases, where agreement on a prefix of a word (e.g. in derivations) yields a partial scoring. The overall score assigned to each system is calculated in two different ways. The 'isomorphic' setting rewards both precision and recall, and performing well on it requires accurately reproducing as many of the gold paraphrases as possible, and in much the same order. The 'non-isomorphic' setting rewards only precision, and performing well on it requires accurately reproducing the top-ranked gold paraphrases, with no importance to order.

Baselines. We compare our method with the published results from the SemEval task. The SemEval 2013 baseline generates for each noun-compound a list of prepositional paraphrases in an arbitrary fixed order. It achieves a moderately good score in the non-isomorphic setting by generating a fixed set of paraphrases which are both common and generic. The MELODI system performs similarly: it represents each noun-compound using a compositional distributional vector (Mitchell and Lapata, 2010) which is then used to predict paraphrases from the corpus. The performance of MELODI indicates that the system was rather conservative, yielding a few common paraphrases rather than many specific ones. SFS and IIITH, on the other hand, show a more balanced trade-off between recall and precision.

As a sanity check, we also report the results of a baseline that retrieves ranked paraphrases from the training data collected in Section 3.2. This baseline has no generalization abilities, therefore it is expected to score poorly on the recall-aware isomorphic setting.

[4] https://www.cs.york.ac.uk/semeval-2013/task4

	Method	isomorphic	non-isomorphic
Baselines	SFS (Versley, 2013)	23.1	17.9
	IIITH (Surtani et al., 2013)	23.1	25.8
	MELODI (Van de Cruys et al., 2013)	13.0	54.8
	SemEval 2013 Baseline (Hendrickx et al., 2013)	13.8	40.6
This paper	Baseline	3.8	16.1
	Our method	**28.2**	28.4

Table 2: Results of the proposed method and the baselines on the SemEval 2013 task.

Category	**%**
False Positive	
(1) Valid paraphrase missing from gold	44%
(2) Valid paraphrase, slightly too specific	15%
(3) Incorrect, common prepositional paraphrase	14%
(4) Incorrect, other errors	14%
(5) Syntactic error in paraphrase	8%
(6) Valid paraphrase, but borderline grammatical	5%
False Negative	
(1) Long paraphrase (more than 5 words)	30%
(2) Prepositional paraphrase with determiners	25%
(3) Inflected constituents in gold	10%
(4) Other errors	35%

Table 3: Categories of false positive and false negative predictions along with their percentage.

Results. Table 2 displays the performance of the proposed method and the baselines in the two evaluation settings. Our method outperforms all the methods in the isomorphic setting. In the non-isomorphic setting, it outperforms the other two systems that score reasonably on the isomorphic setting (SFS and IIITH) but cannot compete with the systems that focus on achieving high precision.

The main advantage of our proposed model is in its ability to generalize, and that is also demonstrated in comparison to our baseline performance. The baseline retrieved paraphrases only for a third of the noun-compounds (61/181), expectedly yielding poor performance on the isomorphic setting. Our model, which was trained on the very same data, retrieved paraphrases for all noun-compounds. For example, *welfare system* was not present in the training data, yet the model predicted the correct paraphrases "system of welfare benefits", "system to provide welfare" and others.

Error Analysis. We analyze the causes of the false positive and false negative errors made by the model. For each error type we sample 10 noun-compounds. For each noun-compound, false positive errors are the top 10 predicted paraphrases which are not included in the gold paraphrases, while false negative errors are the top 10 gold paraphrases not found in the top k predictions made by the model. Table 3 displays the manually annotated categories for each error type.

Many false positive errors are actually valid paraphrases that were not suggested by the human annotators (error 1, "discussion by group"). Some are borderline valid with minor grammatical changes (error 6, "force of coalition forces") or too specific (error 2, "life *of women in* community" instead of "life *in* community"). Common prepositional paraphrases were often retrieved although they are incorrect (error 3). We conjecture that this error often stem from an n-gram that does not respect the syntactic structure of the sentence, e.g. a sentence such as "rinse away the oil from baby 's head" produces the n-gram "oil from baby".

With respect to false negative examples, they consisted of many long paraphrases, while our model was restricted to 5 words due to the source of the training data (error 1, "holding done in the case of a share"). Many prepositional paraphrases consisted of determiners, which we conflated with the same paraphrases without determiners (error 2, "mutation of a gene"). Finally, in some paraphrases, the constituents in the gold paraphrase appear in inflectional forms (error 3, "holding of shares" instead of "holding of share").

5.2 Classification

Noun-compound classification is defined as a multiclass classification problem: given a pre-defined set of relations, classify w_1w_2 to the relation that holds between w_1 and w_2. Potentially, the corpus co-occurrences of w_1 and w_2 may contribute to the classification, e.g. '[w$_2$] held at [w$_1$]' indicates a TIME relation. Tratz and Hovy (2010) included such features in their classifier, but ablation tests showed that these features had a relatively small contribution, probably due to the sparseness of the paraphrases. Recently, Shwartz and Waterson (2018) showed that paraphrases may contribute to the classification when represented in a continuous space.

Model. We generate a paraphrase vector representation $\vec{par}(w_1 w_2)$ for a given noun-compound $w_1 w_2$ as follows. We predict the indices of the k most likely paraphrases: $\hat{p}_1, ..., \hat{p}_k = \text{argmax}_k \hat{p}$, where $\hat{p}$ is the distribution on the paraphrase vocabulary V_p, as defined in Equation 1. We then encode each paraphrase using the biLSTM, and average the paraphrase vectors, weighted by their confidence scores in $\hat{p}$:

$$\vec{par}(w_1 w_2) = \frac{\sum_{i=1}^{k} \hat{p}^{\hat{p}_i} \cdot \vec{V_p}^{\hat{p}_i}}{\sum_{i=1}^{k} \hat{p}^{\hat{p}_i}} \quad (3)$$

We train a linear classifier, and represent $w_1 w_2$ in a feature vector $f(w_1 w_2)$ in two variants: paraphrase: $f(w_1 w_2) = \vec{par}(w_1 w_2)$, or integrated: concatenated to the constituent word embeddings $f(w_1 w_2) = [\vec{par}(w_1 w_2), \vec{w_1}, \vec{w_2}]$. The classifier type (logistic regression/SVM), k, and the penalty are tuned on the validation set. We also provide a baseline in which we ablate the paraphrase component from our model, representing a noun-compound by the concatenation of its constituent embeddings $f(w_1 w_2) = [\vec{w_1}, \vec{w_2}]$ (distributional).

Datasets. We evaluate on the Tratz (2011) dataset, which consists of 19,158 instances, labeled in 37 fine-grained relations (`Tratz-fine`) or 12 coarse-grained relations (`Tratz-coarse`).

We report the performance on two different dataset splits to train, test, and validation: a random split in a 75:20:5 ratio, and, following concerns raised by Dima (2016) about lexical memorization (Levy et al., 2015), on a lexical split in which the sets consist of distinct vocabularies. The lexical split better demonstrates the scenario in which a noun-compound whose constituents have not been observed needs to be interpreted based on similar observed noun-compounds, e.g. inferring the relation in *pear tart* based on *apple cake* and other similar compounds. We follow the random and full-lexical splits from Shwartz and Waterson (2018).

Baselines. We report the results of 3 baselines representative of different approaches:
1) **Feature-based** (Tratz and Hovy, 2010): we re-implement a version of the classifier with features from WordNet and Roget's Thesaurus.
2) **Compositional** (Dima, 2016): a neural architecture that operates on the distributional representations of the noun-compound and its constituents. Noun-compound representations are learned with the Full-Additive (Zanzotto et al., 2010) and Matrix (Socher et al., 2012) models. We report the results from Shwartz and Waterson (2018).
3) **Paraphrase-based** (Shwartz and Waterson, 2018): a neural classification model that learns an LSTM-based representation of the joint occurrences of w_1 and w_2 in a corpus (i.e. observed paraphrases), and integrates distributional information using the constituent embeddings.

Dataset & Split	Method	F_1
`Tratz fine` Random	Tratz and Hovy (2010)	**0.739**
	Dima (2016)	0.725
	Shwartz and Waterson (2018)	0.714
	distributional	**0.677**
	paraphrase	0.505
	integrated	0.673
`Tratz fine` Lexical	Tratz and Hovy (2010)	0.340
	Dima (2016)	0.334
	Shwartz and Waterson (2018)	**0.429**
	distributional	0.356
	paraphrase	0.333
	integrated	**0.370**
`Tratz coarse` Random	Tratz and Hovy (2010)	0.760
	Dima (2016)	**0.775**
	Shwartz and Waterson (2018)	0.736
	distributional	0.689
	paraphrase	0.557
	integrated	**0.700**
`Tratz coarse` Lexical	Tratz and Hovy (2010)	0.391
	Dima (2016)	0.372
	Shwartz and Waterson (2018)	**0.478**
	distributional	0.370
	paraphrase	0.345
	integrated	**0.393**

Table 4: Classification results. For each dataset split, the top part consists of baseline methods and the bottom part of methods from this paper. The best performance in each part appears in bold.

Results. Table 4 displays the methods' performance on the two versions of the Tratz (2011) dataset and the two dataset splits. The paraphrase model on its own is inferior to the distributional model, however, the integrated version improves upon the distributional model in 3 out of 4 settings, demonstrating the complementary nature of the distributional and paraphrase-based methods. The contribution of the paraphrase component is especially noticeable in the lexical splits.

As expected, the integrated method in Shwartz and Waterson (2018), in which the paraphrase representation was trained with the objective of classification, performs better than our integrated model. The superiority of both integrated models in the lexical splits confirms that paraphrases are beneficial for classification.

Example Noun-compounds	Gold	Distributional	Example Paraphrases
printing plant	PURPOSE	OBJECTIVE	[w$_2$] engaged in [w$_1$]
marketing expert *development expert*	TOPICAL	OBJECTIVE	[w$_2$] in [w$_1$] [w$_2$] knowledge of [w$_1$]
weight/job loss	OBJECTIVE	CAUSAL	[w$_2$] of [w$_1$]
rubber band *rice cake*	CONTAINMENT	PURPOSE	[w$_2$] made of [w$_1$] [w$_2$] is made of [w$_1$]
laboratory animal	LOCATION/PART-WHOLE	ATTRIBUTE	[w$_2$] in [w$_1$], [w$_2$] used in [w$_1$]

Table 5: Examples of noun-compounds that were correctly classified by the integrated model while being incorrectly classified by distributional, along with top ranked indicative paraphrases.

Analysis. To analyze the contribution of the paraphrase component to the classification, we focused on the differences between the distributional and integrated models on the `Tratz-Coarse` lexical split. Examination of the per-relation F_1 scores revealed that the relations for which performance improved the most in the integrated model were TOPICAL (+11.1 F_1 points), OBJECTIVE (+5.5), ATTRIBUTE (+3.8) and LOCATION/PART_WHOLE (+3.5).

Table 5 provides examples of noun-compounds that were correctly classified by the integrated model while being incorrectly classified by the distributional model. For each noun-compound, we provide examples of top ranked paraphrases which are indicative of the gold label relation.

6 Compositionality Analysis

Our paraphrasing approach at its core assumes compositionality: only a noun-compound whose meaning is derived from the meanings of its constituent words can be rephrased using them. In §3.2 we added negative samples to the training data to simulate non-compositional noun-compounds, which are included in the classification dataset (§5.2). We assumed that these compounds, more often than compositional ones would consist of unrelated constituents (*spelling bee*, *sacred cow*), and added instances of random unrelated nouns with '[w$_2$] is unrelated to [w$_1$]'. Here, we assess whether our model succeeds to recognize non-compositional noun-compounds.

We used the compositionality dataset of Reddy et al. (2011) which consists of 90 noun-compounds along with human judgments about their compositionality in a scale of 0-5, 0 being non-compositional and 5 being compositional. For each noun-compound in the dataset, we predicted the 15 best paraphrases and analyzed the errors. The most common error was predicting paraphrases for idiomatic compounds which may have a plausible concrete interpretation or which originated from one. For example, it predicted that *silver spoon* is simply a *spoon* made of *silver* and that *monkey business* is a *business* that buys or raises *monkeys*. In other cases, it seems that the strong prior on one constituent leads to ignoring the other, unrelated constituent, as in predicting "*wedding* made of *diamond*". Finally, the "unrelated" paraphrase was predicted for a few compounds, but those are not necessarily non-compositional (*application form*, *head teacher*). We conclude that the model does not address compositionality and suggest to apply it only to compositional compounds, which may be recognized using compositionality prediction methods as in Reddy et al. (2011).

7 Conclusion

We presented a new semi-supervised model for noun-compound paraphrasing. The model differs from previous models by being trained to predict both a paraphrase given a noun-compound, and a missing constituent given the paraphrase and the other constituent. This results in better generalization abilities, leading to improved performance in two noun-compound interpretation tasks. In the future, we plan to take generalization one step further, and explore the possibility to use the biLSTM for generating completely new paraphrase templates unseen during training.

Acknowledgments

This work was supported in part by an Intel ICRI-CI grant, the Israel Science Foundation grant 1951/17, the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), and Theo Hoffenberg. Vered is also supported by the Clore Scholars Programme (2017), and the AI2 Key Scientific Challenges Program (2017).

References

Regina Barzilay and R. Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P01-1008.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1 .

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2009. Semeval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*. Association for Computational Linguistics, Boulder, Colorado, pages 100–105. http://www.aclweb.org/anthology/W09-2416.

Corina Dima. 2016. *Proceedings of the 1st Workshop on Representation Learning for NLP*, Association for Computational Linguistics, chapter On the Compositionality and Semantic Interpretation of English Noun Compounds, pages 27–39. https://doi.org/10.18653/v1/W16-1604.

Corina Dima and Erhard Hinrichs. 2015. Automatic noun compound interpretation using deep neural networks and word embeddings. *IWCS 2015* page 173.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM* 51(12):68–74.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 758–764. http://aclweb.org/anthology/N13-1092.

Roxana Girju. 2007. Improving the interpretation of noun phrases with cross-linguistic information. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 568–575. http://www.aclweb.org/anthology/P07-1072.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Iris Hendrickx, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, pages 138–143. http://aclweb.org/anthology/S13-2025.

Ralf Herbrich. 2000. Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers* pages 115–132.

Nam Su Kim and Preslav Nakov. 2011. Large-scale noun compound interpretation using bootstrapping and the web as a corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 648–658. http://aclweb.org/anthology/D11-1060.

Su Nam Kim and Timothy Baldwin. 2007. Interpreting noun compounds using bootstrapping and sense collocation. In *Proceedings of Conference of the Pacific Association for Computational Linguistics*. pages 129–136.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 970–976. http://www.aclweb.org/anthology/N15-1098.

Guofu Li, Alejandra Lopez-Fernandez, and Tony Veale. 2010. Ucd-goggle: A hybrid system for noun compound paraphrasing. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 230–233.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 881–893.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Preslav Nakov. 2013. On the interpretation of noun compounds: Syntax, semantics, and entailment. *Natural Language Engineering* 19(03):291–330.

Preslav Nakov and Marti Hearst. 2006. Using verbs to characterize noun-noun relations. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, pages 233–244.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Fifth international workshop on computational semantics (IWCS-5)*. pages 285–301.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*. volume 27, pages 372–376.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Paul Nulty and Fintan Costello. 2010. Ucd-pn: Selecting general paraphrases using conditional probability. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 234–237.

Diarmuid Ó Séaghdha and Ann Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Association for Computational Linguistics, Athens, Greece, pages 621–629. http://www.aclweb.org/anthology/E09-1071.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open ie. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*. Association for Computational Linguistics, San Diego, CA, pages 35–39. http://www.aclweb.org/anthology/W16-1307.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Association for Computational Linguistics, Rochester, New York, pages 564–571. http://www.aclweb.org/anthology/N/N07/N07-1071.

Marius Pasca. 2015. Interpreting compound noun phrases using web search queries. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 335–344. https://doi.org/10.3115/v1/N15-1037.

Ellie Pavlick and Marius Pasca. 2017. Identifying 1950s american jazz musicians: Fine-grained isa extraction via modifier composition. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 2099–2109. http://aclweb.org/anthology/P17-1192.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 210–218. http://www.aclweb.org/anthology/I11-1024.

Vered Shwartz and Chris Waterson. 2018. Olive oil is made of olives, baby oil is made for babies: Interpreting noun compounds using paraphrases in a neural model. In *The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. New Orleans, Louisiana.

Richard Socher, Brody Huval, D. Christopher Manning, and Y. Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1201–1211. http://aclweb.org/anthology/D12-1110.

Nitesh Surtani, Arpita Batra, Urmi Ghosh, and Soma Paul. 2013. Iiit-h: A corpus-driven co-occurrence based probabilistic model for noun compound paraphrasing. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. volume 2, pages 153–157.

Stephen Tratz. 2011. *Semantically-enriched parsing for natural language understanding*. University of Southern California.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 678–687. http://www.aclweb.org/anthology/P10-1070.

Tim Van de Cruys, Stergos Afantenos, and Philippe Muller. 2013. Melodi: A supervised distributional approach for free paraphrasing of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (Se-*

mEval 2013). Association for Computational Linguistics, Atlanta, Georgia, USA, pages 144–147. http://www.aclweb.org/anthology/S13-2026.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research* 15(1):3221–3245.

Yannick Versley. 2013. Sfs-tue: Compound paraphrasing with a language model and discriminative reranking. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. volume 2, pages 148–152.

Sander Wubben. 2010. Uvt: Memory-based pairwise ranking of paraphrasing verbs. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 260–263.

Clarissa Xavier and Vera Lima. 2014. Boosting open information extraction with noun-based relations. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 1263–1271.

Searching for the X-Factor: Exploring Corpus Subjectivity for Word Embeddings

Maksim Tkachenko and **Chong Cher Chia** and **Hady W. Lauw**
School of Information Systems
Singapore Management University
maksim.tkatchenko@gmail.com
{ccchia.2014,hadywlauw}@smu.edu.sg

Abstract

We explore the notion of subjectivity, and hypothesize that word embeddings learnt from input corpora of varying levels of subjectivity behave differently on natural language processing tasks such as classifying a sentence by sentiment, subjectivity, or topic. Through systematic comparative analyses, we establish this to be the case indeed. Moreover, based on the discovery of the outsized role that sentiment words play on subjectivity-sensitive tasks such as sentiment classification, we develop a novel word embedding *SentiVec* which is infused with sentiment information from a lexical resource, and is shown to outperform baselines on such tasks.

1 Introduction

Distributional analysis methods such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) have been critical for the success of many large-scale natural language processing (NLP) applications (Collobert et al., 2011; Socher et al., 2013; Goldberg, 2016). These methods employ distributional hypothesis (i.e., words used in the same contexts tend to have similar meaning) to derive distributional meaning via context prediction tasks and produce dense word embeddings.

While there have been active and ongoing research on improving word embedding methods (see Section 5), there is a relative dearth of study on the impact that an input corpus may have on the quality of the word embeddings. The previous preoccupation centers around corpus size, i.e., a larger corpus is perceived to be richer in statistical information. For instance, popular corpora include Wikipedia, Common Crawl, and Google News.

We postulate that there may be variations across corpora owing to factors that affect language use. Intuitively, the many things we write (a work email, a product review, an academic publication, etc.) may each involve certain stylistic, syntactic, and lexical choices, resulting in meaningfully different distributions of word cooccurrences. Consequently, such factors may be encoded in the word embeddings, and input corpora may be differentially informative towards various NLP tasks.

In this work, we are interested in the notion of *subjectivity*. Some NLP tasks, such as sentiment classification, revolve around subjective expressions of likes or dislikes. Others, such as topic classification, revolve around more objective elements of whether a document belongs to a topic (e.g., science, politics). Our central hypothesis is that word embeddings learnt from input corpora of contrasting levels of subjectivity perform differently when classifying sentences by sentiment, subjectivity, or topic. As the *first contribution*, we outline an experimental scheme to explore this hypothesis in Section 2, and conduct a series of controlled experiments in Section 3 establishing that there exists a meaningful difference between word embeddings derived from objective vs. subjective corpora. We further systematically investigate factors that could potentially explain the differences.

Upon discovering from the investigation that sentiment words play a particularly important role in subjectivity-sensitive NLP tasks, such as sentiment classification, as the *second contribution*, in Section 4 we develop *SentiVec*, a novel word embedding method infused with information from lexical resources such as a sentiment lexicon. We further identify two alternative lexical objectives: *Logistic SentiVec* based on discriminative logistic regression, and *Spherical SentiVec* based on soft clustering effect of von Mises-Fisher distributions. In Section 6, the proposed word embeddings show evident improvements on sentiment classification, as compared to the base model Word2Vec and other baselines using the same lexical resource.

2 Data and Methodology

We lay out the methodology for generating word embeddings of contrasting subjectivity, whose effects are tested on several text classification tasks.

2.1 Generating Word Embeddings

As it is difficult to precisely quantify the degree of subjectivity of a corpus, we resort to generating word embeddings from two corpora that contrast sharply in subjectivity, referring to them as the *Objective Corpus* and the *Subjective Corpus*.

Objective Corpus As virtually all contents are written by humans, an absolutely objective corpus (in the philosophical sense) may prove elusive. There are however exemplars where, by construction, a corpus aspires to be as objective as possible, and probably achieves that in practical terms. We postulate that one such corpus is Wikipedia. Its list of policies and guidelines[1], assiduously enforced by an editorial team, specify that an article must be written from a *neutral point of view*, which among other things means *"representing fairly, proportionately, and, as far as possible, without editorial bias, all of the significant views that have been published by reliable sources on a topic."*. Moreover, it is a common resource for training distributional word embeddings and adopted widely by the research community to solve various NLP problems. Hence, in this study, we use Wikipedia as the *Objective Corpus*.

Subjective Corpus By extension, one may then deem a corpus subjective if its content does not at least meet Wikipedia's *neutral point of view* requirement. In other words, if the content is replete with personal feelings and opinions. We posit that product reviews would be one such corpus. For instance, Amazon's Community Guideline[2] states that *"Amazon values diverse opinions"*, and that *"Content you submit should be relevant and based on your own honest opinions and experience."*. Reviews consist of expressive content written by customers, and may not strive for the neutrality of an encyclopedia. We rely on a large corpus of Amazon reviews from various categories (e.g., electronics, jewelry, books, and etc.) (McAuley et al., 2015) as the *Subjective Corpus*.

Word Embeddings For the comparative analysis in Section 3, we employ *Word2Vec* (reviewed below) to generate word embeddings from each corpus. Later on in Section 4, we will propose a new word embedding method called *SentiVec*.

For Word2Vec, we use the Skip-gram model to train distributional word embeddings on the *Objective Corpus* and the *Subjective Corpus* respectively. Skip-gram aims to find word embeddings that are useful for predicting nearby words. The objective is to maximize the context probability:

$$\log \mathcal{L}(W;C) = \sum_{w \in W} \sum_{w' \in C(w)} \log \mathrm{P}(w'|w), \qquad (1)$$

where W is an input corpus and $C(w)$ is the context of token w. The probability of context word w', given observed word w is defined via softmax:

$$\mathrm{P}(w'|w) = \frac{\exp(v_{w'} \cdot v_w)}{\sum_{\hat{w} \in V} \exp(v_{\hat{w}} \cdot v_w)}, \qquad (2)$$

where v_w and $v_{w'}$ are corresponding embeddings and V is the corpus vocabulary. Though theoretically sound, the formulation is computationally impractical and requires tractable approximation.

Mikolov et al. (2013) propose two efficient procedures to optimize (1): Hierarchical Softmax and Negative Sampling (NS). In this work we focus on the widely adopted NS. The intuition is that a "good" model should be able to differentiate observed data from noise. The differentiation task is defined using logistic regression; the goal is to tell apart real context-word pair (w', w) from randomly generated noise pair $(\hat{w}, w)$. Formally,

$$\log \mathcal{L}_{[w',w]} = \log \sigma(v_{w'} \cdot v_w) + \sum_{i=1}^{k} \log \sigma(-v_{\hat{w}_i} \cdot v_w), \qquad (3)$$

where $\sigma(\cdot)$ is a sigmoid function, and $\{\hat{w}_i\}_{i=1}^{k}$ are negative samples. Summing up all the context-word pairs, we derive the NS Skip-gram objective:

$$\log \mathcal{L}_{word2vec}(W;C) = \sum_{w \in W} \sum_{w' \in C(w)} \log \mathcal{L}_{[w',w]}. \qquad (4)$$

Training word embeddings with Skip-gram, we keep the same hyperparameters across all the runs: 300 dimensions for embeddings, $k = 5$ negative samples, and window of 5 tokens. The *Objective*

[1]https://en.wikipedia.org/wiki/Wikipedia:List_of_policies_and_guidelines

[2]https://www.amazon.com/gp/help/customer/display.html?nodeId=201929730

and *Subjective* corpora undergo the same preprocessing, i.e., discarding short sentences (<5 tokens) and rare words (<10 occurrences), removing punctuation, normalizing Unicode symbols.

2.2 Evaluation Tasks

To compare word embeddings, we need a common yardstick. It is difficult to define an inherent quality to word embeddings. Instead, we put them through several evaluation tasks that can leverage word embeddings and standardize their formulations as binary classification tasks. To boil the comparisons down to the essences of word embeddings (which is our central focus), we rely on standardized techniques so as to attribute as much of the differences as possible to the word embeddings. We use logistic regression for classification, and represent a text snippet (e.g., a sentence) in the feature space as the average of the word embeddings of tokens in the snippet (ignoring out-of-vocabulary tokens). The evaluation metric is the average accuracy from 10-fold cross validation.

There are three evaluation tasks of varying degrees of hypothetical subjectivity, as outlined below. Each may involve multiple datasets.

Sentiment Classification Task This task classifies a sentence into either *positive* or *negative*. We use two groups of datasets as follows.

The first group consists of 24 datasets from *UCSD Amazon product data*[3] corresponding to various product categories. Each review has a rating from 1 to 5, which is transformed into *positive* (ratings 4 or 5) or *negative* (ratings 1 or 2) class. For each dataset respectively, we sample 5000 sentences each from the positive and negative reviews. Note that these sentences used for this evaluation task have not participated in the generation of word embeddings. Due to space constraint, in most cases we present the average accuracy across the datasets, but where appropriate we enumerate the results for each dataset.

The second is *Cornell's sentence polarity dataset v1.0*[4] (Pang and Lee, 2005), made up of 5331 each of positive and negative sentences from Rotten Tomatoes movie reviews. The inclusion of this out-of-domain evaluation dataset is useful for examining whether the performance of word embeddings from the *Subjective Corpus* on the first group above may inadvertently be affected by in-domain advantage arising from its Amazon origin.

Subjectivity Classification Task This task classifies a sentence into *subjective* or *objective*. The dataset is *Cornell's subjectivity dataset v1.0*[5], consisting of 5000 subjective sentences derived from Rotten Tomatoes (RT) reviews and 5000 objective sentences derived from IMDB plot summaries (Pang and Lee, 2004). This task is probably less sensitive to the subjectivity within word embeddings than sentiment classification, as determining whether a sentence is subjective or objective should ideally be an objective undertaking.

Topic Classification Task We use the 20 Newsgroups dataset[6] ("bydate" version), whereby the newsgroups are organized into six subject matter groupings. We extract the message body and split them into sentences. Each group's sentences then form the *in-topic* class, and we randomly sample an equivalent number of sentences from the remaining newsgroups to form the *out-of-topic* class. This results in six datasets, each corresponding to a binary classification task. In most cases, we present the average results, and where appropriate we enumerate the results for each dataset. Hypothetically, this task is the least affected by the subjectivity within word embeddings.

3 Comparative Analyses of Subjective vs. Objective Corpora

We conduct a series of comparative analyses under various setups. For each, we compare the performance in the evaluation tasks when using the *Objective Corpus* and the *Subjective Corpus*. Table 1 shows the results for this series of analyses.

Initial Condition Setup I seeks to answer whether there is any difference between word embeddings derived from the *Objective Corpus* and the *Subjective Corpus*. The word embeddings were trained on the whole data respectively. Table 1 shows the corpus statistics and classification accuracies. Evidently, the *Subjective* word embeddings outperform the *Objective* word embeddings on all the evaluation tasks. The margins are largest for sentiment classification (86.5% vs. 81.5% or +5% Amazon, and 78.2% vs. 75.4% or +2.8% on Rotten Tomatoes or RT). For subjectivity and topic classifications, the differences are smaller.

[3]http://jmcauley.ucsd.edu/data/amazon/

[4]http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.README.1.0.txt

[5]http://www.cs.cornell.edu/people/pabo/movie-review-data/subjdata.README.1.0.txt

[6]http://qwone.com/~jason/20Newsgroups/

Setup	Corpus	Corpus Statistics			Classification (Accuracy)			
		# types	# tokens	# sentences	Sentiment		Subjectivity	Topic
					Amazon	RT		
I	Objective	1.34M	1.81B	89M	81.5	75.4	90.5	83.2
	Subjective	1.47M	5.49B	313M	86.5	78.2	91.1	83.4
II	Objective	1.34M	1.81B	89M	81.5	75.4	90.5	83.2
	Subjective	0.59M	1.56B		85.5	77.9	90.7	82.8
III	Objective	0.29M	1.75B	89M	81.6	75.6	90.6	83.4
	Subjective		1.54B		85.4	77.9	90.6	82.8

Table 1: Controlled comparison of Objective and Subjective corpora

As earlier hypothesized, the sentiment classification task is more sensitive to subjectivity within word embeddings than the other tasks. Therefore, training word embeddings on a subjective corpus may confer an advantage for such tasks. On the other hand, the corpus statistics show a substantial difference in corpus size, which could be an alternative explanation for the outperformance by the *Subjective Corpus* if the larger corpus contains more informative distributional statistics.

Controlling for Corpus Size In Setup II, we keep the number of sentences in both corpora the same, by randomly downsampling sentences in the *Subjective Corpus*. This procedure consequently reduces the number of types and tokens (see Table 1, Setup II, Corpus Statistics). Note that the number of tokens in the *Subjective* corpus is now fewer than in the *Objective*, the latter suffers no change. Yet, even after a dramatic reduction in size, the *Subjective* embeddings still outperform the *Objective* significantly on both datasets of the sentiment classification task (+4% on Amazon and +2.5% on RT), while showing similar performance on subjectivity and topic classifications.

This bolsters the earlier observation that sentiment classification is more sensitive to subjectivity. While there is a small effect due to corpus size difference, the gap in performance between *Subjective* and *Objective* embeddings on sentiment classification is still significant and cannot be explained away by the corpus size alone.

Controlling for Vocabulary While the *Subjective Corpus* has a much smaller vocabulary (i.e., # types), we turn a critical eye on whether its apparent advantage lies in having access to special word types that do not exist in the *Objective Corpus*. In Setup III, we keep the training vocabulary the same for both, removing the types that are present in one corpus but not in the other, so that out-of-vocabulary words are ignored in the training phase. Table 1, Setup III, shows significant reduction in types for both corpora. Yet, the outperformance by the *Subjective* embeddings on the sentiment classification task still stands (+3.8% on Amazon and +2.3% on RT). Moreover, it is so for both Amazon and Rotten Tomatoes datasets, implying that it is not due to close in-domain similarity between the corpora used for training the word embeddings and the classification tasks.

Objective Corpus	*Subjective Corpus*
waste, money, return, love, great, and, loves, refund, Great, This, product, recommend, this, even, Very, returned, easy, not, send, sent, customer, item, broke, defective, her	money, waste, return, and, Great, love, refund, recommend, great, this, loves, even, product, This, Very, easy, item, junk, anyone, Don't, horrible, gift, poor, Do, returned

Table 2: Top words of misclassified sentences

Significant Words To get more insights on the difference between the *Subjective* and *Objective* corpora, we analyze the mistakes word embeddings make on the development folds. At this point we focus on the sentiment classification task and specifically on the Amazon data, which indicates the largest performance differences in the controlled experiments (see Table 1, Setup III).

As words are still the main unit of information in distributional word embeddings, we extract words strongly associated with misclassified sentences. We employed log-odds ratio with informative Dirichlet prior method (Monroe et al., 2008) to quantify this association. It is used to contrast the words in misclassified vs. correctly classified sentences, and accounts for the variance of words and their prior counts taken from a large corpus.

Table 2 shows the top 25 words most associated with the misclassified sentences, sorted by their association scores. On average 50% of the mistakes overlap for both word embeddings, therefore, some of the words are included in both lists. $40 - 44\%$ of these words carry positive or negative sentiment connotations in general (see the underlined words in Table 2), while other words like *return* or *send* may carry sentiment connotation in e-commerce context. We check if a word carries sentiment connotation using sentiment lexicon compiled by Hu and Liu (2004), including 6789 words along with positive or negative labels.

We also observe linguistic negations (i.e., *not*, *Don't*). For instance, the word most associated with the *Objective*-specific mistakes (excluding the *Subjective* misclassified sentences) is *not*, which suggests that perhaps *Subjective* word embedding accommodates better understanding of linguistic negations, which may partially explain the difference. However, our methodology as outlined in Section 2.2 permits exchangeable word order and is not intended to analyze structural interaction between words. We focus on further analysis of sentiment words, leaving linguistic negations in word embeddings for future investigation.

Controlling for Sentiment Words To control for the "amount" of sentiment in the *Subjective* and *Objective* corpora, we use sentiment lexicon compiled by Hu and Liu (2004). For each corpus, we create two subcorpora: *With Sentiment* contains only the sentences with at least one word from the sentiment lexicon, while *Without Sentiment* is the complement. We match the corpora on the number of sentences, downsampling the larger corpus, train word embeddings on each subcorpus, and proceed with the classification experiments. Table 3 shows the results, including that of random word embeddings for reference. Sentiment lexicon has a significant impact on the performance of sentiment and subjectivity classifications, and a smaller impact on topic classification. Without sentiment, the *Subjective* embeddings prove more robust, still outperforming the *Objective* on sentiment classification, while the *Objective* performs close to random word embeddings on Amazon .

In summary, evidences from the series of controlled experiments support the existence of some X-factor to the *Subjective* embeddings, which confers superior performance in subjectivity-sensitive tasks such as sentiment classification.

Corpus	Subcorpus Sentiment?	Sentiment Amazon	Sentiment RT	Subjectivity	Topic
Objective	With	81.8	75.2	90.7	83.1
	Without	76.1	67.2	87.8	82.6
Subjective	With	85.5	78.0	90.3	82.5
	Without	79.8	71.0	89.1	82.2
Random Embeddings		76.1	62.2	80.1	71.5

Table 3: With and without sentiment

4 Sentiment-Infused Word Embeddings

To leverage the consequential sentiment information, we propose a family of methods, called *SentiVec*, for training distributional word embeddings that are infused with information on the sentiment polarity of words. The methods are built upon *Word2Vec* optimization algorithm and make use of available lexical sentiment resources such as SentiWordNet (Baccianella et al., 2010), sentiment lexicon by Hu and Liu (2004), and etc.

SentiVec seeks to satisfy two objectives, namely context prediction and lexical category prediction:

$$\log \mathcal{L} = \log \mathcal{L}_{word2vec}(W;C) + \lambda \log \mathcal{L}_{lex}(W,L), \quad (5)$$

where $\mathcal{L}_{word2vec}(W;C)$ is the Skip-gram objective as in (4); $\mathcal{L}_{lex}(W,L)$ is a lexical objective for corpus W and lexical resource L; and λ is a tradeoff parameter. Lexical resource $L = \{X_i\}_{i=1}^n$ comprises of n word sets, each X_i contains words of the same category. For sentiment classification, we consider *positive* and *negative* word categories.

4.1 Logistic SentiVec

Logistic SentiVec admits lexical resource in the form of two disjoint word sets, $L = \{X_1, X_2\}$, $X_1 \cap X_2 = \emptyset$. The objective is to tell apart which word set of L word w belongs to:

$$\log \mathcal{L}_{lex}(W,L) \quad (6)$$
$$= \sum_{w \in X_1} \log \mathrm{P}(w \in X_1) + \sum_{w \in X_2} \log \mathrm{P}(w \in X_2).$$

We further tie these probabilities together, and cast the objective as a logistic regression problem:

$$\mathrm{P}(w \in X_1) = 1 - \mathrm{P}(w \in X_2) = \sigma(v_w \cdot \tau), \quad (7)$$

where v_w is a word embedding and τ is a direction vector. Since word embeddings are generally invariant to scaling and rotation when used as downstream feature representations, τ can be chosen randomly and fixed during training. We experiment with randomly sampled unit length directions. For simplicity, we also scale embedding v_w to its unit length when computing $v_w \cdot \tau$, which now equals to cosine similarity between v_w and τ.

When v_w is completely aligned with τ, the cosine similarity between them is 1, which maximizes $\mathrm{P}(w \in X_1)$ and favors words in X_1. When v_w is opposite to τ, the cosine similarity equals to -1, which maximizes $\mathrm{P}(w \in X_2)$ and predicts vectors from X_2. Orthogonal vectors have cosine similarity of 0, which makes both $w \in X_1$ and $w \in X_2$ equally probable. Optimizing (6) makes the corresponding word embeddings of X_1 and X_2 gravitate to the opposite semispaces and simulates clustering effect for the words of the same category, while the *Word2Vec* objective prevents words from collapsing to the same directions.

Optimization The objective in (6) permits simple stochastic gradient ascent optimization and can be combined with negative sampling procedure for Skip-gram in (5). The gradient for unnormalized embedding v_w is solved as follows:

$$\left(\log \mathcal{L}_{[w \in X_1]}(D, L)\right)'_{v_{wi}} = \left(\log \mathrm{P}\left(x \in X_1\right)\right)'_{v_{wi}} \\ = \frac{1}{\|v_w\|^2} \sigma\left(-\frac{v_w \cdot \tau}{\|v_w\|}\right)\left(\tau_i \|v_w\| - v_{wi}\frac{v_w \cdot \tau}{\|v_w\|}\right) \tag{8}$$

The optimization equation for v_w, when $w \in X_2$, can be derived analogously.

4.2 Spherical SentiVec

Spherical SentiVec extends *Logistic SentiVec* by dealing with any number of lexical categories, $L = \{X_i\}_{i=1}^n$. As such, the lexical objective takes on generic form:

$$\log \mathcal{L}_{lex}(W, L) = \sum_{i=1}^{n} \sum_{w \in X_i} \log \mathrm{P}\left(w \in X_i\right), \tag{9}$$

Each $\mathrm{P}\left(w \in X_i\right)$ defines embedding generating process. We assume each length-normalized v_w for w of L is generated w.r.t. a mixture model of von Mises-Fisher (vMF) distributions. vMF is a probability distribution on a multidimensional sphere, characterized by parameters μ (mean direction) and κ (concentration parameter). Sampled points are concentrated around μ; the greater the κ, the closer the sampled points are to μ. We consider only unimodal vMF distributions, restricting concentration parameters to be strictly positive. Hereby, each $X_i \in L$ is assigned to vMF distribution parameters (μ_i, κ_i) and the membership probabilities are defined as follows:

$$\mathrm{P}(w \in X_i) = P(v_w; \mu_i, \kappa_i) = \frac{1}{Z_{\kappa_i}} e^{\kappa_i \mu_i \cdot v_w}, \tag{10}$$

where Z_κ is the normalization factor.

The *Spherical SentiVec* lexical objective forces words of every $X_i \in L$ to gravitate towards and concentrate around their direction mean μ_i. As in *Logistic SentiVec*, it simulates clustering effect for the words of the same set. In comparison to the direction vector of *Logistic SentiVec*, mean directions of *Spherical SentiVec* when fixed can substantially influence word embeddings training and must be carefully selected. We optimize the mean directions along with the word embeddings using alternating procedure resembling K-means clustering algorithm. For simplicity, we keep concentration parameters tied, $\kappa_1 = \kappa_2 = ... = \kappa_n = \kappa$, and treat κ as a hyperparameter of this algorithm.

Optimization We derive optimization procedure for updating word embeddings assuming fixed direction means. Like *Logistic SentiVec*, *Spherical SentiVec* can be combined with the negative sampling procedure of Skip-gram. The gradient for unnormalized word embedding v_w is solved by the following equation:

$$\left(\log \mathcal{L}_{[w \in X_i]}\left(W, L\right)\right)'_{v_{wj}} = \kappa_i \frac{\left(\mu_{ij}\|v_w\| - v_{wj}\frac{v_w \cdot \mu_i}{\|v_w\|}\right)}{\|v_w\|^2} \tag{11}$$

Once word embedding v_w $(w \in X_i)$ is updated, we revise direction mean μ_i w.r.t. maximum likelihood estimator:

$$\mu_i = \frac{\sum_{w \in X_i} v_w}{\left\|\sum_{w \in X_i} v_w\right\|}. \tag{12}$$

Updating the direction means in such a way ensures that the lexical objective is non-decreasing. Assuming the stochastic optimization procedure for $\mathcal{L}_{word2vec}$ complies with the same non-decreasing property, the proposed alternating procedure converges.

5 Related Work

There have been considerable research on improving the quality of distributional word embeddings. Bolukbasi et al. (2016) seek to debias word embeddings from gender stereotypes. Rothe and Schütze (2017) incorporate WordNet lexeme and synset information. Mrkšic et al. (2016) encode antonym-synonym relations. Liu et al. (2015) encode ordinal relations such as hypernym and hyponym. Kiela et al. (2015) augment Skip-gram to enforce lexical similarity or relatedness constraints, Bollegala et al. (2016) modify GloVe optimization procedure for the same purpose. Faruqui et al. (2015) employ semantic relations of PPDB, WordNet, FrameNet to *retrofit* word embeddings for various prediction tasks. We use this *Retrofitting* method[7] as a baseline.

Socher et al. (2011) derive multi-word embeddings for sentiment distribution prediction, while we focus on lexical distributional analysis. Maas et al. (2011) and Tang et al. (2016) use document-level sentiment annotations to fit word embeddings, but document annotation might not always be available for distributional analysis on neutral corpora such as Wikipedia. *SentiVec* relies on simple sentiment lexicon instead. *Refining* (Yu et al., 2018) aligns the sentiment scores taken from lexical resource and the cosine similarity scores of corresponding word embeddings. The method generally requires fine-grained sentiment scores for the words, which may not be available in some settings. We use *Refining* as a baseline and adopt coarse-grained sentiment lexicon for this method.

Villegas et al. (2016) compare various distributional word embeddings arising from the same corpus for sentiment classification, whereas we focus on the differentiation in input corpora and propose novel sentiment-infused word embeddings.

6 Experiments

The objective of experiments is to study the efficacy of *Logistic SentiVec* and *Spherical SentiVec* word embeddings on the aforementioned text classification tasks. One natural baseline is *Word2Vec*, as *SentiVec* subsumes its context prediction objective, while further incorporating lexical category prediction. We include two other baselines that can leverage the same lexical resource but in manners different from *SentiVec*, namely: *Retrofitting* (Faruqui et al., 2015) and *Refining* (Yu et al., 2018). For these methods, we generate their word embeddings based on Setup III (see Section 3). All the methods were run multiple times with various hyperparameters, optimized via grid-search; for each we present the best performing setting.

First, we discuss the sentiment classification task. Table 4 shows the unfolded results for the 24 classification datasets of Amazon, as well as for Rotten Tomatoes. For each classification dataset (row), and for the *Objective* and *Subjective* embedding corpora respectively, the best word embedding methods are shown in bold. An asterisk indicates statistically significant[8] results at 5% in comparison to *Word2Vec*. Both *SentiVec* variants outperform *Word2Vec* in the vast majority of the cases. The degree of outperformance is higher for the *Objective* than the *Subjective* word embeddings. This is a reasonable trend given our previous findings in Section 3. As the *Objective Corpus* encodes less information than the *Subjective Corpus* for sentiment classification, the former is more likely to benefit from the infusion of sentiment information from additional lexical resources. Note that the sentiment infusion into the word embeddings comes from separate lexical resources, and does not involve any sentiment classification label.

SentiVec also outperforms the two baselines that benefit from the same lexical resources. *Retrofitting* does not improve upon *Word2Vec*, with the two embeddings essentially indistinguishable (the difference is only noticeable at the second decimal point). *Refining* makes the word embeddings perform worse on the sentiment classification task. One possible explanation is that *Refining* normally requires fine-grained labeled lexicon, where the words are scored w.r.t. the sentiment scale, whereas we use sentiment lexicon of two labels (i.e., positive or negative). *SentiVec* accepts coarse-grained sentiment lexicons, and potentially could be extended to deal with fine-grained labels.

As previously alluded to, topic and subjectivity classifications are less sensitive to the subjectivity within word embeddings than sentiment classification. One therefore would not expect much, if any, performance gain from infusion of sentiment information. However, such infusion should not subtract or harm the quality of word embeddings either. Table 5 shows that the unfolded results for topic classification on the six datasets, and the result for subjectivity classification are similar across methods. Neither the *SentiVec* variants, nor *Retrofitting* and *Refining*, change the subjectivity and topic classification capabilities much, which means that the used sentiment lexicon is targeted only at the sentiment subspace of embeddings.

[7]Original code is available at: `https://github.com/mfaruqui/retrofitting`

[8]We use paired t-test to compute p-value.

Corpus/Category	Objective Embeddings					Subjective Embeddings				
	Word2Vec	Retrofitting	Refining	SentiVec		Word2Vec	Retrofitting	Refining	SentiVec	
				Spherical	Logistic				Spherical	Logistic
Amazon										
Instant Video	84.1	84.1	81.9	**84.9***	**84.9***	87.8	87.8	86.9	88.1	**88.2**
Android Apps	83.0	83.0	80.9	**84.0***	**84.0***	86.3	86.3	85.0	**86.6**	86.5
Automotive	80.7	80.7	78.8	81.0	**81.3**	**85.1**	**85.1**	83.8	84.9	85.0
Baby	80.9	80.9	78.6	82.1	**82.2***	84.2	84.2	82.8	84.4	**84.6**
Beauty	81.8	81.8	79.8	82.4	**82.7***	85.2	85.2	83.5	85.2	**85.4**
Books	80.9	80.9	78.9	81.0	**81.3**	85.3	85.3	83.6	85.3	**85.5**
CD & Vinyl	79.4	79.4	77.6	79.4	**79.9**	83.5	83.5	81.9	**83.7**	83.6
Cell Phones	82.2	82.2	80.0	82.9	**83.0***	86.8	86.8	85.3	86.8	**87.0**
Clothing	82.6	82.6	80.7	83.8	**84.0***	86.3	86.3	84.7	86.4	**86.8**
Digital Music	82.3	82.3	80.5	82.8	**83.0***	**86.3**	**86.3**	84.6	86.1	**86.3**
Electronics	81.0	81.0	78.8	80.9	**81.3**	85.2	85.2	83.6	**85.3**	**85.3**
Grocery & Food	81.7	81.7	79.4	**83.1***	**83.1***	85.0	85.0	83.7	85.1	**85.6***
Health	79.7	79.7	77.9	**80.4***	80.4	84.0	84.0	82.3	84.0	**84.3**
Home & Kitchen	81.6	81.6	79.5	**82.1**	**82.1**	**85.4**	**85.4**	83.9	85.3	**85.4**
Kindle Store	84.7	84.7	83.2	85.2	**85.4***	88.3	88.3	87.2	88.3	**88.6**
Movies & TV	81.4	81.4	78.5	81.9	**81.9**	85.2	85.2	83.5	85.4	**85.5**
Musical Instruments	81.7	81.6	79.7	**82.4**	**82.4**	85.8	85.8	84.1	**85.9**	85.7
Office	82.0	82.0	80.0	**83.0***	82.9	86.1	86.1	84.5	86.4	**86.5***
Garden	80.4	80.4	77.9	81.0	**81.5**	84.1	84.1	82.5	84.3	**84.6***
Pet Supplies	79.7	79.7	77.5	**80.4**	80.2	83.2	83.2	81.5	83.4	**83.8**
Sports & Outdoors	80.8	80.8	79.1	**81.3***	81.2	84.6	84.6	83.1	84.3	**84.7**
Tools	81.0	81.0	79.3	81.0	**81.3**	84.7	84.7	83.2	84.8	**84.9**
Toys & Games	83.8	83.8	82.0	84.7	**84.9***	87.2	87.2	85.7	87.1	**87.5**
Video Games	80.3	80.3	77.4	81.5	**81.7***	84.9	84.9	83.2	**85.0**	84.9
Average	81.6	81.6	79.5	82.2	**82.4**	85.4	85.4	83.9	85.5	**85.7**
Rotten Tomatoes	75.6	75.6	73.4	**75.8***	75.4	**77.9**	**77.9**	76.7	77.7	**77.9**

Table 4: Comparison of Sentiment-Infused Word Embeddings on Sentiment Classification Task

Corpus/Category	Objective Embeddings					Subjective Embeddings				
	Word2Vec	Retrofitting	Refining	SentiVec		Word2Vec	Retrofitting	Refining	SentiVec	
				Spherical	Logistic				Spherical	Logistic
Topic										
Computers	79.8	79.8	79.6	79.6	79.8	79.8	79.8	79.8	79.7	79.7
Misc	89.8	89.8	89.7	89.8	90.0	90.4	90.4	90.6	90.4	90.3
Politics	84.6	84.6	84.4	84.5	84.6	83.8	83.8	83.5	83.6	83.5
Recreation	83.4	83.4	83.1	83.1	83.2	82.6	82.6	82.5	82.7	82.8
Religion	84.6	84.6	84.5	84.5	84.6	84.2	84.2	84.2	84.1	84.2
Science	78.2	78.2	78.2	78.1	78.3	76.4	76.4	76.1	76.7	76.6
Average	83.4	83.4	83.2	83.3	83.4	82.8	82.8	82.8	82.9	82.8
Subjectivity	90.6	90.6	90.0	90.6	90.6	90.6	90.6	90.3	90.7	90.8

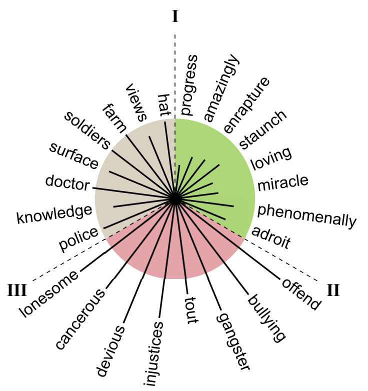
Table 5: Comparison of Word Embeddings on Subjectivity and Topic Classification Tasks

Illustrative Changes in Embeddings To give more insights on the difference between *SentiVec* and *Word2Vec*, we show "flower" diagrams in Figure 1 for *Logistic SentiVec* and Figure 2 for *Spherical SentiVec*. Each is associated with a reference word (e.g., *good* for Figure 1a), and indicates relative changes in cosine distances between the reference word and the testing words surrounding the "flower". Every testing word is associated with a "petal" or black axis extending from the center of the circle. The "petal" length is proportional to the relative distance change in two word embeddings: $\kappa = \frac{d_{SentiVec}(w_{ref}, w_{testing})}{d_{word2vec}(w_{ref}, w_{testing})}$, where $d_{SentiVec}$ and $d_{word2vec}$ are cosine distances between reference w_{ref} and testing $w_{testing}$ words in *SentiVec* and *Word2Vec* embeddings correspondingly. If the distance remains unchanged ($\kappa = 1$), then the "petal" points at the circumference; if the reference and testing words are closer in the *SentiVec* embedding than they are in *Word2Vec* ($\kappa < 1$), the "petal" lies inside the circle; when the distance increases ($\kappa > 1$), the "petal" goes beyond the circle.

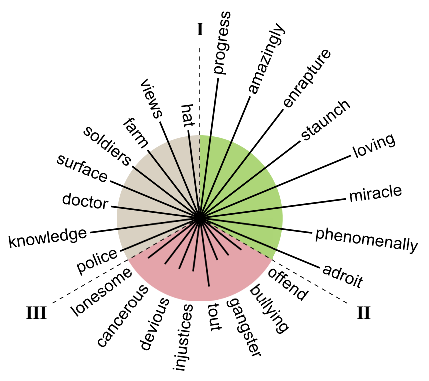
The diagrams are presented for Objective Embeddings[9]. We use three reference words: *good* (positive), *bad* (negative), *time* (neutral); as well as three groups of testing words: green for words randomly sampled from positive lexicon (Sector I-II), red for words randomly sampled from negative lexicon (Sector II-III), and gray for frequent neutral common nouns (Sector III-I).

Figure 1 shows changes produced by *Logistic SentiVec*. For the positive reference word (Figure 1a), the average distance to the green words is shortened, whereas the distance to the red words increases. The reverse is observed for the negative reference word (Figure 1b). This observation

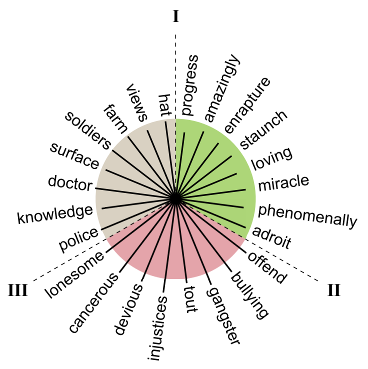
[9]The diagrams for Subjective Embeddings show the same trend, with the moderate changes.



(a) Reference word: **good** (positive)

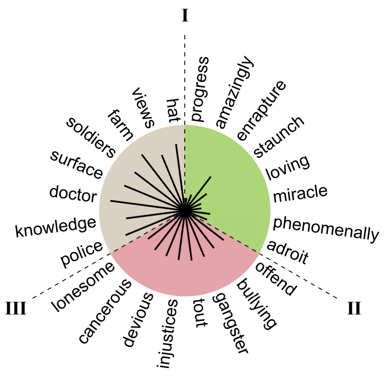


(b) Reference word: **bad** (negative)

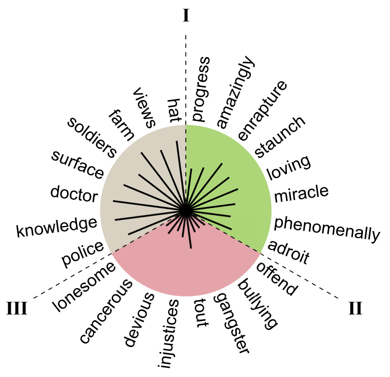


(c) Reference word: **time** (neutral)

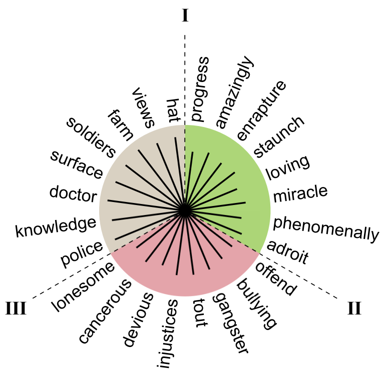
Figure 1: Relative changes in cosine distances in *Logistic SentiVec* contrasted with *Word2Vec*



(a) Reference word: **good** (positive)



(b) Reference word: **bad** (negative)



(c) Reference word: **time** (neutral)

Figure 2: Relative changes in cosine distances in *Spherical SentiVec* contrasted with *Word2Vec*

complies with the lexical objective (7) of *Logistic SentiVec*, which aims to separate the words of two different classes. Note that the gray words suffer only moderate change with respect to positive and negative reference words. For the neutral reference word (Figure 1c), the distances are only moderately affected across all testing groups.

Figure 2 shows that *Spherical SentiVec* tends to make embeddings more compact than *Logistic SentiVec*. As the former's lexical objective (9) is designed for clustering, but not for separation, we look at the comparative strength of the clustering effect on the testing words. For the positive reference word (Figure 2a), the largest clustering effect is achieved for the green words. For the negative reference word (Figure 2b), as expected, the red words are affected the most. The gray words suffer the least change for all the reference words.

In summary, *SentiVec* effectively provides an advantage for subjectivity-sensitive task such as sentiment classification, while not harming the performance of other text classification tasks.

7 Conclusion

We explore the differences between objective and subjective corpora for generating word embeddings, and find that there is indeed a difference in the embeddings' classification task performances. Identifying the presence of sentiment words as one key factor for the difference, we propose a novel method *SentiVec* to train word embeddings that are infused with the sentiment polarity of words derived from a separate sentiment lexicon. We further identify two lexical objectives: *Logistic SentiVec* and *Spherical SentiVec*. The proposed word embeddings show improvements in sentiment classification, while maintaining their performance on subjectivity and topic classifications.

Acknowledgments

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its NRF Fellowship Programme (Award No. NRF-NRFF2016-07).

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*. volume 10.

Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *Proceedings of AAAI*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of NIPS*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12(Aug).

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *JAIR* 57.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL-IJCNLP*. volume 1.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis* 16(4).

Nikola Mrkšic, Diarmuid OSéaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Sascha Rothe and Hinrich Schütze. 2017. Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics* 43(3).

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE TKDE* 28(2).

María Paula Villegas, María José Garciarena Ucelay, Juan Pablo Fernández, Miguel A Álvarez Carmona, Marcelo Luis Errecalde, and Leticia Cagnina. 2016. Vector-based word representations for sentiment analysis: a comparative study. In *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016).*.

L. C. Yu, J. Wang, K. R. Lai, and X. Zhang. 2018. Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26(3).

Word Embedding and WordNet Based Metaphor Identification and Interpretation

Rui Mao, Chenghua Lin and **Frank Guerin**
Department of Computing Science
University of Aberdeen
Aberdeen, United Kingdom
{r03rm16, chenghua.lin, f.guerin}@abdn.ac.uk

Abstract

Metaphoric expressions are widespread in natural language, posing a significant challenge for various natural language processing tasks such as Machine Translation. Current word embedding based metaphor identification models cannot identify the exact metaphorical words within a sentence. In this paper, we propose an unsupervised learning method that identifies and interprets metaphors at word-level without any preprocessing, outperforming strong baselines in the metaphor identification task. Our model extends to interpret the identified metaphors, paraphrasing them into their literal counterparts, so that they can be better translated by machines. We evaluated this with two popular translation systems for English to Chinese, showing that our model improved the systems significantly.

1 Introduction

Metaphor enriches language, playing a significant role in communication, cognition, and decision making. Relevant statistics illustrate that about one third of sentences in typical corpora contain metaphor expressions (Cameron, 2003; Martin, 2006; Steen et al., 2010; Shutova, 2016). Linguistically, metaphor is defined as a language expression that uses one or several words to represent another concept, rather than taking their literal meanings of the given words in the context (Lagerwerf and Meijers, 2008). Computational metaphor processing refers to modelling non-literal expressions (e.g., metaphor, metonymy, and personification) and is useful for improving many NLP tasks such as Machine Translation (MT) and Sentiment Analysis (Rentoumi et al., 2012). For instance, Google Translate failed in translating *devour* within a sentence, *"She devoured his novels."* (Mohammad et al., 2016), into Chinese. The term was translated into 吞噬, which takes the literal sense of *swallow* and is not understandable in Chinese. Interpreting metaphors allows us to paraphrase them into literal expressions which maintain the intended meaning and are easier to translate.

Metaphor identification approaches based on word embeddings have become popular (Tsvetkov et al., 2014; Shutova et al., 2016; Rei et al., 2017) as they do not rely on hand-crafted knowledge for training. These models follow a similar paradigm in which input sentences are first parsed into phrases and then the metaphoricity of the phrases is identified; they do not tackle word-level metaphor. E.g., given the former sentence *"She devoured his novels."*, the aforementioned methods will first parse the sentence into a verb-direct object phrase *devour novel*, and then detect the clash between *devour* and *novel*, flagging this phrase as a likely metaphor. However, which component word is metaphorical cannot be identified, as important contextual words in the sentence were excluded while processing these phrases. Discarding contextual information also leads to a failure to identify a metaphor when both words in the phrase are metaphorical, but taken out of context they appear literal. E.g., *"This young man knows how to climb the social ladder."* (Mohammad et al., 2016) is a metaphorical expression. However, when the sentence is parsed into a verb-direct object phrase, *climb ladder*, it appears literal.

In this paper, we propose an unsupervised metaphor processing model which can identify and interpret linguistic metaphors at the word-level. Specifically, our model is built upon word embedding methods (Mikolov et al., 2013) and uses WordNet (Fellbaum, 1998) for lexical relation acquisition. Our model is distinguished from existing methods in two aspects. First, our model is generic which does not constrain the source domain of metaphor. Second, the developed model does not rely on any labelled data for model training, but rather captures metaphor in an unsupervised, data-driven manner. Linguistic metaphors are identified by modelling the distance (in vector space) between the target word's literal and metaphorical senses. The metaphorical sense within a sentence is identified by its surrounding context within the sentence, using word embedding representations and WordNet. This novel approach allows our model to operate at the sentence level without any preprocessing, e.g., dependency parsing. Taking contexts into account also addresses the issue that a two-word phrase appears literal, but it is metaphoric within a sentence (e.g., the *climb ladder* example).

We evaluate our model against three strong baselines (Melamud et al., 2016; Shutova et al., 2016; Rei et al., 2017) on the task of metaphor identification. Extensive experimentation conducted on a publicly available dataset (Mohammad et al., 2016) shows that our model significantly outperforms the unsupervised learning baselines (Melamud et al., 2016; Shutova et al., 2016) on both phrase and sentence evaluation, and achieves equivalent performance to the state-of-the-art deep learning baseline (Rei et al., 2017) on phrase-level evaluation. In addition, while most of the existing works on metaphor processing solely evaluate the model performance in terms of metaphor classification accuracy, we further conducted another set of experiments to evaluate how metaphor processing can be used for supporting the task of MT. Human evaluation shows that our model improves the metaphoric translation significantly, by testing on two prominent translation systems, namely, Google Translate[1] and Bing Translator[2]. To our best knowledge, this is the first metaphor processing model that is evaluated on MT.

To summarise, the contributions of this paper are two-fold: (1) we proposed a novel framework for metaphor identification which does not require any preprocessing or annotated corpora for training; (2) we conducted, to our knowledge, the first metaphor interpretation study of applying metaphor processing for supporting MT. We describe related work in §2, followed by our labelling method in §4, experimental design in §5, results in §6 and conclusions in §7.

[1]https://translate.google.co.uk

[2]https://www.bing.com/translator

2 Related Work

A wide range of methods have been applied for computational metaphor processing. Turney et al. (2011); Neuman et al. (2013); Assaf et al. (2013) and Tsvetkov et al. (2014) identified metaphors by modelling the abstractness and concreteness of metaphors and non-metaphors, using a machine usable dictionary called MRC Psycholinguistic Database (Coltheart, 1981). They believed that metaphorical words would be more abstract than literal ones. Some researchers used topic models to identify metaphors. For instance, Heintz et al. (2013) used Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to model source and target domains, and assumed that sentences containing words from both domains are metaphorical. Strzalkowski et al. (2013) assumed that metaphorical terms occur out of the topic chain, where a topic chain is constructed by topical words that reveal the core discussion of the text. Shutova et al. (2017) performed metaphorical concept mappings between the source and target domains in multi-languages using both unsupervised and semi-supervised learning approaches. The source and target domains are represented by semantic clusters, which are derived through the distribution of the co-occurrences of words. They also assumed that when contextual vocabularies are from different domains then there is likely to be a metaphor.

There is another line of approaches based on word embeddings. Generally, these works are not limited by conceptual domains and hand-crafted knowledge. Shutova et al. (2016) proposed a model that identified metaphors by employing word and image embeddings. The model first parses sentences into phrases which contain target words. In their word embedding based approach, the metaphoricity of a phrase was identified by measuring the cosine similarity of two component words in the phrase, based on their input vectors from Skip-gram word embeddings. If the cosine similarity is higher than a threshold, the phrase is identified as literal; otherwise metaphorical. Rei et al. (2017) identified metaphors by introducing a deep learning architecture. Instead of using word input vectors directly, they filtered out noisy in-

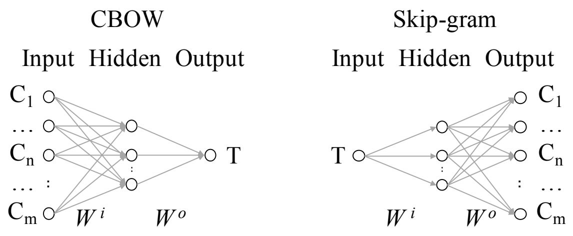


Figure 1: CBOW and Skip-gram framework.

formation in the vector of one word in a phrase, projecting the word vector into another space via a sigmoid activation function. The metaphoricity of the phrases was learnt via training a supervised deep neural network.

The above word embedding based models, while demonstrating some success in metaphor identification, only explored using input vectors, which might hinder their performance. In addition, metaphor identification is highly dependent on its context. Therefore, phrase-level models (e.g., Tsvetkov et al. (2014); Shutova et al. (2016); Rei et al. (2017)) are likely to fail in the metaphor identification task if important contexts are excluded. In contrast, our model can operate at the sentence level which takes into account rich context and hence can improve the performance of metaphor identification.

3 Preliminary: CBOW and Skip-gram

Our metaphor identification framework is built upon word embedding, which is based on Continuous Bag of Words (CBOW) and Skip-gram (Mikolov et al., 2013).

In CBOW (see Figure 1), the input and output layers are context (C) and centre word (T) one-hot encodings, respectively. The model is trained by maximizing the probability of predicting a centre word, given its context (Rong, 2014):

$$\arg\max p(t|c_1, ..., c_n, ..., c_m) \tag{1}$$

where t is a centre word, c_n is the nth context word of t within a sentence, totally m context words. CBOW's hidden layer is defined as:

$$\begin{aligned} H_{CBOW} &= \frac{1}{m} \times W^{i\top} \times \sum_{n=1}^{m} C_n \\ &= \frac{1}{m} \times \sum_{n=1}^{m} v_{c,n}^{i\top} \end{aligned} \tag{2}$$

where C_n is the one-hot encoding of the nth context word, $v_{c,n}^i$ is the nth context word row vector (input vector) in W^i which is a weight matrix between input and hidden layers. Thus, the hidden layer is the transpose of the average of input vectors of context words. The probability of predicting a centre word in its context is given by a softmax function below:

$$u_t = W_t^{o\top} \times H_{CBOW} = v_t^{o\top} \times H_{CBOW} \tag{3}$$

$$p(t|c_1, ..., c_n, ..., c_m) = \frac{\exp(u_t)}{\sum_{j=1}^{V} \exp(u_j)} \tag{4}$$

where W_t^o is equivalent to the output vector v_t^o which is essentially a column vector in a weight matrix W^o that is between hidden and output layers, aligning with the centre word t. V is the size of vocabulary in the corpus.

The output is a one-hot encoding of the centre word. W^i and W^o are updated via back propagation of errors. Therefore, only the value of the position that represents the centre word's probability, i.e., $p(t|c_1, ..., c_n, ..., c_m)$, will get close to the value of 1. In contrast, the probability of the rest of the words in the vocabulary will be close to 0 in every centre word training. W^i embeds context words. Vectors within W^i can be viewed as context word embeddings. W^o embeds centre words, vectors in W^o can be viewed as centre word embeddings.

Skip-gram is the reverse of CBOW (see Figure 1). The input and output layers are centre word and context word one-hot encodings, respectively. The target is to maximize the probability of predicting each context word, given a centre word:

$$\arg\max p(c_1, ..., c_n, ..., c_m|t) \tag{5}$$

Skip-gram's hidden layer is defined as:

$$H_{SG} = W^{i\top} \times T = v_t^{i\top} \tag{6}$$

where T is the one-hot encoding of the centre word t. Skip-gram's hidden layer is equal to the transpose of a centre word's input vector v_t, as only the tth row are kept by the operation. The probability of a context word is:

$$u_{c,n} = W_{c,n}^{o\top} \times H_{SG} = v_{c,n}^{o\top} \times H_{SG} \tag{7}$$

$$p(c_n|t) = \frac{\exp(u_{c,n})}{\sum_{j=1}^{V} \exp(u_j)} \tag{8}$$

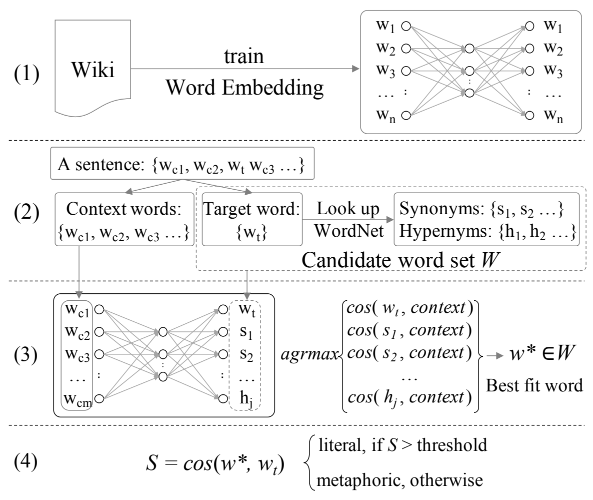


Figure 2: Metaphor identification framework. NB: w^* = best fit word, w_t = target word.

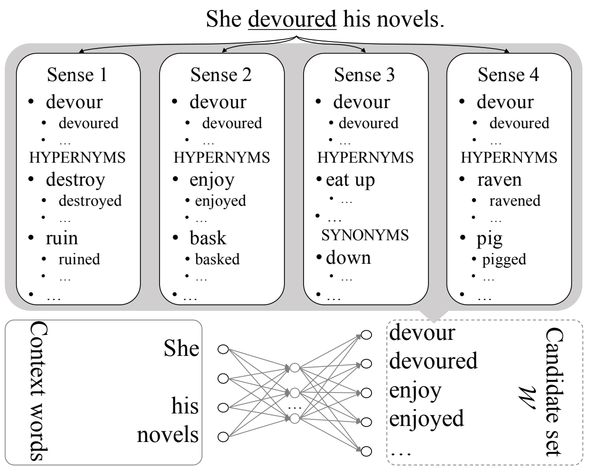


Figure 3: Given CBOW trained input and output vectors, a target word of *devoured*, and a context of *She* [ ] *his novels*, $cos(v^o_{devoured}, v^i_{context}) = -0.01$, $cos(v^o_{enjoyed}, v^i_{context}) = 0.02$.

where c, n is the nth context word, given a centre word. In Skip-gram, W^i aligns to centre words, while W^o aligns to context words. Because the names of centre word and context word embeddings are reversed in CBOW and Skip-gram, we will uniformly call vectors in W^i input vectors v^i, and vectors in W^o output vectors v^o in the remaining sections. Word embeddings represent both input and output vectors.

4 Methodology

In this section, we present the technical details of our metaphor processing framework, built upon two hypotheses. Our first hypothesis (**H1**) is that a metaphorical word can be identified, if the sense the word takes within its context and its literal sense come from different domains. Such a hypothesis is based on the theory of Selectional Preference Violation (Wilks, 1975, 1978) that a metaphorical item can be found in a violation of selectional restrictions, where a word does not satisfy its semantic constrains within a context. Our second hypothesis (**H2**) is that the literal senses of words occur more commonly in corpora than their metaphoric senses (Cameron, 2003; Martin, 2006; Steen et al., 2010; Shutova, 2016).

Figure 2 depicts an overview of our metaphor identification framework. The workflow of our framework is as follows. Step (1) involves training word embeddings based on a Wikipedia dump[3] for obtaining input and output vectors of words. In Step (2), given an input sentence, the target word (i.e., the word in the original text whose metaphoricity is to be determined) and its context words (i.e., all other words in the sentence excluding the target word) are separated. We construct a candidate word set $\mathcal{W}$ which represents all the possible senses of the target word. This is achieved by first extracting the synonyms and direct hypernyms of the target word from WordNet, and then augmenting the set with the inflections of the extracted synonyms and hypernyms, as well as the target word and its inflections. Auxiliary verbs are excluded from this set, as these words frequently appear in most sentences with little lexical meaning. In Step (3), we identify the best fit word, which is defined as the word that represents the literal sense that the target word is most likely taking given its context. Finally, in Step (4), we compute the cosine similarity between the target word and the best fit word. If the similarity is above a threshold, the target word will be identified as literal, otherwise metaphoric (i.e., based on **H1**). We will discuss in detail Step (3) and Step (4) in §4.1.

4.1 Metaphor identification

Step (3): One of the key steps of our metaphor identification framework is to identify the best fit word for a target word given its surrounding context. The intuition is that the best fit word will represent the literal sense that the target word is most likely taking. E.g., for the sentence *"She devoured his novels."* and the corresponding target word *devoured*, the best fit word is *enjoyed*, as shown in

[3]https://dumps.wikimedia.org/enwiki/20170920/

Figure 3. Also note that the best fit word could be the target word itself if the target word is used literally.

Given a sentence s, let w_t be the target word of the sentence, $w^* \in \mathcal{W}$ the best fit word for w_t, and $w_{context}$ the surrounding context for w_t, i.e., all the words in s excluding w_t. We compute the context embedding $v^i_{context}$ by averaging out the input vectors of each context word of $w_{context}$, based on Eq. 2. Next, we rank each candidate word $k \in \mathcal{W}$ by measuring its similarity to the context input vector $v^i_{context}$ in the vector space. The candidate word with the highest similarity to the context is then selected as the best fit word.

$$w^* = \arg\max_k \text{SIM}(v_k, v_{context}) \qquad (9)$$

where v_k is the vector of a candidate word $k \in \mathcal{W}$. In contrast to existing word embedding based methods for metaphor identification which only make use of input vectors (Shutova et al., 2016; Rei et al., 2017), we explore using both input and output vectors of CBOW and Skip-gram embeddings when measuring the similarity between a candidate word and the context. We expect that using a combination of input and output vectors might work better. Specifically, we have experimented with four different model variants as shown below.

$$\text{SIM-CBOW}_I = \cos(v^i_{k,cbow}, v^i_{context,cbow}) \qquad (10)$$

$$\text{SIM-CBOW}_{I+O} = \cos(v^o_{k,cbow}, v^i_{context,cbow}) \qquad (11)$$

$$\text{SIM-SG}_I = \cos(v^i_{k,sg}, v^i_{context,sg}) \qquad (12)$$

$$\text{SIM-SG}_{I+O} = \cos(v^o_{k,sg}, v^i_{context,sg}) \qquad (13)$$

Here, $\cos(\cdot)$ is cosine similarity, $cbow$ is CBOW word embeddings, sg is Skip-gram word embeddings. We have also tried other model variants using output vectors for $v_{context}$. However, we found that the models using output vectors for $v_{context}$ (both CBOW and Skip-gram embeddings) do not improve our framework performance. Due to the page limit we omitted the results of those models in this paper.

Step (4): Given a predicted best fit word w^* identified in Step (3), we then compute the cosine similarity between the lemmatizations of w^* and the target word w_t using their input vectors.

$$\text{SIM}(w^*, w_t) = \cos(v^i_{w^*}, v^i_{w_t}) \qquad (14)$$

We give a detailed discussion in §4.2 of our rationale for using input vectors for Eq. 14.

If the similarity is higher than a threshold (τ) the target word is considered as literal, otherwise, metaphorical (based on **H1**). One benefit of our approach is that it allows one to paraphrase the identified metaphorical target word into the best fit word, representing its literal sense in the context. Such a feature is useful for supporting other NLP tasks such as Machine Translation, which we will explore in §6. In terms of the value of threshold (τ), it is empirically determined based on a development set. Please refer to §5 for details.

To better explain the workflow of our framework, we now go through an example as illustrated in Figure 3. The target word of the input sentence, *"She devoured his novels."* is *devoured*, and its the lemmatised form *devour* has four verbal senses in WordNet, i.e., *destroy completely*, *enjoy avidly*, *eat up completely with great appetite*, and *eat greedily*. Each of these senses has a set of corresponding synonyms and hypernyms. E.g., Sense 3 (*eat up completely with great appetite*) has synonyms *demolish*, *down*, *consume*, and hypernyms *go through*, *eat up*, *finish*, and *polish off*. We then construct a candidate word set $\mathcal{W}$ by including the synonyms and direct hypernyms of the target word from WordNet, and then augmenting the set with the inflections of the extracted synonyms and hypernyms, as well as the target word *devour* and its inflections. We then identify the best fit word given the context *she* [ ] *his novels* based on Eq. 9. Based on **H2**, literal expressions are more common than metaphoric ones in corpora. Therefore, the best fit word is expected to frequently appear within the given context, and thus represents the most likely sense of the target word. For example, the similarity between *enjoy* (i.e., the best fit word) and the the context is higher than that of *devour* (i.e., the target word), as shown in Figure 3.

4.2 Word embedding: output vectors vs. input vectors

Typically, input vectors are used after training CBOW and Skip-gram, with output vectors being abandoned by practical models, e.g., original word2vec model (Mikolov et al., 2013) and Gensim toolkit (Řehůřek and Sojka, 2010), as these models are designed for modelling similarities in semantics. However, we found that using input vectors to measure cosine similarity between two words with different POS types in a phrase is sub-

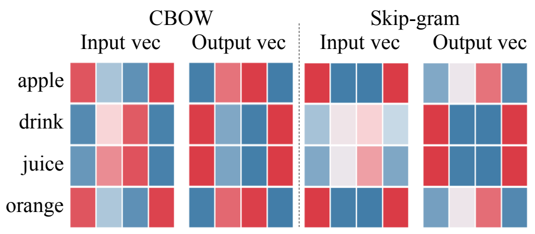


Figure 4: Input and output vector visualization. The bluer, the more negative. The redder, the more positive.

optimal, as words with different POS normally have different semantics. They tend to be distant from each other in the input vector space. Taking Skip-gram for example, empirically, input vectors of words with the same POS, occurring within the same contexts tend to be close in the vector space (Mikolov et al., 2013), as they are frequently updated by back propagating the errors from the same context words. In contrast, input vectors of words with different POS, playing different semantic and syntactic roles tend to be distant from each other, as they seldom occur within the same contexts, resulting in their input vectors rarely being updated equally. Our observation is also in line with Nalisnick et al. (2016), who examine IN-IN, OUT-OUT and IN-OUT vectors to measure similarity between two words. Nalisnick et al. discovered that two words which are similar by function or type have higher cosine similarity with IN-IN or OUT-OUT vectors, while using input and output vectors for two words (IN-OUT) that frequently co-occur in the same context (e.g., a sentence) can obtain a higher similarity score.

For illustrative purpose, we visualize the CBOW and Skip-gram updates between 4-dimensional input and output vectors by Wevi[4] (Rong, 2014), using a two-sentence corpus, *"Drink apple juice."* and *"Drink orange juice."*. We feed these two sentences to CBOW and Skip-gram with 500 iterations. As seen Figure 4, the input vectors of *apple* and *orange* are similar in both CBOW and Skip-gram, which are different from the input vectors of their context words (*drink* and *juice*). However, the output vectors of *apple* and *orange* are similar to the input vectors of *drink* and *juice*.

To summarise, using input vectors to compare similarity between the best fit word and the target word is more appropriate (cf. Eq.14), as they tend to have the same types of POS. When measuring the similarity between candidate words and the context, using output vectors for the former and input vectors for the latter seems to better predict the best fit word.

5 Experimental settings

Baselines. We compare the performance of our framework for metaphor identification against three strong baselines, namely, an unsupervised word embedding based model by Shutova et al. (2016), a supervised deep learning model by Rei et al. (2017), and the Context2Vec model[5] (Melamud et al., 2016) which achieves the best performance on Microsoft Sentence Completion Challenge (Zweig and Burges, 2011). Context2Vec was not designed for processing metaphors, in order to use it for this we plug it into a very similar framework to that described in Figure 2. We use Context2Vec to predict the best fit word from the candidate set, as it similarly uses context to predict the most likely centre word but with bidirectional LSTM based context embedding method. After locating the best fit word with Context2Vec, we identify the metaphoricity of a target word with the same method (see Step (4) in §4), so that we can also apply it for metaphor interpretation. Note that while Shutova et al. and Rei et al. detect metaphors at the phrase level by identifying metaphorical phrases, Melamud et al.'s model can perform metaphor identification and interpretation on sentences.

Dataset. Evaluation was conducted based on a dataset developed by Mohammad et al. (2016). This dataset[6], containing 1,230 literal and 409 metaphor sentences, has been widely used for metaphor identification related research (Shutova et al., 2016; Rei et al., 2017). There is a verbal target word annotated by 10 annotators in each sentence. We use two subsets of the Mohammad et al. set, one for phrase evaluation and one for sentence evaluation. The phrase evaluation dataset was kindly provided by Shutova, which consists of 316 metaphorical and 331 literal phrases (subject-verb and verb-direct object word pairs), parsed from Mohammad et al.'s dataset. Similar to Shutova et al. (2016), we use 40 metaphoric and 40 literal phrases as a development set and the rest as a test

[4]https://ronxin.github.io/wevi/

[5]http://u.cs.biu.ac.il/~nlp/resources/downloads/context2vec/

[6]http://saifmohammad.com/WebPages/metaphor.html

	Method	P	R	F1
Phrase	Shutova et al. (2016)	0.67	0.76	0.71
	Rei et al. (2017)	**0.74**	0.76	**0.74**
	SIM-CBOW$_{I+O}$	0.66	0.78	0.72
	SIM-SG$_{I+O}$	0.68	**0.82**	**0.74***
Sent.	Melamud et al. (2016)	0.60	0.80	0.69
	SIM-SG$_I$	0.56	**0.95**	0.70
	SIM-SG$_{I+O}$	0.62	0.89	0.73
	SIM-CBOW$_I$	0.59	0.91	0.72
	SIM-CBOW$_{I+O}$	**0.66**	0.88	**0.75***

Table 1: Metaphor identification results. NB: * denotes that our model outperforms the baseline significantly, based on two-tailed paired t-test with $p < 0.001$.

set.

For sentence evaluation, we select 212 metaphorical sentences whose target words are annotated with at least 70% agreement. We also add 212 literal sentences with the highest agreement. Among the 424 sentences, we form our development set with 12 randomly selected metaphoric and 12 literal instances to identify the threshold for detecting metaphors. The remaining 400 sentences are our testing set.

Word embedding training. We train CBOW and Skip-gram models on a Wikipedia dump with the same settings as Shutova et al. (2016) and Rei et al. (2017). That is, CBOW and Skip-gram models are trained iteratively 3 times on Wikipedia with a context window of 5 to learn 100-dimensional word input and output vectors. We exclude words with total frequency less than 100. 10 negative samples are randomly selected for each centre word training. The word down-sampling rate is 10^{-5}. We use Stanford CoreNLP (Manning et al., 2014) lemmatized Wikipedia to train word embeddings for phrase level evaluation, which is in line with Shutova et al. (2016). In sentence evaluation, we use the original Wikipedia for training word embeddings.

6 Experimental Results

6.1 Metaphor identification

Table 1 shows the performance of our model and the baselines on the task of metaphor identification. All the results for our models are based on a threshold of 0.6, which is empirically determined based on the developing set. For sentence level metaphor identification, it can be observed that all our models outperform the baseline (Melamud et al., 2016), with SIM-CBOW$_{I+O}$ giving the highest F1 score of 75% which is a 6% gain over the baseline. We also see that models based on both input and output vectors (i.e., SIM-CBOW$_{I+O}$ and SIM-SG$_{I+O}$) yield better performance than the models based on input vectors only (i.e., SIM-CBOW$_I$ and SIM-SG$_I$). Such an observation supports our assumption that using input and output vectors can better model similarity between words that have different types of POS, than simply using input vectors. When comparing CBOW and Skip-gram based models, we see that CBOW based models generally achieve better performance in precision whereas Skip-gram based models perform better in recall.

In terms of phrase level metaphor identification, we compare our best performing models (i.e., SIM-CBOW$_{I+O}$ and SIM-SG$_{I+O}$) against the approaches of Shutova et al. (2016) and Rei et al. (2017). In contrast to the sentence level evaluation in which SIM-CBOW$_{I+O}$ gives the best performance, SIM-SG$_{I+O}$ performs best for the phrase level evaluation. This is likely due to the fact that Skip-gram is trained by using a centre word to maximise the probability of each context word, whereas CBOW uses the average of context word input vectors to maximise the probability of the centre word. Thus, Skip-gram performs better in modelling one-word context, while CBOW has better performance in modelling multi-context words. When comparing to the baselines, our model SIM-SG$_{I+O}$ significantly outperforms the word embedding based approach by Shutova et al. (2016), and gives the same performance as the deep supervised method (Rei et al., 2017) which requires a large amount of labelled data for training and cost in training time.

SIM-CBOW$_{I+O}$ and SIM-SG$_{I+O}$ are also evaluated with different thresholds for both phrase and sentence level metaphor identification. As can be seen from Table 2, the results are fairly stable when the threshold is set between 0.5 and 0.9 in terms of F1.

6.2 Metaphor processing for MT

We believe that one of the key purposes of metaphor processing is to support other NLP tasks. Therefore, we conducted another set of experiments to evaluate how metaphor processing can be used to support English-Chinese machine translation.

The evaluation task was designed as follows. From the test set for sentence-level metaphor identification which contains 200 metaphoric and

τ	Sentence			Phrase	
	P	**R**	**F1**	**F1**$_{\text{SIM-CBOW}_{I+O}}$	**F1**$_{\text{SIM-SG}_{I+O}}$
0.3	0.75	0.60	0.67	0.56	0.46
0.4	0.69	0.75	0.72	0.65	0.63
0.5	0.67	0.82	0.74	0.71	0.72
0.6	0.66	0.88	**0.75**	**0.72**	**0.74**
0.7	0.64	0.88	0.74	0.72	0.73
0.8	0.63	0.89	0.74	0.72	0.73
0.9	0.63	0.89	0.74	0.71	0.73
1.0	0.50	1.00	0.67	0.65	0.65

Table 2: Model performance vs. different threshold (τ) settings. NB: the sentence level results are based on SIM-CBOW$_{I+O}$.

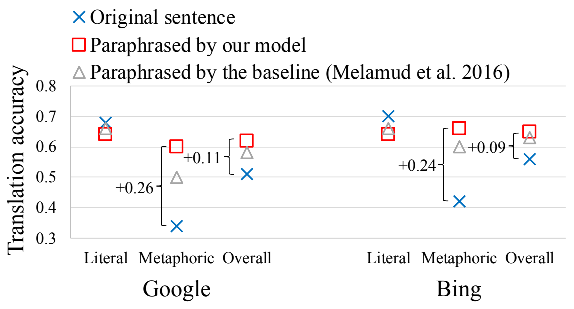
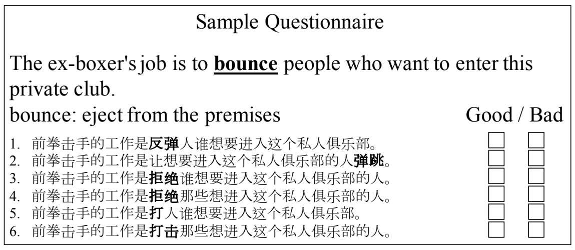


Figure 5: Accuracy of metaphor interpretation, evaluated on Google and Bing Translation.

200 literal sentences, we randomly selected 50 metaphoric and 50 literal sentences to construct a set $\mathcal{S}_{\mathcal{M}}$ for the Machine Translation (MT) evaluation task. For each sentence in $\mathcal{S}_{\mathcal{M}}$, if it is predicted as literal by our model, the sentence is kept unchanged; otherwise, the target word of the sentence is paraphrased with the best fit word (refer to §4.1 for details). The metaphor identification step resulted in 42 True Positive (TP) instances where the ground truth label is metaphoric and 19 False Positive (FP) instances where the ground truth label is literal, resulting in a total of 61 instances predicted as metaphorical by our model. We also run one of our baseline models, Context2Vec, on the 61 sentences to predict the best fit words for comparison. Our hypothesis is that by paraphrasing the metaphorically used target word with the best fit word which expresses the target word's real meaning, the performance of translation engines can be improved.

We test our hypothesis on two popular English-Chinese MT systems, i.e., the Google and Bing Translators. We recruited from a UK university 5 Computing Science postgraduate students who are Chinese native speakers to participate the English-Chinese MT evaluation task. During the evaluation, subjects were presented with a questionnaire


Sample Questionnaire

The ex-boxer's job is to **bounce** people who want to enter this private club.

bounce: eject from the premises

	Good	Bad
1. 前拳击手的工作是**反弹**人谁想要进入这个私人俱乐部。	☐	☐
2. 前拳击手的工作是让想要进入这个私人俱乐部的人**弹跳**。	☐	☐
3. 前拳击手的工作是**拒绝**谁想要进入这个私人俱乐部的人。	☐	☐
4. 前拳击手的工作是**拒绝**那些想进入这个私人俱乐部的人。	☐	☐
5. 前拳击手的工作是**打**人谁想要进入这个私人俱乐部。	☐	☐
6. 前拳击手的工作是**打击**那些想进入这个私人俱乐部的人。	☐	☐

Figure 6: MT-based metaphor interpretation questionnaire.

		Acc-met.	**Acc-lit.**	**Acc-overall**
Google	Orig. Sent.	0.34	**0.68**	0.51
	Context2Vec	0.50	0.66	0.58
	SIM-CBOW$_{I+O}$	**0.60**	0.64	**0.62**
Bing	Orig. Sent.	0.42	**0.70**	0.56
	Context2Vec	0.60	0.66	0.63
	SIM-CBOW$_{I+O}$	**0.66**	0.64	**0.65**

Table 3: Accuracy of metaphor interpretation, evaluated on Google and Bing Translation.

containing English-Chinese translations of each of the 100 randomly selected sentences. For each sentence predicted as literal (39 out of 100 sentences), there are two corresponding translations by Google and Bing respectively. For each sentence predicted as metaphoric (61 out of 100 sentences), there are 6 corresponding translations.

An example of the evaluation task is shown in Figure 6, in which *"The ex-boxer's job is to bounce people who want to enter this private club."* is the original sentence, followed by an WordNet explanation of the target word of the sentence (i.e., bounce: eject from the premises). There are 6 translations. No. 1-2 are the original sentence translations, translated by Google Translate (GT) and Bing Translator (BT). The target word, *bounce*, is translated, taking the sense of (1) *physically rebounding like a ball* (反弹), (2) *jumping* (弹跳). No. 3-4 are SIM-CBOW$_{I+O}$ paraphrased sentences, translated by GT and BT, respectively, taking the sense of *refusing* (拒绝). No. 5-6 are Context2Vec paraphrased sentences, translated by GT and BT, respectively, taking the sense of *hitting* (5.打; 6.打击).

Subjects were instructed to determine if the translation of a target word can correctly represent its sense within the translated sentence, matching its context (cohesion) in Chinese. Note that we evaluate the translation of the target word, therefore, errors in context word translations are ignored by the subjects. Finally, a label is taken agreed by more than half annotators. Noticeably, based on our observation, there is always a Chinese word corresponding to an English target word in MT, as the annotated target word normally represents important information in the sentence in the applied dataset.

We use translation accuracy as a measure to evaluate the improvement on MT systems after metaphor processing. The accuracy is calculated by dividing the number of correctly translated instances by the total number of instances. As can be seen in Figure 5 and Table 3, after paraphrasing the metaphorical sentences with the SIM-CBOW$_{I+O}$ model, the translation improvement for the metaphorical class is dramatic for both MT systems, i.e., 26% improvement for Google Translate and 24% for Bing Translate. In terms of the literal class, there is some small drop (i.e., 4-6%) in accuracy. This is due to the fact that some literals were wrongly identified as metaphors and hence error was introduced during paraphrasing. Nevertheless, with our model, the overall translation performance of both Google and Bing Translate are significantly improved by 11% and 9%, respectively. Our baseline model Context2Vec also improves the translation accuracy, but is 2-4 % lower than our model in terms of overall accuracy. In summary, the experimental results show the effectiveness of applying metaphor processing for supporting Machine Translation.

7 Conclusion

We proposed a framework that identifies and interprets metaphors at word-level with an unsupervised learning approach. Our model outperforms the unsupervised baselines in both sentence and phrase evaluations. The interpretation of the identified metaphorical words given by our model also contributes to Google and Bing translation systems with 11% and 9% accuracy improvements.

The experiments show that using words' hypernyms and synonyms in WordNet can paraphrase metaphors into their literal counterparts, so that the metaphors can be correctly identified and translated. To our knowledge, this is the first study that evaluates a metaphor processing method on Machine Translation. We believe that compared with simply identifying metaphors, metaphor processing applied in practical tasks, can be more valuable in the real world. Additionally, our experiments demonstrate that using a candidate word output vector instead of its input vector to model the similarity between the candidate word and its context yields better results in the best fit word (the literal counterpart of the metaphor) identification.

CBOW and Skip-gram do not consider the distance between a context word and a centre word in a sentence, i.e., context word contributes to predict the centre word equally. Future work will introduce weighted CBOW and Skip-gram to learn positional information within sentences.

Acknowledgments

This work is supported by the award made by the UK Engineering and Physical Sciences Research Council (Grant number: EP/P005810/1).

References

Dan Assaf, Yair Neuman, Yohai Cohen, Shlomo Argamon, Newton Howard, Mark Last, Ophir Frieder, and Moshe Koppel. 2013. Why "dark thoughts" aren't really dark: A novel algorithm for metaphor identification. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2013 IEEE Symposium on*. IEEE, pages 60–65.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Lynne Cameron. 2003. *Metaphor in educational discourse*. A&C Black.

Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology* 33(4):497–505.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Ilana Heintz, Ryan Gabbard, Mahesh Srinivasan, David Barner, Donald S Black, Marjorie Freedman, and Ralph Weischedel. 2013. Automatic extraction of linguistic metaphor with LDA topic modeling. In *Proceedings of the First Workshop on Metaphor in NLP (ACL 2013)*. pages 58–66.

Luuk Lagerwerf and Anoe Meijers. 2008. Openness in metaphorical and straightforward advertisements: Appreciation effects. *Journal of Advertising* 37(2):19–30.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 55–60.

James H Martin. 2006. A corpus-based analysis of context effects on metaphor comprehension. Technical Report CU-CS-738-94, Boulder: University of Colorado: Computer Science Department.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*. pages 51–61.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of International Conference on Learning Representations (ICLR 2013)* .

Saif M Mohammad, Ekaterina Shutova, and Peter D Turney. 2016. Metaphor as a medium for emotion: An empirical study. *Proceedings of the Joint Conference on Lexical and Computational Semantics (*SEM 2016)* page 23.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 83–84.

Yair Neuman, Dan Assaf, Yohai Cohen, Mark Last, Shlomo Argamon, Newton Howard, and Ophir Frieder. 2013. Metaphor identification in large texts corpora. *PloS one* 8(4):e62343.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. http://is.muni.cz/publication/884893/en.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)* pages 1537–1546.

Vassiliki Rentoumi, George A Vouros, Vangelis Karkaletsis, and Amalia Moser. 2012. Investigating metaphorical language in sentiment analysis: A sense-to-sentiment perspective. *ACM Transactions on Speech and Language Processing (TSLP)* 9(3):6.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* .

Ekaterina Shutova. 2016. Design and evaluation of metaphor processing systems. *Computational Linguistics* .

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)* pages 160–170.

Ekaterina Shutova, Lin Sun, Elkin Darío Gutiérrez, Patricia Lichtenstein, and Srini Narayanan. 2017. Multilingual metaphor processing: Experiments with semi-supervised and unsupervised learning. *Computational Linguistics* 43(1):71–123.

Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

Tomek Strzalkowski, George Aaron Broadwell, Sarah Taylor, Laurie Feldman, Samira Shaikh, Ting Liu, Boris Yamrom, Kit Cho, Umit Boz, Ignacio Cases, et al. 2013. Robust extraction of metaphor from novel data. In *Proceedings of the First Workshop on Metaphor in NLP (ACL 2013)*. pages 67–76.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)* pages 248–258.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)* pages 680–690.

Yorick Wilks. 1975. A preferential, pattern-seeking, semantics for natural language inference. *Artificial Intelligence* 6(1):53–74.

Yorick Wilks. 1978. Making preferences more active. *Artificial Intelligence* 11(3):197–223.

Geoffrey Zweig and Christopher JC Burges. 2011. The Microsoft research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft.

Incorporating Latent Meanings of Morphological Compositions to Enhance Word Embeddings

Yang Xu[†], **Jiawei Liu**[†], **Wei Yang**[‡*], and **Liusheng Huang**[‡]
School of Computer Science and Technology,
University of Science and Technology of China, Hefei, 230027, China
[†]{smallant, ustcljw}@mail.ustc.edu.cn
[‡]{qubit, lshuang}@ustc.edu.cn

Abstract

Traditional word embedding approaches learn semantic information at word level while ignoring the meaningful internal structures of words like morphemes. Furthermore, existing morphology-based models directly incorporate morphemes to train word embeddings, but still neglect the latent meanings of morphemes. In this paper, we explore to employ the latent meanings of morphological compositions of words to train and enhance word embeddings. Based on this purpose, we propose three Latent Meaning Models (LMMs), named LMM-A, LMM-S and LMM-M respectively, which adopt different strategies to incorporate the latent meanings of morphemes during the training process. Experiments on word similarity, syntactic analogy and text classification are conducted to validate the feasibility of our models. The results demonstrate that our models outperform the baselines on five word similarity datasets. On Wordsim-353 and RG-65 datasets, our models nearly achieve 5% and 7% gains over the classic CBOW model, respectively. For the syntactic analogy and text classification tasks, our models also surpass all the baselines including a morphology-based model.

1 Introduction

Word embedding, which is also termed distributed word representation, has been a hot topic in the area of Natural Language Processing (NLP). The derived word embeddings have been used in plenty of tasks such as text classification (Liu et al., 2015), information retrieval (Manning et al., 2008), sentiment analysis (Shin et al., 2016), machine translation (Cho et al., 2014) and so on. Recently, some classic word embedding methods have been proposed, like Continuous Bag-of-Word (CBOW), Skip-gram (Mikolov et al., 2013a), Global Vectors (GloVe) (Pennington et al., 2014). These methods can usually capture word-level semantic information but ignore the meaningful inner structures of words like English morphemes or Chinese characters.

The effectiveness of exploiting the internal compositions of words has been validated by some previous work (Luong et al., 2013; Botha and Blunsom, 2014; Chen et al., 2015; Cotterell et al., 2016). Some of them compute the word embeddings by directly adding the representations of morphemes/characters to context words or optimizing a joint objective over distributional statistics and morphological properties (Qiu et al., 2014; Botha and Blunsom, 2014; Chen et al., 2015; Luong et al., 2013; Lazaridou et al., 2013), while others introduce some probabilistic graphical models to build relationship between words and their internal compositions. *e.g.*, Bhatia et al. (2016) treat word embeddings as latent variables for a prior distribution, which reflects words' morphological properties, and feed the latent variables into a neural sequence model to obtain final word embeddings. Cotterell et al. (2016) construct a Gaussian graphical model that binds the morphological analysis to pre-trained word embeddings, which can help to smooth the noisy embeddings. Besides, these two methods also have the ability to predict embeddings for unseen words.

Different from all the above models (we regard them as *Explicit* models in Fig. 1) where internal compositions are directly used to encode morphological regularities into words and the

[*]This is the corresponding author.

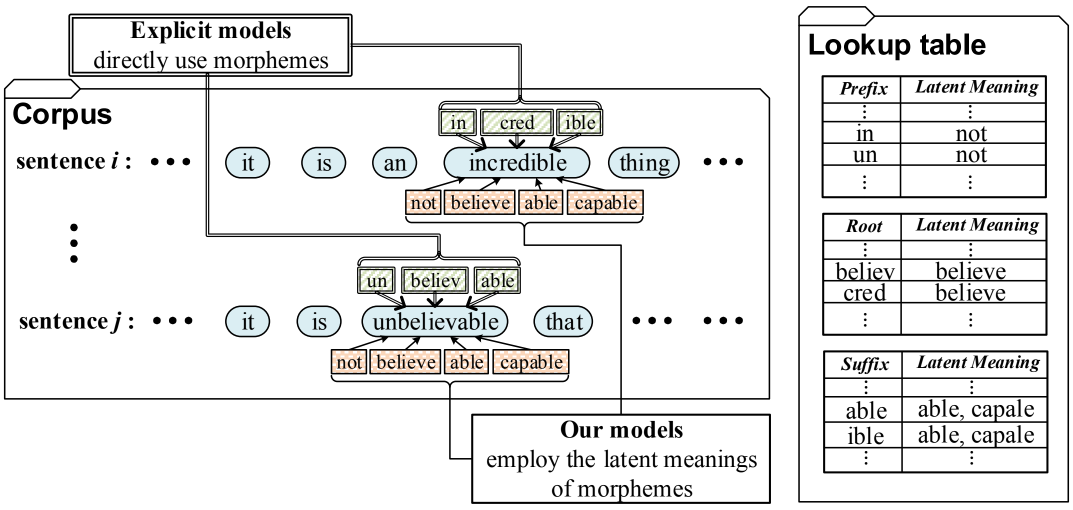


Figure 1: An illustration of explicit models and our models in an English corpus. Although *incredible* and *unbelievable* have different morphemes, their morphemes have the same latent meanings.

composition embeddings like morpheme embeddings are generated as by-products, we explore a new way to employ the latent meanings of morphological compositions rather than the compositions themselves to train word embeddings. As shown in Fig. 1, according to the distributional semantics hypothesis (Sahlgren, 2008), *incredible* and *unbelievable* probably have similar word embeddings because they have similar context. As a matter of fact, *incredible* is a synonym of *unbelievable* and their embeddings are expected to be close enough. Since the morphemes of the two words are different, especially the roots *cred* and *believ*, the explicit models may not significantly shorten the distance between the words in the vector space. Fortunately, the latent meanings of the different morphemes are the same (*e.g.*, the latent meanings of roots *cred*, *believ* are "*believe*") as listed in the lookup table (derived from the resources provided by Michigan State University),[1] which evidently implies that *incredible* and *unbelievable* share the same meanings. In addition, by replacing morphemes with their latent meanings, we can directly and simply quantize the similarities between words and their sub-compositions with the same metrics used in most NLP tasks, *e.g.*, cosine similarity. Subsequently, the similarities are utilized to calculate the weights of latent meanings of morphemes for each word.

In this paper, we try different strategies to modify the input layer and update rules of a neural language model, *e.g.*, CBOW, Skip-gram, and propose three lightweight and efficient models, which are termed Latent Meaning Models (LMMs), to not only encode morphological properties into words but also enhance the semantic similarities among word embeddings. Usually, the vocabulary derived from the corpus contains vast majority or even all of the latent meanings. Rather than generating and training extra embeddings for latent meanings, we directly override the embeddings of the corresponding words in the vocabulary. Moreover, a word map is created to describe the relations between words and the latent meanings of their morphemes.

For comparison, our models together with the state-of-the-art baselines are tested on two basic NLP tasks, which are word similarity and syntactic analogy, and one downstream text classification task. The results show that LMMs outperform the baselines and get satisfactory improvement on these tasks. In all, the main contributions of this paper are summarized as follows.

- Rather than directly incorporating the morphological compositions (surface forms) of words, we decide to employ the latent meanings of the compositions (underlying forms) to train the word embeddings. To validate the feasibility of our purpose, three specific models, named LMMs, are proposed with different strategies to incorporate the latent meanings.

[1]`https://msu.edu/~defores1/gre/roots/gre_rts_afx1.htm`

- We utilize a medium-sized English corpus to train LMMs and the state-of-the-art baselines, and evaluate their performance on two basic NLP tasks, *i.e.*, word similarity and syntactic analogy, and one downstream text classification task. The results show that LMMs outperform the baselines on five word similarity datasets. On the golden standard Wordsim-353 and RG-65, LMMs approximately achieve 5% and 7% gains over CBOW, respectively. For the syntactic analogy and text classification tasks, LMMs also surpass all the baselines.

- We conduct experiments to analyze the impacts of parameter settings, and the results demonstrate that the performance of LMMs on the smallest corpus is similar to the performance of CBOW on the corpus that is five times as large, which convinces us that LMMs are of great advantages to enhance word embeddings compared with traditional methods.

2 Background and Related Work

Considering the high efficiency of CBOW proposed by Mikolov et al. (2013a), our LMMs are built upon CBOW. Here, we first review some backgrounds of CBOW, and then present some related work on recent word-level and morphology-based word embedding methods.

CBOW with Negative Sampling With a sliding window, CBOW utilizes the context words in the window to predict the target word. Given a sequence of tokens $T = \{t_1, t_2, \cdots, t_n\}$, where n is the size of a training corpus, the objective of CBOW is to maximize the following average log probability equation:

$$L = \frac{1}{n} \sum_{i=1}^{n} \log p\big(t_i | context(t_i)\big), \qquad (1)$$

where $context(t_i)$ represents the context words of t_i in the slide window, $p\big(t_i | context(t_i)\big)$ is derived by softmax. Due to huge size of English vocabulary, $p\big(t_i | context(t_i)\big)$ can not be calculated in a tolerable time. Therefore, negative sampling and hierarchical softmax are proposed to solve this problem. Owing to the efficiency of negative sampling, all our models are trained based on it. In terms of negative sampling, the log probability $\log p(t_O|t_I)$ is transformed as:

$$\begin{aligned} &\log \delta\big(vec'(t_O)^T vec(t_I)\big) + \\ &\sum_{i=1}^{m} \log \big[1 - \delta\big(vec'(t_i)^T vec(t_I)\big)\big], \end{aligned} \qquad (2)$$

where m denotes the number of negative samples, and $\delta(\cdot)$ is the sigmoid function. The first item of Eq. (2) is the probability of target word when its context is given. The second item indicates the probability that negative samples do not share the same context as the target word.

Word-level Word Embedding In general, word embedding models can mainly be divided into two branches. One is based on neural network like the classic CBOW model (Mikolov et al., 2013a), while the other is based on matrix factorization. Besides CBOW, Skip-gram (Mikolov et al., 2013a) is another widely used neural-network-based model, which predicts the context by using the target word (Mikolov et al., 2013a). As for matrix factorization, Dhillon et al. (2015) proposed a spectral word embedding method to measure the correlation between word information matrix and context information matrix. In order to combine the advantages of models based on neural network and matrix factorization, Pennington et al. (2014) proposed a famous word embedding model named GloVe, which is reported to outperform the CBOW and Skip-gram models on some tasks. These models are effective to capture word-level semantic information while neglecting inner structures of words. In contrast, the unheeded inner structures are utilized in both our LMMs and other morphology-based models.

Morphology-based Word Embedding Recently, some more fine-grained word embedding models are proposed by exploiting the morphological compositions of words, *e.g.*, root and affixes. These morphology-based models can be divided into two main categories.

The first category directly adds the representations of internal structures to word embeddings or optimizes a joint objective over distributional statistics and morphological properties (Luong et al., 2013; Qiu et al., 2014; Botha and Blunsom, 2014; Lazaridou et al., 2013; Chen et al., 2015; Kim et al., 2016; Cotterell and Schütze, 2015). Chen et al. (2015) proposed a character-enhanced Chinese word embedding model, which splits a Chinese word into several characters and add the characters into the input layer of their models.

Luong et al. (2013) utilized the morpheme segments produced by Morfessor (Creutz and Lagus, 2007) and constructed morpheme trees for words to learn morphologically-aware word embeddings by the recursive neural network. Kim et al. (2016) incorporated the convolutional character information into English words. Their model can learn character-level semantic information for embeddings, which is proved to be effective for some morpheme-rich languages. However, with a huge size architecture, it's very time-consuming. Cotterell et al. (2015) augmented the log linear model to make the words, which share similar morphemes, gather together in vector space.

The other category tries to use probabilistic graphical models to connect words with their morphological compositions, and further learns word embeddings (Bhatia et al., 2016; Cotterell et al., 2016). Bhatia et al. (2016) employed morphemes and made them as prior knowledge of the latent word embeddings, then fed the latent variables into a neural sequence model to obtain final word embeddings. Cotterell et al. (2016) proposed a morpheme-based post-processor for pre-trained word embeddings. They constructed a Gaussian graphical model which can extrapolate continuous representations for unknown words.

However, these morphology-based models directly exploit the internal compositions of words to encode morphological regularities into word embeddings, and some by-products are also produced like morpheme embeddings. In contrast, we employ the latent meanings of morphological compositions to provide deeper insights for training better word embeddings. Furthermore, since the latent meanings are included in the vocabulary, there is no extra embedding being generated.

3 Our Latent Meaning Models

We leverage different strategies to modify the input layer and update rules of CBOW when incorporating the latent meanings of morphemes. Three specific models, named Latent Meaning Model-Average (LMM-A), LMM-Similarity (LMM-S) and LMM-Max (LMM-M), are proposed. It should be stated that, for now, our models mainly concern the derivational morphemes, which can be interpreted to some meaningful words or phrases (*i.e.*, latent meanings), not the inflectional morphemes like tense, number, gender, *etc.*

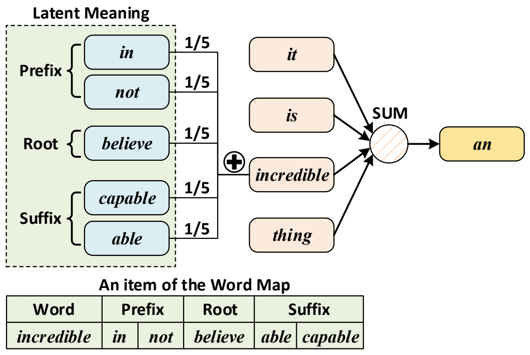
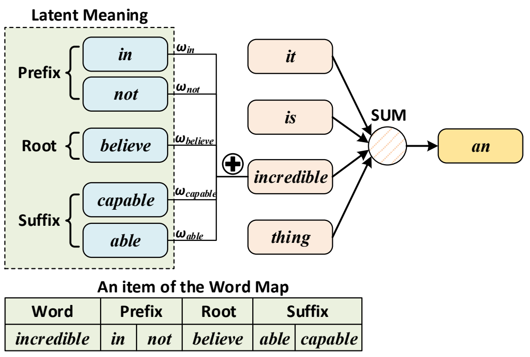


Figure 2: A paradigm of LMM-A. The sentence "*it is an incredible thing*" is selected as an example. When calculating the input vector of "*incredible*", we first find out the latent meanings of its morphemes in the word map, and add the vectors of all latent meanings to the vector of "*incredible*" with equal weights.

LMM-A assumes that all latent meanings of morphemes of a word have equal contributions to the word. LMM-A is applicable to the condition where words are correctly segmented into morphemes and each morpheme is interpreted to appropriate latent meanings. However, refining the latent meanings for morphemes is time-consuming and needs vast human annotations. To address this concern, LMM-S is proposed. Motivated by the attention scheme, LMM-S holds the assumption that all latent meanings have different contributions, and assigns the outliers small weights to let them have little impact on the representation of the target word. Furthermore, in LMM-M, we only keep the latent meanings which have the greatest contributions to the corresponding word. In what follows, we are going to introduce each of our LMMs in detail. At the end of this section, we will introduce the update rules of the models.

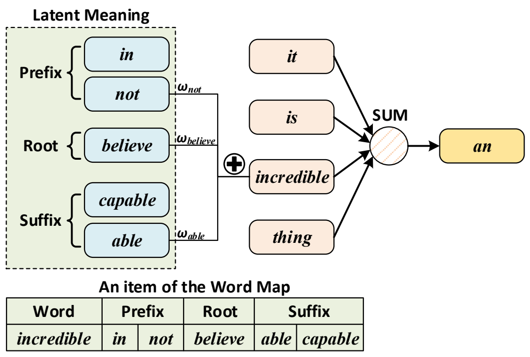
3.1 LMM-A

Given a sequence of tokens $T = \{t_1, t_2, \cdots, t_n\}$, LMM-A assumes that morphemes' latent meanings of token t_i ($i \in [1, n]$) have equal contributions to t_i, as shown in Fig. 2. The item for t_i in the word map is $t_i \mapsto M_i$. M_i is a set of latent meanings of t_i's morphemes, and it consists of three sub-parts P_i, R_i and S_i corresponding to the latent meanings of prefixes, roots and suffixes of t_i, respectively. Hence, at the input layer, the



Word	Prefix		Root	Suffix	
incredible	*in*	*not*	*believe*	*able*	*capable*

Figure 3: A paradigm of LMM-S. In this model, all latent meanings of morphemes of "*incredible*" are added together with different weights.



Word	Prefix		Root	Suffix	
incredible	*in*	*not*	*believe*	*able*	*capable*

Figure 4: A paradigm of LMM-M. The latent meanings with maximum similarities towards "*incredible*" are selected.

modified embedding of t_i can be expressed as

$$\widehat{v}_{t_i} = \frac{1}{2}\big(v_{t_i} + \frac{1}{N_i}\sum_{w \in M_i} v_w\big), \tag{3}$$

where v_{t_i} is the original word embedding of t_i, N_i denotes the length of M_i and v_w indicates the embedding of latent meaning w. Meanwhile, we assume the original word embedding and the average embeddings of v_w ($w \in M_i$) have equal weights, *i.e.*, 0.5. Eventually, $\widehat{v}_{t_i}$ rather than v_{t_i} is utilized for training in CBOW.

3.2 LMM-S

This model is proposed based on the attention scheme. We observe that many morphemes have more than one latent meaning. For instance, prefix *in-* means "*in*" and "*not*", and suffix *-ible* means "*able*" and "*capable*".[2] As Fig. 3 shows, for the item *incredible* $\mapsto \big\{[in, not]$, $[believe]$, $[able, capable]\big\}$ in the word map, the latent meanings have different biases towards "*incredible*". Therefore, we assign different weights to latent meanings. We measure the weights of latent meanings by calculating the normalized similarities between token t_i and the corresponding latent meanings. For LMM-S, the modified embedding of t_i can be rewritten as

$$\widehat{v}_{t_i} = \frac{1}{2}\big[v_{t_i} + \sum_{w \in M_i} \omega_{<t_i,w>} \cdot v_w\big], \tag{4}$$

where v_{t_i} is the original vector of t_i, and $\omega_{<t_i,w>}$ denotes the weight between t_i and the latent meaning w ($w \in M_i$). We use $cos(v_a, v_b)$ to denote the cosine similarity between v_a and v_b, then $\omega_{<t_i,w>}$ is expressed as follows:

$$\omega_{<t_i,w>} = \frac{cos(v_{t_i}, v_w)}{\sum\limits_{x \in M_i} cos(v_{t_i}, v_x)}. \tag{5}$$

3.3 LMM-M

To further eliminate the impacts of some uncorrelated latent meanings to a word, in LMM-M, we only select the latent meanings that have maximum similarities to the token t_i from P_i, R_i and S_i. As is shown in Fig. 4, the latent meaning "*not*" of prefix *in* is finally selected since the similarity between "*not*" and "*incredible*" is larger than that between "*in*" and "*incredible*". For token t_i, LMM-M is mathematically defined as

$$\widehat{v}_{t_i} = \frac{1}{2}\big[v_{t_i} + \sum_{w \in M^i_{max}} \omega_{<t_i,w>} \cdot v_w\big], \tag{6}$$

where $M^i_{max} = \{P^i_{max}, R^i_{max}, S^i_{max}\}$ is the set of latent meanings with maximum similarities towards token t_i, and P^i_{max}, R^i_{max}, S^i_{max} are obtained by the following equations:

$$\begin{aligned} P^i_{max} &= arg\max_{w} cos(v_{t_i}, v_w), w \in P_i, \\ R^i_{max} &= arg\max_{w} cos(v_{t_i}, v_w), w \in R_i, \\ S^i_{max} &= arg\max_{w} cos(v_{t_i}, v_w), w \in S_i. \end{aligned} \tag{7}$$

The normalized weight $\omega_{<t_i,w>}$ ($w \in M^i_{max}$) can similarly be derived like Eq. (5).

[2]All the latent meanings of roots and affixes are referred to the resources we mentioned before.

3.4 Update Rules for LMMs

After modifying the input layer of CBOW, Eq. (1) can be rewritten as

$$\widehat{L} = \frac{1}{n}\sum_{i=1}^{n}\log p\big(v_{t_i}|\sum_{t_j\in context(t_i)}\widehat{v}_{t_j}\big), \qquad (8)$$

where $\widehat{v}_{t_j}$ is the modified vector of v_{t_j} ($t_j \in context(t_i)$). Since the word map describes top-level relations between words and the latent meanings, these relations don't change during the training period. All parameters introduced by our models can be directly derived using the word map and word vectors, thus no extra parameter needs to be trained. When the gradient is propagated back to the input layer, we update not just the word vector v_{t_j} ($t_j \in context(t_i)$) but the vectors of the latent meanings in the vocabulary with the same weights as they are added to the vector v_{t_j}.

4 Experimental Setup

Before conducting experiments, some experimental settings are firstly introduced in this section.

4.1 Corpus and Word Map

We utilize a medium-sized English corpus to train all word embedding models. The corpus stems from the website of the 2013 ACL Workshop on Machine Translation[3] and is used in (Kim et al., 2016). We choose the news corpus of 2009 whose size is about 1.7GB. It contains approximately 500 million tokens and 600,000 words in the vocabulary. To get better quality of the word embeddings, we filter all digits and some punctuation marks out of the corpus.

For many languages, there exist large morphological lexicons or morphological tools that can analyze any word form (Cotterell and Schütze, 2015). To create the word map, we need to obtain the morphemes of each word and interpret them with the lookup table mentioned above to get the latent meanings. Usually, the lookup table can also be derived from the morphological lexicons for different languages, although it costs some time and manpower, we can create the lookup table once for all since it represents the common knowledge with respect to a certain language. Specifically, we first perform an unsupervised morpheme segmentation using Morefessor (Creutz and Lagus, 2007) for the vocabularies. Then we execute matching between the segmentation results and the morphological compositions in the lookup table, and the character sequence with largest overlap ratio will be viewed as a final morpheme and further be replaced by its latent meanings. Although the lookup table employed in this paper contains latent meanings for only 90 prefixes, 382 roots and 67 suffixes, we focus on validating the feasibility of enhancing word embeddings with the latent meanings of morphemes, and expending the lookup table is left as future work.

[3]http://www.statmt.org/wmt13/translation-task.html

4.2 Baselines

For comparison, we choose three word-level state-of-the-art word embedding models including CBOW, Skip-gram (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014), and we also implement an Explicitly Morpheme-related Model (EMM), which is a variant version of the previous work (Qiu et al., 2014). The architecture of EMM is based on our LMM-A, where latent meanings are replaced back to morphemes and the embeddings of morphemes are also learned when training word embeddings. This enables our evaluation to focus on the critical difference between our models and the explicit model (Bhatia et al., 2016). We utilize the source code of word2vec[4] to train CBOW and Skip-gram. GloVe is trained based on the code[5] provided by Pennington et al. (2014). We modify the source of word2vec and train our models and EMM.

4.3 Parameter Settings

Parameter settings have a great effect on the performance of word embeddings (Levy et al., 2015). For fairness, all models are trained based on equal parameter settings. In order to accelerate the training process, CBOW, Skip-gram and EMM together with our models are trained by using the negative sampling technique. It is suggested that the number of negative samples in the range 5-20 is useful for small corpus (Mikolov et al., 2013b). If large corpus is used, the number of negative samples can be as small as 2-5. According to the size of corpus we used, the number of negative samples is empirically set to be 20 in this paper.

[4]https://github.com/dav/word2vec

[5]http://nlp.stanford.edu/projects/glove

Name	Pairs	Name	Pairs
RG-65	65	RW	2034
SCWS	2003	Men-3k	3000
Wordsim-353	353	WS-353-REL	252

Table 1: Details of datasets. The column "Pairs" shows the number of word pairs in each dataset.

The dimension of word embedding is set as 200 like that in (Dhillon et al., 2015). We set the context window size as 5 which is equal to the setting in (Mikolov et al., 2013b).

4.4 Evaluation Benchmarks

4.4.1 Word Similarity

This experiment is conducted to evaluate the ability of word embeddings to capture semantic information from corpus. For English word similarity, we employ two gold standard datasets including Wordsim-353 (Finkelstein et al., 2001) and RG-65 (Rubenstein and Goodenough, 1965) as well as some other widely-used datasets including Rare-Word (Luong et al., 2013), SCWS (Huang et al., 2012), Men-3k (Bruni et al., 2014) and WS-353-Related (Agirre et al., 2009). More details of these datasets are shown in Table 1. Each dataset consists of three columns. The first two columns stand for word pairs and the last column is human score. We utilize the cosine similarity, which is used in many previous works (Mikolov et al., 2013b; Pennington et al., 2014), as the metric to measure the distance between two words. The Spearman's rank correlation coefficient (ρ) is employed to evaluate the similarity between our results and human scores. Higher ρ means better performance.

4.4.2 Syntactic Analogy

Based on the learned word embeddings, the core task of syntactic analogy is to answer the analogy question "a is to b as c is to _ ". We utilize the Microsoft Research Syntactic Analogies dataset, which is created by Mikolov (Mikolov et al., 2013c) with size of 8000. To answer the syntactic analogy question "a is to b as c is to d" where d is unknown, we assume that the word representations of a, b, c, d are v_a, v_b, v_c, v_d, respectively. To get d, we first calculate $\widehat{v}_d = v_b - v_a + v_c$. Then, we find out the word d' whose cosine similarity to $\widehat{v}_d$ is the largest. Finally, we set d as d'.

4.4.3 Text Classification

To further evaluate the learned word embeddings, we also conduct 4 text classification tasks using the 20 Newsgroups dataset.[6] The dataset totally contains around 19000 documents of 20 different newsgroups, and each corresponding to a different topic, such as guns, motorcycles, electronics and so on. For each task, we randomly select the documents of 10 topics and split them into training/validation/test subsets at the ratio of 6:2:2, which are emplyed to train, validate and test an L2-regularized 10-categorization logistic regression (LR) classifier. As mentioned in (Tsvetkov et al., 2015), here we also regard the average word embedding of words (excluding stop words and out-of-vocabulary words) in each document as the feature vector (the input of the classifier) of that document. The LR classifier is implemented with the scikit-learn toolkit (Pedregosa et al., 2011), which is an open-source Python module integrating many state-of-the-art machine learning algorithms.

5 Experimental Results

5.1 The Results on Word Similarity

Word similarity is conducted to test the semantic information which is encoded in word embeddings, and the results are listed in Table 2 (first 6 rows). We observe that our models surpass the comparative baselines on five datasets. Compared with the base model CBOW, it is remarkable that our models approximately achieve improvements of more than 5% and 7%, respectively, in the performance on the golden standard Wordsim-353 and RG-65. On WS-353-REL, the difference between CBOW and LMM-S even reaches 8%. The advantage demonstrates the effectiveness of our methods. Based on our strategy, more semantic information will be captured in corpus when adding more latent meanings in the context window. By incorporating mophemes, EMM also performs better than other baselines but fails to get the performance as well as ours. Actually, EMM mainly tunes the distributions of words in vector space to let the morpheme-similar words gather closer, which means it just encodes more morphological properties into word embeddings but lacks the ability to capture more semantic information. Specially, because of the medium-

[6] `http://qwone.com/~jason/20Newsgroups`

	CBOW	Skip-gram	GloVe	EMM	LMM-A	LMM-S	LMM-M
Wordsim-353	58.77	61.94	49.40	60.01	62.05	**63.13**	61.54
RW	40.58	36.42	33.40	40.83	**43.12**	42.14	40.51
RG-65	56.50	62.81	59.92	60.85	62.51	62.49	**63.07**
SCWS	**63.13**	60.20	47.98	60.28	61.86	61.71	63.02
Men-3k	68.07	66.30	60.56	66.76	66.26	**68.36**	64.65
WS-353-REL	49.72	57.05	47.46	54.48	56.14	**58.47**	55.19
Syntactic Analogy	13.46	13.14	13.94	17.34	**20.38**	17.59	18.30
Text Classification	78.26	79.40	77.01	80.00	80.67	80.59	**81.28**

Table 2: Performance comparison (%) of our LMMs and the baselines on two basic NLP tasks (word similarity & syntactic analogy) and one downstream task (text classification). The bold digits indicate the best performances.

size corpus and the experimental settings, GloVe doesn't perform as well as that described in (Pennington et al., 2014).

5.2 The Results on Syntactic Analogy

In (Mikolov et al., 2013c), the dataset is divided into adjectives, nouns and verbs. For brevity, we only report performance on the whole dataset. As the middle row of Table 2 shows, all of our models outperform the comparative baselines to a great extent. Compared with CBOW, the advantage of LMM-A even reaches to 7%. Besides, we observe that the suffix of "*b*" usually is the same as the suffix of "*d*" when answering question "*a* is to *b* as *c* is to *d*". Based on our strategy, morpheme-similar words will not only gather closer but have a trend to group near the latent meanings of their morphemes, which makes our embeddings have the advantage to deal with the syntactic analogy problem. EMM also performs well on this task but is still weaker than our models. Actually, syntactic analogy is also a semantics-related task because "*c*" and "*d*" are with similar meanings. Since our models are better to capture semantic information, they lead to higher performance than the explicitly morphology-based models.

5.3 The Results on Text Classification

For each one of the 4 text classification tasks, we report the classification accuracy over the test set. The average classification accuracy across the 4 tasks is utilized as the evaluation metric for different models. The results are displayed in the bottom row of Table 2. Since we simply use the average embedding of words as the feature vector for 10-categorization classification, the overall classification accuracies of all models are merely aroud 80%. However, the classification accuracies of our LMMs still surpass all the baselines, especailly CBOW and GloVe. Moreover, it can be found that incorporating morphological knowledge (morphemes or latent meanings of morphemes) into word embeddings can contribute to enhancing the performance of word embeddings in the downstream NLP tasks.

5.4 The Impacts of Parameter Settings

Parameter settings can affect the performance of word embeddings. For example, the corpus with larger corpus size (the ratio of tokens used for training) contains more semantic information, which can improve the performance on word similarity. We analyze the impacts of corpus size and window size on the performance of word embeddings. In the analysis of corpus size, we hold the same parameter settings as before. The sizes of tokens used for training are separately 1/5, 2/5, 3/5, 4/5 and 5/5 of the entire corpus mentioned above. We utilize the result of word similarity on Wordsim-353 as the evaluation criterion. From Fig. 5, we observe several phenomena. Firstly, the performance of our LMMs is better than CBOW at each corpus size. Secondly, the performance of CBOW is sensitive to the corpus size. In contrast, LMMs' performance is more stable than CBOW. As we analyzed in word similarity experiment, LMMs can increase the semantic information of word embeddings. It is worth noting that the performance of LMMs on the smallest corpus is even better than CBOW's performance on the largest corpus. In the analysis of window size, we observe that the performance of all word embeddings trained by different models has a trend to ascend with the increasing of window size as illustrated in Fig. 6. Our LMMs outperform CBOW under all the pre-set conditions. Besides, the worst performance of LMMs is nearly equal to the best performance of CBOW.

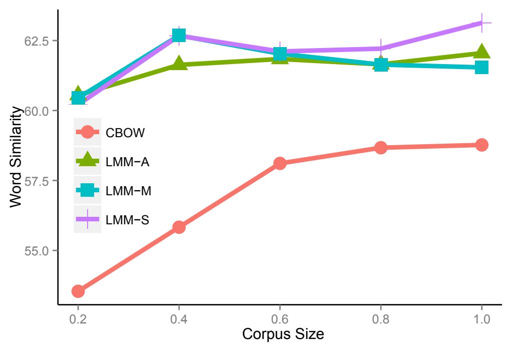


Figure 5: Parameter analysis of corpus size. X-axis denotes the ratio of tokens used for training, and Y-axis denotes the Spearman rank (%) of word similarity.

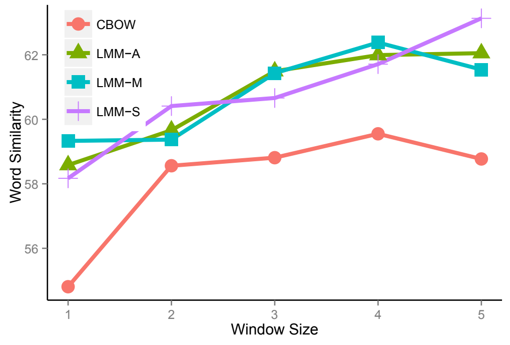


Figure 6: Parameter analysis of window size. X-axis and Y-axis denote the window size and Spearman rank (%) of word similarity, respectively.

5.5 Word Embedding Visualization

To visualize the embeddings of our models, we randomly select several words from the results of LMM-A. The dimensions of the selected word embeddings are reduced from 200 to 2 using Principal Component Analysis (PCA), and the 2-D word embeddings are illustrated in Fig. 7. The words with different colors reflect that they have different morphemes. It is apparent that words with similar morphemes have a trend to group together and stay near the latent meanings of their morphemes. In addition, we can also find some syntactic regularities in Fig. 7, for example, "*physics*" is to "*physicist*" as "*science*" is to "*scientist*", and "*physicist*" and "*scientist*" stay near the latent meaning, *i.e.*, "*human*", of the suffix *-ist*.

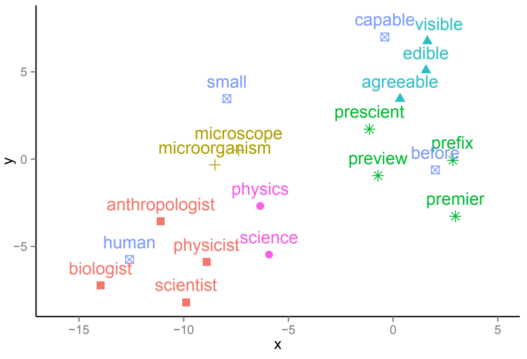


Figure 7: The visualization of word embeddings. Based on PCA, we randomly select several words from word embedding of LMM-A and illustrate them in this figure, "⊠" indicates the latent meanings of morphemes.

6 Conclusion

In this paper, we explored a new direction to employ the latent meanings of morphological compositions rather than the internal compositions themselves to train word embeddings. Three specific models named LMM-A, LMM-S and LMM-M were proposed by modifying the input layer and update rules of CBOW. The source code of LMMs is avaliable at `https://github.com/Y-Xu/lmm`.

To test the performance of our models, we chose three word-level word embedding models and implemented an Explicitly Morpheme-related Model (EMM) as comparative baselines, and tested them on two basic NLP tasks of word similarity and syntactic analogy, and one downstream text classification task. The experimental results demonstrate that our models outperform the baselines on five word similarity datasets. On the syntactic analogy as well as the text classification tasks, our models also surpass all the baselines including the EMM. In the future, we intend to evaluate our models for some morpheme-rich languages like Russian, German and so on.

Acknowledgments

The authors are grateful to the reviewers for constructive feedback. This work was supported by the National Natural Science Foundation of China (No.61572456), the Anhui Province Guidance Funds for Quantum Communication and Quantum Computers and the Natural Science Foundation of Jiangsu Province of China (No.BK20151241).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 490–500. Association for Computational Linguistics.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificail Intelligence Research (JAIR)*, 49(1–47).

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *International Conference on Artificial Intelligence*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1287–1292.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. 2015. Eigenwords: spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Meeting of the Association for Computational Linguistics: Long Papers*, pages 873–882.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *The Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1517–1526. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 141–150.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Magnus Sahlgren. 2008. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53.

Bonggun Shin, Timothy Lee, and Jinho D Choi. 2016. Lexicon integrated cnn models with attention for sentiment analysis. *arXiv preprint arXiv:1610.06272*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054.

A Stochastic Decoder for Neural Machine Translation[*]

Philip Schulz
Amazon Research[†]
phschulz@amazon.com

Wilker Aziz
University of Amsterdam
w.aziz@uva.nl

Trevor Cohn
University of Melbourne
trevor.cohn@unimelb.edu.au

Abstract

The process of translation is ambiguous, in that there are typically many valid translations for a given sentence. This gives rise to significant variation in parallel corpora, however, most current models of machine translation do not account for this variation, instead treating the problem as a deterministic process. To this end, we present a deep generative model of machine translation which incorporates a chain of latent variables, in order to account for local lexical and syntactic variation in parallel corpora. We provide an in-depth analysis of the pitfalls encountered in variational inference for training deep generative models. Experiments on several different language pairs demonstrate that the model consistently improves over strong baselines.

1 Introduction

Neural architectures have taken the field of machine translation by storm and are in the process of replacing phrase-based systems. Based on the encoder-decoder framework (Sutskever et al., 2014) increasingly complex neural systems are being developed at the moment. These systems find new ways of extracting information from the source sentence and the target sentence prefix for example by using convolutions (Gehring et al., 2017) or stacked self-attention layers (Vaswani et al., 2017). These architectural changes have led to great performance improvements over classical RNN-based neural translation systems (Bahdanau et al., 2014).

Surprisingly, there have been almost no efforts to change the probabilistic model wich is used to train the neural architectures. A notable exception is the work of Zhang et al. (2016) who introduce a sentence-level latent Gaussian variable.

In this work, we propose a more expressive latent variable model that extends the attention-based architecture of Bahdanau et al. (2014). Our model is motivated by the following observation: translations by professional translators vary across translators but also within a single translator (the same translator may produce different translations on different days, depending on his state of health, concentration etc.). Neural machine translation (NMT) models are incapable of capturing this variation, however. This is because their likelihood function incorporates the statistical assumption that there is one (and only one) output[1] for a given source sentence, i.e.,

$$P(y_1^n|x_1^m) = \prod_{i=1}^{n} P(y_i|x_1^m, y_{<i}) \ . \tag{1}$$

Our proposal is to augment this model with latent sources of variation that are able to represent more of the variation present in the training data. The noise sources are modelled as Gaussian random variables.

The contributions of this work are:

- The introduction of an NMT system that is capable of capturing word-level variation in translation data.
- A thorough discussions of issues encountered when training this model. In particular, we motivate the use of KL scaling as introduced by Bowman et al. (2016) theoretically.

[*] Code and a workflow that reproduces the experiments are available at https://github.com/philschulz/stochastic-decoder.

[†] Work done prior to joining Amazon.

[1] Notice that from a statistical perspective the output of an NMT system is a distribution over target sentences and not any particular sentence. The mapping from the output distribution to a sentence is performed by a decision rule (e.g. argmax decoding) which can be chosen independently of the NMT system.

- An empirical demonstration of the improvements achievable with the proposed model.

2 Neural Machine Translation

The NMT system upon which we base our experiments is based on the work of Bahdanau et al. (2014). The likelihood of the model is given in Equation (1). We briefly describe its architecture.

Let $x_1^m = (x_1, \ldots, x_m)$ be the source sentence and y_1^n the target sentence. Let $\text{RNN}(\cdot)$ be any function computed by a recurrent neural network (we use a bi-LSTM for the encoder and an LSTM for the decoder). We call the decoder state at the ith target position t_i; $1 \leq i \leq n$. The computation performed by the baseline system is summarised below.

$$[h_1, \ldots, h_m] = \text{RNN}(x_1^m) \tag{2a}$$

$$\tilde{t}_i = \text{RNN}(t_{i-1}, y_{i-1}) \tag{2b}$$

$$e_{ij} = v_a^\top \tanh\left(W_a[\tilde{t}_i, h_j]^\top + b_a\right) \tag{2c}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j=1}^m \exp(e_{ij})} \tag{2d}$$

$$c_i = \sum_{j=1}^m \alpha_{ij} h_j \tag{2e}$$

$$t_i = W_t[\tilde{t}_i, c_i]^\top + b_t \tag{2f}$$

$$\phi_i = \text{softmax}(W_o t_i + b_o) \tag{2g}$$

The parameters $\{W_a, W_t, W_o, b_a, b_t, b_o, v_a\} \subseteq \theta$ are learned during training. The model is trained using maximum likelihood estimation. This means that we employ a cross-entropy loss whose input is the probability vector returned by the softmax.

3 Stochastic Decoder

This section introduces our stochastic decoder model for capturing word-level variation in translation data.

3.1 Motivation

Imagine an idealised translator whose translations are always perfectly accurate and fluent. If an MT system was provided with training data from such a translator, it would still encounter variation in that data. After all, there are several perfectly accurate and fluent translations for each source sentence. These can be highly different in both their lexical as well as their syntactic realisations.

In practice, of course, human translators' performance varies according to their level of education, their experience on the job, their familiarity with the textual domain and myriads of other factors. Even within a single translator variation may occur due to level of stress, tiredness or status of health. That translation corpora contain variation is acknowledged by the machine translation community in the design of their evaluation metrics which are geared towards comparing one machine-generated translation against several human translations (see e.g. Papineni et al., 2002).

Prior to our work, the only attempt at modelling the latent variation underlying these different translations was made by Zhang et al. (2016) who introduced a sentence level Gaussian variable. Intuitively, however, there is more to latent variation than a unimodal density can capture, for example, there may be several highly likely clusters of plausible variations. A cluster may e.g. consist of identical syntactic structures that differ in word choice, another may consist of different syntactic constructs such as active or passive constructions. Multimodal modelling of these variations is thus called for—and our results confirm this intuition.

An example of variation comes from free word order and agreement phenomena in morphologically rich languages. An English sentence with rigid word order may be translated into several orderings in German. However, all orderings need to respect the agreement relationship between the main verb and the subject (indicated by underlining) as well as the dative case of the direct object (dashes) and the accusative of the indirect object (dots). The agreement requirements are fixed and independent of word order.

1. I can't imagine you naked.
   (a) Ich kann mir dich nicht nackt vorstellen.
   (b) Ich kann dich mir nicht nackt vorstellen.
   (c) Dich kann ich mir nicht nackt vorstellen.

Stochastically encoding the word order variation allows the model to learn the same agreement phenomenon from different translation variants as it does not need to encode the word order and agreement relationships jointly in the decoder state.

Further examples of VP and NP variation from an actual translation corpus are shown in Figure 1.

We aim to address these word-level variation phenomena with a stochastic decoder model.

预计听证会将进行两天。	VOM19981105_0700_0262
The hearing is expected to last two days. The hearing will last two days. The hearings are expected to last two days. It is expected that the hearing will go on for two days.	
众议院共和党的起诉人则希望传唤莱温斯基等多达15个人出庭作证。	VOM19981230_0700_0515
However, the Republican complainant in the House wanted to summon 15 people including Lewinsky to testify in court. The prosecutor of Republican Party in House of Representative hoped to summons more than 15 persons, including Lewinsky, to court. The House of Representatives republican prosecution hopes to summon over fifteen witnesses including Monica Lewinsky to appear in court.	

Figure 1: Examples from the multiple-translation Chinese corpus (LDC2002T01), where the translations come from different translators. These demonstrate the lexical variation of the verb and variation between passive and raising structures (top), and lexical variation on the agent NP (bottom). Both examples also exhibit appreciable length variation.

3.2 Model formulation

The model contains a latent Gaussian variable for each target position. This variable depends on the previous latent states and the decoder state. Through the use of recurrent networks, the conditioning context does not need to be restricted and the likelihood factorises exactly.

$$P(y_1^n|x_1^m) = \int \mathrm{d}z_0^n \, p(z_0|x_1^m) \times \prod_{i=1}^{n} p(z_i|z_{<i}, y_{<i}, x_1^m) P(y_i|z_1^i, y_{<i}, x_1^m) \tag{3}$$

As can be seen from Equation (3), the model also contains a 0th latent variable that is meant to initialise the chain of latent variables based solely on the source sentence. Contrast this with the model of Zhang et al. (2016) which uses *only* that 0th variable.

A graphical representation of the stochastic decoder model is given in Figure 2a. Its generative story is as follows

$$Z_0|x_1^m \sim \mathcal{N}(\mu_0, \sigma_0^2) \tag{4a}$$
$$Z_i|z_{<i}, y_{<i}, x_1^m \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{4b}$$
$$Y_i|z_0^i, y_{<i}, x_1^m \sim \mathrm{Cat}(\phi_i) \tag{4c}$$

where $i = 1, \ldots, n$ and both the Gaussian and the Categorical parameters are predicted by neural network architectures whose inputs vary per time step. This probabilistic formulation can be implemented with a multitude of different architectures. We present ours in the next section.

3.3 Neural Architecture

Since the model contains latent variables and is parametrised by a neural network, it falls into the class of deep generative models (DGMs). We use a reparametrisation of the Gaussian variables (Kingma and Welling, 2014; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014) to enable backpropagation inside a stochastic computation graph (Schulman et al., 2015). In order to sample d-dimensional Gaussian variable $z \in \mathbb{R}^d$ with mean μ and variance σ^2, we first sample from a standard Gaussian distribution and then transform the sample,

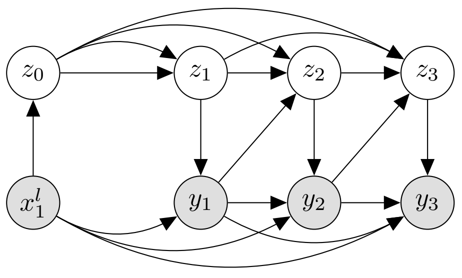
$$z = \mu + \sigma \odot \epsilon \qquad \epsilon \sim \mathcal{N}(0, \mathrm{I}) \ . \tag{5}$$

Here $\mu, \sigma \in \mathbb{R}^d$ and $\odot$ denotes element-wise multiplication (also known as Hadamard product). See the supplement for details on the Gaussian reparametrisation.

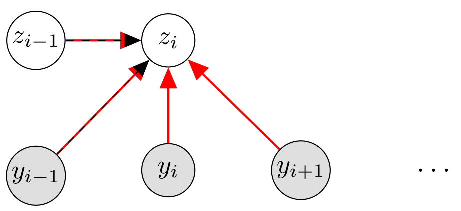
We use neural networks with one hidden layer with a tanh activation to compute the mean and standard deviation of each Gaussian distribution. A softplus transformation is applied to the output of the standard deviation's network to ensure positivity. Let us denote the functions that these networks compute by f.

For the initial latent state z_0 we compute the mean and standard deviation as

$$\mu_0 = f_{\mu_0}(h_m) \qquad \sigma_0 = f_{\sigma_0}(h_m) \ . \tag{6}$$



(a)



(b)

Figure 2: Graphical representation of 2a the generative model and 2b the inference model. Black lines indicate generative parameters (θ) and red lines variational parameters (λ). Dashed red-black lines indicate that the inference model uses feature representations computed by the generative model as inputs. Through the recurrent net, the generative model (2a) also conditions its outputs on all previous latent assignments. We omit these arrows to avoid clutter. The inference model (2b) is only used at training time. Dots indicate further conditioning context.

The parameters of all other latent distributions are computed by functions f_μ and f_σ whose inputs vary per target position.

$$\mu_i = f_\mu\left(t_{i-1}, z_{i-1}\right) \quad \sigma_i = f_\sigma\left(t_{i-1}, z_{i-1}\right) \tag{7}$$

Using these values, each latent variable is sampled according to Equation (5). The sampled latent variables are then used to modify the update of the decoder hidden state (Equation (2b)) as follows:

$$\tilde{t}_i = \text{RNN}\left(t_{i-1}, y_{i-1}, z_i\right) \tag{8}$$

The remaining computations stay unchanged. Notice that the latent values are used directly in updating the decoder state. This makes the decoder state a function of a random variable and thus the decoder state is itself random. Applying this argument recursively shows that also the attention mechanism is random, making the decoder entirely stochastic.

4 Inference and Training

We use variational inference (see e.g. Blei et al., 2017) to train the model. In variational inference, we employ a variational distribution $q(z)$ that approximates the true posterior $p(z|x)$ over the latent variables. The distribution $q(z)$ has its own set of parameters λ that is disjoint from the set of model parameters θ. It is used to maximise the evidence lower bound (ELBO) which is a lower bound on the marginal likelihood $p(x)$. The ELBO is maximised with respect to both the model parameters θ and the variational parameters λ.

Most NLP models that use DGMs only use one latent variable (e.g. Bowman et al., 2016). Models that use several variables usually employ a mean field approximation under which all latent variables are independent. This turns the ELBO into a sum of expectations (e.g. Zhou and Neubig, 2017). For our stochastic decoder we design a more flexible approximation posterior family which respects the dependencies between the latent variables,

$$q(z_0^n) = q(z_0)\prod_{i=1}^{n} q(z_i|z_{<i}) \ . \tag{9}$$

Our stochastic decoder can be viewed as a stack of conditional DGMs (Sohn et al., 2015) in which the latent variables depend on one another. The ELBO thus consists of nested positional ELBOs,

$$\begin{aligned}&\text{ELBO}_0 + \mathbb{E}_{q(z_0)}[\text{ELBO}_1 \\ &+\mathbb{E}_{q(z_1)}[\text{ELBO}_2 + \ldots]] \ ,\end{aligned} \tag{10}$$

where for a given target position i the ELBO is

$$\begin{aligned}\text{ELBO}_i = \ &\mathbb{E}_{q(z_i)}\left[\log p(y_i|x_1^m, y_{<i}, z_{<i}, z_i)\right] \\ &- \text{KL}\left(q(z_i) \mid\mid p(z_i|x_1^m, y_{<i}, z_{<i})\right) \ .\end{aligned} \tag{11}$$

The first term is often called *reconstruction* or *likelihood* term whereas the second term is called the *KL* term. Since the KL term is a function of two Gaussian distributions, and the Gaussian is an exponential family, we can compute it analytically (Michalowicz et al., 2014), without the need for sampling. This is very similar to the hierarchical latent variable model of Rezende et al. (2014).

Following common practice in DGM research, we employ a neural network to compute the variational distributions. To discriminate it from the generative model, we call this neural net the *inference model*. At training time both the source and target sentence are observed. We exploit this by endowing our inference model with a "look-ahead" mechanism. Concretely, samples from the inference network condition on the information available to the generation network (Section 3.3) and also on the target words that are yet to be processed by the generative decoder. This allows the latent distribution to not only encode information about the currently modelled word but also about the target words that follow it. The conditioning of the inference network is illustrated graphically in Figure 2b.

The inference network produces additional representations of the target sentence. One representation encodes the target sentence bidirectionally (12a), in analogy to the source sentence encoding. The second representation is built by encoding the target sentence in reverse (12b). This reverse encoding can be used to provide information about future context to the decoder. We use the symbols b and r for the bidirectional and reverse target encodings, respectively. In our experiments, we again use LSTMs to compute these encodings.

$$\left[b_1, \ldots, b_n\right] = \text{RNN}\left(y_1^n\right) \tag{12a}$$

$$\left[r_1, \ldots, r_n\right] = \text{RNN}\left(y_1^n\right) \tag{12b}$$

In analogy to the generative model (Section 3.3), the inference network uses single hidden layer networks to compute the mean and standard deviations of the latent variable distributions. We denote these functions g and again employ different functions for the initial latent state and all other latent states.

$$\mu_0 = g_{\mu_0}\left(h_m, b_n\right) \tag{13a}$$

$$\sigma_0 = g_{\sigma_0}\left(h_m, b_n\right) \tag{13b}$$

$$\mu_i = g_{\mu}\left(t_{i-1}, z_{i-1}, r_i, y_i\right) \tag{13c}$$

$$\sigma_i = g_{\sigma}\left(t_{i-1}, z_{i-1}, r_i, y_i\right) \tag{13d}$$

As before, we use Equation (5) to sample from the variational distribution.

During training, all samples are obtained from the inference network. Only at test time do we sample from the generator. Notice that since the inference network conditions on representations produced by the generator network, a naïve application of backpropagation would update parts of the generator network with gradients computed for the inference network. We prevent this by blocking gradient flow from the inference net into the generator.

4.1 Analysis of the Training Procedure

The training procedure as outlined above does not work well empirically. This is because our model uses a **strong generator**. By this we mean that the generation model (that is the baseline NMT model) is a very good density model in and by itself and does not need to rely on latent information to achieve acceptable likelihood values during training. DGMs with strong generators have a tendency to not make use of latent information (Bowman et al., 2016). This problem went initially unnoticed because early DGMs (Kingma and Welling, 2014; Rezende et al., 2014) used **weak generators**[2], i.e. models that made very strong independence assumptions and were not able to capture contextual information without making use of the information encoded by the latent variable.

Why DGMs would ignore the latent information can be understood by considering the KL-term of the ELBO. In order for the latent variable to be informative about the observed data, we need them to have high mutual information $I(Z;Y)$.

$$I(Z;Y) = \mathbb{E}_{p(z,y)}\left[\log \frac{p(Z,Y)}{p(Z)p(Y)}\right] \tag{14}$$

Observe that we can rewrite the mutual information as an expected KL divergence by applying the definition of conditional probability.

$$I(Z;Y) = \mathbb{E}_{p(y)}\left[\text{KL}\left(p(Z|Y) \;||\; p(Z)\right)\right] \tag{15}$$

Since we cannot compute the posterior $p(z|y)$ exactly, we approximate it with the variational distribution $q(z|y)$ (the joint is approximated by $q(z|y)p(y)$ where the latter factor is the data distribution). To the extent that the variational distribution recovers the true posterior, the mutual information can be computed this way. In fact, if we take the learned prior $p(z)$ to be an approximation of the marginal $\int q(z|y)p(y)\mathrm{d}y$ it can easily be shown that the thus computed KL term is an upper bound on mutual information (Alemi et al., 2017).

The trouble is that the ELBO (Equation (11)) can be trivially maximised by setting the KL-term to 0 and maximising only the reconstruction term.

[2]The term *weak generator* has first been coined by Alemi et al. (2017).

This is especially likely at the beginning of training when the variational approximation does not yet encode much useful information. We can only hope to learn a useful variational distribution if a) the variational approximation is allowed to move away from the prior and b) the resulting increase in the reconstruction term is higher than the increase in the KL-term (i.e. the ELBO increases overall).

Several schemes have been proposed to enable better learning of the variational distribution (Bowman et al., 2016; Kingma et al., 2016; Alemi et al., 2017). Here we use KL scaling and increase the scale gradually until the original objective is recovered. This has the following effect: during the initial learning stage, the KL-term barely contributes to the objective and thus the updates to the variational parameters are driven by the signal from the reconstruction term and hardly restricted by the prior.

Once the scale factor approaches 1 the variational distribution will be highly informative to the generator (assuming sufficiently slow increase of the scale factor). The KL-term can now be minimised by matching the prior to the variational distribution. Notice that up to this point, the prior has hardly been updated. Thus moving the variational approximation back to the prior would likely reduce the reconstruction term since the standard normal prior is not useful for inference purposes. This is in stark contrast to Bowman et al. (2016) whose prior was a *fixed* standard normal distribution. Although they used KL scaling, the KL term could only be decreased by moving the variational approximation back to the fixed prior. This problem disappears in our model where priors are learned.

Moving the prior towards the variational approximation has another desirable effect. The prior can now learn to emulate the variational "lookahead" mechanism without having access to future contexts itself (recall that the inference model has access to future target tokens). At test time we can thus hope to have learned latent variable distributions that encode information not only about the output at the current position but about future outputs as well.

5 Experiments

We report experiments on the IWSLT 2016 data set which contains transcriptions of TED talks and their respective translations. We trained models to translate from English into Arabic, Czech, French and German. The number of sentences for each language after preprocessing is shown in Table 1.

Data	Arabic	Czech	French	German
Train	224,125	114,389	220,399	196,883
Dev	6,746	5,326	5,937	6,996
Test	2,762	2,762	2,762	2,762

Table 1: Number of parallel sentence pairs for each language paired with English for IWSLT data.

The vocabulary was split into 50,000 subword units using Google's sentence piece[3] software in its standard settings. As our baseline NMT systems we use Sockeye (Hieber et al., 2017)[4]. Sockeye implements several different NMT models but here we use the standard recurrent attentional model described in Section 2. We report baselines with and without dropout (Srivastava et al., 2014). For dropout a retention probability of 0.5 was used.

As a second baseline we use our own implementation of the model of Zhang et al. (2016) which contains a single sentence-level Gaussian latent variable (SENT). Our implementation differs from theirs in three aspects. First, we feed the last hidden state of the bidirectional encoding into encoding of the source and target sentence into the inference network (Zhang et al. (2016) use the average of all states). Second, the latent variable is smaller in size than the one used by (Zhang et al., 2016).[5] This was done to make their model and the stochastic decoder proposed here as similar as possible. Finally, their implementation was based on groundhog whereas ours builds on Sockeye.

Our stochastic decoder model (SDEC) is also built on top of the basic Sockeye model. It adds the components described in Sections 3 and 4. Recall that the functions that compute the means and standard deviations are implemented by neural nets with a single hidden layer with tanh activation. The width of that layer is twice the size of the latent variable. In our experiments we tested different latent variable sizes and used KL scaling (see Section 4.1). The scale started from 0 and was increased by $^1/_{20,000}$ after each mini-batch. Thus, at iteration t the scale is $\min(^t/_{20,000}, 1)$.

All models use 1028 units for the LSTM hid-

[3]https://github.com/google/sentencepiece
[4]https://github.com/awslabs/sockeye
[5]We did, however, find that increasing the latent variable size actually hurt performance in our implementation.

den state (or 512 for each direction in the bidirectional LSTMs) and 256 for the attention mechansim. Training is done with Adam (Kingma and Ba, 2015). In decoding we use a beam of size 5 and output the most likely word at each position. We deterministically set all latent variables to their mean values during decoding. Monte Carlo decoding (Gal, 2016) is difficult to apply to our setting as it would require sampling entire translations.

Results We show the BLEU scores for all models that we tested on the IWSLT data set in Table 2. The stochastic decoder dominates the Sockeye baseline across all 4 languages, and outperforms SENT on most languages. Except on German, there is a trend towards smaller latent variable sizes being more helpful. This is in line with findings by Chung et al. (2015) and Fraccaro et al. (2016) who also used relatively small latent variables. This observation also implies that our model does not improve simply because it has more parameters than the baseline.

That the margin between the SDEC and SENT models is not large was to be expected for two reasons. First, Chung et al. (2015) and Fraccaro et al. (2016) have shown that stochastic RNNs lead to enormous improvements in modelling continuous sequences but only modest increases in performance for discrete sequences (such as natural language). Second, translation performance is measured in BLEU score. We observed that SDEC often reached better ELBO values than SENT indicating a better model fit. How to fully leverage the better modelling ability of stochastic RNNs when producing discrete outputs is a matter of future research.

Qualitative Analysis Finally, we would like to demonstrate that our model does indeed capture variation in translation. To this end, we randomly picked sentences from the IWSLT test set and had our model translate them several times, however, the values of the latent variables were sampled instead of fixed. Contrary to the BLEU-based evaluation, beam search was not used in this evaluation in order to avoid interaction between different latent variable samples. See Figure 3 for examples of syntactic and lexical variation. It is important to note that we do not sample from the categorical output distribution. For each target position we pick the most likely word. A non-stochastic NMT system would always yield the same translation in this scenario. Interestingly, when we applied the sampling procedure to the SENT model it did not produce any variation at all, thus behaving like a deterministic NMT system. This supports our initial point that the SENT model is likely insensitive to local variation, a problem that our model was designed to address. Like the model of Bowman et al. (2016), SENT presumably tends to ignore the latent variable.

6 Related Work

The stochastic decoder is strongly influenced by previous work on stochastic RNNs. The first such proposal was made by Bayer and Osendorfer (2015) who introduced i.i.d. Gaussian latent variables at each output position. Since their model neglects any sequential dependence of the noise sources, it underperformed on several sequence modeling tasks. Chung et al. (2015) made the latent variables depend on previous information by feeding the previous decoder state into the latent variable sampler. Their inference model did not make use of future elements in the sequence.

Using a "look-ahead" mechanism in the inference net was proposed by Fraccaro et al. (2016) who had a separate stochastic and deterministic RNN layer which both influence the output. Since the stochastic layer in their model depends on the deterministic layer but not vice versa, they could first run the deterministic layer at inference time and then condition the inference net's encoding of the future on the thus obtained features. Like us, they used KL scaling during training.

More recently, Goyal et al. (2017) proposed an auxiliary loss that has the inference net predict future feature representations. This approach yields state-of-the-art results but is still in need of a theoretical justification.

Within translation, Zhang et al. (2016) were the first to incorporate Gaussian variables into an NMT model. Their approach only uses one sentence-level latent variable (corresponding to our z_0) and can thus not deal with word-level variation directly. Concurrently to our work, Su et al. (2018) have also proposed a recurrent latent variable model for NMT. Their approach differs from ours in that they do not use a 0^{th} latent variable nor a look-ahead mechanism during inference time. Furthermore, their underlying recurrent model is a GRU.

In the wider field of NLP, deep generative mod-

Model	Dropout	LatentDim	Arabic	Czech	French	German
Sockeye	None	None	8.2	6.9	23.5	14.3
Sockeye	0.5	None	8.4	7.4	24.4	15.1
SENT	0.5	64	8.4	7.3	24.8	15.3
SENT	0.5	128	8.7	7.4	24.0	15.7
SENT	0.5	256	8.9	7.4	24.7	15.5
SDEC	0.5	64	8.2	7.7	25.3	15.4
SDEC	0.5	128	8.8	7.5	24.2	15.6
SDEC	0.5	256	8.7	7.5	23.2	15.9

Table 2: BLEU scores for different models on the IWSLT data for translation into English. Recall that all SDEC and SENT models used KL scaling during training.

Source	Coincidentally, at the same time, the first easy-to-use clinical tests for diagnosing autism were introduced.
SENT	Im gleichen Zeitraum wurden die ersten einfachen klinischen Tests für Diagnose getestet.
SDEC	Übrigens, zur gleichen Zeit, wurden die ersten einfache klinische Tests für die Diagnose von Autismus eingeführt.
SDEC	Übrigens, zur gleichen Zeit, waren die ersten einfache klinische Tests für die Diagnose von Autismus eingeführt worden.
Source	They undertook a study of autism prevalence in the general population.
SENT	Sie haben eine Studie von Autismus in der allgemeinen Population übernommen.
SDEC	Sie entwarfen eine Studie von Autismus in der allgemeinen Bevölkerung.
SDEC	Sie führten eine Studie von Autismus in der allgemeinen Population ein.

Figure 3: Sampled translations from our model (SDEC) and the sentent-level latent variable model (SENT). The first SDEC example shows alternation between the German simple past and past perfect. The past perfect introduces a long range dependency between the main and auxiliary verb (underlined) that the model handles well. The second example shows variation in the lexical realisation of the verb. The second variant uses a particle verb and we again observe a long range dependency between the main verb and its particle (underlined).

els have been applied mostly in monolingual settings such as text generation (Bowman et al., 2016; Semeniuta et al., 2017), morphological analysis (Zhou and Neubig, 2017), dialogue modelling (Wen et al., 2017), question selection (Miao et al., 2016) and summarisation (Miao and Blunsom, 2016).

7 Conclusion and Future Work

We have presented a recurrent decoder for machine translation that uses word-level Gaussian variables to model underlying sources of variation observed in translation corpora. Our experiments confirm our intuition that modelling variation is crucial to the success of machine translation. The proposed model consistently outperforms strong baselines on several language pairs.

As this is the first work that systematically considers word-level variation in NMT, there are lots of research ideas to explore in the future. Here, we list the three which we believe to be most promising.

- Latent factor models: our model only contains one source of variation per word. A latent factor model such as DARN (Gregor et al., 2014) would consider several sources simultaneously. This would also allow us to perform a better analysis of the model behaviour as we could correlate the factors with observed linguistic phenomena.
- Richer prior and variational distributions: The diagonal Gaussian is likely too simple a

distribution to appropriately model the variation in our data. Richer distributions computed by normalising flows (Rezende and Mohamed, 2015; Kingma et al., 2016) will likely improve our model.
- Extension to other architectures: Introducing latent variables into non-autoregressive translation models such as the transformer (Vaswani et al., 2017) should increase their translation ability further.

8 Acknowledgements

Philip Schulz and Wilker Aziz were supported by the Dutch Organisation for Scientific Research (NWO) VICI Grant nr. 277-89-002. Trevor Cohn is the recipient of an Australian Research Council Future Fellowship (project number FT130101105).

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. 2017. An information theoretic analysis of deep latent variable models. *arxiv preprint* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Justin Bayer and Christian Osendorfer. 2015. Learning stochastic recurrent networks. In *ICLR*.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association* 112(518):859–877.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL 2016*. pages 10–21.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *NIPS 28*, pages 2980–2988.

Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NIPS 29*, pages 2199–2207.

Yarin Gal. 2016. *Uncertainty in Deep Learning*. Ph.D. thesis, University of Cambridge.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*. pages 1243–1252.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *NIPS 30*, pages 6716–6726.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. 2014. Deep autoregressive networks. In *ICML*. Bejing, China, pages 1242–1250.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *ArXiv e-prints* .

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *NIPS 29*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *ICLR*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*. pages 319–328.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*. New York, New York, USA, pages 1727–1736.

Joseph Victor Michalowicz, Jonathan M. Nichols, and Frank Bucholtz. 2014. *Handbook of Differential Entropy*. CRC Press.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*. pages 311–318.

Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *ICML*. volume 37, pages 1530–1538.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *NIPS 28*, pages 3528–3536.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *EMNLP*. pages 627–637.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NIPS 28*, pages 3483–3491.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Ly, Xianpei Han, and Biao Zhang. 2018. Variational recurrent neural machine translation. In *AAAI*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 27*, pages 3104–3112.

Michalis Titsias and Miguel Lázaro-Gredilla. 2014. Doubly stochastic Variational Bayes for non-conjugate inference. In *ICML*. pages 1971–1979.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 30*, pages 6000–6010.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *ICML*. pages 3732–3741.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *EMNLP*. pages 521–530.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *ACL*. pages 310–320.

Forest-Based Neural Machine Translation

Chunpeng Ma[1,3][*], **Akihiro Tamura**[2], **Masao Utiyama**[3], **Tiejun Zhao**[1], **Eiichiro Sumita**[3]
[1]Harbin Institute of Technology, Harbin, China
[2]Ehime University, Matsuyama, Japan
[3]National Institute of Information and Communications Technology, Kyoto, Japan
{cpma, tjzhao}@hit.edu.cn, tamura@cs.ehime-u.ac.jp
{mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

Tree-based neural machine translation (NMT) approaches, although achieved impressive performance, suffer from a major drawback: they only use the 1-best parse tree to direct the translation, which potentially introduces translation mistakes due to parsing errors. For statistical machine translation (SMT), forest-based methods have been proven to be effective for solving this problem, while for NMT this kind of approach has not been attempted. This paper proposes a forest-based NMT method that translates a linearized packed forest within a simple sequence-to-sequence framework (i.e., a forest-to-string NMT model). The BLEU score of the proposed method is higher than that of the string-to-string NMT, tree-based NMT, and forest-based SMT systems.

1 Introduction

NMT has witnessed promising improvements recently. Depending on the types of input and output, these efforts can be divided into three categories: *string-to-string* systems (Sutskever et al., 2014; Bahdanau et al., 2014); *tree-to-string* systems (Eriguchi et al., 2016, 2017); and *string-to-tree* systems (Aharoni and Goldberg, 2017; Nadejde et al., 2017). Compared with string-to-string systems, tree-to-string and string-to-tree systems (henceforth, *tree-based systems*) offer some attractive features. They can use more syntactic information (Li et al., 2017), and can conveniently incorporate prior knowledge (Zhang et al., 2017). Because of these advantages, tree-based methods become the focus of many researches of NMT nowadays.

Based on how to represent trees, there are two main categories of tree-based NMT methods: representing trees by a tree-structured neural network (Eriguchi et al., 2016; Zaremoodi and Haffari, 2017), representing trees by linearization (Vinyals et al., 2015; Dyer et al., 2016; Ma et al., 2017). Compared with the former, the latter method has a relatively simple model structure, so that a larger corpus can be used for training and the model can be trained within reasonable time, hence is preferred from the viewpoint of computation. Therefore we focus on this kind of methods in this paper.

In spite of impressive performance of tree-based NMT systems, they suffer from a major drawback: they only use the 1-best parse tree to direct the translation, which potentially introduces translation mistakes due to parsing errors (Quirk and Corston-Oliver, 2006). For SMT, forest-based methods have employed a *packed forest* to address this problem (Huang, 2008), which represents exponentially many parse trees rather than just the 1-best one (Mi et al., 2008; Mi and Huang, 2008). But for NMT, (computationally efficient) forest-based methods are still being explored.[1]

Because of the structural complexity of forests, the lack of appropriate topological ordering, and the hyperedge-attachment nature of weights (see Section 3.1 for details), it is not trivial to linearize a forest. This hinders the development of forest-based NMT to some extent.

Inspired by the tree-based NMT methods based on linearization, we propose an efficient forest-based NMT approach (Section 3), which can encode the syntactic information of a *packed for-*

[*] Contribution during internship at National Institute of Information and Communications Technology.

[1]Zaremoodi and Haffari (2017) have proposed a forest-based NMT method based on a forest-structured neural network recently, but it is computationally *in*efficient (see Section 5).

est on the basis of a novel *weighted* linearization method for a packed forest (Section 3.1), and can decode the linearized packed forest within the simple sequence-to-sequence framework (Section 3.2). Experiments demonstrate the effectiveness of our method (Section 4).

2 Preliminaries

We first review the general sequence-to-sequence model (Section 2.1), then describe tree-based NMT systems based on linearization (Section 2.2), and finally introduce the packed forest, through which exponentially many trees can be represented in a compact manner (Section 2.3).

2.1 Sequence-to-sequence model

Current NMT systems usually resort to a simple framework, i.e., the sequence-to-sequence model (Cho et al., 2014; Sutskever et al., 2014). Given a source sequence $(x_0, \ldots, x_T)$, in order to find a target sequence $(y_0, \ldots, y_{T'})$ that maximizes the conditional probability $p(y_0, \ldots, y_{T'} \mid x_0, \ldots, x_T)$, the sequence-to-sequence model uses one RNN to encode the source sequence into a fixed-length context vector c and another RNN to decode this vector and generate the target sequence. Formally, the probability of the target sequence can be calculated as follows:

$$\begin{aligned} p(y_0, \ldots, &y_{T'} \mid x_0, \ldots, x_T) \\ &= \prod_{t=0}^{T'} p(y_t \mid c, y_0, \ldots, y_{t-1}), \end{aligned} \tag{1}$$

where

$$p(y_t \mid c, y_0, \ldots, y_{t-1}) = g(y_{t-1}, s_t, c), \tag{2}$$

$$s_t = f(s_{t-1}, y_{t-1}, c), \tag{3}$$

$$c = q(h_0, \ldots, h_T), \tag{4}$$

$$h_t = f(e_t, h_{t-1}). \tag{5}$$

Here, g, f, and q are nonlinear functions; h_t and s_t are the hidden states of the source-side RNN and target-side RNN, respectively, c is the context vector, and e_t is the embedding of x_t.

Bahdanau et al. (2014) introduced an attention mechanism to deal with the issues related to long sequences (Cho et al., 2014). Instead of encoding the source sequence into a fixed vector c, the attention model uses different c_i when calculating the target-side output y_i at time step i:

$$c_i = \sum_{j=0}^{T} \alpha_{ij} h_j, \tag{6}$$

$$\alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(a(s_{i-1}, h_k))}. \tag{7}$$

The function $a(s_{i-1}, h_j)$ can be regarded as the soft alignment between the target-side RNN hidden state s_{i-1} and the source-side RNN hidden state h_j.

Depending on the format of the source/target sequences, this framework can be regarded as a string-to-string NMT system (Sutskever et al., 2014), a tree-to-string NMT system (Li et al., 2017), or a string-to-tree NMT system (Aharoni and Goldberg, 2017).

2.2 Linear-structured tree-based NMT systems

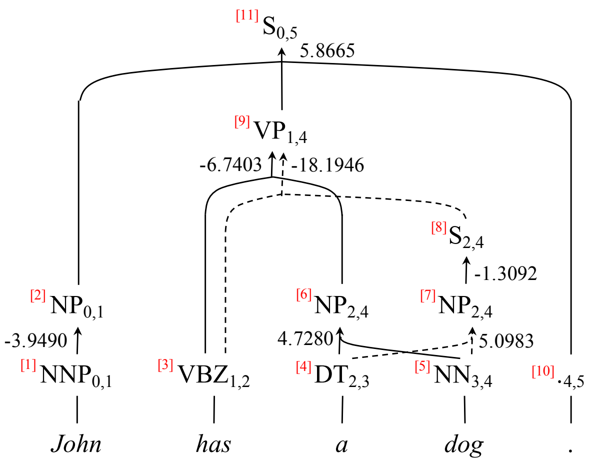
Regarding the linearization adopted for tree-to-string NMT (i.e., linearization of the source side), Sennrich and Haddow (2016) encoded the sequence of dependency labels and the sequence of words simultaneously, partially utilizing the syntax information, while Li et al. (2017) traversed the constituent tree of the source sentence and combined this with the word sequence, utilizing the syntax information completely.

Regarding the linearization used for string-to-tree NMT (i.e., linearization of the target side), Nadejde et al. (2017) used a CCG supertag sequence as the target sequence, while Aharoni and Goldberg (2017) applied a linearization method in a top-down manner, generating a sequence ensemble for the annotated tree in the Penn Treebank (Marcus et al., 1993). Wu et al. (2017) used transition actions to linearize a dependency tree, and employed the sequence-to-sequence framework for NMT.

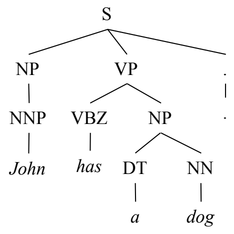
All the current tree-based NMT systems use only one tree for encoding or decoding. In contrast, we hope to utilize multiple trees (i.e., a forest). This is not trivial, on account of the lack of a fixed traversal order and the need for a compact representation.

2.3 Packed forest

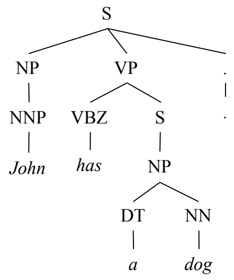
The *packed forest* gives a representation of exponentially many parse trees, and can compactly encode many more candidates than the n-best list



(a) Packed forest



(b) Correct constituent tree, $score = -46.2389$



(c) Incorrect constituent tree, $score = -58.6321$

Figure 1: An example of (a) a packed forest. The numbers in the brackets located at the upper-left corner of each node in the packed forest show one correct topological ordering of the nodes. The packed forest is a compact representation of two trees: (b) the correct constituent tree, and (c) an incorrect constituent tree. Note that the terminal nodes (i.e., words in the sentence) in the packed forest are shown only for illustration, and they do not belong to the packed forest.

(Huang, 2008). Figure 1a shows a packed forest, which can be unpacked into two constituent trees (Figure 1b and Figure 1c).

Formally, a packed forest is a pair $\langle V, E \rangle$, where V is the set of nodes and E is the set of hyperedges. Each $v \in V$ has the form $X_{i,j}$, where X is a constituent label and $i, j \in [0, n]$ are indices of words, showing that the node spans the words ranging from i (inclusive) to j (exclusive). Here, n is the length of the input sentence. Each $e \in E$ is a three-tuple $\langle head(e), tails(e), score(e) \rangle$, where $head(e) \in V$ is similar to the head node in a constituent tree, and $tails(e) \in V^*$ is similar to the set of child nodes in a constituent tree. $score(e) \in \mathbb{R}$ is the log probability that $tails(e)$ represents the tails of $head(e)$ calculated by the parser. Based on $score(e)$, the score of a constituent tree $\mathcal{T}$ can be calculated as follows:

$$score(\mathcal{T}) = -\lambda n + \sum_{e \in E(\mathcal{T})} score(e), \quad (8)$$

where $E(\mathcal{T})$ is the set of hyperedges appearing in tree $\mathcal{T}$, and λ is a regularization coefficient for the sentence length.[2]

[2]Following the configuration of Charniak and Johnson (2005), for all the experiments in this paper, we fixed λ to $\log_2 600$.

3 Forest-based NMT

We first propose a linearization method for the packed forest (Section 3.1), then describe how to encode the linearized forest (Section 3.2), which can then be translated by the conventional decoder (see Section 2.1).

3.1 Forest linearization

Recently, several studies have focused on the linearization methods of a syntax tree, both in the area of tree-based NMT (Section 2.2) and parsing (Vinyals et al., 2015; Dyer et al., 2016; Ma et al., 2017). Basically, these methods follow a fixed traversal order (e.g., depth-first). This does not exist for the packed forest which is a directed acyclic graph (DAG). Furthermore, the weights are attached to edges of a packed forest instead of the nodes. This further increases the difficulty of linearization.

Topological ordering algorithms for DAG (Kahn, 1962; Tarjan, 1976) are not good solutions, because the topological ordering outputted by algorithms is not always optimal for machine trans-

Algorithm 1 Linearization of a packed forest

```
1:  function LINEARIZEFOREST($\langle V, E\rangle, \mathbf{w}$)
2:      $v \leftarrow$ FINDROOT($V$)
3:      $\mathbf{r} \leftarrow []$
4:      EXPANDSEQ($v, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
5:      return r
6:  function FINDROOT($V$)
7:      for $v \in V$ do
8:          if $v$ has no parent then
9:              return $v$
10: procedure EXPANDSEQ($v, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
11:     for $e \in E$ do
12:         if $head(e) = v$ then
13:             if $tails(e) \neq \emptyset$ then
14:                 for $t \in$ SORT($tails(e)$) do        ▷ Sort $tails(e)$ by word indices.
15:                     EXPANDSEQ($t, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
16:                 $l \leftarrow$ LINEARIZEEDGE($head(e), \mathbf{w}$)
17:                 r.append($\langle l, \sigma(0.0)\rangle$) ▷ $\sigma$ is the sigmoid function, i.e., $\sigma(x) = \frac{1}{1+e^{-x}}, x \in \mathbb{R}$.
18:                 $l \leftarrow$ ⓒLINEARIZEEDGES($tails(e), \mathbf{w}$) ▷ ⓒ is a unary operator.
19:                 r.append($\langle l, \sigma(score(e))\rangle$)
20:             else
21:                 $l \leftarrow$ LINEARIZEEDGE($head(e), \mathbf{w}$)
22:                 r.append($\langle l, \sigma(0.0)\rangle$)
23: function LINEARIZEEDGE($X_{i,j}, \mathbf{w}$)
24:     return $X \otimes (\odot_{k=i}^{j-1} w_k)$
25: function LINEARIZEEDGES($\mathbf{v}, \mathbf{w}$)
26:     return $\oplus_{v \in \mathbf{v}}$LINEARIZEEDGE($v, \mathbf{w}$)
```

NNP⊗*John* / NP⊗*John* / ⓒNNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ⓒNP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ⓒVBZ⊗*has*⊕NP⊗*a*⊙*dog* / ⓒVBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ⓒNP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.

Figure 2: Linearization result of the packed forest in Figure 1a.

lation. In particular, a topological ordering could ignore "word sequential information" and "parent-child information."

For example, for the packed forest in Figure 1a, although "[10]→[1]→[2]→ ··· →[9]→[11]" is a valid topological ordering, the *word sequential information* of the words (e.g., "John" should be located ahead of the period), which is fairly crucial for translation of languages with fixed pragmatic word order such as Chinese or English, is lost.

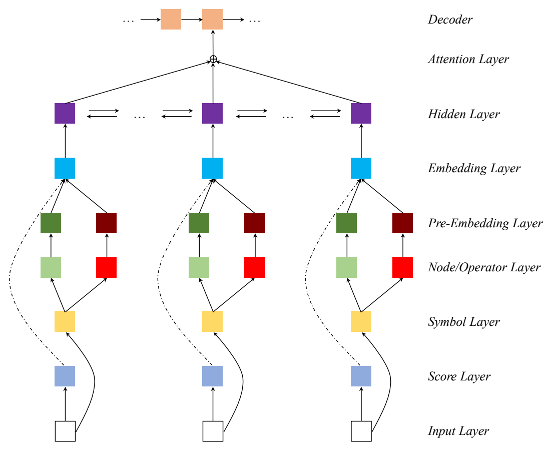
As another example, for the packed forest above, nodes [2], [9], and [10] are all the children of node [11]. However, in the topological order "[1]→[2]→ ··· →[9]→[10]→[11]," node [2] is quite far from node [11], while nodes [9] and [10] are both close to node [11]. The *parent-child information* cannot be reflected in this topological order, which is not what we would expect.

To address the above two problems, we propose a novel linearization algorithm for a packed forest (Algorithm 1). The algorithm linearizes the packed forest from the root node (Line 2) to leaf nodes by calling the EXPANDSEQ procedure (Line 15) recursively, while preserving the word order in the sentence (Line 14). In this way, *word sequential information* is preserved. Within the EXPANDSEQ procedure, once a hyperedge is linearized (Line 16), the tails are also linearized immediately (Line 18). In this way, *parent-child information* is preserved. Intuitively, different parts of constituent trees should be combined in different ways, therefore we define different operators (ⓒ, ⊗, ⊕, or ⊙) to represent the relationships, so that the representations of these parts can be combined in different ways (see Section 3.2 for details). Words are concatenated by the operator "⊙" with each other, a word and a constituent label is concatenated by the operator "⊗", the linearization results of child nodes are concatenated by the operator "⊕" with each other, while the unary operator "ⓒ" is used to indicate that the node is the child node of the previous part. Furthermore, each token in the linearized sequence is related to a score, representing the confidence of the parser.

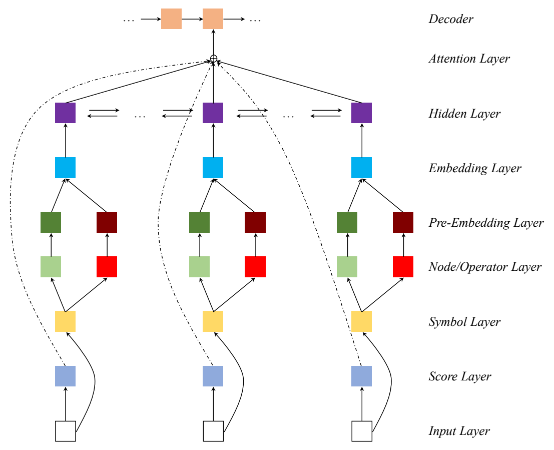
The linearization result of the packed forest in Figure 1a is shown in Figure 2. Tokens in the linearized sequence are separated by slashes. Each token in the sequence is composed of different types of symbols and combined by different operators. We can see that word sequential information is preserved. For example, "NNP⊗*John*" (linearization result of node [1]) is in front of "VBZ⊗*has*" (linearization result of node [3]), which is in front of "DT⊗*a*" (linearization result of node [4]). Moreover, parent-child information is also preserved. For example, "NP⊗*John*" (linearization result of node [2]) is followed by "ⓒNNP⊗*John*" (linearization result of node [1], the child of node [2]).

Note that our linearization method does *not* output fully recoverable packed forests. What we *do* want to do is to encode syntax information as much as possible, so that we can improve the performance of NMT.

Also note that there is one more advantage of our linearization method: the linearized sequence is a weighted sequence, while all the previous studies ignored the weights during linearization.



(a) Score-on-Embedding (SoE)



(b) Score-on-Attention (SoA)

Figure 3: The architecture of the forest-based NMT system.

By preserving only the nodes and hyperedges in the 1-best tree and removing all others, our linearization method can be regarded as a tree-linearization method. Compared with other tree-linearization methods, our method combines several different kinds of information within one symbol, retaining the parent-child information, and incorporating the confidence of the parser in the sequence. We examine whether the weights can be useful not only for linear structured tree-based NMT but also for our forest-based NMT in Section 4.

Furthermore, although our method is non-reversible for packed forests, it is reversible for constituent trees, in that the linearization is processed exactly in the depth-first traversal order and all necessary information in the tree nodes has been encoded. As far as we know, there is no previous work on linearization of packed forests.

3.2 Encoding the linearized forest

The linearized packed forest forms the input of the encoder, which has two major differences from the input of a sequence-to-sequence NMT system. First, the input sequence of the encoder consists of two parts: the symbol sequence and the score sequence. Second, the symbol sequence consists of three types of symbols: words, constituent labels, and operators (Ⓒ, $\otimes$, $\oplus$, or $\odot$) that connect the other two types of symbols. Based on these characteristics, we propose a method of encoding the linearized forest.

Formally, the input layer receives two sequences: the symbol sequence $\mathbf{l} = (l_0, \dots, l_T)$ and the score sequence $\xi = (\xi_0, \dots, \xi_T)$, where l_i denotes the i-th symbol and ξ_i its score. Then, the two sequences are fed into the symbol layer and the score layer, respectively. Any item $l \in \mathbf{l}$ in the symbol layer has the form

$$l = o_0 x_1 o_1 \dots x_{m-1} o_{m-1} x_m, \tag{9}$$

where each x_k $(k = 1, \dots, m)$ is a word or a constituent label, m is the total number of words and constituent labels in a symbol, o_0 is "Ⓒ" or empty, and each o_k $(k = 1, \dots, m-1)$ is either "$\otimes$", "$\oplus$", or "$\odot$". Then, in the node/operator layer, these x and o are separated and rearranged as $\mathbf{x} = (x_1, \dots, x_m, o_0, \dots, o_{m-1})$, which is fed to the pre-embedding layer. The pre-embedding layer generates a sequence $\mathbf{p} = (p_1, \dots, p_m, \dots, p_{2m})$, which is calculated as follows:

$$\mathbf{p} = W_{emb}[I(\mathbf{x})]. \tag{10}$$

Here, the function $I(\mathbf{x})$ returns a list of the indices in the dictionary for all the elements in $\mathbf{x}$, including words, constituent labels, and operators. In addition, W_{emb} is the embedding matrix of size $(|w_{word}| + |w_{label}| + 4) \times d_{word}$, where $|w_{word}|$ and $|w_{label}|$ are the vocabulary size of words and constituent labels, respectively, d_{word} is the dimension of the word embedding, and there are four possible operators: "Ⓒ," "$\otimes$," "$\oplus$," and "$\odot$." Note that $\mathbf{p}$ is a list of $2m$ vectors, and the dimension of each vector is d_{word}. Hereafter, $\mathbf{p}$ for the k-th symbol l_k is denoted by $\mathbf{p}_k$.

Depending on where the score layer is incorporated, we propose two frameworks: Score-on-Embedding (SoE) and Score-on-Attention (SoA), which are illustrated in Figure 3. In SoE, the k-th element of the embedding layer is calculated as follows:

$$e_k = \xi_k \sum_{p \in \mathbf{p}_k} p, \tag{11}$$

while in SoA, the k-th element of the embedding layer is calculated as

$$e_k = \sum_{p \in \mathbf{p}_k} p, \tag{12}$$

where $k = 0, \ldots, T$. Note that $e_k \in \mathbb{R}^{d_{word}}$. In this manner, the proposed forest-to-string NMT framework is connected with the conventional sequence-to-sequence NMT framework.

After calculating the embedding vectors in the embedding layer, the hidden vectors are calculated using Equation (5). When calculating the context vector c_i, SoE and SoA differ from each other. For SoE, the c_i is calculated using Equations (6) and (7), while for SoA, the α_{ij} used to calculate the c_i is determined as follows:

$$\alpha_{ij} = \frac{\exp(\xi_j a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(\xi_k a(s_{i-1}, h_k))}. \tag{13}$$

Then, using the decoder of the sequence-to-sequence framework, the sentence of the target language can be generated.

4 Experiments

4.1 Setup

We evaluated the effectiveness of our forest-based NMT systems on English-to-Chinese and English-to-Japanese translation tasks.[3] The statistics of the corpora used in our experiments are summarized in Table 1.

The packed forests of English sentences were obtained by the constituent parser proposed by Huang (2008).[4] We filtered out the sentences for which the parser was not able to generate any packed forests and those longer than 80 words. For NIST datasets, we simply chose the first reference among the four English references of NIST corpora. For Chinese sentences, we used Stanford segmenter[5] for segmentation. For Japanese sentences, we followed the preprocessing steps recommended in WAT 2017.[6]

Language	Corpus	Usage	#Sent.
English-Japanese	ASPEC	train	100,000
		dev.	1790
		test	1812
English-Chinese	LDC[7]	train	1,423,695
	FBIS		233,510
	NIST MT 02	dev.	876
	NIST MT 03	test	919
	NIST MT 04		1,788
	NIST MT 05		1,082

Table 1: Statistics of the corpora.

We implemented our framework based on `nematus`[8] (Sennrich et al., 2017). For optimization, following previous research such as (Bahdanau et al., 2014), we used the ADADELTA algorithm (Zeiler, 2012). In order to avoid overfitting, we used dropout (Srivastava et al., 2014) on the embedding layer and hidden layer, with the dropout probability set to 0.2. We used the gated recurrent unit (Cho et al., 2014) as the recurrent unit of RNNs, which are bi-directional, with one hidden layer.

Based on the tuning result, we set the maximum length of the input sequence to 300, the hidden layer size as 512, the dimension of word embedding as 620, and the batch size for training as 40. We pruned the packed forest using the algorithm of Huang (2008), removing all hyperedges e which satisfy $\delta(e) > 10^{-5}$, where $\delta(e)$ is the difference between the cost of hyperedge e and that of the globally best derivation. If the linearization of the pruned forest is still longer than 300, then we linearize the 1-best parsing tree instead of the forest. As for the stopping criterion of training process, we evaluated the BLEU score on the development set every 10,000 updates. If BLEU score was not increased in ten consecutive evaluations, then training was stopped. During decoding, we performed beam search with the beam size of 12.

[3]English is commonly chosen as the *target* language. We chose English as the *source* language because a high-performance forest parser is not available for other languages.

[4]`http://web.engr.oregonstate.edu/~huanlian/software/forest-reranker/forest-charniak-v0.8.tar.bz2`

[5]`https://nlp.stanford.edu/software/stanford-segmenter-2017-06-09.zip`

[6]`http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2017/baseline/dataPreparationJE.html`

[7]LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08, and LDC2005T06

[8]`https://github.com/EdinburghNLP/nematus`

System Types		Systems & Configurations	MT 03 FBIS	MT 03 LDC	MT 04 FBIS	MT 04 LDC	MT 05 FBIS	MT 05 LDC	p value (w.r.t. s2s)	p value (w.r.t. 1-best)
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36	-	-
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33	-	-
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59	-	-
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84	-	-
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80	-	-
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01	< 0.01	-
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16	< 0.05	-
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61	< 0.005	-
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75	< 0.01	< 0.1
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41	< 0.001	No
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**	< 0.001	< 0.05

Table 2: English-Chinese experimental results (character-level BLEU). "FS," "SN," "TN," and "FN" denote forest-based SMT, string-based NMT, tree-based NMT, and forest-based NMT systems, respectively. The p values were obtained by the paired bootstrap resampling significance test (Koehn, 2004) over the NIST MT 03 to 05 corpus, with respect to the baselines: s2s or 1-best.

System Types		Systems & Configurations	BLEU (test)	p value (w.r.t. s2s)	p value (w.r.t. 1-best)
Previous	FS	Mi et al. (2008)	34.13	-	-
	TN	Eriguchi et al. (2016)	37.52	-	-
		Chen et al. (2017)	36.94	-	-
		Li et al. (2017)	36.21	-	-
Ours	SN	s2s	37.10	-	-
	TN	1-best (No score)	38.01	< 0.05	-
		1-best (SoE)	38.53	< 0.01	-
		1-best (SoA)	39.42	< 0.001	-
	FN	Forest (No score)	37.92	< 0.1	No
		Forest (SoE)	41.35	< 0.01	< 0.1
		Forest (SoA)	**42.17**	< 0.005	< 0.05

Table 3: English-Japanese experimental results (character-level BLEU).

4.2 Experimental results

Tables 2 and 3 summarize the experimental results. To avoid the effect of segmentation errors, the performance was evaluated by character-level BLEU (Papineni et al., 2002). We compared our proposed models (i.e., Forest (SoE) and Forest (SoA)) with three types of baseline: a string-to-string model (s2s), forest-based models that do not use score sequences (Forest (No score)), and tree-based models that use the 1-best parsing tree (1-best (No score, SoE, SoA)). For the 1-best models, we preserved the nodes and hyperedges that were used in the 1-best constituent tree in the packed forest, while removing all other nodes and hyperedges. For the "No score" configurations, we forced the input score sequence to be a sequence of 1.0 with the same length as the input symbol sequence, so that neither the embedding layer nor the attention layer were affected by the score sequence.

In addition, we also made a comparison with some state-of-the-art tree-based systems. As the SMT system, we examined Mi et al. (2008). Specifically, we used the implementation of `cicada`.[9] For NMT systems, we compared with three systems: Eriguchi et al. (2016)[10] and Chen et al. (2017),[11] both are publicly available, and we reimplemented the "Mixed RNN Encoder" model of Li et al. (2017), because of its outstanding performance on the NIST MT corpus.

We can see that for both English-Chinese and English-Japanese, compared with the s2s baseline system, both the 1-best and forest-based configurations yield better results. This indicates syntactic information contained in the constituent trees or forests is indeed useful for machine translation. Specifically, we observed the following facts.

First, among the three different frameworks, i.e., SoE, SoA, and No-score, the SoA framework performed the best, while the No-score framework

[9]`https://github.com/tarowatanabe/cicada`

[10]`https://github.com/tempra28/tree2seq`

[11]`https://github.com/howardchenhd/Syntax-awared-NMT`

[Source] In the Czech Republic , which was ravaged by serious floods last summer , the temperatures in its border region adjacent to neighboring Slovakia plunged to minus 18 degrees Celsius .

[Reference]	去年	夏季	曾	出现	严重	水患	的	捷克	共和国	,	其	邻近	斯洛伐克	的	边界	地区	气温	低	至		摄氏	零下	18	度	。
	last	summer	ever	appear	serious	floods	of	Czech	Republic	,	its	adjacent	Slovakia	of	border	region	temperature	decrease	to		Celsius	minus	18	degree	.
[s2s]	去年	夏天	,	捷克	地区	遭受		严重	洪灾		的		捷克		边境	地区	气温	下降	了	18	摄氏			度	。
	last	summer	,	Czech	region	suffer		serious	floods		of		Czech		border	region	temperature	decrease	-ed	18	Celsius			degree	.
[1best Tree]	去年	夏天	,	遭受	特大	洪灾	的	捷克	边境				地区	的			气温	下降	了	18	摄氏			度	。
	last	summer	,	suffer	serious	floods	of	Czech	border				region	of			temperature	decrease	-ed	18	Celsius			degree	.
[Forest]	去年	夏天		发生	严重	水灾	的	捷克	共和国	,		毗邻	斯洛伐克	的	边境	地区	温度	下降	至			零下	18	度	。
	last	summer		occur	serious	floods	of	Czech	Republic	,		adjacent	Slovakia	of	border	region	temperature	decrease	to			minus	18	degree	.

Figure 4: Chinese translation results of an English sentence.

performed the worst. This indicates that the scores of the edges in constituent trees or packed forests, which reflect the confidence of the correctness of the edges, are indeed useful. In fact, for the 1-best constituent parsing tree, the score of the edge reflects the confidence of the parser. With this information, the NMT system succeeded to learn a better attention, paying more attention to the confident structure and less attention to the unconfident structure, which improved the translation performance. This fact was ignored by previous studies on tree-based NMT. Furthermore, it is better to use the scores to adjust the values of attention instead of rescaling the word embeddings, because modifying word embeddings may alter the semantic meanings of words.

Second, compared with the cases that only use the 1-best constituent trees, with some exceptions, using packed forests yielded statistical significantly better results for the SoE and SoA frameworks. This shows the effectiveness of using more syntactic information. Compared with one constituent tree, the packed forest, which contains multiple different trees, describes the syntactic structure of the sentence in different aspects, which together increase the accuracy of machine translation. However, without using the scores, the 1-best constituent tree is preferred. This is because without using the scores, all trees in the packed forest are treated equally, which makes it easy to import noise into the encoder.

Compared with other types of state-of-the-art systems, our systems using only the 1-best tree (1-best (SoE, SoA)) were better than the other tree-based systems. Moreover, our NMT systems using the packed forests achieved the best performance. These results also support the usefulness of the scores of the edges and packed forests in NMT.

As for the efficiency, the training time of the SoA system was slightly longer than that of the SoE system, which was about twice of the s2s baseline. The training time of the tree-based system was about 1.5 times of the baseline. For the case of Forest (SoA), with 1 core of Tesla P100 GPU and LDC corpus as the training data, training spent about 10 days, and decoding speed was about 10 sentences per second. The reason for the relatively low efficiency is that the linearized sequences of packed forests were much longer than word sequences, enlarging the scale of the inputs. Despite this, the training process ended within reasonable time.

4.3 Qualitative analysis

Figure 4 shows the translation results of an English sentence using several different configurations: the s2s baseline, using only the 1-best tree (SoE), and using the packed forest (SoE). This is a sentence from NIST MT 03, and the training corpus is the LDC corpus.

For the s2s case, no syntactic information was utilized, and therefore the output of the system was not a grammatical Chinese sentence. The attributive phrase of "Czech border region" (i.e., "last summer ... floods") is a complete sentence. However, this is not grammatically allowed in Chinese.

For the case of using 1-best constituent tree, the output was a grammatical Chinese sentence. However, the phrase "adjacent to neighboring Slovakia" was completely ignored in the translation result. Analysis of the constituent tree revealed that this phrase was incorrectly parsed as an "adverb phrase," and consequently the NMT system paid a little attention to it, because of the low confidence given by the parser.

In contrast, the packed forest did not ignore this phrase and translated it correctly. Actually, besides "adverb phrase," this phrase was also correctly parsed as an "adjective phrase," and covered by multiple different nodes in the forest. Because of the wide coverage, it is difficult for the encoder to ignore the phrase.

We also noticed that our method performed better on learning attention. For example, in Figure 4, we observed that for s2s model, the decoder paid attention to the word "Czech" twice, which caused the output sentence containing the corresponding Chinese translation twice. On the other hand, for our forest model, by using the syntax information, the decoder paid an attention to the phrase "In the Czech Republic" only once; therefore the decoder generated the correct output.

5 Related work

Incorporating syntactic information into NMT systems is attracting widespread attention nowadays. Compared with conventional string-to-string NMT systems, tree-based systems demonstrate a better performance with the help of constituent trees or dependency trees.

The first noteworthy study was Eriguchi et al. (2016), which used Tree-structured LSTM (Tai et al., 2015) to encode the HPSG syntax tree of the sentence in the source-side in a bottom-up manner. Then, Chen et al. (2017) enhanced the encoder with a top-down tree encoder.

As a simple extension of Eriguchi et al. (2016), very recently, Zaremoodi and Haffari (2017) proposed a forest-based NMT method by representing the packed forest with a forest-structured neural network. However, their method was evaluated in small-scale MT settings (each training dataset consists of under 10k parallel sentences). In contrast, our proposed method is effective in a large-scale MT setting, and we present qualitative analysis regarding the effectiveness of using forests in NMT.

Although these methods obtained good results, the tree-structured network used by the encoder made the training and decoding relatively slow, restricting the scope of application.

Other attempts at encoding syntactic trees have also been proposed. Eriguchi et al. (2017) combined the Recurrent Neural Network Grammar (Dyer et al., 2016) with NMT systems, while Li et al. (2017) linearized the constituent tree and encoded it using RNNs. The training of these methods is fast, because of the linear structures of RNNs. However, all these syntax-based NMT systems used only the 1-best parsing tree, making the systems sensitive to parsing errors.

Instead of using trees to represent syntactic information, some studies used other data structures to represent the latent syntax of the input sentence. For example, Hashimoto and Tsuruoka (2017) proposed translating using a latent graph. However, such systems do not enjoy the benefit of handcrafted syntactic knowledge, because they do not use a parser trained from a large treebank with human annotations.

Compared with these related studies, our framework utilizes a linearized packed forest, meaning the encoder can encode exponentially many trees in an efficient manner. The experimental results demonstrated these advantages.

6 Conclusion and future work

We proposed a new encoding method for NMT, which encodes a packed forest for the source sentence using linear-structured neural networks, such as RNN. When introducing packed forest, we confirmed that the score of each edge is indispensable. Compared with conventional string-to-string NMT systems and tree-to-string NMT systems, our framework can utilize exponentially many linearized parsing trees during encoding, without significantly decreasing the efficiency. This represents the first attempt to use a forest within the string-to-string NMT framework. The experimental results demonstrate the effectiveness of our method.

As future work, we plan to design some more elaborate structures to incorporate the score layer into the encoder. We will also apply the proposed linearization method to other tasks.

Acknowledgements

We are grateful to the anonymous reviewers for their insightful comments and suggestions. We thank Lemao Liu from Tencent AI Lab for his suggestions about the experiments. We thank Atsushi Fujita whose suggestions greatly improve the readability and the logical soundness of this paper. This work was done during the internship of Chunpeng Ma at NICT. Akihiro Tamura is supported by JSPS KAKENHI Grant Number JP18K18110. Tiejun Zhao is supported by the National Natural Science Foundation of China (NSFC) via grant 91520204 and State High-Tech Development Plan of China (863 program) via grant 2015AA015405.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. *arXiv preprint arXiv:1702.02265*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594.

Arthur B Kahn. 1962. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697.

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Sumita Eiichiro. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3237–3243.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational linguistics*, 19(2):313–330.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Brich. 2017. Syntax-aware neural machine translation using CCG. *arXiv preprint arXiv:1702.01147*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 62–69.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Robert Endre Tarjan. 1976. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707.

Poorya Zaremoodi and Gholamreza Haffari. 2017. Incorporating syntactic uncertainty in neural machine translation with forest-to-sequence model. *arXiv preprint arXiv:1711.07019*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1514–1523.

Context-Aware Neural Machine Translation Learns Anaphora Resolution

Elena Voita
Yandex, Russia
University of Amsterdam, Netherlands
lena-voita@yandex-team.ru

Pavel Serdyukov
Yandex, Russia
pavser@yandex-team.ru

Rico Sennrich
University of Edinburgh, Scotland
University of Zurich, Switzerland
rico.sennrich@ed.ac.uk

Ivan Titov
University of Edinburgh, Scotland
University of Amsterdam, Netherlands
ititov@inf.ed.ac.uk

Abstract

Standard machine translation systems process sentences in isolation and hence ignore extra-sentential information, even though extended context can both prevent mistakes in ambiguous cases and improve translation coherence. We introduce a context-aware neural machine translation model designed in such way that the flow of information from the extended context to the translation model can be controlled and analyzed. We experiment with an English-Russian subtitles dataset, and observe that much of what is captured by our model deals with improving pronoun translation. We measure correspondences between induced attention distributions and coreference relations and observe that the model implicitly captures anaphora. It is consistent with gains for sentences where pronouns need to be gendered in translation. Beside improvements in anaphoric cases, the model also improves in overall BLEU, both over its context-agnostic version (+0.7) and over simple concatenation of the context and source sentences (+0.6).

1 Introduction

It has long been argued that handling discourse phenomena is important in translation (Mitkov, 1999; Hardmeier, 2012). Using extended context, beyond the single source sentence, should in principle be beneficial in ambiguous cases and also ensure that generated translations are coherent. Nevertheless, machine translation systems typically ignore discourse phenomena and translate sentences in isolation.

Earlier research on this topic focused on handling specific phenomena, such as translating pronouns (Le Nagard and Koehn, 2010; Hardmeier and Federico, 2010; Hardmeier et al., 2015), discourse connectives (Meyer et al., 2012), verb tense (Gong et al., 2012), increasing lexical consistency (Carpuat, 2009; Tiedemann, 2010; Gong et al., 2011), or topic adaptation (Su et al., 2012; Hasler et al., 2014), with special-purpose features engineered to model these phenomena. However, with traditional statistical machine translation being largely supplanted with neural machine translation (NMT) models trained in an end-to-end fashion, an alternative is to directly provide additional context to an NMT system at training time and hope that it will succeed in inducing relevant predictive features (Jean et al., 2017; Wang et al., 2017; Tiedemann and Scherrer, 2017; Bawden et al., 2018).

While the latter approach, using context-aware NMT models, has demonstrated to yield performance improvements, it is still not clear what kinds of discourse phenomena are successfully handled by the NMT systems and, importantly, how they are modeled. Understanding this would inform development of future discourse-aware NMT models, as it will suggest what kind of inductive biases need to be encoded in the architecture or which linguistic features need to be exploited.

In our work we aim to enhance our understanding of the modelling of selected discourse phenomena in NMT. To this end, we construct a simple discourse-aware model, demonstrate that it achieves improvements over the discourse-agnostic baseline on an English-Russian subtitles dataset (Lison et al., 2018) and study which context information is being captured in the model. Specifically, we start with the Transformer (Vaswani et al., 2017), a state-of-the-art model for context-agnostic NMT, and modify it in such way that it can handle additional context. In our model, a source sentence and a context sentence are first encoded independently, and then a single attention layer, in a combination with a gating function, is used to produce a context-aware representation of the source sentence. The information from context can only flow through this attention layer. When compared to simply concatenating input sentences, as proposed by Tiedemann and Scherrer (2017), our architecture appears both more accurate (+0.6 BLEU) and also guarantees that the contextual information cannot bypass the attention layer and hence remain undetected in our analysis.

We analyze what types of contextual information are exploited by the translation model. While studying the attention weights, we observe that much of the information captured by the model has to do with pronoun translation. It is not entirely surprising, as we consider translation from a language without grammatical gender (English) to a language with grammatical gender (Russian). For Russian, translated pronouns need to agree in gender with their antecedents. Moreover, since in Russian verbs agree with subjects in gender and adjectives also agree in gender with pronouns in certain frequent constructions, mistakes in translating pronouns have a major effect on the words in the produced sentences. Consequently, the standard cross-entropy training objective sufficiently rewards the model for improving pronoun translation and extracting relevant information from the context.

We use automatic co-reference systems and human annotation to isolate anaphoric cases. We observe even more substantial improvements in performance on these subsets. By comparing attention distributions induced by our model against co-reference links, we conclude that the model implicitly captures coreference phenomena, even without having any kind of specialized features which could help it in this subtask. These observations also suggest potential directions for future work. For example, effective co-reference systems go beyond relying simply on embeddings of contexts. One option would be to integrate 'global' features summarizing properties of groups of mentions predicted as linked in a document (Wiseman et al., 2016), or to use latent relations to trace entities across documents (Ji et al., 2017). Our key contributions can be summarized as follows:

- we introduce a context-aware neural model, which is effective and has a sufficiently simple and interpretable interface between the context and the rest of the translation model;
- we analyze the flow of information from the context and identify pronoun translation as the key phenomenon captured by the model;
- by comparing to automatically predicted or human-annotated coreference relations, we observe that the model implicitly captures anaphora.

2 Neural Machine Translation

Given a source sentence $\mathbf{x} = (x_1, x_2, \dots, x_S)$ and a target sentence $\mathbf{y} = (y_1, y_2, \dots, y_T)$, NMT models predict words in the target sentence, word by word.

Current NMT models mainly have an encoder-decoder structure. The encoder maps an input sequence of symbol representations $\mathbf{x}$ to a sequence of distributed representations $\mathbf{z} = (z_1, z_2, \dots, z_S)$. Given $\mathbf{z}$, a neural decoder generates the corresponding target sequence of symbols $\mathbf{y}$ one element at a time.

Attention-based NMT The encoder-decoder framework with attention has been proposed by Bahdanau et al. (2015) and has become the defacto standard in NMT. The model consists of encoder and decoder recurrent networks and an attention mechanism. The attention mechanism selectively focuses on parts of the source sentence during translation, and the attention weights specify the proportions with which information from different positions is combined.

Transformer Vaswani et al. (2017) proposed an architecture that avoids recurrence completely. The Transformer follows an encoder-decoder architecture using stacked self-attention and fully connected layers for both the encoder and decoder. An important advantage of the Transformer is that it is more parallelizable and faster to train than recurrent encoder-decoder models.

From the source tokens, learned embeddings are generated and then modified using positional encodings. The encoded word embeddings are then used as input to the encoder which consists of N

layers each containing two sub-layers: (a) a multi-head attention mechanism, and (b) a feed-forward network.

The self-attention mechanism first computes attention weights: i.e., for each word, it computes a distribution over all words (including itself). This distribution is then used to compute a new representation of that word: this new representation is set to an expectation (under the attention distribution specific to the word) of word representations from the layer below. In multi-head attention, this process is repeated h times with different representations and the result is concatenated.

The second component of each layer of the Transformer network is a feed-forward network. The authors propose using a two-layered network with the ReLU activations.

Analogously, each layer of the decoder contains the two sub-layers mentioned above as well as an additional multi-head attention sub-layer that receives input from the corresponding encoding layer.

In the decoder, the attention is masked to prevent future positions from being attended to, or in other words, to prevent illegal leftward information flow. See Vaswani et al. (2017) for additional details.

The proposed architecture reportedly improves over the previous best results on the WMT 2014 English-to-German and English-to-French translation tasks, and we verified its strong performance on our data set in preliminary experiments. Thus, we consider it a strong state-of-the-art baseline for our experiments. Moreover, as the Transformer is attractive in practical NMT applications because of its parallelizability and training efficiency, integrating extra-sentential information in Transformer is important from the engineering perspective. As we will see in Section 4, previous techniques developed for recurrent encoder-decoders do not appear effective for the Transformer.

3 Context-aware model architecture

Our model is based on Transformer architecture (Vaswani et al., 2017). We leave Transformer's decoder intact while incorporating context information on the encoder side (Figure 1).

Source encoder: The encoder is composed of a stack of N layers. The first $N-1$ layers are identical and represent the original layers of Transformer's encoder. The last layer incorporates contextual information as shown in Figure 1. In addition to multi-head self-attention it has a block which performs multi-head attention over the output of the context encoder stack. The outputs of the two attention mechanisms are combined via a gated sum. More precisely, let $c_i^{(s-attn)}$ be the output of the multi-head self-attention, $c_i^{(c-attn)}$ the output of the multi-head attention to context, c_i their gated sum, and σ the logistic sigmoid function, then

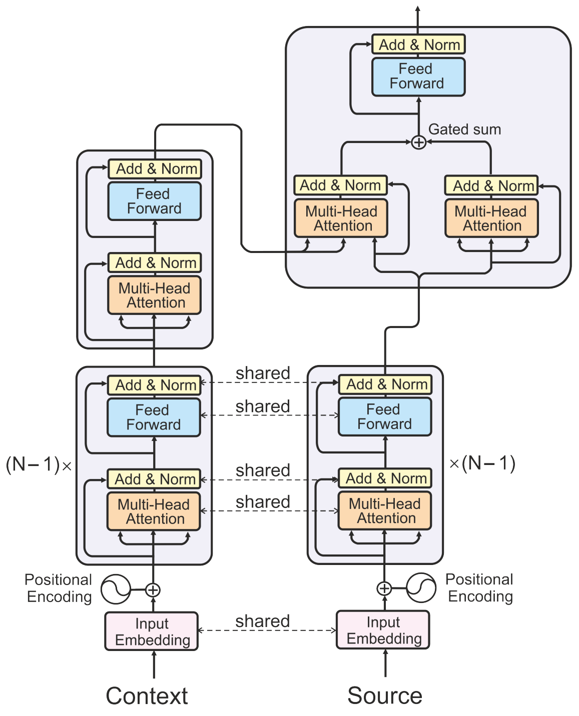


Figure 1: Encoder of the discourse-aware model

$$g_i = \sigma\left(W_g\left[c_i^{(s-attn)}, c_i^{(c-attn)}\right] + b_g\right) \quad (1)$$

$$c_i = g_i \odot c_i^{(s-attn)} + (1 - g_i) \odot c_i^{(c-attn)} \quad (2)$$

Context encoder: The context encoder is composed of a stack of N identical layers and replicates the original Transformer encoder. In contrast to related work (Jean et al., 2017; Wang et al., 2017), we found in preliminary experiments that using separate encoders does not yield an accurate model. Instead we share the parameters of the first $N-1$ layers with the source encoder.

Since major proportion of the context encoder's parameters are shared with the source encoder, we add a special token (let us denote it `<bos>`) to the beginning of context sentences, but not source sentences, to let the shared layers know whether it is encoding a source or a context sentence.

4 Experiments

4.1 Data and setting

We use the publicly available OpenSubtitles2018 corpus (Lison et al., 2018) for English and Russian.[1] As described in the appendix, we apply data cleaning and randomly choose 2 million training instances from the resulting data. For development and testing, we randomly select two subsets of 10000 instances from movies not encountered in training.[2] Sentences were encoded using byte-pair encoding (Sennrich et al., 2016), with source and target vocabularies of about 32000 tokens. We generally used the same parameters and optimizer as in the original Transformer (Vaswani et al., 2017). The hyperparameters, preprocessing and training details are provided in the supplementary material.

5 Results and analysis

We start by experiments motivating the setting and verifying that the improvements are indeed genuine, i.e. they come from inducing predictive features of the context. In subsequent section 5.2, we analyze the features induced by the context encoder and perform error analysis.

5.1 Overall performance

We use the traditional automatic metric BLEU on a general test set to get an estimate of the overall performance of the discourse-aware model, before turning to more targeted evaluation in the next section. We provide results in Table 1.[3] The 'baseline' is the discourse-agnostic version of the Transformer. As another baseline we use the standard Transformer applied to the concatenation of the previous and source sentences, as proposed by Tiedemann and Scherrer (2017). Tiedemann and Scherrer (2017) only used a special symbol to mark where the context sentence ends and the source sentence begins. This technique performed badly with the non-recurrent Transformer architecture in preliminary experiments, resulting in a substantial degradation of performance (over 1 BLEU). Instead, we use a binary flag at every word position in our concatenation baseline telling the encoder whether the word belongs to the context sentence or to the source sentence.

model	**BLEU**
baseline	29.46
concatenation (previous sentence)	29.53
context encoder (previous sentence)	**30.14**
context encoder (next sentence)	29.31
context encoder (random context)	29.69

Table 1: Automatic evaluation: BLEU. Significant differences at $p < 0.01$ are in bold.

We consider two versions of our discourse-aware model: one using the previous sentence as the context, another one relying on the next sentence. We hypothesize that both the previous and the next sentence provide a similar amount of additional clues about the topic of the text, whereas for discourse phenomena such as anaphora, discourse relations and elliptical structures, the previous sentence is more important.

First, we observe that our best model is the one using a context encoder for the previous sentence: it achieves 0.7 BLEU improvement over the discourse-agnostic model. We also notice that, unlike the previous sentence, the next sentence does not appear beneficial. This is a first indicator that discourse phenomena are the main reason for the observed improvement, rather than topic effects. Consequently, we focus solely on using the previous sentence in all subsequent experiments.

Second, we observe that the concatenation baseline appears less accurate than the introduced context-aware model. This result suggests that our model is not only more amendable to analysis but also potentially more effective than using concatenation.

In order to verify that our improvements are genuine, we also evaluate our model (trained with the previous sentence as context) on the same test set with shuffled context sentences. It can be seen that the performance drops significantly when a real context sentence is replaced with a random one. This confirms that the model does rely on context information to achieve the improvement in translation quality, and is not merely better regularized. However, the model is robust towards being shown a random context and obtains a performance similar to the context-agnostic baseline.

[1] http://opus.nlpl.eu/OpenSubtitles2018.php
[2] The resulting data sets are freely available at http://data.statmt.org/acl18_contextnmt_data/
[3] We use bootstrap resampling (Riezler and Maxwell, 2005) for significance testing

5.2 Analysis

In this section we investigate what types of contextual information are exploited by the model. We study the distribution of attention to context and perform analysis on specific subsets of the test data. Specifically the research questions we seek to answer are as follows:

- For the translation of which words does the model rely on contextual history most?
- Are there any non-lexical patterns affecting attention to context, such as sentence length and word position?
- Can the context-aware NMT system implicitly learn coreference phenomena without any feature engineering?

Since all the attentions in our model are multi-head, by *attention weights* we refer to an average over heads of per-head attention weights.

First, we would like to identify a *useful* attention mass coming to context. We analyze the attention maps between source and context, and find that the model mostly attends to `<bos>` and `<eos>` context tokens, and much less often attends to words. Our hypothesis is that the model has found a way to take no information from context by looking at uninformative tokens, and it attends to words only when it wants to pass some contextual information to the source sentence encoder. Thus we define *useful* contextual attention mass as sum of attention weights to context words, excluding `<bos>` and `<eos>` tokens and punctuation.

5.2.1 Top words depending on context

We analyze the distribution of attention to context for individual source words to see for which words the model depends most on contextual history. We compute the overall average attention to context words for each source word in our test set. We do the same for source words at positions higher than first. We filter out words that occurred less than 10 times in a test set. The top 10 words with the highest average attention to context words are provided in Table 2.

An interesting finding is that contextual attention is high for the translation of "it", "yours", "ones", "you" and "I", which are indeed very ambiguous out-of-context when translating into Russian. For example, "it" will be translated as third person singular masculine, feminine or neuter, or third person plural depending on its antecedent.

word	attn	pos	word	attn	pos
it	0.376	5.5	it	0.342	6.8
yours	0.338	8.4	yours	0.341	8.3
yes	0.332	2.5	ones	0.318	7.5
i	0.328	3.3	'm	0.301	4.8
yeah	0.314	1.4	you	0.287	5.6
you	0.311	4.8	am	0.274	4.4
ones	0.309	8.3	i	0.262	5.2
'm	0.298	5.1	's	0.260	5.6
wait	0.281	3.8	one	0.259	6.5
well	0.273	2.1	won	0.258	4.6

Table 2: Top-10 words with the highest average attention to context words. *attn* gives an average attention to context words, *pos* gives an average position of the source word. Left part is for words on all positions, right — for words on positions higher than first.

"You" can be second person singular impolite or polite, or plural. Also, verbs must agree in gender and number with the translation of "you".

It might be not obvious why "I" has high contextual attention, as it is not ambiguous itself. However, in past tense, verbs must agree with "I" in gender, so to translate past tense sentences properly, the source encoder must predict speaker gender, and the context may provide useful indicators.

Most surprising is the appearance of "yes", "yeah", and "well" in the list of context-dependent words, similar to the finding by Tiedemann and Scherrer (2017). We note that these words mostly appear in sentence-initial position, and in relatively short sentences. If only words after the first are considered, they disappear from the top-10 list. We hypothesize that the amount of attention to context not only depends on the words themselves, but also on factors such as sentence length and position, and we test this hypothesis in the next section.

5.2.2 Dependence on sentence length and position

We compute useful attention mass coming to context by averaging over source words. Figure 2 illustrates the dependence of this average attention mass on sentence length. We observe a disproportionally high attention on context for short sentences, and a positive correlation between the average contextual attention and context length.

It is also interesting to see the importance given to the context at different positions in the source

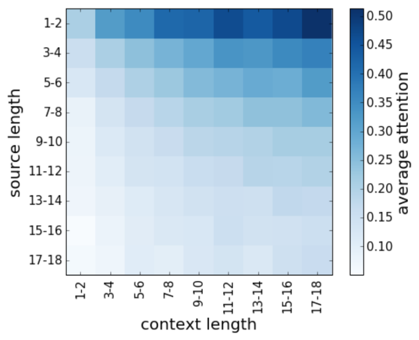


Figure 2: Average attention to context words vs. both source and context length

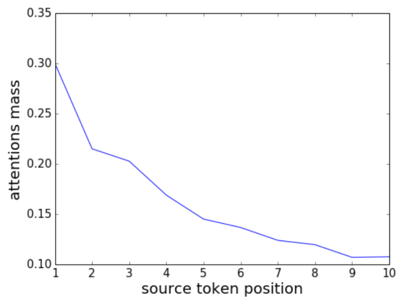


Figure 3: Average attention to context vs. source token position

sentence. We compute an average attention mass to context for a set of 1500 sentences of the same length. As can be seen in Figure 3, words at the beginning of a source sentence tend to attend to context more than words at the end of a sentence. This correlates with standard view that English sentences present hearer-old material before hearer-new.

There is a clear (negative) correlation between sentence length and the amount of attention placed on contextual history, and between token position and the amount of attention to context, which suggests that context is especially helpful at the beginning of a sentence, and for shorter sentences. However, Figure 4 shows that there is no straightforward dependence of BLEU improvement on source length. This means that while attention on context is disproportionally high for short sentences, context does not seem disproportionally more useful for these sentences.

5.3 Analysis of pronoun translation

The analysis of the attention model indicates that the model attends heavily to the contextual history for the translation of some pronouns. Here, we investigate whether this context-aware modelling results in empirical improvements in translation

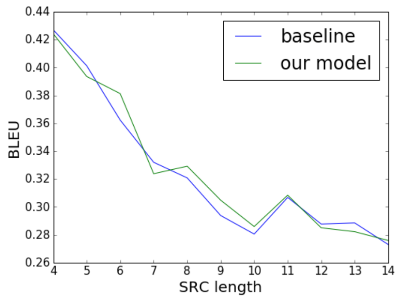


Figure 4: BLEU score vs. source sentence length

quality, and whether the model learns structures related to anaphora resolution.

5.3.1 Ambiguous pronouns and translation quality

Ambiguous pronouns are relatively sparse in a general-purpose test set, and previous work has designed targeted evaluation of pronoun translation (Hardmeier et al., 2015; Miculicich Werlen and Popescu-Belis, 2017; Bawden et al., 2018). However, we note that in Russian, grammatical gender is not only marked on pronouns, but also on adjectives and verbs. Rather than using a pronoun-specific evaluation, we present results with BLEU on test sets where we hypothesize context to be relevant, specifically sentences containing co-referential pronouns. We feed Stanford CoreNLP open-source coreference resolution system (Manning et al., 2014a) with pairs of sentences to find examples where there is a link between one of the pronouns under consideration and the context. We focus on anaphoric instances of “it” (this excludes, among others, pleonastic uses of ”it”), and instances of the pronouns “I”, “you”, and “yours” that are coreferent with an expression in the previous sentence. All these pronouns express ambiguity in the translation into Russian, and the model has learned to attend to context for their translation (Table 2). To combat data sparsity, the test sets are extracted from large amounts of held-out data of OpenSubtitles2018. Table 3 shows BLEU scores for the resulting subsets.

First of all, we see that most of the antecedents in these test sets are also pronouns. Antecedent pronouns should not be particularly informative for translating the source pronoun. Nevertheless, even with such contexts, improvements are generally larger than on the overall test set.

When we focus on sentences where the antecedent for pronoun under consideration contains

pronoun	N	#pronominal antecedent	baseline	our model	difference
it	11128	6604	25.4	26.6	**+1.2**
you	6398	5795	29.7	30.8	**+1.1**
yours	2181	2092	24.1	25.2	**+1.1**
I	8205	7496	30.1	30.0	-0.1

Table 3: BLEU for test sets with coreference between pronoun and a word in context sentence. We show both *N*, the total number of instances in a particular test set, and number of instances with *pronominal antecedent*. Significant BLEU differences are in bold.

word	N	baseline	our model	diff.
it	4524	23.9	26.1	**+2.2**
you	693	29.9	31.7	**+1.8**
I	709	29.1	29.7	**+0.6**

Table 4: BLEU for test sets of pronouns having a nominal antecedent in context sentence. *N*: number of examples in the test set.

type	N	baseline	our model	diff.
masc.	2509	26.9	27.2	**+0.3**
fem.	2403	21.8	26.6	**+4.8**
neuter	862	22.1	24.0	**+1.9**
plural	1141	18.2	22.5	**+4.3**

Table 5: BLEU for test sets of pronoun "it" having a nominal antecedent in context sentence. *N*: number of examples in the test set.

a noun, we observe even larger improvements (Table 4). Improvement is smaller for "I", but we note that verbs with first person singular subjects mark gender only in the past tense, which limits the impact of correctly predicting gender. In contrast, different types of "you" (polite/impolite, singular/plural) lead to different translations of the pronoun itself plus related verbs and adjectives, leading to a larger jump in performance. Examples of nouns co-referent with "I" and "you" include names, titles ("Mr.", "Mrs.", "officer"), terms denoting family relationships ("Mom", "Dad"), and terms of endearment ("honey", "sweetie"). Such nouns can serve to disambiguate number and gender of the speaker or addressee, and mark the level of familiarity between them.

The most interesting case is translation of "it", as "it" can have many different translations into Russian, depending on the grammatical gender of the antecedent. In order to disentangle these cases, we train the Berkeley aligner on 10m sentences and use the trained model to divide the test set with "it" referring to a noun into test sets specific to each gender and number. Results are in Table 5.

pronoun	agreement (in %)			
	random	first	last	attention
it	69	66	72	69
you	76	85	71	80
I	74	81	73	78

Table 6: Agreement with CoreNLP for test sets of pronouns having a nominal antecedent in context sentence (%).

We see an improvement of 4-5 BLEU for sentences where "it" is translated into a feminine or plural pronoun by the reference. For cases where "it" is translated into a masculine pronoun, the improvement is smaller because the masculine gender is more frequent, and the context-agnostic baseline tends to translate the pronoun "it" as masculine.

5.3.2 Latent anaphora resolution

The results in Tables 4 and 5 suggest that the context-aware model exploits information about the antecedent of an ambiguous pronoun. We hypothesize that we can interpret the model's attention mechanism as a latent anaphora resolution, and perform experiments to test this hypothesis.

For test sets from Table 4, we find an antecedent noun phrase (usually a determiner or a possessive pronoun followed by a noun) using Stanford CoreNLP (Manning et al., 2014b). We select only examples where a noun phrase contains a single noun to simplify our analysis. Then we identify which token receives the highest attention weight (excluding `<bos>` and `<eos>` tokens and punctuation). If this token falls within the antecedent span, then we treat it as agreement (see Table 6).

One natural question might be: does the attention component in our model genuinely learn to perform anaphora resolution, or does it capture some simple heuristic (e.g., pointing to the last noun)? To answer this question, we consider several baselines: choosing a random, last or first

pronoun	agreement (in %)			
	random	first	last	attention
it	40	36	52	58
you	42	63	29	67
I	39	56	35	62

Table 7: Agreement with CoreNLP for test sets of pronouns having a nominal antecedent in context sentence (%). Examples with ≥1 noun in context sentence.

noun from the context sentence as an antecedent.

Note that an agreement of the last noun for "it" or the first noun for "you" and "I" is very high. This is partially due to the fact that most context sentences have only one noun. For these examples a random and last predictions are always correct, meanwhile attention does not always pick a noun as the most relevant word in the context. To get a more clear picture let us now concentrate only on examples where there is more than one noun in the context (Table 7). We can now see that the attention weights are in much better agreement with the coreference system than any of the heuristics. This indicates that the model is indeed performing anaphora resolution.

While agreement with CoreNLP is encouraging, we are aware that coreference resolution by CoreNLP is imperfect and partial agreement with it may not necessarily indicate that the attention is particularly accurate. In order to control for this, we asked human annotators to manually evaluate 500 examples from the test sets where CoreNLP predicted that "it" refers to a noun in the context sentence. More precisely, we picked random 500 examples from the test set with "it" from Table 7. We marked the pronoun in a source which CoreNLP found anaphoric. Assessors were given the source and context sentences and were asked to mark an antecedent noun phrase for a marked pronoun in a source sentence or say that there is no antecedent at all. We then picked those examples where assessors found a link from "it" to some noun in context (79% of all examples). Then we evaluated agreement of CoreNLP and our model with the ground truth links. We also report the performance of the best heuristic for "it" from our previous analysis (i.e. last noun in context). The results are provided in Table 8.

The agreement between our model and the ground truth is 72%. Though 5% below the coreference system, this is a lot higher than the best heuristic (+18%). This confirms our conclusion that our model performs latent anaphora resolution. Interestingly, the patterns of mistakes are quite different for CoreNLP and our model (Table 9). We also present one example (Figure 5) where the attention correctly predicts anaphora while CoreNLP fails. Nevertheless, there is room for improvement, and improving the attention component is likely to boost translation performance.

	agreement (in %)
CoreNLP	77
attention	72
last noun	54

Table 8: Performance of CoreNLP and our model's attention mechanism compared to human assessment. Examples with ≥1 noun in context sentence.

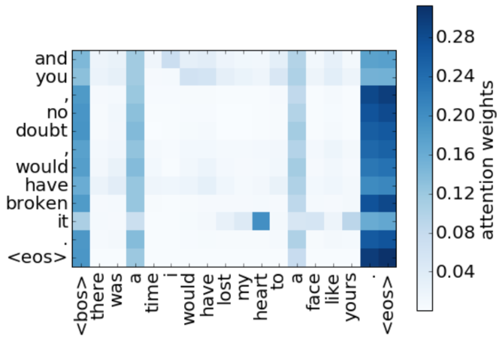


Figure 5: An example of an attention map between source and context. On the y-axis are the source tokens, on the x-axis the context tokens. Note the high attention between "it" and its antecedent "heart".

	CoreNLP	
	right	wrong
attn right	53	19
attn wrong	24	4

Table 9: Performance of CoreNLP and our model's attention mechanism compared to human assessment (%). Examples with ≥1 noun in context sentence.

6 Related work

Our analysis focuses on how our context-aware neural model implicitly captures anaphora. Early work on anaphora phenomena in statistical machine translation has relied on external systems for coreference resolution (Le Nagard and Koehn, 2010; Hardmeier and Federico, 2010). Results were mixed, and the low performance of coreference resolution systems was identified as a problem for this type of system. Later work by Hardmeier et al. (2013) has shown that cross-lingual pronoun prediction systems can implicitly learn to resolve coreference, but this work still relied on external feature extraction to identify anaphora candidates. Our experiments show that a context-aware neural machine translation system can implicitly learn coreference phenomena without any feature engineering.

Tiedemann and Scherrer (2017) and Bawden et al. (2018) analyze the attention weights of context-aware NMT models. Tiedemann and Scherrer (2017) find some evidence for above-average attention on contextual history for the translation of pronouns, and our analysis goes further in that we are the first to demonstrate that our context-aware model learns latent anaphora resolution through the attention mechanism. This is contrary to Bawden et al. (2018), who do not observe increased attention between a pronoun and its antecedent in their recurrent model. We deem our model more suitable for analysis, since it has no recurrent connections and fully relies on the attention mechanism within a single attention layer.

7 Conclusions

We introduced a context-aware NMT system which is based on the Transformer architecture. When evaluated on an En-Ru parallel corpus, it outperforms both the context-agnostic baselines and a simple context-aware baseline. We observe that improvements are especially prominent for sentences containing ambiguous pronouns. We also show that the model induces anaphora relations. We believe that further improvements in handling anaphora, and by proxy translation, can be achieved by incorporating specialized features in the attention model. Our analysis has focused on the effect of context information on pronoun translation. Future work could also investigate whether context-aware NMT systems learn other discourse phenomena, for example whether they improve the translation of elliptical constructions, and markers of discourse relations and information structure.

Acknowledgments

We would like to thank Bonnie Webber for helpful discussions and annonymous reviewers for their comments. The authors also thank David Talbot and Yandex Machine Translation team for helpful discussions and inspiration. Ivan Titov acknowledges support of the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518). Rico Sennrich has received funding from the Swiss National Science Foundation (105212_169888).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*. San Diego.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating Discourse Phenomena in Neural Machine Translation. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. New Orleans, USA.

Marine Carpuat. 2009. One Translation Per Discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, Boulder, Colorado, pages 19–27. http://www.aclweb.org/anthology/W09-2404.

Zhengxian Gong, Min Zhang, Chew Lim Tan, and Guodong Zhou. 2012. N-gram-based tense models for statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 276–285. http://www.aclweb.org/anthology/D12-1026.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based Document-level Statistical Machine Translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 909–919. http://www.aclweb.org/anthology/D11-1084.

Christian Hardmeier. 2012. Discourse in statistical machine translation: A survey and a case study. *Discours* 11.

Christian Hardmeier and Marcello Federico. 2010. Modelling Pronominal Anaphora in Statistical Machine Translation. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*. pages 283–289.

Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, and Mauro Cettolo. 2015. Pronoun-Focused MT and Cross-Lingual Pronoun Prediction: Findings of the 2015 DiscoMT Shared Task on Pronoun Translation. In *Proceedings of the Second Workshop on Discourse in Machine Translation*. Association for Computational Linguistics, Lisbon, Portugal, pages 1–16. https://doi.org/10.18653/v1/W15-2501.

Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2013. Latent anaphora resolution for cross-lingual pronoun prediction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 380–391. http://www.aclweb.org/anthology/D13-1037.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based mt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 328–337. https://doi.org/10.3115/v1/E14-1035.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does Neural Machine Translation Benefit from Larger Context? In *arXiv:1704.05135*. ArXiv: 1704.05135.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1830–1839. https://doi.org/10.18653/v1/D17-1195.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*. Association for Computational Linguistics, Uppsala, Sweden, pages 252–261. http://www.aclweb.org/anthology/W10-1737.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. Opensubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014a. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014b. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Thomas Meyer, Andrei Popescu-Belis, Najeh Hajlaoui, and Andrea Gesmundo. 2012. Machine Translation of Labeled Discourse Connectives. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*. http://www.mt-archive.info/AMTA-2012-Meyer.pdf.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017. Validation of an automatic metric for the accuracy of pronoun translation (apt). In *Proceedings of the Third Workshop on Discourse in Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark, pages 17–25. https://doi.org/10.18653/v1/W17-4802.

Ruslan Mitkov. 1999. Introduction: Special issue on anaphora resolution in machine translation and multilingual nlp. *Machine Translation* 14(3/4):159–161. http://www.jstor.org/stable/40006919.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 57–64. https://www.aclweb.org/anthology/W05-0908.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. https://doi.org/10.18653/v1/P16-1162.

Jinsong Su, Hua Wu, Haifeng Wang, Yidong Chen, Xiaodong Shi, Huailin Dong, and Qun Liu. 2012. Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 459–468. http://www.aclweb.org/anthology/P12-1048.

Jörg Tiedemann. 2010. Context Adaptation in Statistical Machine Translation Using Models with Exponentially Decaying Cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*. Association for Computational Linguistics, Uppsala, Sweden, pages 8–15. http://www.aclweb.org/anthology/W10-2602.

Jörg Tiedemann and Yves Scherrer. 2017. Neural Machine Translation with Extended Context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark, DISCOMT'17, pages 82–92. https://doi.org/10.18653/v1/W17-4811.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. Los Angeles. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting Cross-Sentence Context for Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Denmark, Copenhagen, EMNLP'17, pages 2816–2821. https://doi.org/10.18653/v1/D17-1301.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 994–1004. https://doi.org/10.18653/v1/N16-1114.

Document Context Neural Machine Translation with Memory Networks

Sameen Maruf and **Gholamreza Haffari**
Faculty of Information Technology, Monash University, Australia
{firstname.lastname}@monash.edu

Abstract

We present a document-level neural machine translation model which takes both source and target document context into account using memory networks. We model the problem as a structured prediction problem with interdependencies among the observed and hidden variables, i.e., the source sentences and their unobserved target translations in the document. The resulting structured prediction problem is tackled with a neural translation model equipped with two memory components, one each for the source and target side, to capture the documental interdependencies. We train the model end-to-end, and propose an iterative decoding algorithm based on block coordinate descent. Experimental results of English translations from French, German, and Estonian documents show that our model is effective in exploiting both source and target document context, and statistically significantly outperforms the previous work in terms of BLEU and METEOR.

1 Introduction

Neural machine translation (NMT) has proven to be powerful (Sutskever et al., 2014; Bahdanau et al., 2015). It is on-par, and in some cases, even surpasses the traditional statistical MT (Luong et al., 2015) while enjoying more flexibility and significantly less manual effort for feature engineering. Despite their flexibility, most neural MT models translate sentences independently. Discourse phenomenon such as pronominal anaphora and lexical consistency, may depend on long-range dependency going farther than a few previous sentences, are neglected in sentence-based translation (Bawden et al., 2017).

There are only a handful of attempts to document-wide machine translation in statistical and neural MT camps. Hardmeier and Federico (2010); Gong et al. (2011); Garcia et al. (2014) propose document translation models based on statistical MT but are restrictive in the way they incorporate the document-level information and fail to gain significant improvements. More recently, there have been a few attempts to incorporate source side context into neural MT (Jean et al., 2017; Wang et al., 2017; Bawden et al., 2017); however, these works only consider a very local context including a few previous source/target sentences, ignoring the global source and target documental contexts. The latter two report deteriorated performance when using the target-side context.

In this paper, we present a document-level machine translation model which combines sentence-based NMT (Bahdanau et al., 2015) with memory networks (Sukhbaatar et al., 2015). We capture the global source and target document context with two memory components, one each for the source and target side, and incorporate it into the sentence-based NMT by changing the decoder to condition on it as the sentence translation is generated. We conduct experiments on three language pairs: French-English, German-English and Estonian-English. The experimental results and analysis demonstrate that our model is effective in exploiting both source and target document context, and statistically significantly outperforms the previous work in terms of BLEU and METEOR.

2 Background

2.1 Neural Machine Translation (NMT)

Our document NMT model is grounded on sentence-based NMT model (Bahdanau et al.,

2015) which contains an encoder to *read* the source sentence as well as an attentional decoder to *generate* the target translation.

Encoder It is a bidirectional RNN consisting of two RNNs running in opposite directions over the source sentence:

$$\overrightarrow{\boldsymbol{h}_i} = \overrightarrow{\text{RNN}}(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{E}_S[x_i]), \overleftarrow{\boldsymbol{h}}_i = \overleftarrow{\text{RNN}}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{E}_S[x_i])$$

where $\boldsymbol{E}_S[x_i]$ is embedding of the word x_i from the embedding table $\boldsymbol{E}_S$ of the source language, and $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$ are the hidden states of the forward and backward RNNs which can be based on the LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014) units. Each word in the source sentence is then represented by the concatenation of the corresponding bidirectional hidden states, $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$.

Decoder The generation of each word y_j is conditioned on all of the previously generated words $\boldsymbol{y}_{<j}$ via the state of the RNN decoder $\boldsymbol{s}_j$, and the source sentence via a *dynamic* context vector $\boldsymbol{c}_j$:

$$\begin{aligned} y_j &\sim \text{softmax}(\boldsymbol{W}_y \cdot \boldsymbol{r}_j + \boldsymbol{b}_r) \\ \boldsymbol{r}_j &= \tanh(\boldsymbol{s}_j + \boldsymbol{W}_{rc} \cdot \boldsymbol{c}_j + \boldsymbol{W}_{rj} \cdot \boldsymbol{E}_T[y_{j-1}]) \\ \boldsymbol{s}_j &= \tanh(\boldsymbol{W}_s \cdot \boldsymbol{s}_{j-1} + \boldsymbol{W}_{sj} \cdot \boldsymbol{E}_T[y_{j-1}] + \boldsymbol{W}_{sc} \cdot \boldsymbol{c}_j) \end{aligned}$$

where $\boldsymbol{E}_T[y_j]$ is embedding of the word y_j from the embedding table $\boldsymbol{E}_T$ of the target language, and $\boldsymbol{W}$ matrices and $\boldsymbol{b}_r$ vector are the parameters. The dynamic context vector $\boldsymbol{c}_j$ is computed via $\boldsymbol{c}_j = \sum_i \alpha_{ji} \boldsymbol{h}_i$, where

$$\begin{aligned} \boldsymbol{\alpha}_j &= \text{softmax}(\boldsymbol{a}_j) \\ a_{ji} &= \boldsymbol{v} \cdot \tanh(\boldsymbol{W}_{ae} \cdot \boldsymbol{h}_i + \mathbf{W}_{at} \cdot \boldsymbol{s}_{j-1}) \end{aligned}$$

This is known as the *attention* mechanism which dynamically attends to relevant parts of the source necessary for generating the next target word.

2.2 Memory Networks (MemNets)

Memory Networks (Weston et al., 2015) are a class of neural models that use external memories to perform inference based on long-range dependencies. A memory is a collection of vectors $\boldsymbol{M} = \{\boldsymbol{m}_1, .., \boldsymbol{m}_K\}$ constituting the memory cells, where each cell $\boldsymbol{m}_k$ may potentially correspond to a discrete object $\boldsymbol{x}_k$. The memory is equipped with a *read* and optionally a *write* operation. Given a query vector $\boldsymbol{q}$, the output vector generated by reading from the memory is $\sum_{i=1}^{|\boldsymbol{M}|} p_i \boldsymbol{m}_i$, where p_i represents the relevance of the query to the i-th memory cell $\boldsymbol{p} =$ softmax$(\boldsymbol{q}^T \cdot \boldsymbol{M})$. For the rest of the paper, we denote the read operation by MemNet$(\boldsymbol{M}, \boldsymbol{q})$.

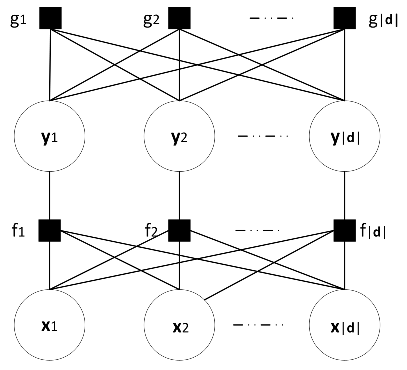


Figure 1: Factor graph for document-level MT

3 Document NMT as Structured Prediction

We formulate document-wide machine translation as a *structured* prediction problem. Given a set of sentences $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\boldsymbol{d}|}\}$ in a source document $\boldsymbol{d}$, we are interested in generating the collection of their translations $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|\boldsymbol{d}|}\}$ taking into account *interdependencies* among them imposed by the document. We achieve this by the factor graph in Figure 1 to model the probability of the target document given the source document. Our model has two types of factors:

- $f_{\boldsymbol{\theta}}(\boldsymbol{y}_t; \boldsymbol{x}_t, \boldsymbol{x}_{-t})$ to capture the interdependencies between the translation $\boldsymbol{y}_t$, the corresponding source sentence $\boldsymbol{x}_t$ and all the other sentences in the source document $\boldsymbol{x}_{-t}$, and
- $g_{\boldsymbol{\theta}}(\boldsymbol{y}_t; \boldsymbol{y}_{-t})$ to capture the interdependencies between the translation $\boldsymbol{y}_t$ and all the other translations in the document $\boldsymbol{y}_{-t}$.

Hence, the probability of a document translation given the source document is

$$\begin{aligned} &P(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|\boldsymbol{d}|} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\boldsymbol{d}|}) \propto \\ &\exp\Big(\sum_t f_{\boldsymbol{\theta}}(\boldsymbol{y}_t; \boldsymbol{x}_t, \boldsymbol{x}_{-t}) + g_{\boldsymbol{\theta}}(\boldsymbol{y}_t; \boldsymbol{y}_{-t})\Big). \end{aligned}$$

The factors $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\theta}}$ are realised by neural architectures whose parameters are collectively denoted by $\boldsymbol{\theta}$.

Training It is challenging to train the model parameters by maximising the (regularised) likelihood since computing the partition function is hard. This is due to the enormity of factors

$g_{\boldsymbol{\theta}}(\boldsymbol{y}_t; \boldsymbol{y}_{-t})$ over a large number of translation variables $\boldsymbol{y}_t$'s (i.e., the number of sentences in the document) as well as their unbounded domain (i.e., all sentences in the target language). Thus, we resort to maximising the *pseudo-likelihood* (Besag, 1975) for training the parameters:

$$\arg\max_{\boldsymbol{\theta}} \prod_{\boldsymbol{d}\in\mathcal{D}} \prod_{t=1}^{|\boldsymbol{d}|} P_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t}) \qquad (1)$$

where $\mathcal{D}$ is the set of bilingual training documents, and $|\boldsymbol{d}|$ denotes the number of (bilingual) sentences in the document $\boldsymbol{d} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\boldsymbol{d}|}$. We directly model the document-conditioned NMT model $P_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t})$ using a neural architecture which subsumes both the $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\theta}}$ factors (covered in the next section).

Decoding To generate the best translation for a document according to our model, we need to solve the following optimisation problem:

$$\arg\max_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{|\boldsymbol{d}|}} \prod_{t=1}^{|\boldsymbol{d}|} P_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t})$$

which is hard (due to similar reasons as mentioned earlier). We hence resort to a block coordinate descent optimisation algorithm. More specifically, we initialise the translation of each sentence using the base neural MT model $P(\boldsymbol{y}_t|\boldsymbol{x}_t)$. We then repeatedly visit each sentence in the document, and update its translation using our document-context dependent NMT model $P(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t})$ while the translations of other sentences are kept fixed.

4 Context Dependent NMT with MemNets

We augment the sentence-level attentional NMT model by incorporating the document context (both source and target) using memory networks when generating the translation of a sentence, as shown in Figure 2.

Our model generates the target translation word-by-word from left to right, similar to the vanilla attentional neural translation model. However, it conditions the generation of a target word not only on the previously generated words and the current source sentence (as in the vanilla NMT model), but also on all the other source sentences of the document and their translations. That is, the generation process is as follows:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t}) = \prod_{j=1}^{|\boldsymbol{y}_t|} P_{\boldsymbol{\theta}}(y_{t,j}|\boldsymbol{y}_{t,<j}, \boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t}) \qquad (2)$$

where $y_{t,j}$ is the j-th word of the t-th target sentence, $\boldsymbol{y}_{t,<j}$ are the previously generated words, and $\boldsymbol{x}_{-t}$ and $\boldsymbol{y}_{-t}$ are as introduced previously.

Our model represents the source and target document contexts as external memories, and *attends* to relevant parts of these external memories when generating the translation of a sentence. Let $\boldsymbol{M}[\boldsymbol{x}_{-t}]$ and $\boldsymbol{M}[\boldsymbol{y}_{-t}]$ denote external memories representing the source and target document context, respectively. These contain memory cells corresponding to all sentences in the document except the t-th sentence (described shortly). Let $\boldsymbol{h}_t$ and $\boldsymbol{s}_t$ be representations of the t-th source sentence and its current translation, from the encoder and decoder respectively. We make use of $\boldsymbol{h}_t$ as the query to get the relevant *context* from the source external memory:

$$\boldsymbol{c}_t^{src} = \text{MemNet}(\boldsymbol{M}[\boldsymbol{x}_{-t}], \boldsymbol{h}_t)$$

Furthermore, for the t-th sentence, we get the relevant information from the target context:

$$\boldsymbol{c}_t^{trg} = \text{MemNet}(\boldsymbol{M}[\boldsymbol{y}_{-t}], \boldsymbol{s}_t + \boldsymbol{W}_{at} \cdot \boldsymbol{h}_t)$$

where the query consists of the representation of the translation $\boldsymbol{s}_t$ from the decoder endowed with that of the source sentence $\boldsymbol{h}_t$ from the encoder to make the query robust to potential noises in the current translation and circumvent error propagation, and $\boldsymbol{W}_{at}$ projects the source representation into the hidden state space.

Now that we have representations of the relevant source and target document contexts, Eq. 2 can be re-written as:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{-t}, \boldsymbol{x}_{-t}) = \prod_{j=1}^{|\boldsymbol{y}_t|} P_{\boldsymbol{\theta}}(y_{t,j}|\boldsymbol{y}_{t,<j}, \boldsymbol{x}_t, \boldsymbol{c}_t^{trg}, \boldsymbol{c}_t^{src}) \qquad (3)$$

More specifically, the memory contexts $\boldsymbol{c}_t^{src}$ and $\boldsymbol{c}_t^{trg}$ are incorporated into the NMT decoder as:

- **Memory-to-Context** in which the memory contexts are incorporated when computing the next decoder hidden state:

$$\mathbf{s}_{t,j} = \tanh(\boldsymbol{W}_s \cdot \mathbf{s}_{t,j-1} + \boldsymbol{W}_{sj} \cdot \boldsymbol{E}_T[y_{t,j}] + \boldsymbol{W}_{sc} \cdot \boldsymbol{c}_{t,j} + \boldsymbol{W}_{sm} \cdot \boldsymbol{c}_t^{src} + \boldsymbol{W}_{st} \cdot \boldsymbol{c}_t^{trg})$$

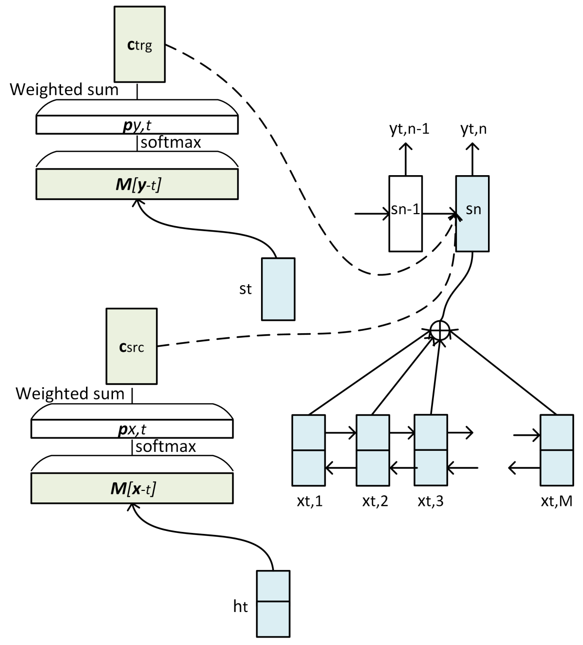


Figure 2: Our Memory-to-Context document-NMT model consisting of sentence-based NMT model with source and target external memories.

- **Memory-to-Output** in which the memory contexts are incorporated in the output layer:

$$y_{t,j} \sim \text{softmax}(\boldsymbol{W}_y \cdot \mathbf{r}_{t,j} + \boldsymbol{W}_{ym} \cdot \boldsymbol{c}_t^{src} + \boldsymbol{W}_{yt} \cdot \boldsymbol{c}_t^{trg} + \mathbf{b}_r)$$

where $\boldsymbol{W}_{sm}$, $\boldsymbol{W}_{st}$, $\boldsymbol{W}_{ym}$, and $\boldsymbol{W}_{yt}$ are the new parameter matrices. We use only the source, only the target, or both external memories as the additional conditioning contexts. Furthermore, we use either the Memory-to-Context or Memory-to-Output architectures for incorporating the document contexts. In the experiments, we will explore these different options to investigate the most effective combination. We now turn our attention to the construction of the external memories for the source and target sides of a document.

The Source Memory We make use of a hierarchical 2-level RNN architecture to construct the external memory of the source document. More specifically, we pass each sentence of the document through a sentence-level bidirectional RNN to get the representation of the sentence (by concatenating the last hidden states of the forward and backward RNNs). We then pass the sentence representations through a document-level bidirectional RNN to propagate sentences' information across the document. We take the hidden states of the document-level bidirectional RNNs as the memory cells of the source external memory.

The source external memory is built once for each minibatch, and does not change throughout the document translation. To be able to fit the computational graph of the document NMT model within GPU memory limits, we pre-train the sentence-level bidirectional RNN using the language modelling training objective. However, the document-level bidirectional RNN is trained together with other parameters of the document NMT model by back-propagating the document translation training objective.

The Target Memory The memory cells of the target external memory represent the current translations of the document. Recall from the previous section that we use coordinate descent iteratively to update these translations. Let $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|\boldsymbol{d}|}\}$ be the current translations, and let $\{\boldsymbol{s}_{|\boldsymbol{y}_1|}, \ldots, \boldsymbol{s}_{|\boldsymbol{y}_{|\boldsymbol{d}|}|}\}$ be the last states of the decoder when these translations were generated. We use these last decoder states as the cells of the external target memory. We could make use of hierarchical sentence-document RNNs to transform the document translations into memory cells (similar to what we do for the source memory); however, it would have been computationally expensive and may have resulted in error propagation. We will show in the experiments that our efficient target memory construction is indeed effective.

5 Experiments and Analysis

Datasets. We conducted experiments on three language pairs: French-English, German-English and Estonian-English. Table 1 shows the statistics of the datasets used in our experiments. The French-English dataset is based on the TED Talks corpus[1] (Cettolo et al., 2012) where each talk is considered a document. The Estonian-English data comes from the Europarl v7 corpus[2] (Koehn, 2005). Following Smith et al. (2013), we split the speeches based on the `SPEAKER` tag and treat them as documents. The French-English and Estonian-English corpora were randomly split into train/dev/test sets. For German-English, we use the News Commentary v9 corpus[3] for training, `news-dev2009` for development,

[1]https://wit3.fbk.eu/
[2]http://www.statmt.org/europarl/
[3]http://statmt.org/wmt14/news-commentary-v9-by-document.tgz

	# docs	# sents	doc len	src/tgt vocab
Fr-En	10/1.2/1.5	123/15/19	123/128/124	25.1/21
Et-En	150/10/18	209/14/25	14/14/14	48.6/24.9
De-En	49/.9/1.1/1.6	191/2/3/3	39/23/27/19	45.1/34.7

Table 1: Training/dev/test corpora statistics: number of documents ($\times$100) and sentences ($\times$1000), average document length (in sentences) and source/target vocabulary size ($\times$1000). For De-En, we report statistics of the two test sets `news-test2011` and `news-test2016`.

and `news-test2011` and `news-test2016` as the test sets. The news-commentary corpus has document boundaries already provided.

We pre-processed all corpora to remove very short documents and those with missing translations. Out-of-vocabulary and rare words (frequency less than 5) are replaced by the `<UNK>` token, following Cohn et al. (2016).[4]

Evaluation Measures We use BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007) scores to measure the quality of the generated translations. We use bootstrap resampling (Clark et al., 2011) to measure statistical significance, $p < 0.05$, comparing to the baselines.

Implementation and Hyperparameters We implement our document-level neural machine translation model in C++ using the DyNet library (Neubig et al., 2017), on top of the basic sentence-level NMT implementation in `mantis` (Cohn et al., 2016). For the source memory, the sentence and document-level bidirectional RNNs use LSTM and GRU units, respectively. The translation model uses GRU units for the bidirectional RNN encoder and the 2-layer RNN decoder. GRUs are used instead of LSTMs to reduce the number of parameters in the main model. The RNN hidden dimensions and word embedding sizes are set to 512 in the translation and memory components, and the alignment dimension is set to 256 in the translation model.

Training We use a stage-wise method to train the variants of our document context NMT model. Firstly, we pre-train the Memory-to-Context/Memory-to-Output models, setting their *readings* from the source and target memories to the zero vector. This effectively learns parameters associated with the underlying sentence-based NMT model, which is then used as initialisation when training *all* parameters in the second stage (including the ones from the first stage). For the first stage, we make use of stochastic gradient descent (SGD)[5] with initial learning rate of 0.1 and a decay factor of 0.5 after the fourth epoch for a total of ten epochs. The convergence occurs in 6-8 epochs. For the second stage, we use SGD with an initial learning rate of 0.08 and a decay factor of 0.9 after the first epoch for a total of 15 epochs[6]. The best model is picked based on the dev-set perplexity. To avoid overfitting, we employ dropout with the rate 0.2 for the single memory model. For the dual memory model, we set dropout for Document RNN to 0.2 and for the encoder and decoder to 0.5. Mini-batching is used in both stages to speed up training. For the largest dataset, the document NMT model takes about 4.5 hours per epoch to train on a single P100 GPU, while the sentence-level model takes about 3 hours per epoch for the same settings.

When training the document NMT model in the second stage, we need the target memory. One option would be to use the ground truth translations for building the memory. However, this may result in inferior training, since at the test time, the decoder iteratively updates the translation of sentences based on the noisy translations of other sentences (accessed via the target memory). Hence, while training the document NMT model, we construct the target memory from the translations *generated* by the pre-trained sentence-level model[7]. This effectively exposes the model to its potential test-time mistakes during the training time, resulting in more robust learned parameters.

5.1 Main Results

We have three variants of our model, using: (i) only the source memory (*S-NMT+src mem*), (ii) only the target memory (*S-NMT+trg mem*), or

[4]We do not split words into subwords using BPE (Sennrich et al., 2016) as that increases sentence lengths resulting in removing long documents due to GPU memory limitations, which would heavily reduce the amount of data that we have.

[5]In our initial experiments, we found SGD to be more effective than Adam/Adagrad; an observation also made by Bahar et al. (2017).

[6]For the document NMT model training, we did some preliminary experiments using different learning rates and used the scheme which converged to the best perplexity in the least number of epochs while for sentence-level training we follow Cohn et al. (2016).

[7]We report results for two-pass decoding, i.e., we only update the translations once using the initial translations generated from the base model. We tried multiple passes of decoding at test-time but it was not helpful.

	Memory-to-Context								Memory-to-Output							
	BLEU				METEOR				BLEU				METEOR			
	Fr→En	De→En		Et→En	Fr→En	De→En		Et→En	Fr→En	De→En		Et→En	Fr→En	De→En		Et→En
		NC-11	NC-16			NC-11	NC-16			NC-11	NC-16			NC-11	NC-16	
S-NMT	20.85	5.24	9.18	20.42	23.27	10.90	14.35	24.65	20.85	5.24	9.18	20.42	23.27	10.90	14.35	24.65
+src	21.91†	6.26†	10.20†	22.10†	24.04†	11.52†	15.45†	25.92†	**21.80**†	6.10†	9.98†	21.50†	23.99†	11.53†	15.29†	25.44†
+trg	21.74†	6.24†	9.97†	21.94†	23.98†	11.58†	15.32†	25.89†	21.76†	**6.31**†	10.04†	21.82†	24.06†	**12.10**†	15.75†	25.93†
+both	**22.00**†	**6.57**†	**10.54**†	**22.32**†	**24.40**†	**12.24**†	**16.18**†	**26.34**†	21.77†	6.20†	**10.23**†	**22.20**†	**24.27**†	11.84†	**15.82**†	**26.10**†

Table 2: BLEU and METEOR scores for the sentence-level baseline (S-NMT) vs. variants of our Document NMT model. **bold**: Best performance, †: Statistically significantly better than the baseline.

Lang. Pair	Memory-to-Context Fr→En	De→En	Et→En	Memory-to-Output Fr→En	De→En	Et→En
S-NMT	42.5	66.8	58.4	42.5	66.8	58.5
+src mem	48.8	73.1	64.8	68.7	107.1	88.7
+trg mem	43.8	68.1	59.8	53.8	85.1	71.8
+both mems	50.1	74.4	66.1	80	125.4	102

Table 3: Number of model parameters (millions).

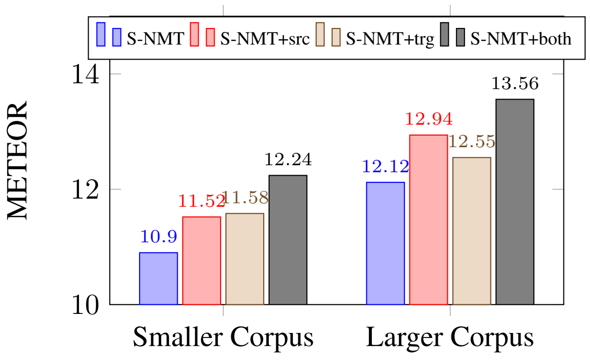
(iii) both the source and target memories (*S-NMT+both mems*). We compare these variants against the standard sentence-level NMT model (*S-NMT*). We also compare the source memory variants of our model to the local context-NMT models[8] of Jean et al. (2017) and Wang et al. (2017), which use a few previous source sentences as context, added to the decoder hidden state (similar to our Memory-to-Context model).

Memory-to-Context We consistently observe +1.15/+1.13 BLEU/METEOR score improvements across the three language pairs upon comparing our best model to *S-NMT* (see Table 2). Overall, our document NMT model with both memories has been the most effective variant for all of the three language pairs.

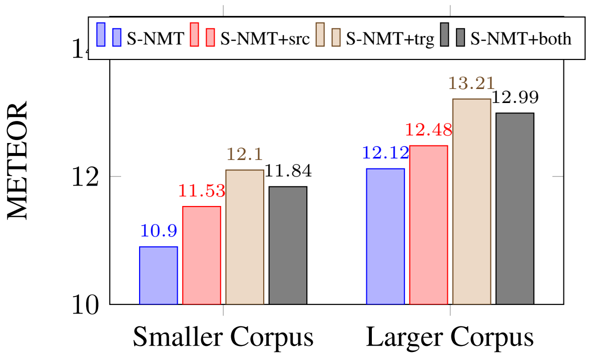
We further experiment to train the target memory variants using *gold* translations instead of the generated ones for German-English. This led to −0.16 and −0.25 decrease[9] in the BLEU scores for the target-only and both-memory variants, which confirms the intuition of constructing the target memory by exposing the model to its noises during training time.

Memory-to-Output From Table 2, we consistently see +.95/+1.00 BLEU/METEOR improvements between the best variants of our model and the sentence-level baseline across the three language pairs. For French→English, all variants of document NMT model show comparable performance when using BLEU; however, when evaluated using METEOR, the dual memory model is the best. For German→English, the target memory variants give comparable results, whereas for Estonian→English, the dual memory variant proves to be the best. Overall, the Memory-to-Context model variants perform better than their Memory-to-Output counterparts. We attribute this to the large number of parameters in the latter architecture (Table 3) and limited amount of data.

We further experiment with more data for train-



(a) Memory-to-Context model



(b) Memory-to-Output model

Figure 3: METEOR scores on De→En (NC-11) while training S-NMT with smaller vs. larger corpus.

[8]We implemented and trained the baseline local context models using the same hyperparameters and training procedure that we used for training our memory models.

[9]Latter is statistically significant decrease w.r.t. the both memory model trained on generated target translations.

	BLEU				METEOR			
	Fr→En	De→En		Et→En	Fr→En	De→En		Et→En
		NC-11	NC-16			NC-11	NC-16	
Jean et al. (2017)	21.95	6.04	10.26	21.67	24.10	11.61	15.56	25.77
Wang et al. (2017)	21.87	5.49	10.14	22.06	24.13	11.05	15.20	26.00
S-NMT	20.85	5.24	9.18	20.42	23.27	10.90	14.35	24.65
+*src mem*	21.91†	6.26$^{\clubsuit}$	10.20	22.10$^{\spadesuit}$	24.04†	11.52$^{\clubsuit}$	15.45$^{\clubsuit}$	25.92$^{\spadesuit}$
+*both mems*	**22.00**†	**6.57**$^{\diamond}$	**10.54**$^{\clubsuit}$	**22.32**$^{\diamond}$	**24.40**$^{\diamond}$	**12.24**$^{\diamond}$	**16.18**$^{\diamond}$	**26.34**$^{\diamond}$

Table 4: Our Memory-to-Context Source Memory NMT variants vs. S-NMT and Source context NMT baselines. **bold**: Best performance, †, ♠, ♣, ◇: Statistically significantly better than only S-NMT, S-NMT & Jean et al. (2017), S-NMT & Wang et al. (2017), all baselines, respectively.

	BLEU-1			
	Fr→En	De→En		Et→En
		NC-11	NC-16	
Jean et al. (2017)	52.8	30.6	39.2	51.9
Wang et al. (2017)	52.6	28.2	38.3	52.3
S-NMT	51.4	28.7	36.9	50.4
+*src mem*	53.0	30.5	39.1	52.6
+*both mems*	**53.5**	**33.1**	**41.3**	**53.2**

Table 5: Unigram BLEU for our Memory-to-Context Document NMT models vs. S-NMT and Source context NMT baselines. **bold**: Best performance.

ing the sentence-based NMT to investigate the extent to which document context is useful in this setting. We randomly choose an additional 300K German-English sentence pairs from WMT'14 data to train the base NMT model in stage 1. In stage 2, we use the same document corpus as before to train the document-level models. As seen from Figure 3, the document MT variants still benefit from the document context even when the base model is trained on a larger bilingual corpus. For the Memory-to-Context model, we see massive improvements of +0.72 and +1.44 METEOR scores for the source memory and dual memory model respectively, when compared to the baseline. On the other hand, for the Memory-to-Output model, the target memory model's METEOR score increases significantly by +1.09 compared to the baseline, slightly differing from the corresponding model using the smaller corpus (+1.2).

Local Source Context Models Table 4 shows comparison of our Memory-to-Context model variants to local source context-NMT models (Jean et al., 2017; Wang et al., 2017). For French→English, our source memory model is comparable to both baselines. For German→English, our *S-NMT+src mem* model is comparable to Jean et al. (2017) but outperforms Wang et al. (2017) for one test set according to BLEU, and for both test sets according to METEOR. For Estonian→English, our model outperforms Jean et al. (2017). Our global source context model has only surface-level sentence information, and is oblivious to the individual words in the context since we do an offline training to get the sentence representations (as previously mentioned). However, the other two context baselines have access to that information, yet our model's performance is either better or quite close to those models. We also look into the unigram BLEU scores to see how much our global source memory variants lead to improvement at the word-level. From Table 5, it can be seen that our model's performance is better than the baselines for majority of the cases. The *S-NMT+both mems* model gives the best results for all three language pairs, showing that leveraging both source and target document context is indeed beneficial for improving MT performance.

5.2 Analysis

Using Global/Local Target Context We first investigate whether using a local target context would have been equally sufficient in comparison to our global target memory model for the three datasets. We condition the decoder on the previous target sentence representation (obtained from the last hidden state of the decoder) by adding it as an additional input to all decoder states (*PrevTrg*) similar to our Memory-to-Context model. From Table 6, we observe that for French→English and Estonian→English, using all sentences in the target context or just the previous target sentence gives comparable results. We may attribute this to these specific datasets, that is documents from TED talks or European Parliament Proceedings may depend more on the local than on the global context. However, for German→English (NC-11), the target memory model performs the best show-

	BLEU			METEOR		
Lang. Pair	Fr→En	De→En	Et→En	Fr→En	De→En	Et→En
S-NMT	20.85	5.24	20.42	23.27	10.90	24.65
+*prev trg*	**21.75**	5.93	**22.08**	**24.03**	11.40	**25.94**
+*trg mem*	21.74	**6.24**	21.94	23.98	**11.58**	25.89

Table 6: Analysis of target context model.

ing that for documents with richer context (e.g. news articles) we do need the global target document context to improve MT performance.

Output Analysis To better understand the dual memory model, we look at the first sentence example in Table 7. It can be seen that the source sentence has the noun "Qimonda" but the sentence-level NMT model fails to attend to it when generating the translation. On the other hand, the single memory models are better in delivering some, if not all, of the underlying information in the source sentence but the dual memory model's translation quality surpasses them. This is because the word "Qimonda" was being repeated in this specific document, providing a strong contextual signal to our global document context model while the local context model by Wang et al. (2017) is still unable to correctly translate the noun even when it has access to the word-level information of previous sentences.

We resort to manual evaluation as there is no standard metric which evaluates document-level discourse information like consistency or pronominal anaphora. By manual inspection, we observe that our models can identify nouns in the source sentence to resolve coreferent pronouns, as shown in the second example of Table 7. Here the topic of the sentence is "*the country under the dictatorship of Lukashenko*" and our target and dual memory models are able to generate the appropriate pronoun/determiner as well as accurately translate the word '*diktatuur*', hence producing much better translation as compared to both baselines. Apart from these improvements, our models are better in improving the readability of sentences by generating more context appropriate grammatical structures such as verbs and adverbs.

Furthermore, to validate that our model improves the consistency of translations, we look at five documents (roughly 70 sentences) from the test set of Estonian-English, each of which had a word being repeated in the gold translation. Our model is able to resolve the consistency in 22 out of 32 cases as compared to the sentence-based model which only accurately translates 16 of those. Following Wang et al. (2017), we also investigate the extent to which our model can correct errors made by the baseline system. We randomly choose five documents from the test set. Out of the 20 words/phrases which were incorrectly translated by the sentence-based model, our model corrects 85% of them while also generating 10% new errors.

Source	qimonda täidab lissaboni strateegia eesmärke.
Target	qimonda meets the objectives of the lisbon strategy.
S-NMT	`<UNK>` is the objectives of the lisbon strategy.
+Src Mem	the millennium development goals are fulfilling the millennium goals of the lisbon strategy.
+Trg Mem	in writing. - (ro) the lisbon strategy is fulfilling the objectives of the lisbon strategy.
+Both Mems	qimonda fulfils the aims of the lisbon strategy.
Wang et al. (2017)	`<UNK>` fulfils the objectives of the lisbon strategy.
Source	... et riigis kehtib endiselt lukašenka diktatuur, mis rikub inim- ning etnilise vähemuse õigusi.
Target	... this country is still under the dictatorship of lukashenko, breaching human rights and the rights of ethnic minorities.
S-NMT	... the country still remains in a position of lukashenko to violate human rights and ethnic minorities.
+Src Mem	... the country still applies to the brutal dictatorship of human and ethnic minority rights.
+Trg Mem	... the country still keeps the `<UNK>` dictatorship that violates human rights and ethnic rights.
+Both Mems	... the country still persists in lukashenko's dictatorship that violate human rights and ethnic minority rights.
Wang et al. (2017)	... there is still a regime in the country that is violating the rights of human and ethnic minority in the country.

Table 7: Example Et→En sentence translations (Memory-to-Context) from two test documents.

6 Related Work

Document-level Statistical MT There have been a few SMT-based attempts to document MT, but they are either restrictive or do not lead to significant improvements. Hardmeier and Federico (2010) identify links among words in the source document using a word-dependency model to improve translation of anaphoric pronouns. Gong et al. (2011) make use of a cache-based system to save relevant information from the previously generated translations and use that to enhance document-level translation. Garcia et al. (2014) propose a two-pass approach to improve the translations already obtained by a sentence-level model.

Docent is an SMT-based document-level decoder (Hardmeier et al., 2012, 2013), which tries to modify the initial translation generated by the Moses decoder (Koehn et al., 2007) through stochastic local search and hill-climbing. Garcia et al. (2015) make use of neural-based continuous word representations to incorporate distributional semantics into Docent. In another work, Garcia et al. (2017) incorporate new word embedding features into Docent to improve the lexical consistency of translations. The proposed methods fail to yield improvements upon automatic evaluation.

Larger Context Neural MT Jean et al. (2017)

extend the vanilla attention-based neural MT model (Bahdanau et al., 2015) by conditioning the decoder on the previous sentence via attention over its words. Extending their model to consider the global source document context would be challenging due to the large size of computation graph over all the words in the source document. Wang et al. (2017) employ a 2-level hierarichal RNN to summarise three previous source sentences, which is then used as an additional input to the decoder hidden state. Bawden et al. (2017) use multi-encoder NMT models to exploit context from the previous source and target sentence. They highlight the importance of target-side context but report deteriorated BLEU scores when using it. All these works consider a very local source/target context and completely ignore the global source and target document contexts.

7 Conclusion

We have proposed a document-level neural MT model that captures global source and target document context. Our model augments the vanilla sentence-based NMT model with external memories to incorporate documental interdependencies on both source and target sides. We show statistically significant improvements of the translation quality on three language pairs. For future work, we intend to investigate models which incorporate specific discourse-level phenomena.

Acknowledgments

The authors are grateful to André Martins and the anonymous reviewers for their helpful comments and corrections. This work was supported by the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) (`www.massive.org.au`), and partially supported by a Google Faculty Award to GH and the Australian Research Council through DP160102686.

References

Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, and Hermann Ney. 2017. Empirical investigation of optimization algorithms in neural machine translation. In *Conference of the European Association for Machine Translation*, pages 13–26, Prague, Czech Republic.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2017. Evaluating discourse phenomena in neural machine translation. In *arXiv:1711.00513*.

Julian Besag. 1975. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):179–195.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation*, pages 261–268.

Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Short Papers)*, pages 176–181. Association for Computational Linguistics.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885. Association for Computational Linguistics.

Eva Martínez Garcia, Carles Creus, Cristina España-Bonet, and Lluís Màrquez. 2017. Using word embeddings to enforce document-level lexical consistency in machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108:85–96.

Eva Martínez Garcia, Cristina España-Bonet, and Lluís Màrquez. 2014. Document-level machine translation as a re-translation process. *Procesamiento del Lenguaje Natural*, 53:103–110.

Eva Martínez Garcia, Cristina España-Bonet, and Lluís Màrquez. 2015. Document-level machine translation with word vector models. In *Proceedings of the18th Conference of the European Association for Machine Translation*, pages 59–66.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 909–919. Association for Computational Linguistics.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *International Workshop on Spoken Language Translation*, pages 283–289.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1179–1190. Association for Computational Linguistics.

Christian Hardmeier, Sara Stymne, Jörg Tiedemann, and Joakim Nivre. 2013. Docent: A document-level decoder for phrase-based statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 193–198. Association for Computational Linguistics.

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

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? In *arXiv:1704.05135*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Conference Proceedings: the 10th Machine Translation Summit*, pages 79–86. AAMT.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Jason R. Smith, Herve Saint-Amand, Chris Callison-Burch, Magdalena Plamada, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2440–2448. MIT Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112. MIT Press.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2816–2821. Association for Computational Linguistics.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the International Conference on Learning Representations*.

Which Melbourne? Augmenting Geocoding with Maps

Milan Gritta, Mohammad Taher Pilehvar and Nigel Collier

Language Technology Lab
Department of Theoretical and Applied Linguistics
University of Cambridge

{mg711,mp792,nhc30}@cam.ac.uk

Abstract

The purpose of text geolocation is to associate geographic information contained in a document with a set (or sets) of coordinates, either *implicitly* by using linguistic features and/or *explicitly* by using geographic metadata combined with heuristics. We introduce a geocoder (location mention disambiguator) that achieves state-of-the-art (SOTA) results on three diverse datasets by exploiting the implicit lexical clues. Moreover, we propose a new method for systematic encoding of geographic metadata to generate two *distinct* views of the same text. To that end, we introduce the *Map Vector* (*MapVec*), a sparse representation obtained by plotting prior geographic probabilities, derived from population figures, on a World Map. We then integrate the implicit (language) and explicit (map) features to significantly improve a range of metrics. We also introduce an open-source dataset for geoparsing of news events covering global disease outbreaks and epidemics to help future evaluation in geoparsing.

1 Introduction

Geocoding[1] is a specific case of text geolocation, which aims at disambiguating place references in text. For example, *Melbourne* can refer to more than ten possible locations and a geocoder's task is to identify the place coordinates for the intended *Melbourne* in a context such as "*Melbourne* hosts one of the four annual Grand Slam tennis tournaments." This is central to the success of tasks such as indexing and searching documents by geography (Bhargava et al., 2017), geospatial analysis of social media (Buchel and Pennington, 2017), mapping of disease risk using integrated data (Hay et al., 2013), and emergency response systems (Ashktorab et al., 2014). Previous geocoding methods (Section 2) have leveraged lexical semantics to associate the implicit geographic information in natural language with coordinates. These models have achieved good results in the past. However, focusing *only* on lexical features, to the exclusion of other feature spaces such as the Cartesian Coordinate System, puts a ceiling on the amount of semantics we are able to extract from text. Our proposed solution is the *Map Vector* (*MapVec*), a sparse, geographic vector for explicit modelling of geographic distributions of location mentions. As in previous work, we use population data and geographic coordinates, observing that the most populous *Melbourne* is also the most likely to be the intended location. However, MapVec is the first instance, to our best knowledge, of the topological semantics of context locations explicitly isolated into a standardized vector representation, which can then be easily transferred to an independent task and combined with other features. MapVec is able to encode the prior geographic distribution of any number of locations into a single vector. Our extensive evaluation shows how this representation of context locations can be integrated with linguistic features to achieve a significant improvement over a SOTA lexical model. MapVec can be deployed as a standalone neural geocoder, significantly beating the population baseline, while remaining effective with simpler machine learning algorithms.

This paper's contributions are: (1) *Lexical Geocoder* outperforming existing systems by analysing only the textual context; (2) *MapVec*, a geographic representation of locations using a sparse, probabilistic vector to extract and isolate spatial features; (3) *CamCoder*, a novel geocoder

[1]Also called Toponym Resolution in related literature.

that exploits both lexical and geographic knowledge producing SOTA results across multiple datasets; and (4) *GeoVirus*, an open-source dataset for the evaluation of geoparsing (Location Recognition *and* Disambiguation) of news events covering global disease outbreaks and epidemics.

2 Background

Depending on the task objective, geocoding methodologies can be divided into two *distinct* categories: (1) *document geocoding*, which aims at locating a piece of text as a whole, for example geolocating Twitter users (Rahimi et al., 2016, 2017; Roller et al., 2012; Rahimi et al., 2015), Wikipedia articles and/or web pages (Cheng et al., 2010; Backstrom et al., 2010; Wing and Baldridge, 2011; Dredze et al., 2013; Wing and Baldridge, 2014). This is an active area of NLP research (Hulden et al., 2015; Melo and Martins, 2017, 2015; Iso et al., 2017); (2) *geocoding of place mentions*, which focuses on the disambiguation of location (named) entities i.e. this paper and (Karimzadeh et al., 2013; Tobin et al., 2010; Grover et al., 2010; DeLozier et al., 2015; Santos et al., 2015; Speriosu and Baldridge, 2013; Zhang and Gelernter, 2014). Due to the differences in evaluation and objective, the categories cannot be directly or fairly compared. Geocoding is typically the second step in *Geoparsing*. The first step, usually referred to as *Geotagging*, is a Named Entity Recognition component which extracts all location references in a given text. This phase may optionally include metonymy resolution, see (Zhang and Gelernter, 2015; Gritta et al., 2017a). The goal of geocoding is to choose the correct coordinates for a location mention from a set of candidates. Gritta et al. (2017b) provided a comprehensive survey of five recent geoparsers. The authors established an evaluation framework, with a new dataset, for their experimental analysis. We use this evaluation framework in our experiments. We briefly describe the methodology of each geocoder featured in our evaluation (names are capitalised and appear in italics) as well as survey the related work in geocoding.

Computational methods in geocoding broadly divide into rule-based, statistical and machine learning-based. *Edinburgh Geoparser* (Tobin et al., 2010; Grover et al., 2010) is a fully rule-based geocoder that uses hand-built heuristics combined with large lists from Wikipedia and the Geonames[2] gazetteer. It uses metadata (feature type, population, country code) with heuristics such as contextual information, spatial clustering and user locality to rank candidates. *GeoTxT* (Karimzadeh et al., 2013) is another rule-based geocoder with a free web service[3] for identifying locations in unstructured text and grounding them to coordinates. Disambiguation is driven by multiple heuristics and uses the administrative level (country, province, city), population size, the Levenshtein Distance of the place referenced and the candidate's name and spatial minimisation to resolve ambiguous locations. (Dredze et al., 2013) is a rule-based Twitter geocoder using only metadata (coordinates in tweets, GPS tags, user's reported location) and custom place lists for fast and simple geocoding. *CLAVIN* (Cartographic Location And Vicinity INdexer)[4] is an open-source geocoder, which offers context-based entity recognition and linking. It seems to be mostly rule-based though details of its algorithm are underspecified, short of reading the source code. Unlike the Edinburgh Parser, this geocoder seems to overly rely on population data, seemingly mirroring the behaviour of a naive population baseline. Rule-based systems can perform well though the variance in performance is high (see Table 1). *Yahoo! Placemaker* is a free web service with a proprietary geo-database and algorithm from Yahoo![5] letting anyone geoparse text in a globally-aware and language-independent manner. It is unclear how geocoding is performed, however, the inclusion of proprietary methods makes evaluation broader and more informative.

The statistical geocoder *Topocluster* (DeLozier et al., 2015) divides the world surface into a grid (0.5 x 0.5 degrees, approximately 60K tiles) and uses lexical features to model the geographic distribution of context words over this grid. Building on the work of Speriosu and Baldridge (2013), it uses a window of 15 words (our approach scales this up by more than 20 times) to perform hot spot analysis using Getis-Ord Local Statistic of individual words' association with geographic space. The classification decision was made by finding the grid square with the strongest overlap of

[2]http://www.geonames.org/
[3]http://www.geotxt.org/
[4]https://clavin.bericotechnologies.com
[5]https://developer.yahoo.com/geo/

individual geo-distributions. Hulden et al. (2015) used Kernel Density Estimation to learn the word distribution over a world grid with a resolution of 0.5 x 0.5 degrees and classified documents with Kullback-Leibler divergence or a Naive Bayes model, reminiscent of an earlier approach by Wing and Baldridge (2011). Roller et al. (2012) used the Good-Turing Frequency Estimation to learn document probability distributions over the vocabulary with Kullback-Leibler divergence as the similarity function to choose the correct bucket in the k-d tree (world representation). Iso et al. (2017) combined Gaussian Density Estimation with a CNN-model to geolocate Japanese tweets with Convolutional Mixture Density Networks.

Among the recent machine learning methods, bag-of-words representations combined with a Support Vector Machine (Melo and Martins, 2015) or Logistic Regression (Wing and Baldridge, 2014) have also achieved good results. For Twitter-based geolocation (Zhang and Gelernter, 2014), bag-of-words classifiers were successfully augmented with social network data (Jurgens et al., 2015; Rahimi et al., 2016, 2015). The machine learning-based geocoder by *Santos et al. (2015)* supplemented lexical features, represented as a bag-of-words, with an exhaustive set of manually generated geographic features and spatial heuristics such as geospatial containment and geodesic distances between entities. The ranking of locations was learned with LambdaMART (Burges, 2010). Unlike our geocoder, the addition of geographic features did not significantly improve scores, reporting: “The geo-specific features seem to have a limited impact over a strong baseline system.” Unable to obtain a codebase, their results feature in Table 1. The latest neural network approaches (Rahimi et al., 2017) with normalised bag-of-word representations have achieved SOTA scores when augmented with social network data for Twitter document (user’s concatenated tweets) geolocation (Bakerman et al., 2018).

3 Methodology

Figure 1 shows our new geocoder *CamCoder* implemented in Keras (Chollet, 2015). The lexical part of the geocoder has three inputs, from the top: Context Words (location mentions excluded), Location Mentions (context words excluded) and the Target Entity (up to 15 words long) to be

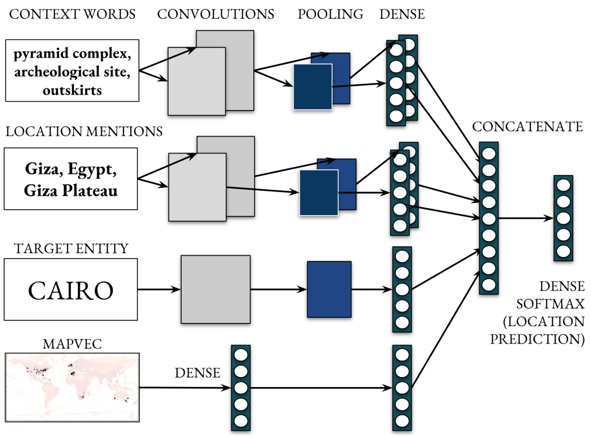

CONTEXT WORDS
CONVOLUTIONS
POOLING
DENSE
pyramid complex, archeological site, outskirts
LOCATION MENTIONS
Giza, Egypt, Giza Plateau
CONCATENATE
TARGET ENTITY
CAIRO
DENSE SOFTMAX (LOCATION PREDICTION)
MAPVEC
DENSE

Figure 1: The *CamCoder* neural architecture. It is possible to split CamCoder into a *Lexical* (top 3 inputs) model and a *MapVec* model (see Table 2).

geocoded. Consider an example disambiguation of *Cairo* in a sentence: *“The Giza pyramid complex is an archaeological site on the Giza Plateau, on the outskirts of Cairo, Egypt.”*. Here, *Cairo* is the Target Entity; *Egypt*, *Giza* and *Giza Plateau* are the Location Mentions; the rest of the sentence forms the Context Words (excluding stopwords). The *context window* is up to 200 words each side of the Target Entity, approximately an order of magnitude larger than most previous approaches.

We used separate layers, convolutional and/or dense (fully-connected), with ReLu activations (Nair and Hinton, 2010) to break up the task into smaller, focused modules in order to learn distinct lexical feature patterns, phrases and keywords for different types of inputs, concatenating only at a higher level of abstraction. Unigrams and bigrams were learned for context words and location mentions (1,000 filters of size 1 and 2 for each input), trigrams for the target entity (1,000 filters of size 3). Convolutional Neural Networks (CNNs) with Global Maximum Pooling were chosen for their position invariance (detecting location-indicative words anywhere in context) and efficient input size scaling. The dense layers have 250 units each, with a dropout layer ($p = 0.5$) to prevent overfitting. The fourth input is MapVec, the geographic vector representation of location mentions. It feeds into two dense layers with 5,000 and 1,000 units respectively. The concatenated hidden layers then get fully connected to the softmax layer. The model is optimised with RMSProp (Tieleman and Hinton, 2012). We approach geocoding as a classification task where the model predicts one of

7,823 classes (units in the softmax layer in Figure 1), each being a 2x2 degree tile representing part of the world's surface, slightly coarser than MapVec (see Section 3.1 next). The coordinates of the location candidate with the smallest *FD* (Equation 1) are the model's final output.

$$FD = error - error\,\frac{candidatePop}{maximumPop}\,Bias \quad (1)$$

FD for each candidate is computed by reducing the prediction *error* (the distance from predicted coordinates to candidate coordinates) by the value of *error* multiplied by the estimated prior probability (candidate population divided by maximum population) multiplied by the *Bias* parameter. The value of $Bias = 0.9$ was determined to be optimal for highest development data scores and is *identical for all* highly diverse test datasets. Equation 1 is designed to bias the model towards more populated locations to reflect real-world data.

3.1 The Map Vector (MapVec)

Word embeddings and/or distributional vectors encode a word's meaning in terms of its *linguistic* context. However, location (named) entities also carry explicit *topological semantic knowledge* such as a coordinate position and a population count for all places with an identical name. Until now, this knowledge was only used as part of simple disparate heuristics and manual disambiguation procedures. However, it is possible to plot this spatial data on a world map, which can then be reshaped into a 1D *feature vector*, or a *Map Vector*, the geographic representation of location mentions. MapVec is a novel standardised method for generating geographic features from text documents *beyond* lexical features. This enables a strong geocoding classification performance gain by extracting additional spatial knowledge that would normally be ignored. Geographic semantics cannot be inferred from language alone (too imprecise and incomplete). Word embeddings and distributional vectors use language/words as an *implicit* container of geographic information. Map Vector uses a low-resolution, probabilistic world map as an *explicit* container of geographic information, giving us two types of semantic features from the same text. In related papers on the generation of location representations, Rahimi et al. (2017) inverted the task of geocoding Twitter users to predict word probability from a set of coordinates. A continuous representation of a *region* was generated by using the hidden layer of the neural network. However, all locations in the same region will be assigned an identical vector, which assumes that their semantics are also identical. Another way to obtain geographic representations is by generating embeddings directly from Geonames data using heuristics-driven DeepWalk (Perozzi et al., 2014) with geodesic distances (Kejriwal and Szekely, 2017). However, to assign a vector, places must first be disambiguated (catch-22). While these generation methods are original and interesting in theory, deploying them in the real-world is infeasible, hence we invented the Map Vector.

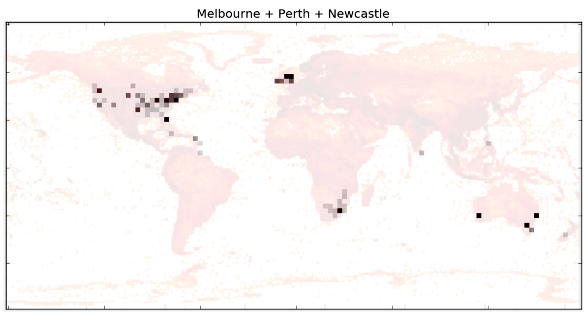


Figure 2: MapVec visualisation (before reshaping into a 1D vector) for *Melbourne*, *Perth* and *Newcastle*, showing their combined prior geographic probabilities. Darker tiles have higher probability.

MapVec initially begins as a 180x360 world map of geodesic tiles. There are other ways of representing the surface of the Earth such as using nested hierarchies (Melo and Martins, 2015) or k-dimensional trees (Roller et al., 2012), however, this is beyond the scope of this work. The 1x1 tile size, in degrees of geographic coordinates, was empirically determined to be optimal to keep MapVec's size computationally efficient while maintaining meaningful resolution. This map is then populated with the *prior geographic distribution* of each location mentioned in context (see Figure 2 for an example). We use population count to *estimate* a location's prior probability as more populous places are more likely to be mentioned in common discourse. For each location mention and for each of its ambiguous candidates, their prior probability is added to the correct tile indicating its geographic position (see Algorithm 1). Tiles that cover areas of open water (64.1%) were removed to reduce size. Finally,

```
Data: Text ← article, paragraph, tweet, etc.
Result: MapVec location(s) representation
Locs ← extractLocations(Text);
MapVec ← new array(length=23,002);
for each l in Locs do
    Cands ← queryCandidatesFromDB($l$);
    maxPop ← maxPopulationOf(Cands);
    for each c in Cands do
        prior ← populationOf($c$) / maxPop;
        i ← coordinatesToIndex($c$);
        MapVec[i] ← MapVec[i] + prior;
    end
end
m ← max(MapVec);
return MapVec / m;
```

Algorithm 1: MapVec generation. For each extracted location ***l*** in *Locs*, estimate the prior probability of each candidate ***c***. Add ***c***'s prior probability to the appropriate array position at index ***i*** representing its geographic position/tile. Finally, normalise the array (to a $[0 - 1]$ range) by dividing by the maximum value of the MapVec array.

this world map is reshaped into a one-dimensional *Map Vector* of length 23,002.

The following features of MapVec are the most salient: *Interpretability:* Word vectors typically need intrinsic (Gerz et al., 2016) and extrinsic tasks (Senel et al., 2017) to interpret their semantics. MapVec generation is a fully transparent, human readable and modifiable method. *Efficiency:* MapVec is an efficient way of embedding *any number* of locations using the same standardised vector. The alternative means creating, storing, disambiguating and computing with millions of unique location vectors. *Domain Independence:* Word vectors vary depending on the source, time, type and language of the training data and the parameters of generation. MapVec is language-independent and stable over time, domain, size of dataset since the world geography is objectively measured and changes very slowly.

3.2 Data and Preprocessing

Training data was generated from geographically annotated Wikipedia pages (dumped February 2017). Each page provided up to 30 training instances, limited to avoid bias from large pages. This resulted in collecting approximately 1.4M training instances, which were uniformly subsampled down to 400K to shorten training cycles as further increases offer diminishing returns. We used the Python-based NLP toolkit Spacy[6] (Honnibal and Johnson, 2015) for text preprocessing. All words were lowercased, lemmatised, any stopwords, dates, numbers and so on were replaced with a special token ("0"). Word vectors were initialised with pretrained word embeddings[7] (Pennington et al., 2014). We do not employ explicit feature selection as in (Bo et al., 2012), only a minimum frequency count, which was shown to work almost as well as deliberate selection (Van Laere et al., 2014). The vocabulary size was limited to the most frequent 331K words, minimum ten occurrences for words and two for location references in the 1.4M training corpus. A final training instance comprises four types of context information: Context Words (excluding location mentions, up to 2x200 words), Location Mentions (excluding context words, up to 2x200 words), Target Entity (up to 15 words) and the MapVec geographic representation of context locations. We have also checked for any overlaps between our Wikipedia-based training data and the WikToR dataset. Those examples were removed. The aforementioned 1.4M Wikipedia training corpus was once again uniformly subsampled to generate a *disjoint* development set of 400K instances. While developing our models mainly on this data, we also used small subsets of LGL (18%), GeoVirus (26%) and WikToR (9%) described in Section 4.2 to verify that development set improvements generalised to target domains.

4 Evaluation

Our evaluation compares the geocoding performance of six systems from Section 2, our geocoder (CamCoder) and the population baseline. Among these, our CNN-based model is the only neural approach. We have included all open-source/free geocoders *in working order* we were able to find and they are the most up-to-date versions. Tables 1 and 2 feature several machine learning algorithms including Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to reproduce context2vec (Melamud et al., 2016), Naive Bayes (Zhang, 2004) and Random Forest (Breiman, 2001) using three diverse datasets.

[6] https://spacy.io/
[7] https://nlp.stanford.edu/

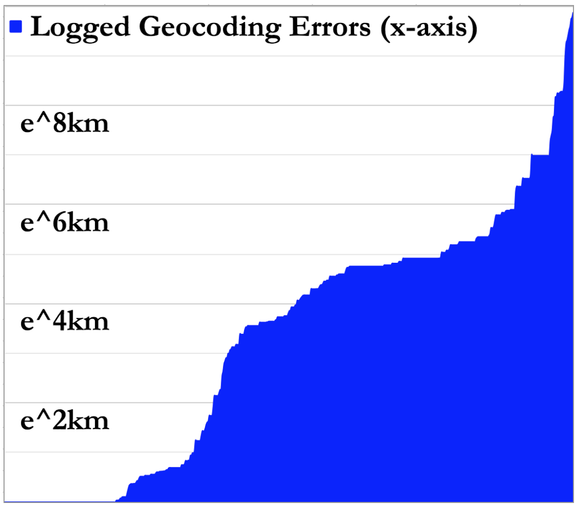


Figure 3: The AUC (range $[0-1]$) is calculated using the Trapezoidal Rule. Smaller errors mean a smaller (blue) area, which means a lower score and therefore better geocoding results.

4.1 Geocoding Metrics

We use the three standard and comprehensive metrics, each measuring an important aspect of geocoding, giving an accurate, holistic evaluation of performance. A more detailed cost-benefit analysis of geocoding metrics is available in (Karimzadeh, 2016) and (Gritta et al., 2017b). (1) *Average (Mean) Error* is the sum of all geocoding errors per dataset divided by the number of errors. It is an informative metric as it also indicates the total error but treats all errors as equivalent and is sensitive to outliers; (2) *Accuracy@161km* is the percentage of errors that are smaller than 161km (100 miles). While it is easy to interpret, giving fast and intuitive understanding of geocoding performance in percentage terms, it ignores all errors greater than 161km; (3) *Area Under the Curve* (*AUC*) is a comprehensive metric, initially introduced for geocoding in (Jurgens et al., 2015). AUC reduces the importance of large errors (1,000km+) since accuracy on successfully resolved places is more desirable. While it is not an intuitive metric, AUC is robust to outliers and measures *all* errors. A versatile geocoder should be able to maximise all three metrics.

4.2 Evaluation Datasets

News Corpus: The Local Global Corpus (LGL) by Lieberman et al. (2010) contains 588 news articles (4460 test instances), which were collected from geographically distributed newspaper sites. This is the most frequently used geocoding evaluation dataset to date. The toponyms are mostly smaller places no larger than a US state. Approximately 16% of locations in the corpus do not have any coordinates assigned; hence, we do not use those in the evaluation, which is also how the previous figures were obtained. *Wikipedia Corpus*: This corpus was deliberately designed for ambiguity hence the population heuristic is not effective. Wikipedia Toponym Retrieval (WikToR) by Gritta et al. (2017b) is a programmatically created corpus and although not necessarily representative of the real world distribution, it is a test of ambiguity for geocoders. It is also a large corpus (25,000+ examples) containing the first few paragraphs of 5,000 Wikipedia pages. High quality, free and open datasets are not readily available (GeoVirus tries to address this). The following corpora could not be included: WoTR (DeLozier et al., 2016) due to limited coverage (southern US) and domain type (historical language, the 1860s), (De Oliveira et al., 2017) contains fewer than 180 locations, GeoCorpora (Wallgrün et al., 2017) could not be retrieved in full due to deleted Twitter users/tweets, GeoText (Eisenstein et al., 2010) only allows for user geocoding, SpatialML (Mani et al., 2010) involves prohibitive costs, GeoSemCor (Buscaldi and Rosso, 2008) was annotated with WordNet senses (rather than coordinates).

4.3 GeoVirus: a New Test Dataset

We now introduce GeoVirus, an open-source test dataset for the evaluation of geoparsing of news events covering global disease outbreaks and epidemics. It was constructed from free WikiNews[8] and collected during 08/2017 - 09/2017. The dataset is suitable for the evaluation of Geotagging/Named Entity Recognition and Geocoding/Toponym Resolution. Articles were identified using the WikiNews search box and keywords such as Ebola, Bird Flu, Swine Flu, AIDS, Mad Cow Disease, West Nile Disease, etc. Off-topic articles were not included. Buildings, POIs, street names and rivers were not annotated.

Annotation Process. (1) The WikiNews contributor(s) who wrote the article annotated most, but not all location references. The first author checked those annotations and identified further references, then proceeded to extract the place name, indices of the start and end characters in

[8]`https://en.wikinews.org`

Geocoder	Area Under Curve†			Average Error‡			Accuracy@161km		
	LGL	WIK	GEO	LGL	WIK	GEO	LGL	WIK	GEO
CamCoder	**22 (18)**	**33 (37)**	**31 (32)**	7 **(5)**	**11 (9)**	**3 (3)**	**76 (83)**	**65 (57)**	**82 (80)**
Edinburgh	25 (22)	53 (58)	33 (34)	8 (8)	31 (30)	5 (4)	**76** (80)	42 (36)	78 (78)
Yahoo!	34 (35)	44 (53)	40 (44)	**6 (5)**	23 (25)	**3 (3)**	72 (75)	52 (39)	70 (65)
Population	27 (22)	68 (71)	32 **(32)**	12 (10)	45 (42)	5 **(3)**	70 (79)	22 (14)	80 **(80)**
CLAVIN	26 (20)	70 (69)	32 (33)	13 (9)	43 (39)	6 (5)	71 (80)	16 (16)	79 **(80)**
GeoTxt	29 (21)	70 (71)	33 (34)	14 (9)	47 (45)	6 (5)	68 (80)	18 (14)	79 (79)
Topocluster	38 (36)	63 (66)	NA	12 (8)	38 (35)	NA	63 (71)	26 (20)	NA
Santos et al.	NA	NA	NA	8	NA	NA	71	NA	NA

Table 1: Results on LGL, WikToR (WIK) and GeoVirus (GEO). Lower AUC and Average Error are better while higher Acc@161km is better. Figures in *brackets* are scores on identical subsets of each dataset. †Only the AUC decimal part shown. ‡Average Error rounded up to the nearest 100km.

text, assigned coordinates and the Wikipedia page URL for each location. (2) A second pass over the entire dataset by the first author to check and/or remedy annotations. (3) A computer program checked that locations were tagged correctly, checking coordinates against the Geonames Database, URL correctness, eliminating any duplicates and validating XML formatting. Places without a Wikipedia page (0.6%) were assigned Geonames coordinates. (4) The second author annotated a random 10% sample to obtain an Inter-Annotator Agreement, which was 100% for geocoding and an F-Score of 92.3 for geotagging. *GeoVirus in Numbers:* Annotated locations: 2,167, Unique: 685, Continents: 94, Number of articles: 229, Most frequent places (21% of total): US, Canada, China, California, UK, Mexico, Kenya, Africa, Australia, Indonesia; Mean location occurrence: 3.2, Total word count: 63,205.

5 Results

All tested models (except CamCoder) operate as end-to-end systems; therefore, it is not possible to perform geocoding separately. Each system geoparses its particular majority of the dataset to obtain a representative data sample, shown in Table 1 as strongly correlated scores for subsets of different sizes, with which to assess model performance. Table 1 also shows scores in *brackets* for the overlapping partition of all systems in order to compare performance on identical instances: GeoVirus 601 (26%), LGL 787 (17%) and WikToR 2,202 (9%). The geocoding difficulty based on the ambiguity of each dataset is: LGL (moderate to hard), WIK (very hard), GEO (easy to moderate). A population baseline also features in the evaluation. The baseline is conceptually simple: choose the candidate with the highest population, akin to the most frequent word sense in WSD. Table 1 shows the effectiveness of this heuristic, which is competitive with many geocoders, even *outperforming* some. However, the baseline is not effective on WikToR as the dataset was deliberately constructed as a tough ambiguity test. Table 1 shows how several geocoders mirror the behaviour of the population baseline. This simple but effective heuristic is rarely used in system comparisons, and where evaluated (Santos et al., 2015; Leidner, 2008), it is inconsistent with expected figures (due to unpublished resources, we are unable to investigate).

We note that no single computational paradigm dominates Table 1. The rule-based (Edinburgh, GeoTxt, CLAVIN), statistical (Topocluster), machine learning (CamCoder, Santos) and other (Yahoo!, Population) geocoders occupy different ranks across the three datasets. Due to space constraints, Table 1 does not show figures for another type of scenario we tested, a shorter lexical context, using 200 words instead of the standard 400. CamCoder proved to be robust to reduced context, with only a small performance decline. Using the same format as Table 1, AUC errors for LGL increased from 22 (18) to 23 (19), WIK from 33 (37) to 37 (40) and GEO remained the same at 31 (32). This means that reducing model input size to save computational resources would still deliver accurate results. Our CNN-based lexical model performs at SOTA levels (Table 2) proving the effectiveness of linguistic features while being

Geocoder	System configuration			Dataset			Average
	Language Features	+	MapVec Features	LGL	WIK	GEO	
CamCoder	CNN		MLP	**0.22**	**0.33**	**0.31**	**0.29**
Lexical Only	CNN		–	0.23	0.39	0.33	0.32
MapVec Only	–		MLP	0.25	0.41	0.32	0.33
Context2vec[†]	LSTM		MLP	0.24	0.38	0.33	0.32
Context2vec	LSTM		–	0.27	0.47	0.39	0.38
Random Forest	MapVec features only, no lexical input			0.26	0.36	0.33	0.32
Naive Bayes	MapVec features only, no lexical input			0.28	0.56	0.36	0.40
Population	–		–	0.27	0.68	0.32	0.42

Table 2: AUC scores for *CamCoder* and its *Lexical* and *MapVec* components (model ablation). Lower AUC scores are better. [†]Standard context2vec model augmented with MapVec representation.

the outstanding geocoder on the highly ambiguous WikToR data. The Multi-Layer Perceptron (MLP) model using only MapVec with no lexical features is almost as effective but more importantly, it is significantly better than the population baseline (Table 2). This is because the Map Vector benefits from wide contextual awareness, encoded in Algorithm 1, while a simple population baseline does not. When we combined the lexical *and* geographic feature spaces in one model (CamCoder[9]), we observed a substantial increase in the SOTA scores. We have also reproduced the *context2vec* model to obtain a continuous context representation using bidirectional LSTMs to encode lexical features, denoted as LSTM[10] in Table 2. This enabled us to test the effect of integrating MapVec into another deep learning model as opposed to CNNs. Supplemented with MapVec, we observed a significant improvement, demonstrating how enriching various neural models with a geographic vector representation boosts classification results.

Deep learning is the dominant paradigm in our experiments. However, it is important that MapVec is still effective with simpler machine learning algorithms. To that end, we have evaluated it with the Random Forest *without* using any lexical features. This model was well suited to the geocoding task despite training with only half of the 400K training data (due to memory constraints, partial fit is unavailable for batch training in SciKit Learn). Scores were on par with more sophisticated systems. The Naive Bayes was less effective with MapVec though still somewhat viable as a geocoder given the lack of lexical features and a naive algorithm, narrowly beating population. GeoVirus scores remain highly competitive across most geocoders. This is due to the nature of the dataset; locations skewed towards their dominant "senses" simulating ideal geocoding conditions, enabling high accuracy for the population baseline. GeoVirus alone may not serve as the best scenario to assess a geocoder's performance, however, it is nevertheless important and valuable to determine behaviour in a standard environment. For example, GeoVirus helped us diagnose Yahoo! Placemaker's lower accuracy in what should be an easy test for a geocoder. The figures show that while the average error is low, the accuracy@161km is noticeably lower than most systems. When coupled with other complementary datasets such as LGL and WikToR, it facilitates a comprehensive assessment of geocoding behaviour in many types of scenarios, exposing potential domain dependence. We note that GeoVirus has a dual function, NER (not evaluated but useful for future work) and Geocoding. We made all of our resources freely available[11] for full reproducibility (Goodman et al., 2016).

5.1 Discussion and Errors

The Pearson correlation coefficient of the target entity *ambiguity* and the *error size* was only $r \approx 0.2$ suggesting that CamCoder's geocoding errors do not simply rise with location ambiguity. Errors were also not correlated ($r \approx 0.0$) with population size with all types of locations geocoded to various degrees of accuracy. All error curves follow

[9]Single model settings/parameters for all tests.

[10]`https://keras.io/layers/recurrent/`

[11]`https://github.com/milangritta/`

a power law distribution with between 89% and 96% of errors less than 1500km, the rest rapidly increasing into thousands of kilometers. Errors also appear to be uniformly geographically distributed across the world. The strong lexical component shown in Table 2 is reflected by the lack of a relationship between *error size* and the *number of locations* found in the context. The *number of total words* in context is also independent of geocoding accuracy. This suggests that CamCoder learns strong linguistic cues beyond simple association of place names with the target entity and is able to cope with flexible-sized contexts. A CNN Geocoder would expect to perform well for the following reasons: Our context window is 400 words rather than 10-40 words as in previous approaches. The model learns 1,000 feature maps per input and per feature type, tracking 5,000 different word patterns (unigrams, bigrams and trigrams), a significant text processing capability. The lexical model also takes advantage of our own 50-dimensional word embeddings, tuned on geographic Wikipedia pages only, allowing for greater generalisation than bag-of-unigrams models; and the large training/development datasets (400K each), optimising geocoding over a diverse global set of places allowing our model to generalise to unseen instances. We note that MapVec generation is sensitive to NER performance with higher F-Scores leading to better quality of the geographic vector representation(s). Precision errors can introduce noise while recall errors may withhold important locations. The average F-Score for the featured geoparsers is $F \approx 0.7$ (standard deviation ≈ 0.1). Spacy's NER performance over the three datasets is also $F \approx 0.7$ with a similar variation between datasets. In order to further interpret scores in Tables 1 and 2, with respect to maximising geocoding performance, we briefly discuss the *Oracle* score. Oracle is the geocoding performance upper bound given by the Geonames data, i.e. the *highest possible score(s)* using Geonames coordinates as the geocoding output. In other words, it quantifies the *minimum error* for each dataset given the *perfect* location disambiguation. This means it quantifies the difference between "gold standard" coordinates and the coordinates in the Geonames database. The following are the Oracle scores for LGL (*AUC=0.04, a@161km=99*) annotated with Geonames, WikToR (*AUC=0.14, a@161km=92*) and GeoVirus (*AUC=0.27, a@161km=88*), which are annotated with Wikipedia data. Subtracting the Oracle score from a geocoder's score quantifies the scope of its *theoretical future improvement*, given a particular database/gazetteer.

6 Conclusions and Future Work

Geocoding methods commonly employ lexical features, which have proved to be very effective. Our lexical model was the best language-only geocoder in extensive tests. It is possible, however, to go *beyond* lexical semantics. Locations also have a rich topological meaning, which has not yet been successfully isolated and deployed. We need a means of extracting and encoding this additional knowledge. To that end, we introduced MapVec, an algorithm and a container for encoding context locations in geodesic vector space. We showed how CamCoder, using lexical *and* MapVec features, outperformed both approaches, achieving a new SOTA. MapVec remains effective with various machine learning frameworks (Random Forest, CNN and MLP) and substantially improves accuracy when combined with other neural models (LSTMs). Finally, we introduced GeoVirus, an open-source dataset that helps facilitate geoparsing evaluation across more diverse domains with different lexical-geographic distributions (Flatow et al., 2015; Dredze et al., 2016). Tasks that could benefit from our methods include social media placing tasks (Choi et al., 2014), inferring user location on Twitter (Zheng et al., 2017), geolocation of images based on descriptions (Serdyukov et al., 2009) and detecting/analyzing incidents from social media (Berlingerio et al., 2013). Future work may see our methods applied to *document* geolocation to assess the effectiveness of scaling geodesic vectors from paragraphs to entire documents.

Acknowledgements

We gratefully acknowledge the funding support of the Natural Environment Research Council (NERC) PhD Studentship NE/M009009/1 (Milan Gritta, DREAM CDT), EPSRC (Nigel Collier) Grant Number EP/M005089/1 and MRC (Mohammad Taher Pilehvar) Grant Number MR/M025160/1 for PheneBank. We also gratefully acknowledge NVIDIA Corporation's donation of the Titan Xp GPU used for this research.

References

Zahra Ashktorab, Christopher Brown, Manojit Nandi, and Aron Culotta. 2014. Tweedr: Mining twitter to inform disaster response. In *ISCRAM*.

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 61–70.

Jordan Bakerman, Karl Pazdernik, Alyson Wilson, Geoffrey Fairchild, and Rian Bahran. 2018. Twitter geolocation: A hybrid approach. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12(3):34.

Michele Berlingerio, Francesco Calabrese, Giusy Di Lorenzo, Xiaowen Dong, Yiannis Gkoufas, and Dimitrios Mavroeidis. 2013. Safercity: a system for detecting and analyzing incidents from social media. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*. IEEE, pages 1077–1080.

Preeti Bhargava, Nemanja Spasojevic, and Guoning Hu. 2017. Lithium nlp: A system for rich information extraction from noisy user generated text on social media. *arXiv preprint arXiv:1707.04244* .

Han Bo, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. In *Proceedings of COLING*. pages 1045–1062.

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

Olga Buchel and Diane Pennington. 2017. Geospatial analysis. *The SAGE Handbook of Social Media Research Methods* pages 285–303.

Chris J.C. Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. Technical report. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

Davide Buscaldi and Paulo Rosso. 2008. A conceptual density-based approach for the disambiguation of toponyms. *International Journal of Geographical Information Science* 22(3):301–313.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, pages 759–768.

Jaeyoung Choi, Bart Thomee, Gerald Friedland, Liangliang Cao, Karl Ni, Damian Borth, Benjamin Elizalde, Luke Gottlieb, Carmen Carrano, Roger Pearce, et al. 2014. The placing task: A large-scale geo-estimation challenge for social-media videos and images. In *Proceedings of the 3rd ACM Multimedia Workshop on Geotagging and Its Applications in Multimedia*. ACM, pages 27–31.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Maxwell Guimarães De Oliveira, Cláudio de Souza Baptista, Cláudio EC Campelo, and Michela Bertolotto. 2017. A gold-standard social media corpus for urban issues. In *Proceedings of the Symposium on Applied Computing*. ACM, pages 1011–1016.

Grant DeLozier, Jason Baldridge, and Loretta London. 2015. Gazetteer-independent toponym resolution using geographic word profiles. In *AAAI*. pages 2382–2388.

Grant DeLozier, Ben Wing, Jason Baldridge, and Scott Nesbit. 2016. Creating a novel geolocation corpus from historical texts. *LAW X* page 188.

Mark Dredze, Miles Osborne, and Prabhanjan Kambadur. 2016. Geolocation for twitter: Timing matters. In *HLT-NAACL*. pages 1064–1069.

Mark Dredze, Michael J Paul, Shane Bergsma, and Hieu Tran. 2013. Carmen: A twitter geolocation system with applications to public health. In *AAAI workshop on expanding the boundaries of health informatics using AI (HIAI)*. volume 23, page 45.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1277–1287.

David Flatow, Mor Naaman, Ke Eddie Xie, Yana Volkovich, and Yaron Kanza. 2015. On the accuracy of hyper-local geotagging of social media content. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, pages 127–136.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869* .

Steven N Goodman, Daniele Fanelli, and John PA Ioannidis. 2016. What does research reproducibility mean? *Science translational medicine* 8(341):341ps12–341ps12.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2017a. Vancouver welcomes you! minimalist location metonymy resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1248–1259.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2017b. Whats missing in geographical parsing? .

Claire Grover, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. Use of the edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 368(1925):3875–3889.

Simon I Hay, Katherine E Battle, David M Pigott, David L Smith, Catherine L Moyes, Samir Bhatt, John S Brownstein, Nigel Collier, Monica F Myers, Dylan B George, et al. 2013. Global mapping of infectious disease. *Phil. Trans. R. Soc. B* 368(1614):20120250.

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

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1373–1378. https://aclweb.org/anthology/D/D15/D15-1162.

Mans Hulden, Miikka Silfverberg, and Jerid Francom. 2015. Kernel density estimation for text-based geolocation. In *AAAI*. pages 145–150.

Hayate Iso, Shoko Wakamiya, and Eiji Aramaki. 2017. Density estimation for geolocation via convolutional mixture density network. *arXiv preprint arXiv:1705.02750* .

David Jurgens, Tyler Finethy, James McCorriston, Yi Tian Xu, and Derek Ruths. 2015. Geolocation prediction in twitter using social networks: A critical analysis and review of current practice. *ICWSM* 15:188–197.

Morteza Karimzadeh. 2016. Performance evaluation measures for toponym resolution. In *Proceedings of the 10th Workshop on Geographic Information Retrieval*. ACM, page 8.

Morteza Karimzadeh, Wenyi Huang, Siddhartha Banerjee, Jan Oliver Wallgrün, Frank Hardisty, Scott Pezanowski, Prasenjit Mitra, and Alan M MacEachren. 2013. Geotxt: a web api to leverage place references in text. In *Proceedings of the 7th workshop on geographic information retrieval*. ACM, pages 72–73.

Mayank Kejriwal and Pedro Szekely. 2017. Neural embeddings for populated geonames locations. In *International Semantic Web Conference*. Springer, pages 139–146.

Jochen L Leidner. 2008. *Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names*. Universal-Publishers.

Michael D Lieberman, Hanan Samet, and Jagan Sankaranarayanan. 2010. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, pages 201–212.

Inderjeet Mani, Christy Doran, Dave Harris, Janet Hitzeman, Rob Quimby, Justin Richer, Ben Wellner, Scott Mardis, and Seamus Clancy. 2010. Spatialml: annotation scheme, resources, and evaluation. *Language Resources and Evaluation* 44(3):263–280.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*. pages 51–61.

Fernando Melo and Bruno Martins. 2015. Geocoding textual documents through the usage of hierarchical classifiers. In *Proceedings of the 9th Workshop on Geographic Information Retrieval*. ACM, page 7.

Fernando Melo and Bruno Martins. 2017. Automated geocoding of textual documents: A survey of current approaches. *Transactions in GIS* 21(1):3–38.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 701–710.

Afshin Rahimi, Timothy Baldwin, and Trevor Cohn. 2017. Continuous representation of location for geolocation and lexical dialectology using mixture density networks. *arXiv preprint arXiv:1708.04358* .

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2016. pigeo: A python geotagging tool .

Afshin Rahimi, Duy Vu, Trevor Cohn, and Timothy Baldwin. 2015. Exploiting text and network context for geolocation of social media users. *arXiv preprint arXiv:1506.04803* .

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1500–1510.

João Santos, Ivo Anastácio, and Bruno Martins. 2015. Using machine learning methods for disambiguating place references in textual documents. *GeoJournal* 80(3):375–392.

Lutfi Kerem Senel, Ihsan Utlu, Veysel Yucesoy, Aykut Koc, and Tolga Cukur. 2017. Semantic structure and interpretability of word embeddings. *arXiv preprint arXiv:1711.00331* .

Pavel Serdyukov, Vanessa Murdock, and Roelof Van Zwol. 2009. Placing flickr photos on a map. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 484–491.

Michael Speriosu and Jason Baldridge. 2013. Text-driven toponym resolution using indirect supervision. In *ACL (1)*. pages 1466–1476.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Richard Tobin, Claire Grover, Kate Byrne, James Reid, and Jo Walsh. 2010. Evaluation of georeferencing. In *proceedings of the 6th workshop on geographic information retrieval*. ACM, page 7.

Olivier Van Laere, Jonathan Quinn, Steven Schockaert, and Bart Dhoedt. 2014. Spatially aware term selection for geotagging. *IEEE transactions on Knowledge and Data Engineering* 26(1):221–234.

Jan Oliver Wallgrün, Morteza Karimzadeh, Alan M MacEachren, and Scott Pezanowski. 2017. Geocorpora: building a corpus to test and train microblog geoparsers. *International Journal of Geographical Information Science* pages 1–29.

Benjamin Wing and Jason Baldridge. 2014. Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 336–348.

Benjamin P Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 955–964.

Harry Zhang. 2004. The optimality of naive bayes. *AA* 1(2):3.

Wei Zhang and Judith Gelernter. 2014. Geocoding location expressions in twitter messages: A preference learning method. *Journal of Spatial Information Science* 2014(9):37–70.

Wei Zhang and Judith Gelernter. 2015. Exploring metaphorical senses and word representations for identifying metonyms. *arXiv preprint arXiv:1508.04515* .

Xin Zheng, Jialong Han, and Aixin Sun. 2017. A survey of location prediction on twitter. *arXiv preprint arXiv:1705.03172* .

Learning Prototypical Goal Activities for Locations

Tianyu Jiang and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
{tianyu, riloff}@cs.utah.edu

Abstract

People go to different places to engage in activities that reflect their goals. For example, people go to restaurants to eat, libraries to study, and churches to pray. We refer to an activity that represents a common reason *why* people typically go to a location as a *prototypical goal activity (goal-act)*. Our research aims to learn goal-acts for specific locations using a text corpus and semi-supervised learning. First, we extract activities and locations that co-occur in goal-oriented syntactic patterns. Next, we create an *activity profile matrix* and apply a semi-supervised label propagation algorithm to iteratively revise the activity strengths for different locations using a small set of labeled data. We show that this approach outperforms several baseline methods when judged against goal-acts identified by human annotators.

1 Introduction

Every day, people go to different places to accomplish goals. People go to stores to buy clothing, go to restaurants to eat, and go to the doctor for medical services. People travel to specific destinations to enjoy the beach, go skiing, or see historical sites. For most places, people typically go there for a common set of reasons, which we will refer to as *prototypical goal activities (goal-acts)* for a location. For example, a prototypical goal-act for restaurants would be "*eat food*" and for IKEA would be "*buy furniture*".

Previous research has established that recognizing people's goals is essential for narrative text understanding and story comprehension (Schank and Abelson, 1977; Wilensky, 1978; Lehnert, 1981; Elson and McKeown, 2010; Goyal et al., 2013). Goals and plans are essential to understand people's behavior and we use our knowledge of prototypical goals to make inferences when reading. For example, consider the following pair of sentences: *"Mary went to the supermarket. She needed milk."* Most people will infer that Mary purchased milk, unless told otherwise. But a purchase event is not explicitly mentioned. In contrast, a similar sentence pair *"Mary went to the theatre. She needed milk."* feels incongruent and does not produce that inference. Recognizing goals is also critical for conversational dialogue systems. For example, if a friend tells you that they went to a restaurant, you might reply *"What did you eat?"*, but if a friend says that they went to Yosemite, a more appropriate response might be *"Did you hike?"* or *"Did you see the waterfalls?"*.

Our knowledge of prototypical goal activities also helps us resolve semantic ambiguity. For example, consider the following sentences:

(a) *She went to the kitchen and got chicken.*
(b) *She went to the supermarket and got chicken.*
(c) *She went to the restaurant and got chicken.*

In sentence (a), we infer that she retrieved chicken (e.g., from the refrigerator) but did not pay for it. In (b), we infer that she paid for the chicken but probably did not eat it at the supermarket. In (c), we infer that she ate the chicken at the restaurant. Note how the verb "*got*" maps to different presumed events depending on the location.

Our research aims to learn the prototypical goal-acts for locations using a text corpus. First, we extract activities that co-occur with locations in goal-oriented syntactic patterns. Next, we construct an *activity profile matrix* that consists of an activity vector (profile) for each of the locations. We then apply a semi-supervised label propagation algorithm to iteratively revise the activity profile strengths based on a small set of labeled locations.

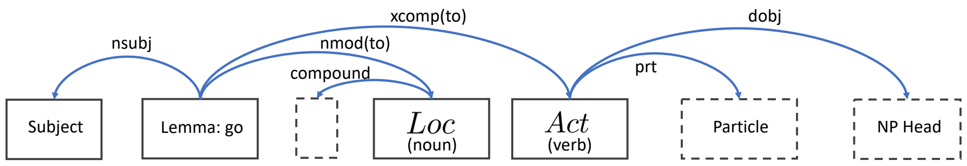


Figure 1: Dependency relation structure for "go to" pattern.

We also incorporate external resources to measure similarity between different activity expressions. Our results show that this semi-supervised learning approach outperforms several baseline methods in identifying the prototypical goal activities for locations.

2 Related Work

Recognizing plans and goals is fundamental to narrative story understanding (Schank and Abelson, 1977; Bower, 1982). Conceptual knowledge structures developed in prior work have shown the importance of this type of knowledge, including plans (Wilensky, 1978), goal trees (Carbonell, 1979), and plot units (Lehnert, 1981). Wilensky's research aimed to understand the actions of characters in stories by analyzing their goals, and their plans to accomplish those goals. For example, someone's goal might be to obtain food with a plan to go to a restaurant. Our work aims to learn prototypical goals associated with a location, to support similar inference capabilities during story understanding.

Goals and plans can also function to trigger *scripts* (Cullingford, 1978), such as the $RESTAURANT script. There has been growing interest in learning narrative event chains and script knowledge from large text corpora (e.g., (Chambers and Jurafsky, 2008, 2009; Jans et al., 2012; Pichotta and Mooney, 2014, 2016)). In addition, Goyal et al. (2010; 2013) developed a system to automatically produce plot unit representations for short stories. A manual analysis of their stories revealed that 61% of Positive/Negative Affect States originated from completed plans and goals, and 46% of Mental Affect States originated from explicitly stated or inferred plans and goals.

Elson & McKeown (2010) included plans and goals in their work on creating extensive story bank annotations that capture the knowledge needed to understand narrative structure. Researchers have also begun to explore NLP methods for recognizing the goals, desires, and plans of characters in stories. Recent work has explored techniques to detect wishes (desires) in natural language text (Goldberg et al., 2009) and identify desire fulfillment (Chaturvedi et al., 2016; Rahimtoroghi et al., 2017).

Graph-based semi-supervised learning has been successfully used for many tasks, including sentiment analysis (Rao and Ravichandran, 2009; Feng et al., 2013), affective event recognition (Ding and Riloff, 2016) and class-instance extraction (Talukdar and Pereira, 2010). The semi-supervised learning algorithm used in this paper is modeled after a framework developed by Zhu et al. (2003) based on harmonic energy minimization and a label propagation algorithm described in (Zhu and Ghahramani, 2002).

3 Learning Prototypical Goal Activities

Our aim is to learn the most prototypical goal-acts for locations. To tackle this problem, we first extract locations and related activities from a large text corpus. Then we use a semi-supervised learning method to identify the goal activities for individual locations. In the following sections we describe these processes in detail.

3.1 Location and Activity Extraction

To collect information about locations and activities, we use the 2011 Spinn3r dataset (Burton et al., 2011). Since our interest is learning about the activities of ordinary people in their daily lives, we use the Weblog subset of the Spinn3r corpus, which contains over 133 million blog posts.

We use the text data to identify activities that are potential goal-acts for a location. However we also need to identify locations and want to include both proper names (e.g., Disneyland) as well as nominals (e.g., store, beach), so Named Entity Recognition will not suffice. Consequently, we extract (Loc, Act) pairs using syntactic patterns.

First, we apply the Stanford dependency parser (Manning et al., 2014). We then extract sentences that match the pattern "go to X to Y" with the

	$a_1 =$ buy book	$a_2 =$ eat burger	...	$a_m =$ pray
$l_1 =$ McDonald's	.10	.30		.01
$l_2 =$ Burger King	.12	.50		.02
$l_3 =$ bookstore	.40	.02		.04
$\vdots$		$\vdots$		
$l_n =$ church	.05	.01		.70

Table 1: An illustration of the activity profile matrix Y.

following conditions: (1) there exists a subject connecting to "go", (2) X has an **nmod** (nominal modifier) relation to "go" (lemma), (3) X is a noun or noun compound, (4) Y has an **xcomp** relation (open clausal complement) with "go", and (5) Y is a verb. Figure 1 depicts the intended syntactic structure, which we will informally call the "go to" pattern. For sentences that match this pattern, we extract X as a location and Y as an activity. If the verb is followed by a particle and/or noun phrase (NP), then we also include the particle and head noun of the NP. For example, we extract activities such as "*pray*", "*clean up*", and "*buy sweater*".

This syntactic structure was chosen to identify activities that are described as being the reason why someone went to the location. However it is not perfect. In some cases, X is not a location (e.g., *"go to great lengths to ..."* yields "*lengths*" as a location), or Y is not a goal-act for X (e.g., *"go to the office to retrieve my briefcase ..."* yields "*retrieve briefcase*" which is not a prototypical goal for "*office*"). Interestingly, the pattern extracts some nominals that are not locations in a strict sense, but behave as locations. For example, *"go to the doctor"* extracts "*doctor*" as a location. Literally a doctor is a person, but in this context it really refers to the doctor's office, which is a location. The pattern also extracts entities such as "*roof*", which are not generally thought of as locations but do have a fixed physical location. Other extracted entities are virtual but function as locations, such as "*Internet*". For the purposes of this work, we use the term **location** in a general sense to include any place or object that has a physical, virtual or implied location.

The "go to" pattern worked quite well at extracting (Loc, Act) pairs, but in relatively small quantities due to the very specific nature of the syntactic structure. So we tried to find additional activities for those locations. Initially, we tried harvesting activities that occurred in close proximity (within 5 words) to a known location, but the results were too noisy. Instead, we used the pattern "Y in/at X" with the same syntactic constraints for Y (the extracted activity) and X (a location extracted by the "go to" pattern).

We discovered many sentences in the corpus that were exactly or nearly the same, differing only by a few words, which resulted in artificially high frequency counts for some (Loc, Act) pairs. So we filtered duplicate or near-duplicate sentences by computing the longest common substring of sentence pairs that extracted the same (Loc, Act). If the shared substring had length ≥ 5, then we discarded the "duplicate" sentence.

Finally, we applied three filters. To keep the size of the data manageable, we discarded locations and activities that were each extracted with frequency < 30 by our patterns. And we manually filtered locations that are Named Entities corresponding to cities or larger geo-political regions (e.g., provinces or countries). Large regions defined by government boundaries fall outside the scope of our task because the set of activities that typically occur in (say) a city or country is so broad. Finally, we added a filter to try to remove extremely general activities that can occur almost anywhere (e.g., visit). If an activity co-occurred with $> 20\%$ of the extracted (distinct) locations, then we discarded it.

After these filters, we extracted 451 distinct locations, 5143 distinct activities, roughly $200,000$ distinct (Loc, Act) pairs, and roughly $500,000$ instances of (Loc, Act) pairs.

3.2 Activity Profiles for Locations

We define an *activity profile matrix* Y of size $n \times m$, where n is the number of distinct locations and m is the number of distinct activities. $Y_{i,j}$ represents the strength of the jth activity a_j being a goal-act for l_i. We use $\mathbf{y}_i \in \mathbb{R}^m$ to denote the ith row of Y. Table 1 shows an illustration of (partial) activity profiles for four locations.[1] Our goal is

[1]Not actual values, for illustration only.

to learn the $Y_{i,j}$ values so that activities with high strength are truly goal-acts for location l_i.

We could build the activity profile for location l_i using the co-occurrence data extracted from the blog corpus. For example, we could estimate $P(a_j \mid l_i)$ directly from the frequency counts of the activities extracted for l_i. However, a high co-occurrence frequency doesn't necessarily mean that the activity represents a prototypical goal. For example, the activity *"have appointment"* frequently co-occurs with *"clinic"* but doesn't reveal the underlying reason for going to the clinic (e.g., probably to see a doctor or undergo a medical test). To appreciate the distinction, imagine that you asked a friend why she went to a health clinic, and she responded with "because I had an appointment". You would likely view her response as being snarky or evasive (i.e., she didn't want to tell you the reason). In Section 4, we will evaluate this approach as a baseline and show that it does not perform well.

3.3 Semi-Supervised Learning of Goal-Act Probabilities

Our aim is to learn the activity profiles for locations using a small amount of labeled data, so we frame this problem as a semi-supervised learning task. Given a small number of "seed" locations coupled with predefined goal-acts, we want to learn the goal-acts for new locations.

3.3.1 Location Similarity Graph

We use $l_i \in L$ to represent location l_i, where $|L| = n$. We define an undirected graph $G = (V, E)$ with vertices representing locations ($|V| = n$) and edges $E = V \times V$, such that each pair of vertices v_i and v_k is connected with an edge e_{ik} whose weight represents the similarity between l_i and l_k.

We can then represent the edge weights as an $n \times n$ symmetric weight matrix W indicating the similarity between locations. There could be many ways to define the weights, but for now we use the following definition from (Zhu et al., 2003), where σ^2 is a hyper-parameter[2]:

$$W_{i,k} = \exp\left(-\frac{1}{\sigma^2}\left(1 - sim\left(l_i, l_k\right)\right)\right) \quad (1)$$

To assess the similarity between locations, we measure the cosine similarity between vectors of their co-occurrence frequencies with activities. Specifically, let matrix $F_{n\times m} = [\mathbf{f}_1, ..., \mathbf{f}_n]^{\mathrm{T}}$ where $\mathbf{f}_i$ is a vector of length m capturing the co-occurrence frequencies between location l_i and each activity a_j in the extracted data (i.e., $F_{i,j}$ is the number of times that activity a_j occurred with location l_i). We then define location similarity as:

$$sim(l_i, l_k) = \frac{\mathbf{f}_i^{\mathrm{T}}\mathbf{f}_k}{\|\mathbf{f}_i\|\|\mathbf{f}_k\|} \quad (2)$$

3.3.2 Initializing Activity Profiles

We use semi-supervised learning with a set of "seed" locations from human annotations, and another set of locations that are unlabeled. So we subdivide the set of locations into $S = \{l_1, ..., l_s\}$, which are the seed locations, and $U = \{l_{s+1}, ..., l_{s+u}\}$, which are the unlabeled locations, such that $s + u = n$. For an unlabeled location $l_i \in U$, the initial activity profile is the normalized co-occurrence frequency vector $\overline{\mathbf{f}}_i$.

For each seed location $l_i \in S$, we first automatically construct an activity profile vector $\overline{\mathbf{h}}_i$ based on the gold goal-acts which were obtained from human annotators as described in Section 4.1. All activities not in the gold set are assigned a value of zero. Each activity a_j in the gold set is assigned a probability $P(a_j \mid l_i)$ based on the gold answers. However, the gold goal-acts may not match the activity phrases found in the corpus (see discussion in Section 4.3), so we smooth the vector created with the gold goal-acts by averaging it with the normalized co-occurrence frequency vector $\overline{\mathbf{f}}_i$ extracted from the corpus.

The activity profiles of seed locations stay constant through the learning process. We use $\mathbf{y}_i^0$ to denote the initial activity profiles. So when $l_i \in S$, $\mathbf{y}_i^0 = (\overline{\mathbf{f}}_i + \overline{\mathbf{h}}_i)/2$.

3.3.3 Learning Goal-Act Strengths

We apply a learning framework developed by (Zhu et al., 2003) based on harmonic energy minimization and extend it to multiple labels. Intuitively, we assume that similar locations should share similar activity profiles,[3] which motivates the following objective function over matrix Y:

$$\begin{gathered}\arg\min_{Y} \sum_{i,k} W_{i,k}\|\mathbf{y}_i - \mathbf{y}_k\|^2, \\ \text{s.t. } \mathbf{y}_i = \mathbf{y}_i^0 \text{ for each } l_i \in S\end{gathered} \quad (3)$$

Let $D = (d_i)$ denote an $n \times n$ diagonal matrix where $d_i = \sum_{k=1}^{n} W_{i,k}$. Let's split Y by the sth

[2]We use the same value $\sigma^2 = 0.03$ as (Zhu et al., 2003).

[3]This is a heuristic but is not always true.

row: $Y = \begin{bmatrix} Y_s \\ Y_u \end{bmatrix}$, then split W(similarly for D) into four blocks by the sth row and column:

$$W = \begin{bmatrix} W_{ss} & W_{su} \\ W_{us} & W_{uu} \end{bmatrix} \tag{4}$$

From (Zhu et al., 2003), Eq (3) is given by:

$$Y_u = (D_{uu} - W_{uu})^{-1} W_{us} Y_s \tag{5}$$

We then use the label propagation algorithm described in (Zhu and Ghahramani, 2002) to compute Y:

Algorithm 1

```
repeat
    $Y \leftarrow D^{-1}WY$
    Clamp $\mathbf{y}_i = \mathbf{y}_i^0$ for each $l_i \in S$
until convergence
```

3.3.4 Activity Similarity

One problem with the above algorithm is that it only takes advantage of relations between vertices (i.e., locations). If there are intrinsic relations between activities, they could be exploited as a complementary source of information to benefit the learning. Intuitively, different pairs of activities share different similarities, e.g., "*eat burgers*" should be more similar to "*have lunch*" than "*read books*".

Under this idea, similar to the previous location similarity weight matrix W, we want to define an activity similarity weight matrix $A_{m\times m}$ where $A_{i,k}$ indicates the similarity weight between activity a_i and a_k:

$$A_{i,k} = \exp\left(-\frac{1}{\sigma^2}\left(1 - sim\left(a_i, a_k\right)\right)\right) \tag{6}$$

where σ^2 is the same as in Eq (1).

We explore 3 different similarity functions $sim(a_i, a_k)$ based on co-occurrence with locations, word matching, and embedding similarities.

First, similar to Eq (2), we can use each activity's co-occurrence frequency with all locations as its location profile and define a similarity score based on cosine values of location profile vectors:

$$sim^L(a_i, a_k) = \frac{\mathbf{g}_i^{\mathrm{T}} \mathbf{g}_k}{\|\mathbf{g}_i\| \|\mathbf{g}_k\|} \tag{7}$$

where the predefined co-occurrence frequency matrix $F = [\mathbf{f}_1, ..., \mathbf{f}_n]^{\mathrm{T}} = [\mathbf{g}_1, ..., \mathbf{g}_m]$.

As a second option, the similarity between activities can often be implied by their lexical overlap, e.g., two activities sharing the same verb or noun might be related. For each word belonging to any of our activities, we use WordNet (Miller, 1995) to find its synonyms. We also include the word itself in the synonym set. If the synonym sets of two words overlap, we call these two words "**match**". Then we define the lexical overlap similarity function between a_i and a_k:

$$sim^O(a_i, a_k) = \begin{cases} 1 & \text{if verb and noun match} \\ 0.5 & \text{if verb or noun match} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

As a third option, we can use 300-dimension word embedding vectors (Pennington et al., 2014) trained on 840 billion tokens of web data to compute semantic similarity. We compute an activity's embedding as the average of its words' embeddings. Let $sim^E(a_i, a_k)$ be the cosine value between the embedding vectors of a_i and a_k:

$$sim^E(a_i, a_k) = cos\langle \text{Embed}(a_i), \text{Embed}(a_k)\rangle \tag{9}$$

Finally, we can plug these similarity functions into Eq (6). We use A^L, A^O, A^E to denote the corresponding matrix. We can also plug in multiple similarity metrics such as $(sim^L + sim^E)/2$ and use combination symbols A^{L+E} to denote the matrix.

3.3.5 Injecting Activity Similarity

Once we have a similarity matrix for activities, the next question is how will it help with the activity profile computation? Recall from Eq (5), we know that the activity profile of an unlabeled location can be represented by a linear combination of other locations' activity profiles. The activity profile matrix Y is an $n \times m$ matrix where each row is the activity profile for a location. We can also view Y as a matrix whose each column is the location profile for an activity. Using the same idea, we can make each column approximate a linear combination of its highly related columns (i.e., the location profile of an activity will become more similar to the location profiles of its similar activities). Our expectation is that this approximation will help improve the quality of Y.

By being right multiplied by matrix A, Y gets updated from manipulating its columns (activities) as well. We modify the algorithm accordingly as below:

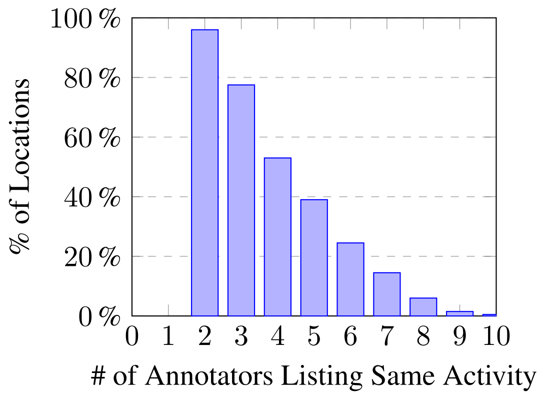


Figure 2: Percentage of locations that have at least one goal-act assigned by multiple annotators.

Algorithm 2

```
repeat
    $Y \leftarrow D^{-1}WYA$
    Clamp $\mathbf{y}_i = \mathbf{y}_i^0$ for each $l_i \in S$
until convergence
```

4 Evaluation

4.1 Gold Standard Data

Since this is a new task and there is no existing dataset for evaluation, we use crowd-sourcing via Amazon Mechanical Turk (AMT) to acquire gold standard data. First, we released a qualification test containing 15 locations along with detailed annotation guidelines. 25 AMT workers finished our assignment, and we chose 15 of them who did the best job following our guidelines to continue. We gave the 15 qualified workers 200 new locations, consisting of 152 nominals and 48 proper names,[4] randomly selected from our extracted data and set aside as test data. For each location, we asked the AMT workers to complete the following sentence:

People go to *LOC* to ______ ______
VERB NOUN

LOC was replaced by one of the 200 locations. Annotators were asked to provide an activity that is the primary reason why a person would go to that location, in the form of just a VERB or a VERB NOUN pair. Annotators also had the option to label a location as an "ERROR" if they felt that the provided term is not a location, since our location extraction was not perfect.

[4]Same distribution as in the whole location set.

Only 10 annotators finished labeling our test cases, so we used their answers as the gold standard. We discarded 12 locations that were labeled as an "ERROR" by ≥ 3 workers.[5] This resulted in a test set of 188 locations paired with 10 manually defined goal-acts for each one.

A key question that we wanted to investigate through this manual annotation effort is to know whether people truly do associate the same prototypical goal activities with locations. To what extent do people agree when asked to list goal-acts? Also, some places clearly have a smaller set of goal-acts than others. For example, the primary reason to go to an airport is to catch a flight, but there's a larger set of common reasons why people go to Yosemite (e.g.,"*hiking camping*", "*rock climbing*", "*see waterfalls*", etc.).

Complicating matters, the AMT workers often described the same activity with different words (e.g., "*buy book*" vs. "*purchase book*"). Automatically recognizing synonymous event phrases is a difficult NLP problem in its own right.[6] So solely for the purpose of analysis, we manually merged activities that have a nearly identical meaning. We were extremely conservative and did not merge similar or related phrases that were not synonymous because the granularity of terms may matter for this task (e.g., we did not merge "*eat burger*" and "*eat lunch*" because one may apply to a specific location while the other does not).

Figure 2 shows the results of our analysis. Only 1 location was assigned exactly the same goal-act by all 10 annotators. But at least half (5) of the annotators listed the same goal-act for 40% of the locations. And nearly 80% of locations had one or more goal-acts listed by ≥ 3 people. These results show that people often do share the same associations between prototypical goal-acts and locations. These results are also very conservative because many different answers were also similar (e.g. "*eat burger*", "*eat meal*").

In Table 2 we show examples of locations and the goal-acts listed for them by the human annotators. If multiple people gave the same answer, we show the number in parentheses. For example, given the location "Toys R Us", 9 people listed "*buy toys*" as a goal-act and 1 person listed "*browse gifts*". We see from Table 2 that

[5]We found that the workers rarely used the "ERROR" label, so setting this threshold to be 3 was a strong signal.

[6]We tried using WordNet synsets to conflate phrases, but it didn't help much.

Location	Gold Goal-Acts
Toys R Us	buy toys (9), browse gifts
sink	wash hands (7), wash dishes (3)
airport	catch flight (7), board planes, take airplane, take trips
bookstore	buy books (6), browse books (2), browse bestsellers, read book
lake	go fishing (3), go swimming (3), drive boat (2), ride boat, see scenery
chiropractor	get treatment (3), adjust backs (3), alleviate pain (2), get adjustment, get aligned
Chinatown	buy goods (2), buy duck, buy souvenirs, eat dim sum, eat rice, eat wontons, find Chinese, speak Chinese, visit restaurants

Table 2: Goal-acts provided by human annotators.

some locations yield very similar sets of goal-acts (e.g., sink, airport, bookstore), while other locations show more diversity (e.g., lake, chiropractor, Chinatown).

4.2 Baselines

To assess the difficulty of this NLP task, we created 3 baseline systems for comparison with our learning approach. All of these methods take the list of activities that co-occurred with a location l_i in our extracted data and rank them.

The first baseline, **FREQ**, ranks the activities based on the co-occurrence frequency $F_{i,j}$ between l_i and a_j in our patterns. The second baseline, **PMI**, ranks the activities using point-wise mutual information. The third baseline, **EMBED**, ranks the activities based on the cosine similarity of the semantic embedding vectors for l_i and a_j. We use GloVe (Pennington et al., 2014) 300-dimension embedding vectors pre-trained on 840 billion tokens of web data. For locations and activities with multiple words, we create an embedding by averaging the vectors of their constituent words.

4.3 Matching Activities

The gold standard contains a set of goal-acts for each location. Since the same activity can be expressed with many different phrases, the only way to truly know whether two phrases refer to the same activity is manual evaluation, which is expensive. Furthermore, many activities are very similar or highly related, but not exactly the same. For example, "*eat burger*" and "*eat food*" both describe eating activities, but the latter is more general than the former. Considering them to be the same is not always warranted (e.g., "*eat burger*" is a logical goal-act for McDonald's but not for Baskin-Robbins which primarily sells ice cream). As another example, "*buy chicken*" and "*eat chicken*" refer to different events (buying and eating) so they are clearly not the same semantically. But at a place like KFC, buying chicken implies eating chicken, and vice versa, so they seem like equally good answers as goal-acts for KFC. Due to the complexities of determining which gold standard answers belong in equivalence classes, we considered all of the goal-acts provided by the human annotators to be acceptable answers.

	MRR_E	MRR_P	TOP1	TOP2	TOP3
EMBED	0.02	0.09	0.05	0.08	0.12
PMI	0.20	0.33	0.25	0.36	0.41
FREQ	0.23	0.34	0.23	0.32	0.40
AP	0.28	0.38	0.29	0.41	0.47
AP+A^L	0.28	0.40	0.32	**0.44**	0.49
AP+A^O	0.23	0.33	0.24	0.35	0.43
AP+A^E	0.25	0.36	0.28	0.40	0.47
AP+A^{L+E}	**0.29**	**0.42**	**0.35**	**0.44**	**0.52**

Table 3: Scores for MRR and Top k results.

To determine whether an activity a_j produced by our system matches any of the gold goal-acts for a location l_i, we report results using two types of matching criteria. **Exact Match** judges a_j to be a correct answer for l_i if (1) it exactly matches (after lemmatization) any activity in l_i's gold set, or (2) a_j's *verb* and *noun* both appear in l_i's gold set, though possibly in different phrases. For example, if a gold set contains "*buy novels*" and "*browse books*", then "*buy books*" will be a match.

Since Exact Match is very conservative, we also define a **Partial Match** criterion to give 50% credit for answers that partially overlap with a gold answer. An activity a_j is a partial match for l_i if either its *verb* or *noun* matches any of the activities in l_i's gold set of goal-acts. For example, "*buy burger*" would be a partial match with "*buy food*" because their verbs match.

4.4 Evaluation Metrics

All of our methods produce a ranked list of hypothesized goal-acts for a location. So we use Mean Reciprocal Rank (MRR) to judge the quality of the top 10 activities in each ranked list. We report two types of MRR scores.

MRR based on the Exact Match criteria (MRR_E) is computed as follows, where n is the number of locations in the test set:

$$\text{MRR}_\text{E} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\text{rank of } 1^{st} \text{ Exact Match}} \tag{10}$$

We also compute MRR using both the Exact Match and Partial Match criteria. First, we need to identify the "best" answer among the 10 activities in the ranked list, which depends both on each activity's ranking and its matching score. The matching score for activity a_j is defined as:

$$\text{score}(a_j) = \begin{cases} 1 & \text{if } a_j \text{ is an Exact Match} \\ 0.5 & \text{if } a_j \text{ is a Partial Match} \\ 0 & \text{otherwise} \end{cases}$$

Given 10 ranked activities a_1 ... a_{10} for l_i, we then compute:

$$\text{best\_score}(l_i) = \max_{j=1..10} \frac{\text{score}(a_j)}{\text{rank}(a_j)}$$

And then finally define MRR_P as follows:

$$\text{MRR}_\text{P} = \frac{1}{n}\sum_{i=1}^{n} \text{best\_score}(l_i) \tag{11}$$

4.5 Experimental Results

Unless otherwise noted, all of our experiments report results using 4-fold cross-validation on the 200 locations in our test set. We used 4 folds to ensure 50 seed locations for each run (i.e., 1 fold for training and 3 folds for testing).

The first two columns of Table 3 show the MRR results under Exact Match and Partial Match conditions. The first 3 rows show the results for the baseline systems, and the remaining rows show results for our Activity Profile (AP) semi-supervised learning method. We show results for 5 variations of the algorithm: **AP** uses Algorithm 1, and the others use Algorithm 2 with different Activity Similarity measures: **AP+A$^\mathbf{L}$** (location profile similarity), **AP+A$^\mathbf{O}$** (overlap similarity), **AP+A$^\mathbf{E}$** (embedding similarity), and **AP+A$^\mathbf{L+E}$** (location profiles plus embeddings).

Table 3 shows that our AP algorithm outperforms all 3 baseline methods. When adding Activity Similarity into the algorithm, we find that A^L slightly improves performance, but A^O and A^E do not. However, we also tried combining them and obtained improved results by using A^L and A^E together, yielding an MRR_P score of 0.42.

To gain more insight about the behavior of the models, Table 3 also shows results for the top-ranked 1, 2, and 3 answers. For these experiments, the system gets full credit if any of its top k answers exactly matches the gold standard, or 50% credit if a partial match is among its top k answers. These results show that our AP method produces more correct answers at the top of the list than the baseline methods.

Table 4 shows six locations with their gold answers and the Top 3 goal-acts hypothesized by our best AP system and the PMI and FREQ baselines. The activities in **boldface** were deemed correct (including Partial Match). For "bookstore" and "pharmacy", all of the methods perform well. Note the challenge of recognizing that different phrases mean essentially the same thing (e.g., "*fill prescription*", "*pick up prescription*", "*find medicine*"). For "university" and "Meijer", the AP method produces more appropriate answers than the baseline methods. For "market" and "phone", all three methods struggle to produce good answers. Since "market" is polysemous, we see activities related to both stores and financial markets. And "phone" arguably is not a location at all, but most human annotators treated it as a virtual location, listing goal-acts related to telephones. However our algorithm considered phones to be similar to computers, which makes sense for today's smartphones. In general, we also observed that Internet sites behave as virtual locations in language (e.g., "*I went to YouTube...*").

4.6 Discussion

The goal-acts learned by our system were extracted from the Spinn3r dataset, while the gold standard answers were provided by human annotators, so the same (or very similar) activities are often expressed in different ways (see Section 4.3). This raises the question: what is the upper bound on system performance when evaluating against human-provided goal-acts? To answer this, we compared all of the activities that co-occurred with each location in the corpus against its gold goal-acts. Only 36% of locations had at least one gold goal-act among its extracted activities when matching identical strings (after lemmatization). Because of this issue, our Exact Match criteria also allowed for combining verbs and nouns from different gold answers. Under this Exact Match criteria, 73% of locations had at least one gold goal-act

Location	Gold Activity List	AP+A^{L+E} Top 3	PMI Top 3	FREQ Top 3
bookstore	buy book (6) browse book (2) browse bestseller read book	**buy book** **purchase book** **see book**	**buy copy** **purchase book** **buy book**	**buy book** **browse** **find book**
pharmacy	get drug (4) fill prescription (3) get prescription (2) buy medicine	**find medicine** **get prescription** **pick up prescription**	**buy pill** **fill prescription** **pick up prescription**	**buy pill** **fill prescription** **pick up prescription**
university	get degree (4) gain education (5) watch sport	**gain education** **further education** **gain knowledge**	study law study psychology pursue study	enrol[7] enroll take class
Meijer	buy grocery (8) buy cream obtain grocery	**buy item** go shopping get item	check out deal have shopping post today	get item save money check out
market	buy grocery (6) buy fresh, buy goods buy shirt, find produce	make money eat out eat lunch	have demand increase competition lead player	trade intervene make money
phone	make call (4), ERROR (2) answer call, call friend have conversation stop ring	play game browse website view website	put number have number put card	plug glance have number

Table 4: Examples of Top 3 hypothesized prototypical goal activities.

among the extracted activities, so this represents an upper bound on performance using this metric. Under the Partial Match criteria, 98% of locations had at least one gold goal-act among the extracted activities, but only 50% credit was awarded for these cases so the maximum score possible would be ∼86%.

We also manually inspected 200 gold locations to analyze their types. We discovered some related groups, but substantial diversity overall. The largest group contains ∼20% of the locations, which are many kinds of stores (e.g., Ikea, WalMart, Apple store, shoe store). Even within a group, different locations often have quite different sets of co-occurring activities. In fact, we discovered some spelling variants (e.g., "WalMart" and "wal mart"), but they also have substantially different activity vectors (e.g., because one spelling is much more frequent), so the model learns about them independently.[8] Other groups include restaurants (∼5%), home-related (e.g., bathroom) (∼5%), education (∼5%), virtual (e.g., Wikipedia) (∼3%), medical (∼3%) and landscape (e.g., hill) (∼3%). It is worth noting that our locations were extracted by two syntactic patterns and it remains to be seen if this has brought in any bias — detecting location nouns (especially nominals) is a challenging problem in its own right.

5 Conclusions and Future Work

We introduced the problem of learning prototypical goal activities for locations. We obtained human annotations and showed that people do associate prototypical goal-acts with locations. We then created an activity profile framework and applied a semi-supervised label propagation algorithm to iteratively update the activity strengths for locations. We demonstrated that our learning algorithm identifies goal-acts for locations more accurately than several baseline methods.

However, this problem is far from solved. Challenges also remain in how to evaluate the accuracy of goal knowledge extracted from text corpora. Nevertheless, our work represents a first step toward learning goal knowledge about locations, and we believe that learning knowledge about plans and goals is an important direction for natural language understanding research. In future work, we hope to see if we can take advantage of more contextual information as well as other external knowledge to improve the recognition of goal-acts.

Acknowledgments

We are grateful to Haibo Ding for valuable comments on preliminary versions of this work.

[7]A lemmatization error for the verb "enrolled".

[8]Of course normalizing location names beforehand may be beneficial in future work.

References

Gordon H Bower. 1982. Plans and Goals in Understanding Episodes. *Advances in Psychology*, 8:2–15.

K. Burton, N. Kasch, and I. Soboroff. 2011. The ICWSM 2011 Spinn3r Dataset. In *Proceedings of the Fifth Annual Conference on Weblogs and Social Media (ICWSM-2011)*.

J. G. Carbonell. 1979. *Subjective Understanding: Computer Models of Belief Systems*. Ph.D. thesis, Yale University.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT-2008)*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Snigdha Chaturvedi, Dan Goldwasser, and Hal Daumé III. 2016. Ask, and Shall You Receive? Understanding Desire Fulfillment in Natural Language Text. In *Processings of the 30th AAAI Conference on Artificial Intelligence (AAAI-2016)*.

Richard Edward Cullingford. 1978. *Script Application: Computer Understanding of Newspaper Stories*. Ph.D. thesis, Yale University.

Haibo Ding and Ellen Riloff. 2016. Acquiring Knowledge of Affective Events from Blogs using Label Propagation. In *Processings of the 30th AAAI Conference on Artificial Intelligence (AAAI-2016)*.

David Elson and Kathleen McKeown. 2010. Building a Bank of Semantically Encoded Narratives. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC-2010)*.

Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation Lexicon: A Dash of Sentiment Beneath the Surface Meaning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-2013)*.

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-2009)*.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods on Natural Language Processing (EMNLP-2010)*.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2013. A Computational Model for Plot Units. *Computational Intelligence*, 29(3):466–488.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2012)*.

Wendy G Lehnert. 1981. Plot Units and Narrative Summarization. *Cognitive Science*, 5(4):293–331.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014) System Demonstrations*.

George A Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP-2014)*.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2014)*.

Karl Pichotta and Raymond J Mooney. 2016. Learning Statistical Scripts with LSTM Recurrent Neural Networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-2016)*.

Elahe Rahimtoroghi, Jiaqi Wu, Ruimin Wang, Pranav Anand, and Marilyn Walker. 2017. Modelling Protagonist Goals and Desires in First-Person Narrative. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL-2017)*.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised Polarity Lexicon Induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2009)*.

Roger C Schank and Robert Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum.

Partha Pratim Talukdar and Fernando Pereira. 2010. Experiments in Graph-based Semi-supervised Learning Methods for Class-instance Acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-2010)*.

Robert Wilensky. 1978. *Understanding Goal-based Stories*. Ph.D. thesis, Yale University.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. Technical report, Carnegie Mellon University.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*.

Guess Me if You Can: Acronym Disambiguation for Enterprises

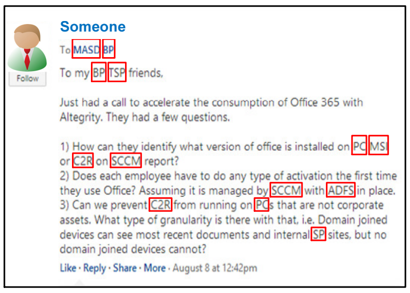
Yang Li[1][*], **Bo Zhao**[2], **Ariel Fuxman**[1], **Fangbo Tao**[3]
[1]Google, Mountain View, CA, USA
[2]Pinterest, San Francisco, CA, USA
[3]Facebook, Menlo Park, CA, USA
{zheda2006liyang, bo.zhao.uiuc, afuxman, fangbo.tao}@gmail.com

Abstract

Acronyms are abbreviations formed from the initial components of words or phrases. In enterprises, people often use acronyms to make communications more efficient. However, acronyms could be difficult to understand for people who are not familiar with the subject matter (new employees, etc.), thereby affecting productivity. To alleviate such troubles, we study how to automatically resolve the true meanings of acronyms in a given context. Acronym disambiguation for enterprises is challenging for several reasons. First, acronyms may be highly ambiguous since an acronym used in the enterprise could have multiple internal and external meanings. Second, there are usually no comprehensive knowledge bases such as Wikipedia available in enterprises. Finally, the system should be generic to work for any enterprise. In this work we propose an end-to-end framework to tackle all these challenges. The framework takes the enterprise corpus as input and produces a high-quality acronym disambiguation system as output. Our disambiguation models are trained via distant supervised learning, without requiring any manually labeled training examples. Therefore, our proposed framework can be deployed to any enterprise to support high-quality acronym disambiguation. Experimental results on real world data justified the effectiveness of our system.

1 Introduction

Acronyms are abbreviations formed from the initial components of words or phrases (e.g., "AI" from "Artificial Intelligence"). As acronyms can shorten long names and make communications more efficient, they are widely used at almost everywhere in enterprises, including notifications, emails, reports and social network posts. Figure 1 shows a sample enterprise social network post. As we can see, acronyms are frequently used there.



Someone
To MASD BP
To my BP TSP friends,

Just had a call to accelerate the consumption of Office 365 with Altegrity. They had a few questions.

1) How can they identify what version of office is installed on PC MSI or C2R on SCCM report?
2) Does each employee have to do any type of activation the first time they use Office? Assuming it is managed by SCCM with ADFS in place.
3) Can we prevent C2R from running on PCs that are not corporate assets. What type of granularity is there with that, i.e. Domain joined devices can see most recent documents and internal SP sites, but no domain joined devices cannot?

Figure 1: Acronyms in Enterprises

Despite the fact that acronyms can make communications more efficient, sometimes they could be difficult to understand, especially for people who are not familiar with the specific areas, such as new employees and patent lawyers. We randomly sampled 1000 documents from a Microsoft question answering forum and found out that only **7%** of the acronyms co-occur with the corresponding meanings in the same document, which means **93%** of the time when the user does not understand an acronym, she will need to find clues outside of the document. Therefore, it is particularly useful to develop a system that can automatically resolve the true meanings of acronyms in enterprise documents. Such system could be run online as a querying tool to handle any adhoc document, or run offline to annotate acronyms with their true meanings in a large corpus. In the offline mode, the true meanings can be further indexed by an enterprise search engine, so that when users search for the true meaning, documents containing the acronym can also be found.

The enterprise acronym disambiguation task is challenging due to the high ambiguity of acronyms, e.g., "SP" could stand for "Service Pack", "SharePoint" or "Surface Pro" in Microsoft. And there is one additional challenge compared with previous disambiguation tasks: in an enterprise document, an acronym could refer

[*]Work done while authors were at Microsoft Research.

to either an internal meaning (concepts created by the enterprise that may or may not be found outside) or an external meaning (all concepts that are not internal). For example, regarding the acronym "AI", "Asset Intelligence" is an internal meaning mainly used only in Microsoft, while "Artificial Intelligence" is an external meaning widely used in public. A good acronym disambiguation system should be able to handle both internal and external meanings. As we will explain in details, it is important to make such distinction and different strategies are needed for such two cases.

For internal meanings, there are some previous work on word sense disambiguation (Navigli, 2009) and acronym disambiguation (Feng et al., 2009; Pakhomov et al., 2005; Pustejovsky et al., 2001; Stevenson et al., 2009; Yu et al., 2006) on a closed-domain corpus. The main challenge here is that there are rarely any domain-specific knowledge bases available in enterprises, therefore all the signals for disambiguation (including potential meanings, and their popularity scores, context representations, etc.) need to be mined from plain text. Training data should also be automatically generated to make the system easily scale out to all enterprises. Compared with previous work, we developed a more comprehensive and advanced set of features in the disambiguation model, and also used a much less restrictive way to discover meaning candidates and training data, so that both precision and recall can be improved. Moreover, one main limitation of all previous work is that they do not distinguish internal and external meanings. They merely rely on the enterprise corpus to discover information about external meanings, which we observe is quite ineffective. The reason is that for popular external meaning like "Artificial Intelligence", people often directly use its acronym in enterprises without explanation, therefore there is limited information about the connection between the acronym and the external meaning in the enterprise corpus. On the other hand, there are much more such information available in the public domain, which should be leveraged by the system.

If we consider utilizing a public knowledge base such as Wikipedia to better handle external meanings of acronyms, the problem becomes very related to the well studied Entity Linking (Ji and Grishman, 2011; Cucerzan, 2007; Dredze et al., 2010; Hoffart et al., 2011; Li et al., 2013, 2016; Ratinov et al., 2011; Shen et al., 2012) problem, which is to map entity mentions in texts to their corresponding entities in a reference knowledge base (e.g. Wikipedia). But our disambiguation task is different from the entity linking task, because the system also needs to handle internal meanings which are not covered by any knowledge bases, and ultimately needs to decide whether an acronym refers to an internal meaning or an external meaning. It is nontrivial to combine the information mined from the enterprise corpus and the public knowledge base so that the system can get the best of both worlds. For instance, we have tried to run an internal disambiguator (leveraging information mined from enterprise corpus) and then resort to a public entity linking system if the internal one's confidence is low, but the performance is very poor. Even for external meanings, it is important to leverage signals from the enterprise corpus since the context surrounding them could be quite different from that in the external world, and context is one of the most important factor for disambiguation. For example, in public world, when people mention "Operating System" they mainly talk about how to install or use it; while within Microsoft, when people mention "Operating System" most of the time they focus on how to design or implement it.

In this work, we design a novel, end-to-end framework to address all the above challenges. Our framework takes the enterprise corpus and certain public knowledge base as input and produces a high-quality acronym disambiguation system as output. The models are all trained via distant supervised learning, therefore our system requires no manually labeled training examples and can be easily deployed to any enterprises.

2 Problem Statement

The *Enterprise Acronym Disambiguation* problem is comprised of two sub-problems. The first one is *Acronym Meaning Mining* (Adar, 2004; Ao and Takagi, 2005; Park and Byrd, 2001; Schwartz and Hearst, 2002; Jain et al., 2007; Larkey et al., 2000; Nadeau and Turney, 2005; Taneva et al., 2013), which aims at mining acronym/meaning pairs from the enterprise corpus. Each meaning m should contain the full name expansion e, popularity score p (indicating how often m is used as the genuine meaning of acronym a) and context words W (i.e. words frequently used in context of the meaning). The popularity score and context words can provide critical information for making disambiguation decisions. The second one is *Meaning Candidate Ranking*, whose goal is to rank the candidate meanings associated with the target acronym a and select the genuine meaning m based on the given context.

In this paper we assume the acronyms for disambiguation are provided as input to the system, either by the user or by an existing acronym detection module. We do not try to optimize the performance of acronym detection (e.g. identifying acronyms beyond the simple capitalized rule, or distinguishing cases where a capitalized term is not an acronym but a regular English word, such as "OK"). The task of acronym detection is also interesting and important. But due to the space limit, it is beyond the scope of this paper.

3 Framework

We propose a novel end-to-end framework to solve the *Enterprise Acronym Disambiguation* problem. Our framework takes the enterprise corpus as input and produces a high-quality acronym disambiguation system as output. Figure 2 shows the details of our proposed framework. In the mining module, we will sequentially perform Candidates Generation, Popularity Calculation, Candidates Deduplication and Context Harvesting on the input enterprise corpus. The details of these steps will be discussed in Section 4. After mining steps, we will get an acronym/meaning repository storing all the mined acronym/meaning pairs. Feed this repository together with the training data (automatically generated via distant supervision from the enterprise corpus) to the training module, we will get a candidate ranking model, a confidence estimation model and a final selection model. These models form the final acronym disambiguator and will be used in the testing module for actual acronym disambiguation. In the testing module, given the target acronym along with some context as input, the system will output the predicted meaning. Note that the mining and training module run offline once for the entire corpus or periodically when the corpus update, while the testing can be run online repeatedly for processing new documents.

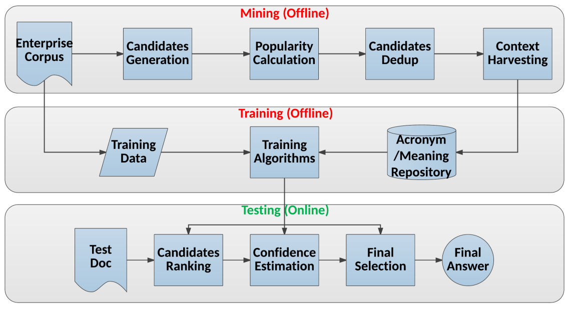


Figure 2: Framework

4 Acronym Meaning Mining

4.1 Candidates Generation

As there is no reference dictionary or knowledge base available in enterprise telling us the potential meanings of acronyms, we have to extract them from plain text. We propose a strategy called *Hybrid Generation* to balance extraction accuracy and coverage. Namely, we treat a phrase as a meaning candidate for an acronym if: (1) the initial letters of the phrase match the acronym **and** the phrase and the acronym co-occur in at least one document in the enterprise corpus; **or** (2) it is a valid candidate for the acronym in public knowledge bases (e.g. Wikipedia). The insight of this strategy is that the valid candidates missed by condition (1) are mainly public meanings which can be found in public knowledge bases. With this strategy we can make our system understand both the internal world and the external world.

4.2 Popularity Calculation

As mentioned in Section 2, for each candidate meaning, we need to calculate its popularity score, which reveals how often the candidate meaning is used as the genuine meaning of the acronym. In previous research on *Entity Linking*, popularity is calculated as the fraction of times a candidate being the target page for an anchor text in a reference knowledge base (e.g. Wikipedia). However, in enterprises, we do not have a knowledge base with anchor links. Thus we cannot calculate popularity in the same way. Here we propose to calculate two types of popularity to mimic the effect.

1. *Marginal Popularity*.
$$MP(m_i) = \frac{Count(m_i)}{\sum_{j=1}^{n} Count(m_j)}, \quad (1)$$
where $m_1, m_2, \ldots, m_n$ are the meaning candidates of acronym a and $Count(m_i)$ is the number of occurrences for m_i in the corpus.

2. *Conditional Popularity*.
$$CP(m_i) = \frac{Count(m_i, a)}{\sum_{j=1}^{n} Count(m_j, a)}, \quad (2)$$
where $m_1, m_2, \ldots, m_n$ are the meaning candidates of acronym a and $Count(m_i, a)$ is the number of document-level co-occurrences for m_i and a in the corpus.

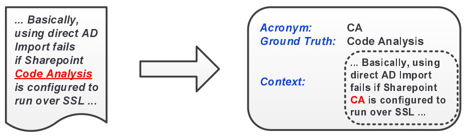

... Basically, using direct AD Import fails if Sharepoint Code Analysis is configured to run over SSL ...
Acronym:
CA
Ground Truth:
Code Analysis
Context:
... Basically, using direct AD Import fails if Sharepoint CA is configured to run over SSL ...

Figure 3: Distant Supervision Example

Conditional Popularity can more reasonably reveal how often the acronym is used to represent each meaning candidate. However, due to the data sparsity issue in enterprises, many valid candidates may get zero value for conditional popularity since they may never co-occur with the acronyms in the enterprise corpus. The Marginal Popularity does not have this problem since it is calculated from the raw counts of the candidates. Yet on the other hand, high marginal popularity score does not necessarily indicate high correlation between the candidate and the acronym. It is unclear how to combine the two scores into one popularity score, so we use both of them as features in the disambiguation model.

4.3 Candidates Deduplication

In enterprises, people often create many variants (including abbreviations, plurals or even misspellings) for the same meaning, therefore many mined meaning candidates are actually equivalent. For example, for the meaning "Certificate Authority" of the acronym "CA", the variants include "Cert Auth", "Certificate Authorities" and many others. It is important to deduplicate these variants before sending them to the disambiguation module. The deduplication helps aggregate disambiguation evidences and reduce noises. We design several heuristic rules[1] to perform the deduplication. Experiments show that the rules can accurately group the variants together. After grouping, we sort the variants within the same group based on their marginal popularity. The candidate with the largest marginal popularity is selected as the canonical candidate for the group. Other variants in the group will be deleted from the candidate list and their popularity scores will be aggregated to the canonical candidate. We maintain a table to record the variants for each canonical candidate.

4.4 Context Harvesting

In this step, we aim to harvest context words for each meaning candidate. These context words could be used to calculate context similarity with the query context. For each meaning candidate m, we put its canonical form and all its variants (from the variants table in Section 4.3) into set S. Then we scan the enterprise corpus, each time we find a match of any $e \in S$, we harvest the words in a width-W word window surrounding e as the context words of m. In our experiments we set window size as 30 after trying to vary the window size from 10 to 50 and finding 30 gives the best result.

As mentioned before, some popular public meanings might be mentioned very rarely by their full names in the enterprise corpus since people directly use their acronyms most of the time. Therefore, the above context harvesting process can only get very few context words for those public meanings. To alleviate this, for each public meaning we add its Wikipedia page's content as complementary context. By doing so, we ensure almost all valid candidates get a reasonable amount of context words.

5 Meaning Candidate Ranking

5.1 Candidate Ranking

We first train a candidate ranking model to rank candidates with respect to the likelihood of being the genuine meaning for the target acronym.

5.1.1 Training Data Generation

In order to train a robust ranking model, we need to get adequate amount of labeled training data. Manually labeling is obviously too expensive and it requires a lot of domain knowledge, which severely limits our framework's generalization capability. To tackle this problem, we propose to automatically generate training data via distant supervision. The intuition is that since acronyms and the corresponding meanings are semantically equivalent, people use them interchangeably in enterprises. Therefore we can fetch documents containing the meaning, replace the meaning with the corresponding acronym and treat the meaning as ground truth. Figure 3 shows an example of this automatic training data generation process.

5.1.2 Training Algorithm

Any learning-to-rank algorithms can be used here. In our system we utilize the LambdaMART algorithm (Burges, 2010) to train the model.

[1]Due to space limitations, the detailed rules are omitted. Example rules are "word overlap percentage after stemming > 0.8", "corresponding component words share same prefix".

5.1.3 Features

Now we explain the features we developed for the candidate ranking model. First, we have the *Marginal Popularity* score and *Conditional Popularity* score as two context-independent features, which could compensate for each other. However, as discussed in the previous section, some popular public meanings (e.g., "Artificial Intelligence") can be rarely mentioned in enterprise corpus by their full names, therefore both their marginal popularity score and conditional popularity score can be very low. To address this, we add a third feature called *Wiki Popularity*, which is calculated from Wikipedia anchor texts to capture how often an acronym refers to a public meaning in Wikipedia. The fourth feature we adopt is *Context Similarity*. We convert the harvested context for the meaning and the query context of the target acronym into TFIDF vectors and then compute their cosine similarity[2]. We also include two features (i.e. *LeftNeighborScore* and *RightNeighborScore*) to capture the effect of the immediate neighboring words, which are more important than further context words since immediate words could form phrases with the acronym. For example, if we see an acronym "SP" followed by the word "2", then likely it stands for "Service Pack". However, if we see "SP" followed by "2003", then probably its genuine meaning is "SharePoint". The last feature we use is *FullNamePercentage*. This feature is defined as the percentage of the meaning candidate's component words appearing in the context of the target acronym. Table 1 summarizes the features used to train the candidate ranking model.

5.2 Confidence Estimation

After getting the ranking results, we propose to apply a confidence estimation step to decide whether to trust the top ranked answer. There are two motivations behind. First, our candidate generation approach is not perfect, therefore we could encounter cases in which the genuine meaning is not in our candidates. For such cases, the top ranked answer is obviously incorrect. Second, our training data is biased towards the internal meanings since external meanings may rarely appear with full names. As a result, the learned ranking model may lack the capability to properly rank the external meanings. In such cases, we would better have the system return nothing rather than directly provide a wrong answer to mislead the user. In this step, we train a confidence estimation model, which will estimate the top result's confidence.

[2]One popular alternative to measure context similarity is using word embeddings (Mikolov et al., 2013; Li et al., 2015). In our system we experimented replacing TFIDF cosine similarity with word embedding similarity, or adding word embedding similarity as an additional feature, but both hurt the disambiguation accuracy. So we only included the TFIDF cosine similarity as the context similarity feature in our system.

5.2.1 Training Data Generation

Similar to the ranker training, here the training data is also automatically generated. We run the learned ranker on some distant labeled data (generated from a different corpus), and then check if the top ranked answer is correct or not. If it is correct, we generate a positive training example; otherwise we make a negative training example.

5.2.2 Training Algorithm

Any classification algorithms can be used here. In our system we utilize the MART boosted tree algorithm (Friedman, 2000) to train the model.

5.2.3 Features

We design 7 features (summarized in Table 2) to train the confidence estimation model. There are two intuitions behind: (1) If the top-ranked answer's ranking score is very small, or the top-ranked answer's score is close to the second-ranked answer's score, then the ranking is not very confident; (2) If the acronym has a dominating candidate in the public domain (e.g., "Personal Computer" is the dominating candidate for "PC"), and the candidates' Wiki popularity distribution is significantly different from their marginal/conditional popularity distributions, then the ranker's output is not very confident. The first intuition covers the first 3 features, while the second intuition covers the last 4 features.

5.3 Final Selection

We have discussed that one particular motivation for confidence estimation is that the candidate ranking stage has some bias so it does not always rank public meanings at top when they are correct. Therefore, assuming the confidence estimation model can remove incorrect top-ranked result, we still need one additional step to decide if any public meaning is correct, which we call a final selection model. In this step, we determine whether to return the most popular public meaning (based on Wiki Popularity) as the final answer, and this step is only triggered when the confidence estimator judges that the ranking result is unconfident.

Feature	Description
MarginalPopularity	The meaning candidate's marginal popularity score
ConditionalPopularity	The meaning candidate's conditional popularity score
WikiPopularity	The meaning candidate's Wiki popularity score
ContextSimilarity	TFIDF cosine similarity between meaning context and acronym context
LeftNeighborScore	Probability of acronym and meaning sharing the same immediate left word
RightNeighborScore	Probability of acronym and meaning sharing the same immediate right word
FullNamePercentage	Percentage of meaning candidate's component words appearing in acronym context

Table 1: Candidate Ranking Features

Feature	Description
Top1Score	Top 1 ranked meaning candidate's ranking score
Top1&2ScoreDiff	Difference between 1st and 2nd ranked meaning candidates' ranking score
Top1&2CtxSimDiff	Difference between 1st and 2nd ranked meaning candidates' context similarity score
Top1WikiPopularity	Top 1 ranked meaning candidate's Wiki popularity score
MaxWikiPopularity	Max Wiki popularity score among all the meaning candidates
MaxWP&MPGap	Max gap between Wiki and marginal popularity among all the meaning candidates
MaxWP&CPGap	Max gap between Wiki and conditional popularity among all the meaning candidates

Table 2: Confidence Estimation Features

The goal of the final selection model is similar to that of the confidence estimation model. In confidence estimation, we judge whether the top-ranked answer is correct; while in final selection, we check whether the most popular external meaning is correct. Thanks to this similarity, we can reuse the data, features and training algorithm in confidence estimation model. We take the same training data in Section 5.2.1 and update the labels correspondingly: if the genuine answer is the most popular external meaning, we generate a positive example; otherwise we make a negative one.

6 Experiments

6.1 Data

6.1.1 Mining and Training Corpus

We use both the Microsoft Answer Corpus (MAC) and the Microsoft Yammer Corpus (MYC) as the mining corpus. These corpus are kindly shared to us by Microsoft for research purpose. MAC contains 0.3 million web pages from a Microsoft internal question answering forum. MYC is consisted of 6.8 million posts from Microsoft's Yammer social network. In total, our mining module harvested 5287 acronyms and 17258 meaning candidates from this joint corpus.

For model training, the confidence estimation model and final selection model need to be trained on a different corpus than the candidate ranking model. So we train the candidate ranking model on MAC, with 12500 training examples being automatically generated; and train the confidence estimation and final selection model on MYC, with 40000 training instances being generated.

6.1.2 Evaluation Datasets

We prepared four datasets[3] for evaluation purposes. The first one ***Manual*** is obtained from the recent pages of Microsoft answer forum. Note these pages are disjoint from those used as mining/training corpus. We randomly sampled 300 pages and filtered out pages which do not contain ambiguous acronyms. After filtering, 240 test cases were left and we manually labeled them.

The second one ***Distant*** is generated via distant labeling on Microsoft Office365 documents. We sampled 2000 documents which contain at least one occurrence of a meaning candidate. Then we replaced the meanings with the corresponding acronyms and treat the meanings as ground truths. We manually checked through this dataset to remove some bad cases (e.g., "AS" for "App Store"). This resulted in a test set of 1949 test cases.

Comparing the ***Manual*** dataset with the ***Distant*** dataset, the ***Manual*** one, though in smaller size, can more accurately evaluate the system performance, since the target acronyms in it are sampled from the real distribution, while in the ***Distant*** dataset acronyms are artificially generated from

[3]Due to data confidentiality issue, we were unable to directly release these datasets.

randomly sampled meanings.

We also want to compare our method with the state-of-the-art Entity Linking (EL) systems based on public knowledge bases such as Wikipedia. However, it is unfair to directly compare as most enterprise specific meanings are unknown to them. Therefore, we need to only consider cases where the true meaning is a public meaning covered by both our system and the compared system. By filtering the distant dataset from Office365, we get the third dataset ***JoinW*** (1659 test cases) for comparing with the Wikifier (Ratinov et al., 2011), and the fourth dataset ***JoinA*** (237 test cases) for comparing with AIDA (Hoffart et al., 2011).

6.2 Compared Methods

6.2.1 Ablations of Our System

We compare the following ablations of our system, to illustrate the effectiveness of the features and components.

- **Internal Popularity (IP)**: Only the internal popularity features (i.e., marginal popularity and conditional popularity).
- **Popularity (P)**: The internal popularity features plus Wiki popularity features.
- **Popularity+Context (P+C)**: The popularity features plus context similarity feature.
- **Popularity+Context+Neighbbor (P+C+N)**: The popularity features, context similarity feature and immediate neighbor features.
- **Popularity+ Context+ Neighbbor+ Fullname (a.k.a. Candidate Ranker, or CR)**: Using all the features in candidate ranking module.
- **Candidate Ranker + Confidence Estimator (CR+ CE)**: Using the candidate ranking model plus the confidence estimation model.
- **Candidate Ranker + Confidence Estimator + Final Selector (a.k.a. Acronym Disambiguator, or AD)**: Using the candidate ranking model, the confidence estimation model and the final selection model. Full version of our system.

6.2.2 State-of-the-art EL Systems

We also compare our method with two state-of-the-art Entity Linking (EL) systems.

- **Wikifier**: a popular EL system using machine learning to combine various features together.
- **AIDA**: a robust EL system using mention-entity graph to find the best mention-entity mapping.

6.3 Quality of Mined Acronyms/Meanings

We first conduct experiments to evaluate the quality of the acronym/meaning pairs harvested through our offline mining module. Out of the 17258 mined pairs, we randomly sampled 2000 of them and asked 5 domain experts to manually check their validness. An acronym/meaning pair is considered as valid if the majority of the experts think the acronym is indeed used to abbreviate the meaning. For example, (*AS*, *Analysis Service*) is a valid pair, but (*AS*, *App Store*) is considered as invalid because people will not actually use *AS* to represent *App Store*. Among the sampled 2000 pairs, **94.5%** are labeled as valid, indicating our offline mining module can accurately extract acronym/meaning pairs from enterprise corpus. It is hard to precisely evaluate the coverage/recall of our mining method, since it is very difficult to obtain the complete meaning list for a given acronym. To get a rough idea, we randomly picked up 100 acronyms and asked the 5 domain experts to enumerate the valid meanings for these acronyms. In total we got 230 valid meanings and **all** of them are covered by the mined pairs.

6.4 Disambiguation Performance

We first conduct experiments to evaluate the disambiguation performance of our ranking model, and compare the helpfulness of the features used in the model. Figure 4 shows the precision (i.e., percentage of correctly disambiguated cases among all predicted cases), recall (i.e., percentage of correctly disambiguated cases among all test cases) and F_1 (i.e., harmonic mean of precision and recall) of the compared methods on the *Manual* dataset and the *Distant* dataset. In terms of the helpfulness of the features, the context similarity feature and the immediate neighbor features contribute most to the performance gain. Other features are less helpful, yet still bring improvements to the overall performance.

Next we conduct experiments to illustrate the effectiveness of the confidence estimation module and the final selection module in our system. Figure 5 shows the precision, recall and F_1 of the compared system configurations on the *Manual* and *Distant* dataset. As can be seen, the confidence estimation module can improve precision at the cost of hurting recall. Fortunately, the final selection module can recover some recall losses without sacrificing too much on precision. In

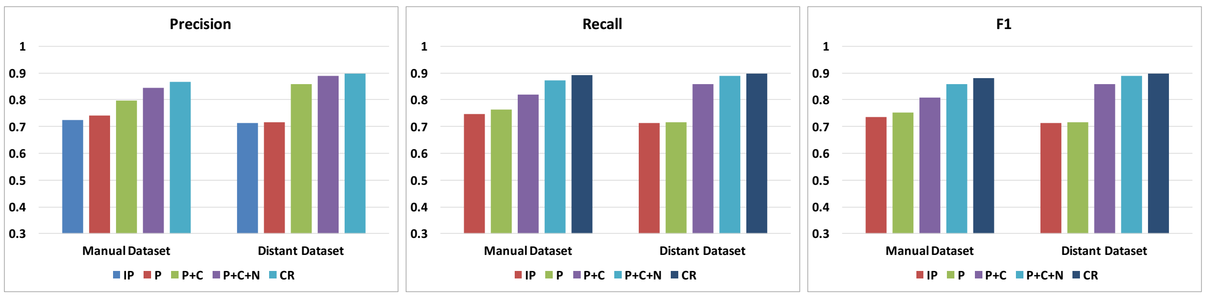


Figure 4: Ranking Performance

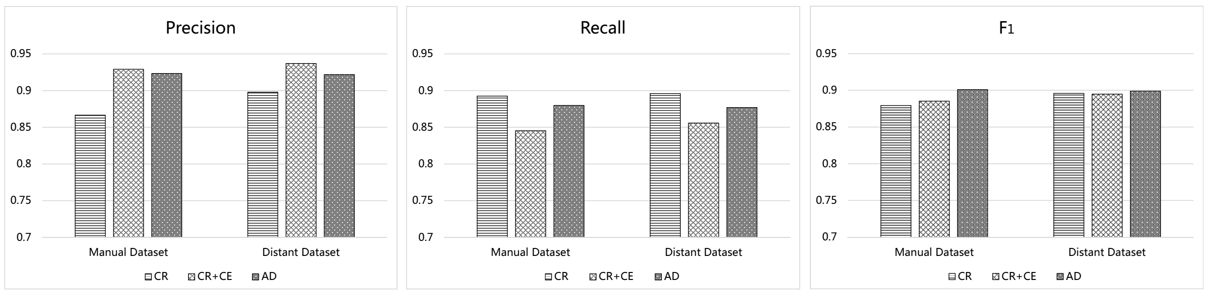
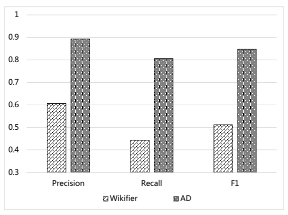


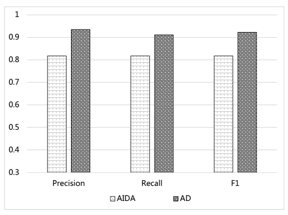
Figure 5: Effectiveness of Confidence Estimator and Final Selector

terms of the F_1 measure, the final system achieves the best performance.

Note that the ablation **P+C** naturally corresponds to the existing acronym disambiguation approaches (Feng et al., 2009; Pakhomov et al., 2005; Pustejovsky et al., 2001; Stevenson et al., 2009; Yu et al., 2006) mainly relying on context words and domain specific resources. These approaches do not specifically distinguish internal and external meanings. They merely rely on the internal corpus to discover information about external meanings, which is quite ineffective in the scenario of enterprise acronym disambiguation (as discussed in Section 1). In comparison, our system (**AD**) is able to leverage public resources together with the internal corpus to better handle the problem and therefore significantly outperforms them.



(a) *Wikifier vs. AD*



(b) *AIDA vs. AD*

Figure 6: Comparison with EL Systems

6.5 Comparison with EL Systems

We also compare our system (**AD**) with two state-of-the-art Entity Linking (EL) systems: **Wikifier** and **AIDA**. As explained in Section 6.1.2, we made two datasets (i.e., *JoinW* and *JoinA*) for fair comparisons. Figure 6(a) and Figure 6(b) present the comparison of our AD system against Wikifer and AIDA, respectively. As we can see from the figures, AD significantly outperforms both Wikifier and AIDA on all three measures. The reason is that even for public meanings (e.g., Operating System) indexed by Wikifier and AIDA, the usage of them could be quite different in enterprises (e.g., inside Microsoft people talk more about designing Operating System rather than how to install it). Wikifier and AIDA utilize information from public knowledge bases (e.g., Wikipedia) to generate features, therefore can hardly capture such enterprise-specific signals. In contrast, our AD system mines disambiguation features directly from the enterprise corpus and utilizes them together with the public signals. As a result, it can more accurately represent the characteristics of the enterprise and lead to much better disambiguation performances.

7 Related Work

Acronym meaning discovery has received a lot of attentions in vertical domains (mainly in biomedical). Most of the proposed approaches (Adar, 2004; Ao and Takagi, 2005; Park and Byrd, 2001; Schwartz and Hearst, 2002; Wren et al., 2002) utilized generic rules or text patterns (e.g. brackets, colons) to discover acronym meanings. These methods are usually based on the assumption that acronyms are co-mentioned with the corresponding meanings in the same document. However, in enterprises, this assumption rarely holds. Enterprises themselves are closed ecosystems, so it is very common for people to define the acronyms somewhere and use them elsewhere. As a result, such methods cannot be used for acronym meaning discovery in enterprises.

Recently, there have been a few works (Jain et al., 2007; Larkey et al., 2000; Nadeau and Turney, 2005; Taneva et al., 2013) on automatically mining acronym meanings by leveraging Web data (e.g., query sessions, click logs). However, it is hard to apply them directly to enterprises, since most data in enterprises are raw text and therefore the query sessions/click logs are rarely available.

Acronym disambiguation can be seen as a special case for the Entity Linking (EL) (Ji and Grishman, 2011; Dredze et al., 2010) problem. Approaches that link entity mentions to Wikipedia date back to Bunescu et. al's work (Bunescu and Paşca, 2006). They computed the cosine similarity between the text around the mention and the entity candidate's Wikipedia page. The referent entity with the maximum similarity score is selected as the disambiguation result. Cucerzan's work (Cucerzan, 2007) is the first one to realize the effectiveness of using topical coherence to globally perform EL. In that work, the topical coherence between the referent entity candidate and other entities within the same context is calculated based on their overlaps in categories and incoming links in Wikipedia. Recently, several methods (Hoffart et al., 2011; Li et al., 2013, 2016; Ratinov et al., 2011; Shen et al., 2012; Cheng and Roth, 2013) also tried to enrich "context similarity" and "topical coherence" using hybrid strategies. Shen et. al (Shen et al., 2015) provided a comprehensive survey for the techniques used in EL. However, these EL techniques cannot be used for acronym disambiguation in enterprises, since most enterprise meanings are not covered by public knowledge bases, and there are rarely any domain-specific knowledge bases available in enterprises. Automatic knowledge base construction (Suchanek et al., 2013) is promising, but the quality is far from applicable. Moreover, the structural information (e.g. entity taxonomy, cross-document hyperlinks) within Wikipedia, is rarely available in enterprises.

Most of the previous work (Feng et al., 2009; Pakhomov et al., 2005; Pustejovsky et al., 2001; Stevenson et al., 2009; Yu et al., 2006) on acronym disambiguation heavily rely on context words and domain specific resources. In comparison, our method explored a more comprehensive set of domain-independent features. Moreover, our method used a much less restrictive way to discover meaning candidates and training data, which is far more general than the methods relying on strict definition patterns (Schwartz and Hearst, 2002). Another particular limitation of all these previous work is that they do not distinguish internal and external meanings. They merely rely on the internal corpus to discover information about external meanings, which is quite ineffective.

8 Conclusions

In this paper, we studied the *Acronym Disambiguation for Enterprises* problem. We proposed a novel, end-to-end framework to solve this problem. Our framework takes the enterprise corpus as input and produces a high-quality acronym disambiguation system as output. The disambiguation models are trained via distant supervised learning, without requiring any manually labeled training examples. Different from all the previous acronym disambiguation approaches, our system is capable of accurately resolving acronyms to both enterprise-specific meanings and public meanings. Experimental results on Microsoft enterprise data demonstrated that our system can effectively construct acronym/meaning repositories from scratch and accurately disambiguate acronyms to their meanings with over 90% precision. Furthermore, our proposed framework can be easily deployed to any enterprises without requiring any domain knowledge.

References

Eytan Adar. 2004. Sarad: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533.

Hiroko Ao and Toshihisa Takagi. 2005. Alice: an algorithm to extract abbreviations from medline. *Journal of the American Medical Informatics Association*, 12(5):576–586.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, pages 9–16.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of EMNLP*, pages 1787–1796.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, pages 708–716.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of COLING*, pages 277–285.

Shicong Feng, Yuhong Xiong, Conglei Yao, Liwei Zheng, and Wei Liu. 2009. Acronym extraction and disambiguation in large-scale organizational web pages. In *Proceedings of CIKM*, pages 1693–1696.

Jerome H Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of EMNLP*, pages 782–792.

Alpa Jain, Silviu Cucerzan, and Saliha Azzam. 2007. Acronym-expansion recognition and ranking on the web. In *Information Reuse and Integration*, pages 209–214.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of ACL*, pages 1148–1158.

Leah S Larkey, Paul Ogilvie, M Andrew Price, and Brenden Tamilio. 2000. Acrophile: an automated acronym extractor and server. In *Proceedings of ACM conference on Digital libraries*, pages 205–214.

Chao Li, Lei Ji, and Jun Yan. 2015. Acronym disambiguation using word embedding. In *Proceedings of AAAI*, pages 4178–4179.

Yang Li, Shulong Tan, Huan Sun, Jiawei Han, Dan Roth, and Xifeng Yan. 2016. Entity disambiguation with linkless knowledge bases. In *Proceedings of WWW*, pages 1261–1270.

Yang Li, Chi Wang, Fangqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. 2013. Mining evidences for named entity disambiguation. In *Proceedings of SIGKDD*, pages 1070–1078.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*, pages 3111–3119.

David Nadeau and Peter D Turney. 2005. A supervised learning approach to acronym identification. In *Proceedings of CSCSI*, pages 319–329.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69.

Serguei Pakhomov, Ted Pedersen, and Christopher G Chute. 2005. Abbreviation and acronym disambiguation in clinical discourse. In *AMIA Annual Symposium Proceedings*, pages 589–593.

Youngja Park and Roy J Byrd. 2001. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of EMNLP*, pages 126–133.

James Pustejovsky, Jose Castano, Brent Cochran, Maciej Kotecki, Michael Morrell, and Anna Rumshisky. 2001. Extraction and disambiguation of acronym-meaning pairs in medline. *Medinfo*, 10(2001):371–375.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of ACL*, pages 1375–1384.

Ariel S Schwartz and Marti A Hearst. 2002. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Biocomputing*, pages 451–462.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *Knowledge and Data Engineering, IEEE Transactions on*, 27(2):443–460.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of WWW*, pages 449–458.

Mark Stevenson, Yikun Guo, Abdulaziz Al Amri, and Robert Gaizauskas. 2009. Disambiguation of biomedical abbreviations. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 71–79.

Fabian Suchanek, James Fan, Raphael Hoffmann, Sebastian Riedel, and Partha Pratim Talukdar. 2013. Advances in automated knowledge base construction. *SIGMOD Records*.

Bilyana Taneva, Tao Cheng, Kaushik Chakrabarti, and Yeye He. 2013. Mining acronym expansions and their meanings using query click log. In *Proceedings of WWW*, pages 1261–1272.

Jonathan D Wren, Harold R Garner, et al. 2002. Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of information in medicine*, 41(5):426–434.

Hong Yu, Won Kim, Vasileios Hatzivassiloglou, and John Wilbur. 2006. A large scale, corpus-based approach for automatically disambiguating biomedical abbreviations. *ACM Transactions on Information Systems*, 24(3):380–404.

A Multi-Axis Annotation Scheme for Event Temporal Relations

Qiang Ning,[1] **Hao Wu,**[2] **Dan Roth**[1,2]
Department of Computer Science
[1]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[2]University of Pennsylvania, Philadelphia, PA 19104, USA
qning2@illinois.edu, {haowu4,danroth}@seas.upenn.edu

Abstract

Existing temporal relation (TempRel) annotation schemes often have low inter-annotator agreements (IAA) even between experts, suggesting that the current annotation task needs a better definition. This paper proposes a new multi-axis modeling to better capture the temporal structure of events. In addition, we identify that event end-points are a major source of confusion in annotation, so we also propose to annotate TempRels based on start-points only. A pilot expert annotation effort using the proposed scheme shows significant improvement in IAA from the conventional 60's to 80's (Cohen's Kappa). This better-defined annotation scheme further enables the use of crowdsourcing to alleviate the labor intensity for each annotator. We hope that this work can foster more interesting studies towards event understanding.[1]

1 Introduction

Temporal relation (TempRel) extraction is an important task for event understanding, and it has drawn much attention in the natural language processing (NLP) community recently (UzZaman et al., 2013; Chambers et al., 2014; Llorens et al., 2015; Minard et al., 2015; Bethard et al., 2015, 2016, 2017; Leeuwenberg and Moens, 2017; Ning et al., 2017, 2018a,b).

Initiated by TimeBank (TB) (Pustejovsky et al., 2003b), a number of TempRel datasets have been collected, including but not limited to the verb-clause augmentation to TB (Bethard et al., 2007), TempEval1-3 (Verhagen et al., 2007, 2010; UzZaman et al., 2013), TimeBank-Dense (TB-Dense) (Cassidy et al., 2014), EventTimeCorpus (Reimers et al., 2016), and datasets with both temporal and other types of relations (e.g., coreference and causality) such as CaTeRs (Mostafazadeh et al., 2016) and RED (O'Gorman et al., 2016). These datasets were annotated by experts, but most still suffered from low inter-annotator agreements (IAA). For instance, the IAAs of TB-Dense, RED and THYME-TimeML (Styler IV et al., 2014) were only below or near 60% (given that events are already annotated). Since a low IAA usually indicates that the task is difficult even for humans (see Examples 1-3), the community has been looking into ways to simplify the task, by reducing the label set, and by breaking up the overall, complex task into subtasks (e.g., getting agreement on which event pairs should have a relation, and then what that relation should be) (Mostafazadeh et al., 2016; O'Gorman et al., 2016). In contrast to other existing datasets, Bethard et al. (2007) achieved an agreement as high as 90%, but the scope of its annotation was narrowed down to a very special verb-clause structure.

(*e1*, *e2*), (*e3*, *e4*), and (*e5*, *e6*): TempRels that are difficult even for humans. Note that only relevant events are highlighted here.
Example 1: Serbian police tried to eliminate the pro-independence Kosovo Liberation Army and (***e1:restore***) order. At least 51 people were (***e2:killed***) in clashes between Serb police and ethnic Albanians in the troubled region.
Example 2: Service industries (***e3:showed***) solid job gains, as did manufacturers, two areas expected to be hardest (***e4:hit***) when the effects of the Asian crisis hit the American economy.
Example 3: We will act again if we have evidence he is (***e5:rebuilding***) his weapons of mass destruction capabilities, senior officials say. In a bit of television diplomacy, Iraq's deputy foreign minister (***e6:responded***) from Baghdad in less than one hour, saying that ...

This paper proposes a new approach to handling

[1]The dataset is publicly available at https://cogcomp.org/page/publication_view/834.

these issues in TempRel annotation. **First**, we introduce *multi-axis modeling* to represent the temporal structure of events, based on which we anchor events to different semantic axes; only events from the same axis will then be temporally compared (Sec. 2). As explained later, those event pairs in Examples 1-3 are difficult because they represent different semantic phenomena and belong to different axes. **Second**, while we represent an event pair using two time intervals (say, $[t^1_{start}, t^1_{end}]$ and $[t^2_{start}, t^2_{end}]$), we suggest that comparisons involving end-points (e.g., t^1_{end} vs. t^2_{end}) are typically more difficult than comparing start-points (i.e., t^1_{start} vs. t^2_{start}); we attribute this to the ambiguity of expressing and perceiving durations of events (Coll-Florit and Gennari, 2011). We believe that this is an important consideration, and we propose in Sec. 3 that TempRel annotation should focus on start-points. Using the proposed annotation scheme, a pilot study done by experts achieved a high IAA of .84 (Cohen's Kappa) on a subset of TB-Dense, in contrast to the conventional 60's.

In addition to the low IAA issue, TempRel annotation is also known to be labor intensive. Our **third contribution** is that we facilitate, for the first time, the use of crowdsourcing to collect a new, high quality (under multiple metrics explained later) TempRel dataset. We explain how the crowdsourcing quality was controlled and how *vague* relations were handled in Sec. 4, and present some statistics and the quality of the new dataset in Sec. 5. A baseline system is also shown to achieve much better performance on the new dataset, when compared with system performance in the literature (Sec. 6). The paper's results are very encouraging and hopefully, this work would significantly benefit research in this area.

2 Temporal Structure of Events

Given a set of events, one important question in designing the TempRel annotation task is: which pairs of events should have a relation? The answer to it depends on the modeling of the overall temporal structure of events.

2.1 Motivation

TimeBank (Pustejovsky et al., 2003b) laid the foundation for many later TempRel corpora, e.g., (Bethard et al., 2007; UzZaman et al., 2013; Cassidy et al., 2014).[2] In TimeBank, the annotators were allowed to label TempRels between any pairs of events. This setup models the overall structure of events using *a general graph*, which made annotators inadvertently overlook some pairs, resulting in low IAAs and many false negatives.

Example 4: Dense Annotation Scheme.
Serbian police (***e7:tried***) to (***e8:eliminate***) the pro-independence Kosovo Liberation Army and (***e1:restore***) order. At least 51 people were (***e2:killed***) in clashes between Serb police and ethnic Albanians in the troubled region.
Given 4 NON-GENERIC events above, the dense scheme presents 6 pairs to annotators one by one: (*e7*, *e8*), (*e7*, *e1*), (*e7*, *e2*), (*e8*, *e1*), (*e8*, *e2*), and (*e1*, *e2*). Apparently, not all pairs are well-defined, e.g., (*e8*, *e2*) and (*e1*, *e2*), but annotators are forced to label all of them.

To address this issue, Cassidy et al. (2014) proposed a dense annotation scheme, TB-Dense, which annotates all event pairs within a sliding, two-sentence window (see Example 4). It requires all TempRels between GENERIC[3] and NON-GENERIC events to be labeled as *vague*, which conceptually models the overall structure by *two disjoint time-axes*: one for the NON-GENERIC and the other one for the GENERIC.

However, as shown by Examples 1-3 in which the highlighted events are NON-GENERIC, the TempRels may still be ill-defined: In Example 1, Serbian police tried to restore order but ended up with conflicts. It is reasonable to argue that the attempt to ***e1:restore*** order happened *before* the conflict where 51 people were ***e2:killed***; or, 51 people had been ***killed*** but order had not been ***restored*** yet, so ***e1:restore*** is *after* ***e2:killed***. Similarly, in Example 2, service industries and manufacturers were originally expected to be hardest ***e4:hit*** but actually ***e3:showed*** gains, so ***e4:hit*** is *before* ***e3:showed***; however, one can also argue that the two areas had ***showed*** gains but had not been ***hit***, so ***e4:hit*** is *after* ***e3:showed***. Again, ***e5:rebuilding*** is a hypothetical event: "we will act if ***rebuilding*** is true". Readers do not know for sure if "he is already rebuilding weapons but we have no evidence", or "he will be building weapons in the future", so annotators may disagree on the relation between ***e5:rebuilding*** and ***e6:responded***. Despite, importantly, minimizing missing annota-

[2]EventTimeCorpus (Reimers et al., 2016) is based on TimeBank, but aims at anchoring events onto explicit time expressions in each document rather than annotating TempRels between events, which can be a good complementary to other TempRel datasets.

[3]For example, *lions eat meat* is GENERIC.

tions, the current dense scheme forces annotators to label many such ill-defined pairs, resulting in low IAA.

2.2 Multi-Axis Modeling

Arguably, an ideal annotator may figure out the above ambiguity by him/herself and mark them as *vague*, but it is not a feasible requirement for all annotators to stay clear-headed for hours; let alone crowdsourcers. What makes things worse is that, after annotators spend a long time figuring out these difficult cases, whether they disagree with each other or agree on the vagueness, the final decisions for such cases will still be *vague*.

As another way to handle this dilemma, TB-Dense resorted to a 80% confidence rule: annotators were allowed to choose a label if one is 80% sure that it was the writer's intent. However, as pointed out by TB-Dense, annotators are likely to have rather different understandings of 80% confidence and it will still end up with disagreements.

In contrast to these annotation difficulties, humans can easily grasp the meaning of news articles, implying a potential gap between the difficulty of the annotation task and the one of understanding the actual meaning of the text. In Examples 1-3, the writers did not intend to explain the TempRels between those pairs, and the original annotators of TimeBank[4] did not label relations between those pairs either, which indicates that both writers and readers did not think the TempRels between these pairs were crucial. Instead, what is crucial in these examples is that "Serbian police ***tried*** to restore order but ***killed*** 51 people", that "two areas were ***expected*** to be hit but ***showed*** gains", and that "***if*** he rebuilds weapons ***then*** we will act." To "***restore*** order", to be "hardest ***hit***", and "if he was ***rebuilding***" were only the intention of police, the opinion of economists, and the condition to ***act***, respectively, and whether or not they actually happen is not the focus of those writers.

This discussion suggests that a single axis is too restrictive to represent the complex structure of NON-GENERIC events. Instead, we need a modeling which is more restrictive than a general graph so that annotators can focus on relation annotation (rather than looking for pairs first), but also more flexible than a single axis so that ill-defined

[4]Recall that they were given the entire article and only salient relations would be annotated.

Event Type	Category
INTENTION, OPINION	On an orthogonal axis
HYPOTHESIS, GENERIC	On a parallel axis
NEGATION	Not on any axis
STATIC, RECURRENT	Other

Table 1: The interpretation of various event types that are not on the main axis in the proposed multi-axis modeling. The names are rather straightforward; see examples for each in Appendix A.

relations are not forcibly annotated. Specifically, we need axes for intentions, opinions, hypotheses, etc. in addition to the main axis of an article. We thus argue for *multi-axis modeling*, as defined in Table 1. Following the proposed modeling, Examples 1-3 can be represented as in Fig. 1. This modeling aims at capturing what the author has explicitly expressed and it only asks annotators to look at comparable pairs, rather than forcing them to make decisions on often vaguely defined pairs.

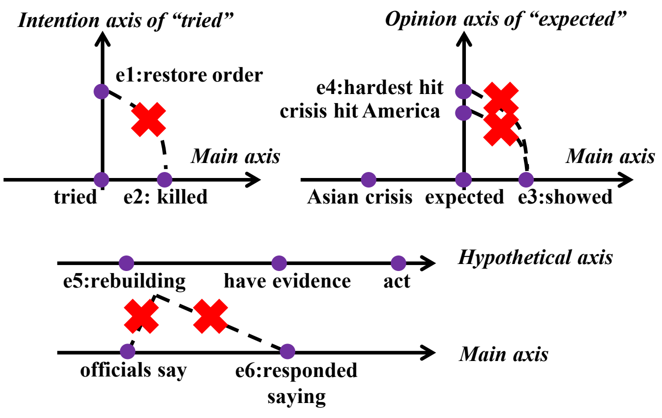


Figure 1: A multi-axis view of Examples 1-3. Only events on the same axis are compared.

In practice, we annotate one axis at a time: we first classify if an event is anchorable onto a given axis (this is also called the anchorability annotation step); then we annotate every pair of anchorable events (i.e., the relation annotation step); finally, we can move to another axis and repeat the two steps above. Note that ruling out cross-axis relations is only a strategy we adopt in this paper to separate well-defined relations from ill-defined relations. We do not claim that cross-axis relations are unimportant; instead, as shown in Fig. 2, we think that cross-axis relations are a different semantic phenomenon that requires additional investigation.

2.3 Comparisons with Existing Work

There have been other proposals of temporal structure modelings (Bramsen et al., 2006; Bethard et al., 2012), but in general, the semantic phenomena handled in our work are very different and complementary to them. (Bramsen et al., 2006) introduces "temporal segments" (a fragment of text that does not exhibit abrupt changes) in the medical domain. Similarly, their temporal segments can also be considered as a special temporal structure modeling. But a key difference is that (Bramsen et al., 2006) only annotates inter-segment relations, ignoring intra-segment ones. Since those segments are usually large chunks of text, the semantics handled in (Bramsen et al., 2006) is in a very coarse granularity (as pointed out by (Bramsen et al., 2006)) and is thus different from ours.

(Bethard et al., 2012) proposes a tree structure for children's stories, which "typically have simpler temporal structures", as they pointed out. Moreover, in their annotation, an event can only be linked to a single nearby event, even if multiple nearby events may exist, whereas we do not have such restrictions.

In addition, some of the semantic phenomena in Table 1 have been discussed in existing work. Here we compare with them for a better positioning of the proposed scheme.

2.3.1 Axis Projection

TB-Dense handled the incomparability between main-axis events and HYPOTHESIS/NEGATION by *treating an event as having occurred* if the event is HYPOTHESIS/NEGATION.[5] In our multi-axis modeling, the strategy adopted by TB-Dense falls into a more general approach, "axis projection". That is, projecting events across different axes to handle the incomparability between any two axes (not limited to HYPOTHESIS/NEGATION). Axis projection works well for certain event pairs like ***Asian crisis*** and ***e4:hardest hit*** in Example 2: as in Fig. 1, ***Asian crisis*** is *before* ***expected***, which is again *before* ***e4:hardest hit***, so ***Asian crisis*** is *before* ***e4:hardest hit***.

Generally, however, since there is no direct evidence that can guide the projection, annotators may have different projections (imagine projecting ***e5:rebuilding*** onto the main axis: is it in the past or in the future?). As a result, axis projection requires many specially designed guidelines or strong external knowledge. Annotators have to rigidly follow the sometimes counter-intuitive guidelines or "guess" a label instead of looking for evidence in the text.

[5]In the case of Example 3, it is to treat ***rebuilding*** as actually happened and then link it to ***responded***.

When strong external knowledge is involved in axis projection, it becomes a reasoning process and the resulting relations are a different type. For example, a reader may reason that in Example 3, it is well-known that they did "act again", implying his ***e5:rebuilding*** had happened and is *before* ***e6:responded***. Another example is in Fig. 2. It is obvious that relations based on these projections are not the same with and more challenging than those same-axis relations, so in the current stage, we should focus on same-axis relations only.

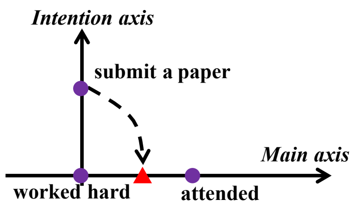


Figure 2: In *I worked hard to submit a paper ... I attended the conference*, the projection of ***submit a paper*** onto the main axis is clearly *before* ***attended***. However, this projection requires strong external knowledge that a paper should be submitted before attending a conference. Again, this projection is only a guess based on our external knowledge and it is still open whether the paper is submitted or not.

2.3.2 Introduction of the Orthogonal Axes

Another prominent difference to earlier work is the introduction of orthogonal axes, which has not been used in any existing work as we know. A special property is that the intersection event of two axes can be compared to events from both, which can sometimes bridge events, e.g., in Fig. 1, ***Asian crisis*** is seemingly *before* ***hardest hit*** due to their connections to ***expected***. Since ***Asian crisis*** is on the main axis, it seems that ***e4:hardest hit*** is on the main axis as well. However, the "***hardest hit***" in "***Asian crisis*** *before* ***hardest hit***" is only a projection of the original ***e4:hardest hit*** onto the real axis and is valid only when this OPINION is true.

Nevertheless, OPINIONS are not always true and INTENTIONS are not always fulfilled. In Example 5, ***e9:sponsoring*** and ***e10:resolve*** are the opinions of the West and the speaker, respectively; whether or not they are true depends on the authors' implications or the readers' understandings, which is often beyond the scope of TempRel annotation.[6] Example 6 demonstrates a similar situation for INTENTIONS: when reading the sentence of ***e11:report***, people are inclined to believe that it is fulfilled. But if we read the sentence of ***e12:report***, we have reason to believe that it is not. When it comes to ***e13:tell***, it is unclear if everyone told the truth. The existence of such examples indicates that orthogonal axes are a better modeling for INTENTIONS and OPINIONS.

Example 5: Opinion events may not always be true.
He is ostracized by the West for (***e9:sponsoring***) terrorism.
We need to (***e10:resolve***) the deep-seated causes that have resulted in these problems.
Example 6: Intentions may not always be fulfilled.
A passerby called the police to (***e11:report***) the body.
A passerby called the police to (***e12:report***) the body. Unfortunately, the line was busy.
I asked everyone to (***e13:tell***) the truth.

2.3.3 Differences from Factuality

Event modality have been discussed in many existing event annotation schemes, e.g., Event Nugget (Mitamura et al., 2015), Rich ERE (Song et al., 2015), and RED. Generally, an event is classified as *Actual* or *Non-Actual*, a.k.a. factuality (Saurí and Pustejovsky, 2009; Lee et al., 2015).

The main-axis events defined in this paper seem to be very similar to *Actual* events, but with several important differences: **First**, future events are *Non-Actual* because they indeed have not happened, but they may be on the main axis. **Second**, events that are not on the main axis can also be *Actual* events, e.g., intentions that are fulfilled, or opinions that are true. **Third**, as demonstrated by Examples 5-6, identifying anchorability as defined in Table 1 is relatively easy, but judging if an event actually happened is often a high-level understanding task that requires an understanding of the entire document or external knowledge.

Interested readers are referred to Appendix B for a detailed analysis of the difference between *Anchorable* (onto the main axis) and *Actual* on a subset of RED.

3 Interval Splitting

All existing annotation schemes adopt the interval representation of events (Allen, 1984) and there are 13 relations between two intervals (for readers who are not familiar with it, please see Fig. 4 in the appendix). To reduce the burden of annotators, existing schemes often resort to a reduced set of the 13 relations. For instance, Verhagen et al. (2007) merged all the overlap relations into a single relation, *overlap*. Bethard et al. (2007); Do et al. (2012); O'Gorman et al. (2016) all adopted this strategy. In Cassidy et al. (2014), they further split *overlap* into *includes*, *included* and *equal*.

Let $[t^1_{start}, t^1_{end}]$ and $[t^2_{start}, t^2_{end}]$ be the time intervals of two events (with the implicit assumption that $t_{start} \le t_{end}$). Instead of reducing the relations between two intervals, we try to explicitly compare the time points (see Fig. 3). In this way, the label set is simply *before, after* and *equal*,[7] while the expressivity remains the same. This interval splitting technique has also been used in (Raghavan et al., 2012).

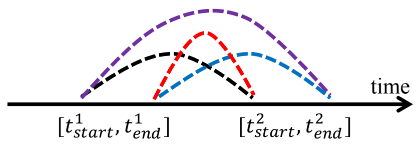


Figure 3: The comparison of two event time intervals, $[t^1_{start}, t^1_{end}]$ and $[t^2_{start}, t^2_{end}]$, can be decomposed into four comparisons t^1_{start} vs. t^2_{start}, t^1_{start} vs. t^2_{end}, t^1_{end} vs. t^2_{start}, and t^1_{end} vs. t^2_{end}, without loss of generality.

In addition to same expressivity, interval splitting can provide even more information when the relation between two events is *vague*. In the conventional setting, imagine that the annotators find that the relation between two events can be either *before* or *before and overlap*. Then the resulting annotation will have to be *vague*, although the annotators actually agree on the relation between t^1_{start} and t^2_{start}. Using interval splitting, however, such information can be preserved.

An obvious downside of interval splitting is the increased number of annotations needed (4 point comparisons vs. 1 interval comparison). In practice, however, it is usually much fewer than 4 comparisons. For example, when we see $t^1_{end} < t^2_{start}$ (as in Fig. 3), the other three can be skipped because they can all be inferred. Moreover, although the number of annotations is increased, the work load for human annotators may still be the same, because even in the conventional scheme, they still need to think of the relations between start- and

[6]For instance, there is undoubtedly a *causal* link between ***e9:sponsoring*** and ***ostracized***.

[7]We will discuss *vague* in Sec. 4.

end-points before they can make a decision.

3.1 Ambiguity of End-Points

During our pilot annotation, the annotation quality dropped significantly when the annotators needed to reason about relations involving end-points of events. Table 2 shows four metrics of task difficulty when only t^1_{start} vs. t^2_{start} or t^1_{end} vs. t^2_{end} are annotated. Non-anchorable events were removed for both jobs. The first two metrics, qualifying pass rate and survival rate are related to the two quality control protocols (see Sec. 4.1 for details). We can see that when annotating the relations between end-points, only one out of ten crowdsourcers (11%) could successfully pass our qualifying test; and even if they had passed it, half of them (56%) would have been kicked out in the middle of the task. The third line is the overall accuracy on gold set from all crowdsourcers (excluding those who did not pass the qualifying test), which drops from 67% to 37% when annotating end-end relations. The last line is the average response time per annotation and we can see that it takes much longer to label an end-end TempRel (52s) than a start-start TempRel (33s). This important discovery indicates that the TempRels between end-points is probably governed by a different linguistic phenomenon.

Metric	t^1_{start} vs. t^2_{start}	t^1_{end} vs. t^2_{end}
Qualification pass rate	50%	11%
Survival rate	74%	56%
Accuracy on gold	67%	37%
Avg. response time	33s	52s

Table 2: Annotations involving the end-points of events are found to be much harder than only comparing the start-points.

We hypothesize that the difficulty is a mixture of how durative events are expressed (by authors) and perceived (by readers) in natural language. In cognitive psychology, Coll-Florit and Gennari (2011) discovered that human readers take longer to perceive durative events than punctual events, e.g., *owe 50 bucks* vs. *lost 50 bucks*. From the writer's standpoint, durations are usually fuzzy (Schockaert and De Cock, 2008), or assumed to be a prior knowledge of readers (e.g., college takes 4 years and watching an NBA game takes a few hours), and thus not always written explicitly. Given all these reasons, we ignore the comparison of end-points in this work, although event duration is indeed, another important task.

4 Annotation Scheme Design

To summarize, with the proposed multi-axis modeling (Sec. 2) and interval splitting (Sec. 3), our annotation scheme is two-step. First, we mark every event candidate as being temporally *Anchorable* or not (based on the time axis we are working on). Second, we adopt the dense annotation scheme to label TempRels only between *Anchorable* events. Note that we only work on verb events in this paper, so non-verb event candidates are also deleted in a preprocessing step. We design crowdsourcing tasks for both steps and as we show later, high crowdsourcing quality was achieved on both tasks. In this section, we will discuss some practical issues.

4.1 Quality Control for Crowdsourcing

We take advantage of the quality control feature in CrowdFlower in our crowdsourcing jobs. For any job, a set of examples are annotated by experts beforehand, which is considered gold and will serve two purposes. (i) Qualifying test: Any crowdsourcer who wants to work on this job has to pass with 70% accuracy on 10 questions randomly selected from the gold set. (ii) Surviving test: During the annotation process, questions from the gold set will be randomly given to crowdsourcers without notice, and one has to maintain 70% accuracy on the gold set till the end of the annotation; otherwise, he or she will be forbidden from working on this job anymore and all his/her annotations will be discarded. At least 5 different annotators are required for every judgement and by default, the majority vote will be the final decision.

4.2 Vague Relations

How to handle *vague* relations is another issue in temporal annotation. In non-dense schemes, annotators usually skip the annotation of a *vague* pair. In dense schemes, a majority agreement rule is applied as a postprocessing step to back off a decision to *vague* when annotators cannot pass a majority vote (Cassidy et al., 2014), which reminds us that annotators often label a *vague* relation as non-vague due to lack of thinking.

We decide to proactively reduce the possibility of such situations. As mentioned earlier, our label set for t^1_{start} vs. t^2_{start} is *before*, *after*, *equal* and *vague*. We ask two questions: Q1=Is it possible that t^1_{start} is before t^2_{start}? Q2=Is it possible that t^2_{start} is before t^1_{start}? Let the answers be A1 and A2. Then we have a one-to-one mapping as follows: A1=A2=yes$\mapsto$*vague*, A1=A2=no$\mapsto$*equal*, A1=yes, A2=no$\mapsto$*before*, and A1=no, A2=yes$\mapsto$*after*. An advantage is that one will be prompted to think about all possibilities, thus reducing the chance of overlook.

Finally, the annotation interface we used is shown in Appendix C.

5 Corpus Statistics and Quality

In this section, we first focus on annotations on the main axis, which is usually the primary storyline and thus has most events. Before launching the crowdsourcing tasks, we checked the IAA between two experts on a subset of TB-Dense (about 100 events and 400 relations). A Cohen's Kappa of .85 was achieved in the first step: anchorability annotation. Only those events that both experts labeled *Anchorable* were kept before they moved onto the second step: relation annotation, for which the Cohen's Kappa was .90 for Q1 and .87 for Q2. Table 3 furthermore shows the distribution, Cohen's Kappa, and F_1 of each label. We can see the Kappa and F_1 of *vague* (κ=.75, F_1=.81) are generally lower than those of the other labels, confirming that temporal *vagueness* is a more difficult semantic phenomenon. Nevertheless, the overall IAA shown in Table 3 is a significant improvement compared to existing datasets.

	b	a	e	v	Overall
Distribution	.49	.23	.02	.26	1
IAA: Cohen's κ	.90	.87	1	.75	.84
IAA: F_1	.92	.93	1	.81	.90

Table 3: IAA of two experts' annotations in a pilot study on the main axis. Notations: **b**efore, **a**fter, **e**qual, and **v**ague.

With the improved IAA confirmed by experts, we sequentially launched the two-step crowdsourcing tasks through CrowdFlower on top of the same 36 documents of TB-Dense. To evaluate how well the crowdsourcers performed on our task, we calculate two quality metrics: accuracy on the gold set and the Worker Agreement with Aggregate (WAWA). WAWA indicates the average number of crowdsourcers' responses agreed with the aggregate answer (we used majority aggregation for each question). For example, if N individual responses were obtained in total, and n of them were correct when compared to the aggregate answer, then WAWA is simply n/N. In the first step, crowdsourcers labeled 28% of the events as *Non-Anchorable* to the main axis, with an accuracy on the gold of .86 and a WAWA of .79.

With *Non-Anchorable* events filtered, the relation annotation step was launched as another crowdsourcing task. The label distribution is b=.50, a=.28, e=.03, and v=.19 (consistent with Table 3). In Table 4, we show the annotation quality of this step using accuracy on the gold set and WAWA. We can see that the crowdsourcers achieved a very good performance on the gold set, indicating that they are consistent with the authors who created the gold set; these crowdsourcers also achieved a high-level agreement under the WAWA metric, indicating that they are consistent among themselves. These two metrics indicate that the annotation task is now well-defined and easy to understand even by non-experts.

No.	Metric	Q1	Q2	All
1	Accuracy on Gold	.89	.88	.88
2	WAWA	.82	.81	.81

Table 4: Quality analysis of the relation annotation step of MATRES. "Q1" and "Q2" refer to the two questions crowdsourcers were asked (see Sec. 4.2 for details). Line 1 measures the level of consistency between crowdsourcers and the authors and line 2 measures the level of consistency among the crowdsourcers themselves.

We continued to annotate INTENTION and OPINION which create orthogonal branches on the main axis. In the first step, crowdsourcers achieved an accuracy on gold of .82 and a WAWA of .89. Since only 16% of the events are in this category and these axes are usually very short (e.g., ***allocate*** *funds to* ***build*** *a museum*.), the annotation task is relatively small and two experts took the second step and achieved an agreement of .86 (F_1).

We name our new dataset *MATRES* for Multi-Axis Temporal RElations for Start-points. Each individual judgement cost us $0.01 and MATRES in total cost about $400 for 36 documents.

5.1 Comparison to TB-Dense

To get another checkpoint of the quality of the new dataset, we compare with the annotations of TB-Dense. TB-Dense has 1.1K verb events, between which 3.4K event-event (EE) relations are annotated. In the new dataset, 72% of the events (0.8K) are anchored onto the main axis, resulting in 1.6K EE relations, and 16% (0.2K) are anchored onto orthogonal axes, resulting in 0.2K EE relations.

The following comparison is based on the 1.8K EE relations in common. Moreover, since TB-Dense annotations are for intervals instead of start-points only, we converted TB-Dense's interval relations to start-point relations (e.g., if ***A*** includes ***B***, then t^{A}_{start} is before t^{B}_{start}).

	b	a	e	v	All
b	455	11	5	42	513
a	45	309	16	68	438
e	13	7	2	10	32
v	450	138	20	192	800
All	963	465	43	312	1783

Table 5: An evaluation of MATRES against TB-Dense. Horizontal: MATRES. Vertical: TB-Dense (with interval relations mapped to start-point relations). Please see explanation of these numbers in text.

The confusion matrix is shown in Table 5. A few remarks about how to understand it: **First**, when TB-Dense labels *before* or *after*, MATRES also has a high-probability of having the same label (b=455/513=.89, a=309/438=.71); when MATRES labels *vague*, TB-Dense is also very likely to label *vague* (v=192/312=.62). This indicates the *high agreement level* between the two datasets if the interval- or point-based annotation difference is ruled out. **Second**, many *vague* relations in TB-Dense are labeled as *before*, *after* or *equal* in MATRES. This is expected because TB-Dense annotates relations between *intervals*, while MATRES annotates *start-points*. When durative events are involved, the problem usually becomes more difficult and interval-based annotation is more likely to label *vague* (see earlier discussions in Sec. 3). Example 7 shows three typical cases, where ***e14:became***, ***e17:backed***, ***e18:rose*** and ***e19:extending*** can be considered durative. If only their start-points are considered, the crowdsourcers were correct in labeling ***e14*** before ***e15***, ***e16*** after ***e17***, and ***e18*** equal to ***e19***, although TB-Dense says *vague* for all of them. **Third**, *equal* seems to be the relation that the two dataset mostly disagree on, which is probably due to crowdsourcers' lack of understanding in time granularity and event coreference. Although *equal* relations only constitutes a small portion in all relations, it needs further investigation.

6 Baseline System

We develop a baseline system for TempRel extraction on MATRES, assuming that all the events and axes are given. The following commonly-

Example 7: Typical cases that TB-Dense annotated *vague* but MATRES annotated *before*, *after*, and *equal*, respectively.
At one point , when it (***e14:became***) clear controllers could not contact the plane, someone (***e15:said***) a prayer. *TB-Dense:* vague*; MATRES:* before
The US is bolstering its military presence in the gulf, as President Clinton (***e16:discussed***) the Iraq crisis with the one ally who has (***e17:backed***) his threat of force, British prime minister Tony Blair. *TB-Dense:* vague*; MATRES:* after
Average hourly earnings of nonsupervisory employees (***e18:rose***) to $12.51. The gain left wages 3.8 percent higher than a year earlier, (***e19:extending***) a trend that has given back to workers some of the earning power they lost to inflation in the last decade. *TB-Dense:* vague*; MATRES:* equal

used features for each event pair are used: (i) The part-of-speech (POS) tags of each individual event and of its neighboring three words. (ii) The sentence and token distance between the two events. (iii) The appearance of any modal verb between the two event mentions in text (i.e., will, would, can, could, may and might). (iv) The appearance of any temporal connectives between the two event mentions (e.g., before, after and since). (v) Whether the two verbs have a common synonym from their synsets in WordNet (Fellbaum, 1998). (vi) Whether the input event mentions have a common derivational form derived from WordNet. (vii) The head words of the preposition phrases that cover each event, respectively. And (viii) event properties such as Aspect, Modality, and Polarity that come with the TimeBank dataset and are commonly used as features.

The proposed baseline system uses the averaged perceptron algorithm to classify the relation between each event pair into one of the four relation types. We adopted the same train/dev/test split of TB-Dense, where there are 22 documents in train, 5 in dev, and 9 in test. Parameters were tuned on the train-set to maximize its F_1 on the dev-set, after which the classifier was retrained on the union of train and dev. A detailed analysis of the baseline system is provided in Table 6. The performance on *equal* and *vague* is lower than on *before* and *after*, probably due to shortage in these labels in the training data and the inherent difficulty in event coreference and temporal vagueness. We can see, though, that the overall performance on MATRES is much better than those in the literature for TempRel extraction, which used to be in the low 50's (Chambers et al., 2014; Ning et al., 2017). The same system was also retrained and tested on the original annotations of TB-Dense (Line "Original"), which confirms the significant improvement if the proposed annotation scheme is used. Note that we *do not* mean to say that the proposed baseline system itself is better than other existing algorithms, but rather that the proposed annotation scheme and the resulting dataset lead to better defined machine learning tasks. In the future, more data can be collected and used with advanced techniques such as ILP (Do et al., 2012), structured learning (Ning et al., 2017) or multi-sieve (Chambers et al., 2014).

	Training			Testing		
	P	R	F_1	P	R	F_1
Before	.74	.91	.82	.71	.80	.75
After	.73	.77	.75	.55	.64	.59
Equal	1	.05	.09	-	-	-
Vague	.75	.28	.41	.29	.13	.18
Overall	**.73**	**.81**	**.77**	**.66**	**.72**	**.69**
Original	.44	.67	.53	.40	.60	.48

Table 6: Performance of the proposed baseline system on MATRES. Line "Original" is the same system retrained on the original TB-Dense and tested on the same subset of event pairs. Due to the limited number of *equal* examples, the system did not make any *equal* predictions on the testset.

7 Conclusion

This paper proposes a new scheme for TempRel annotation between events, simplifying the task by focusing on a single time axis at a time. We have also identified that end-points of events is a major source of confusion during annotation due to reasons beyond the scope of TempRel annotation, and proposed to focus on start-points only and handle the end-points issue in further investigation (e.g., in event duration annotation tasks). Pilot study by expert annotators shows significant IAA improvements compared to literature values, indicating a better task definition under the proposed scheme. This further enables the usage of crowdsourcing to collect a new dataset, MATRES, at a lower time cost. Analysis shows that MATRES, albeit crowdsourced, has achieved a reasonably good agreement level, as confirmed by its performance on the gold set (agreement with the authors), the WAWA metric (agreement with the crowdsourcers themselves), and consistency with TB-Dense (agreement with an existing dataset). Given the fact that existing schemes suffer from low IAAs and lack of data, we hope that the findings in this work would provide a good start towards understanding more sophisticated semantic phenomena in this area.

Acknowledgements

We thank Martha Palmer, Tim O'Gorman, Mark Sammons and all the anonymous reviewers for providing insightful comments and critique in earlier stages of this work. This research is supported in part by a grant from the Allen Institute for Artificial Intelligence (allenai.org); the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network; by DARPA under agreement number FA8750-13-2-0008; and by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053 (the ARL Network Science CTA).

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, of the Army Research Laboratory or the U.S. Government. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the ARL.

References

James F Allen. 1984. Towards a general theory of action and time. *Artificial intelligence* 23(2):123–154.

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 Task 6: Clinical TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 806–814.

Steven Bethard, Oleksandr Kolomiyets, and Marie-Francine Moens. 2012. Annotating story timelines as temporal dependency structures. In *Proceedings of the eighth international conference on language resources and evaluation (LREC)*. ELRA, pages 2721–2726.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *IEEE International Conference on Semantic Computing (ICSC)*. pages 11–18.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. SemEval-2016 Task 12: Clinical TempEval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1052–1062.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. SemEval-2017 Task 12: Clinical TempEval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, pages 565–572.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. pages 189–198.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 501–506.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics* 2:273–284.

Marta Coll-Florit and Silvia P Gennari. 2011. Time in language: Event duration in language comprehension. *Cognitive psychology* 62(1):41–79.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event detection and factuality assessment with non-expert supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1643–1648.

Tuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TEMPEVAL - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 792–800.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, Ruben Urizar, and Fondazione Bruno Kessler. 2015. SemEval-2015 Task 4: TimeLine: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 778–786.

Teruko Mitamura, Yukari Yamakawa, Susan Holm, Zhiyi Song, Ann Bies, Seth Kulick, and Stephanie Strassel. 2015. Event nugget annotation: Processes and issues. In *Proceedings of the Workshop on Events at NAACL-HLT*.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*. pages 51–61.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. Copenhagen, Denmark.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018a. Improving temporal relation extraction with a globally acquired statistical resource. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Qiang Ning, Zhongzhi Yu, Chuchu Fan, and Dan Roth. 2018b. Exploiting partially annotated data for temporal relation extraction. In *The Joint Conference on Lexical and Computational Semantics (*SEM)*. Association for Computational Linguistics.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*. Association for Computational Linguistics, Austin, Texas, pages 47–56.

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. 2003a. TimeML: Robust specification of event and temporal expressions in text. *New directions in question answering* 3:28–34.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003b. The TIMEBANK corpus. In *Corpus linguistics*. volume 2003, pages 647–656.

Preethi Raghavan, Eric Fosler-Lussier, and Albert M Lai. 2012. Learning to temporally order medical events in clinical text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pages 70–74.

Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. 2016. Temporal anchoring of events for the timebank corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2195–2204.

Roser Saurí and James Pustejovsky. 2009. FactBank: a corpus annotated with event factuality. *Language resources and evaluation* 43(3):227.

Steven Schockaert and Martine De Cock. 2008. Temporal reasoning about fuzzy intervals. *Artificial Intelligence* 172(8-9):1158–1193.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: Annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. Association for Computational Linguistics, Denver, Colorado, pages 89–98.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics* 2:143.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics*. volume 2, pages 1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval temporal relation identification. In *SemEval*. pages 75–80.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *SemEval*. pages 57–62.

Exemplar Encoder-Decoder for Neural Conversation Generation

Gaurav Pandey, Danish Contractor, Vineet Kumar and Sachindra Joshi
IBM Research AI
New Delhi, India
{gpandey1, dcontrac, vineeku6, jsachind}@in.ibm.com

Abstract

In this paper we present the Exemplar Encoder-Decoder network (EED), a novel conversation model that learns to utilize *similar* examples from training data to generate responses. Similar conversation examples (context-response pairs) from training data are retrieved using a traditional TF-IDF based retrieval model. The retrieved responses are used to create exemplar vectors that are used by the decoder to generate the response. The contribution of each retrieved response is weighed by the similarity of corresponding context with the input context. We present detailed experiments on two large data sets and find that our method outperforms state of the art sequence to sequence generative models on several recently proposed evaluation metrics. We also observe that the responses generated by the proposed EED model are more informative and diverse compared to existing state-of-the-art method.

1 Introduction

With the availability of large datasets and the recent progress made by neural methods, variants of sequence to sequence learning (seq2seq) (Sutskever et al., 2014) architectures have been successfully applied for building conversational systems (Serban et al., 2016, 2017b). However, despite these methods being the state-of-the art frameworks for conversation generation, they suffer from problems such as lack of diversity in responses and generation of short, repetitive and uninteresting responses (Liu et al., 2016; Serban et al., 2016, 2017b). A large body of recent literature has focused on overcoming such challenges (Li et al., 2016a; Lowe et al., 2017).

In part, such problems arise as all information required to generate responses needs to be captured as part of the model parameters learnt from the training data. These model parameters alone may not be sufficient for generating natural conversations. Therefore, despite providing enormous amount of data, neural generative systems have been found to be ineffective for use in real world applications (Liu et al., 2016).

In this paper, we focus our attention on closed domain conversations. A characteristic feature of such conversations is that over a period of time, some conversation contexts[1] are likely to have occurred previously (Lu et al., 2017b). For instance, Table 1 shows some contexts from the Ubuntu dialog corpus. Each row presents an input dialog context with its corresponding gold response followed by a similar context and response seen in training data – as can be seen, contexts for "*installing dms*", "*sharing files*", "*blocking ufw ports*" have all occurred in training data. We hypothesize that being able to refer to training responses for previously seen similar contexts could be a helpful signal to use while generating responses.

In order to exploit this aspect of closed domain conversations we build our neural encoder-decoder architecture called the *Exemplar Encoder Decoder (EED)*, that learns to generate a response for a given context by exploiting *similar* contexts from training conversations. Thus, instead of having the seq2seq model learn patterns of language only from aligned parallel corpora, we assist the model by providing it *closely* related (*similar*) samples from the training data that it can *refer* to while generating text.

Specifically, given a context c, we retrieve a set

[1]We use the phrase "dialog context", "conversation context" and "context" interchangeably throughout the paper.

	Input Context	Gold Response		Similar Context in training data	Associated Response
U1 **U2**	if you want autologin install a dm of some sort what is a dm	lightdm, **gdm**, **kdm**, **xdm**, slim, etc.	**U1** **U2**	if you're running a dm, it will probably restart x whats a dm?	e.g. **gdm**, **kdm**, **xdm**
U1 **U2**	is it possible to share a file in one user's home directory with another user? if you set permissions (to 'group','other' or with an acl)	so **chmod** 777 should do it, right?	**U1** **U2**	howto set right permission for my home directory? but which is the syntax to set permission for my user in my home user directory ?	**chmod** and chown? u mean that sintax
U1 **U2**	is there a way to block all ports in ufw and only allow the ports that have been allowed? try to get familiar with configuring iptables	do i need to use iptables in order to use **ufw?**	**U1** **U2**	is ufw blocking connections to all ports by default? no, all ports are open by default.	how do i block all ports with **ufw**?
U1 **U2**	how do i upgrade on php beyond 5.3.2 on ubuntu using apt-get ? ? ? this version is a bit old which version of ubuntu are you using?	lucid, **10.04 ubuntu** 10.04.4 lts	**U1** **U2** **U1**	hello!, how can i upgrade apt-get?(i have version 0.7.9 installed but i need to update to latest) sudo apt-get upgrade apt-get what version of ubuntu do you have?	I'm using **ubuntu** server **10.04** 64

Table 1: Sample input contexts and corresponding gold responses from Ubuntu *validation* dataset along with similar contexts seen in training data and their corresponding responses. We refer to training data as training data for the Ubuntu corpus. The highlighted words are common between the gold response and the exemplar response.

of context-response pairs $(c^{(k)}, r^{(k)})$, $1 \leq k \leq K$ using an inverted index of training data. We create an *exemplar vector* $e^{(k)}$ by encoding the response $r^{(k)}$ (also referred to as exemplar response) along with an encoded representation of the current context c. We then learn the *importance* of each exemplar vector $e^{(k)}$ based on the likelihood of it being able to generate the ground truth response. We believe that $e^{(k)}$ may contain information that is helpful in generating the response. Table 1 highlights the words in exemplar responses that appear in the ground truth response as well.

Contributions: We present a novel Exemplar Encoder-Decoder (EED) architecture that makes use of similar conversations, fetched from an index of training data. The retrieved context-response pairs are used to create *exemplar* vectors which are used by the decoder in the EED model, to learn the importance of training context-response pairs, while generating responses. We present detailed experiments on the publicly benchmarked Ubuntu dialog corpus data set (Lowe et al., 2015) as well a large collection of more than 127,000 technical support conversations. We compare the performance of the EED model with the existing state of the art generative models such as HRED (Serban et al., 2016) and VHRED (Serban et al., 2017b). We find that our model out-performs these models on a wide variety of metrics such as the recently proposed *Activity Entity* metrics (Serban et al., 2017a) as well as Embedding-based metrics (Lowe et al., 2015). In addition, we present qualitative insights into our results and we find that exemplar based responses are more informative and diverse.

The rest of the paper is organized as follows. Section 2 briefly describes the recent works in neural dialogue generation The details of the proposed EED model for dialogue generation are described in detail in Section 3. In Section 4, we describe the datasets as well as the details of the models used during training. We present quantitative and qualitative results of EED model in Section 5.

2 Related Work

In this section, we compare our work against other data-driven end-to-end conversation models. End-to-end conversation models can be further classified into two broad categories — *generation* based models and *retrieval* based models.

Generation based models cast the problem of dialogue generation as a sequence to sequence learning problem. Initial works treat the entire context as a single long sentence and learn an encoder-decoder framework to generate response word by word (Shang et al., 2015; Vinyals and Le, 2015). This was followed by work that models context better by breaking it into conversation history and last utterance (Sordoni et al., 2015b). Context was further modeled effectively by using a hierarchical encoder decoder (HRED) model which first learns a vector representation of each utterance and then combines these representations to learn vector representation of context (Serban et al., 2016). Later, an alternative hierarchical model called VHRED (Serban et al., 2017b) was proposed, where generated responses were conditioned on latent variables. This leads to more informative responses and adds diversity to response generation. Models that explicitly incorporate diversity in response generation have also been studied in literature (Li et al., 2016b; Vijayakumar et al., 2016; Cao and Clark, 2017; Zhao et al., 2017).

Our work differs from the above as none of these above approaches utilize similar conversation contexts observed in the training data explicitly.

Retrieval based models on the other hand treat the conversation context as a query and obtain a set of responses using information retrieval (IR) techniques from the conversation logs (Ji et al., 2014). There has been further work where the responses are further ranked using a deep learning based model (Yan et al., 2016a,b; Qiu et al., 2017). On the other hand of the spectrum, end-to-end deep learning based rankers have also been employed to generate responses (Wu et al., 2017; Henderson et al., 2017). Recently a framework has also been proposed that uses a discriminative dialog network that ranks the candidate responses received from a response generator network and trains both the networks in an end to end manner (Lu et al., 2017a).

In contrast to the above models, we use the input contexts as well as the retrieved responses for generating the final responses. Contemporaneous to our work, a generative model for machine translation that employs retrieved translation pairs has also been proposed (Gu et al., 2017). We note that while the underlying premise of both the papers remains the same, the difference lies in the mechanism of incorporating the retrieved data.

3 Exemplar Encoder Decoder

3.1 Overview

A conversation consists of a sequence of utterances. At a given point in the conversation, the utterances expressed prior to it are jointly referred to as the context. The utterance that immediately follows the context is referred to as the response. As discussed in Section 1, given a conversational context, we wish to to generate a response by utilizing similar context-response pairs from the training data. We retrieve a set of K exemplar context-response pairs from an inverted index created using the training data in an off-line manner. The input and the retrieved context-response pairs are then fed to the Exemplar Encoder Decoder (EED) network. A schematic illustration of the EED network is presented in Figure 1. The EED encoder combines the input context and the retrieved responses to create a set of *exemplar vectors*. The EED decoder then uses the exemplar vectors based on the similarity between the input context and retrieved contexts to generate a response. We now provide details of each of these modules.

3.2 Retrieval of Similar Context-Response Pairs

Given a large collection of conversations as $(context,\ response)$ pairs, we index each response and its corresponding context in $tf - idf$ vector space. We further extract the last *turn* of a conversation and index it as an additional attribute of the context-response document pairs so as to allow directed queries based on it.

Given an input context c, we construct a query that weighs the last utterance in the context twice as much as the rest of the context and use it to retrieve the top-k similar context-response pairs from the index based on a BM25 (Robertson et al., 2009) retrieval model. These retrieved pairs form our exemplar context-response pairs $(c^{(k)}, r^{(k)})$, $1 \leq k \leq K$.

3.3 Exemplar Encoder Network

Given the exemplar pairs $(c^{(k)}, r^{(k)})$, $1 \leq k \leq K$ and an input context-response pair (c, r), we feed the input context c and the exemplar contexts $c^{(1)}, \ldots, c^{(K)}$ through an encoder to generate the embeddings as given below:

$$c_e = Encode_c(c)$$
$$c_e^{(k)} = Encode_c(c^{(k)}),\ 1 \leq k \leq K$$

Note that we do not constrain our choice of encoder and that any parametrized differentiable architecture can be used as the encoder to generate the above embeddings. Similarly, we feed the exemplar responses $r^{(1)}, \ldots, r^{(K)}$ through a response encoder to generate response embeddings $r_e^{(1)}, \ldots, r_e^{(K)}$, that is,

$$r_e^{(k)} = Encode_r(r^{(k)}),\ 1 \leq k \leq K \qquad (1)$$

Next, we concatenate the exemplar response encoding $r_e^{(k)}$ with an encoded representation of current context c_e as shown in equation 2 to create the exemplar vector $e^{(k)}$. This allows us to include in-

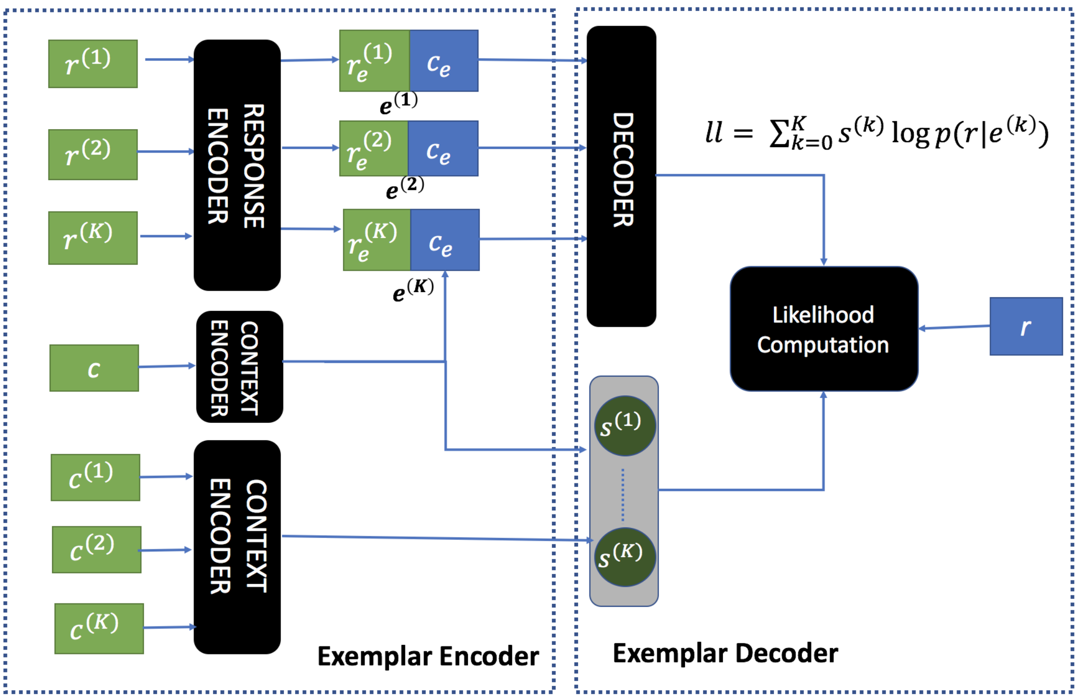


Figure 1: A schematic illustration of the EED network. The input context-response pair is (c, r), while the exemplar context-response pairs are $(c^{(k)}, r^{(k)})$, $1 \leq k \leq K$.

formation about similar responses along with the encoded input context representation.

$$e^{(k)} = [c_e; r_e^{(k)}],\ 1 \leq k \leq K \tag{2}$$

The exemplar vectors $e^{(k)}$, $1 \leq k \leq K$ are further used by the decoder for generating the ground truth response as described in the next section.

3.4 Exemplar Decoder Network

Recall that we want the exemplar responses to help generate the responses based on how similar the corresponding contexts are with the input context. More similar an exemplar context is to the input context, higher should be its effect in generating the response. To this end, we compute the similarity scores $s^{(k)}$, $1 \leq k \leq K$ using the encodings computed in Section 3.3 as shown below.

$$s^{(k)} = \frac{\exp(c_e^{\mathrm{T}} c_e^{(k)})}{\sum_{l=1}^{K} \exp(c_e^{\mathrm{T}} c_e^{(l)})} \tag{3}$$

Next, each exemplar vector $e^{(k)}$ computed in Section 3.3, is fed to a decoder, where the decoder is responsible for predicting the ground truth response from the exemplar vector. Let $p^{dec}(r|e^{(k)})$ be the distribution of generating the ground truth response given the exemplar embedding. The objective function to be maximized, is expressed as a function of the scores $s^{(k)}$, the decoding distribution p^{dec} and the exemplar vectors $e^{(k)}$ as shown below:

$$ll = \sum_{k=1}^{K} s^{(k)} \log p^{dec}(r|e^{(k)}) \tag{4}$$

Note that we weigh the contribution of each exemplar vector to the final objective based on how similar the corresponding context is to the input context. Moreover, the similarities are differentiable function of the input and hence, trainable by back propagation. The model should learn to assign higher similarities to the exemplar contexts, whose responses are helpful for generating the correct response.

The model description uses encoder and decoder networks that can be implemented using any differentiable parametrized architecture. We discuss our choices for the encoders and decoder in the next section.

3.5 The Encoders and Decoder

In this section, we discuss the various encoders and the decoder used by our model. The conversation context consists of an ordered sequence of utterances and each utterance can be further viewed as a sequence of words. Thus, context can be viewed as having multiple levels of hierarchies—at the word level and then at the utterance (sentence) level. We use a hierarchical recurrent encoder—popularly employed as part of the HRED framework for generating responses and query suggestions (Sordoni et al., 2015a; Serban et al., 2016, 2017b). The word-level encoder encodes the vector representations of words of an utterance to an utterance vector. Finally, the utterance-level encoder encodes the utterance vectors to a context vector.

Let $(\mathbf{u}_1, \ldots, \mathbf{u}_N)$ be the utterances present in the context. Furthermore, let $(w_{n1}, \ldots, w_{nM_n})$ be the words present in the n^{th} utterance for $1 \leq n \leq N$. For each word in the utterance, we retrieve its corresponding embedding from an embedding matrix. The word embedding for w_{nm} will be denoted as w_{enm}. The encoding of the n^{th} utterance can be computed iteratively as follows:

$$h_{nm} = f_1(h_{nm-1}, w_{enm}),\ 1 \leq m \leq M_n \qquad (5)$$

We use an LSTM (Hochreiter and Schmidhuber, 1997) to model the above equation. The last hidden state h_{nM_n} is referred to as the utterance encoding and will be denoted as h_n.

The utterance-level encoder takes the utterance encodings $h_1, \ldots, h_N$ as input and generates the encoding for the context as follows:

$$c_{en} = f_2(c_{en-1}, h_n),\ 1 \leq n \leq N \qquad (6)$$

Again, we use an LSTM to model the above equation. The last hidden state c_{eN} is referred to as the context embedding and is denoted as c_e.

A single level LSTM is used for embedding the response. In particular, let $(w_1, \ldots, w_M)$ be the sequence of words present in the response. For each word w, we retrieve the corresponding word embedding w_e from a word embedding matrix. The response embedding is computed from the word embeddings iteratively as follows:

$$r_{em} = g(r_{em-1}, w_{em}),\ 1 \leq m \leq M \qquad (7)$$

Again, we use an LSTM to model the above equation. The last hidden state r_{em} is referred to as the response embedding and is denoted as r_e.

4 Experimental Setup

4.1 Datasets

4.1.1 Ubuntu Dataset

We conduct experiments on Ubuntu Dialogue Corpus (Lowe et al., 2015)(v2.0)[2]. Ubuntu dialogue corpus has about 1M context response pairs along with a label. The label value 1 indicates that the response associated with a context is the correct response and is incorrect otherwise. As we are only interested in positive labeled data we work with $label = 1$. Table 2 depicts some statistics for the dataset.

	Size
Training Pairs	499,873
Validation Pairs	19,560
Test Pairs	18,920
$\lvert V \rvert$	538,328

Table 2: Dataset statistics for Ubuntu Dialog Corpus v2.0 (Lowe et al., 2015), where $|V|$ represents the size of vocabulary.

4.1.2 Tech Support Dataset

We also conduct our experiments on a large technical support dataset with more than 127K conversations. We will refer to this dataset as Tech Support dataset in the rest of the paper. Tech Support dataset contains conversations pertaining to an employee seeking assistance from an agent (technical support) — to resolve problems such as password reset, software installation/licensing, and wireless access. In contrast to Ubuntu dataset, this dataset has clearly two distinct users — employee and agent. In our experiments we model the *agent* responses only.

For each conversation in the tech support data, we sample context and response pairs to create a dataset similar to the Ubuntu dataset format. Note that multiple context-response pairs can be generated from a single conversation. For each conversation, we sample 25% of the possible context-response pairs. We create validation pairs by selecting 5000 conversations randomly and sampling context response pairs). Similarly, we create test pairs from a different subset of 5000 conversations. The remaining conversations are used to

[2]https://github.com/rkadlec/ubuntu-ranking-dataset-creator

create training context-response pairs. Table 3 depicts some statistics for this dataset:

	Size
Conversations	127,466
Training Pairs	204,808
Validation Pairs	8,738
Test Pairs	8,756
$\|V\|$	293,494

Table 3: Dataset statistics for Tech Support dataset.

4.2 Model and Training Details

The EED and HRED models were implemented using the PyTorch framework (Paszke et al., 2017). We initialize the word embedding matrix as well as the weights of context and response encoders from the standard normal distribution with mean 0 and variance 0.01. The biases of the encoders and decoder are initialized with 0. The word embedding matrix is shared by the context and response encoders. For Ubuntu dataset, we use a word embedding size of 600, whereas the size of the hidden layers of the LSTMs in context and response encoders and the decoder is fixed at 1200. For Tech support dataset, we use a word embedding size of 128. Furthermore, the size of the hidden layers of the multiple LSTMs in context and response encoders and the decoder is fixed at 256. A smaller embedding size was chosen for the Tech Support dataset since we observed much less diversity in the responses of the Tech Support dataset as compared to Ubuntu dataset.

Two different encoders are used for encoding the input context (not shown in Figure 1 for simplicity). The output of the first context encoder is concatenated with the exemplar response vectors to generate exemplar vectors as detailed in Section 3.3. The output of the second context encoder is used to compute the scoring function as detailed in Section 3.4. For each input context, we retrieve 5 similar context-response pairs for Ubuntu dataset and 3 context-response pairs for Tech support dataset using the tf-idf mechanism discussed in Section 3.2.

We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $1e-4$ for training the model. A batch size of 20 samples was used during training. In order to prevent overfitting, we use early stopping with log-likelihood on validation set as the stopping criteria. In order to generate the samples using the proposed EED model, we identify the exemplar context that is most similar to the input context based on the learnt scoring function discussed in Section 3.4. The corresponding exemplar vector is fed to the decoder to generate the response. The samples are generated using a beam search with width 5. The average per-word log-likelihood is used to score the beams.

5 Results & Evaluation

5.1 Quantitative Evaluation

5.1.1 Activity and Entity Metrics

A traditional and popular metric used for comparing a generated sentence with a ground truth sentence is BLEU (Papineni et al., 2002) and is frequently used to evaluate machine translation. The metric has also been applied to compute scores for predicted responses in conversations, but it has been found to be less indicative of actual performance (Liu et al., 2016; Sordoni et al., 2015a; Serban et al., 2017a), as it is extremely sensitive to the exact words in the ground truth response, and gives equal importance to stop words/phrases and informative words.

Serban et al. (2017a) recently proposed a new set of metrics for evaluating dialogue responses for the Ubuntu corpus. It is important to highlight that these metrics have been specifically designed for the Ubuntu corpus and evaluate a generated response with the ground truth response by comparing the coarse level representation of an utterance (such as entities, activities, Ubuntu OS commands). Here is a brief description of each metric:

- **Activity**: Activity metric compares the activities present in a predicted response with the ground truth response. Activity can be thought of as a verb. Thus, all the verbs in a response are mapped to a set of manually identified list of 192 verbs.

- **Entity**: This compares the technical entities that overlap with the ground truth response. A total of 3115 technical entities is identified using public resources such as Debian package manager APT.

	Activity			Entity			Tense	Cmd
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**Acc.**	**Acc.**
LSTM*	1.7	1.03	1.18	1.18	0.81	0.87	14.57	94.79
VHRED*	**6.43**	4.31	4.63	3.28	2.41	2.53	20.2	92.02
HRED*	5.93	4.05	4.34	2.81	2.16	2.22	22.2	92.58
EED	6.42	**4.77**	**4.87**	**3.8**	**2.91**	**2.99**	**31.73**	**95.06**

Table 4: Activity & Entity metrics for the Ubuntu corpus. LSTM*, HRED* & VHRED* as reported by Serban et al. (2017a).

- **Tense**: This measure compares the time tense of ground truth with predicted response.

- **Cmd**: This metric computes accuracy by comparing commands identified in ground truth utterance with a predicted response.

Table 4 compares our model with other recent generative models (Serban et al., 2017a) — LSTM (Shang et al., 2015), HRED (Serban et al., 2016) & VHRED (Serban et al., 2017b).We do not compare our model with Multi-Resolution RNN (MRNN) (Serban et al., 2017a), as MRNN explicitly utilizes the activities and entities during the generation process. In contrast, the proposed EED model and the other models used for comparison are agnostic to the activity and entity information. We use the standard script[3] to compute the metrics.

The EED model scores better than generative models on almost all of the metrics, indicating that we generate more *informative* responses than other state-of-the-art generative based approaches for Ubuntu corpus. The results show that responses associated with similar contexts may contain the activities and entities present in the ground truth response, and thus help in response generation. This is discussed further in Section 5.2. Additionally, we compared our proposed EED with a retrieval only baseline. The retrieval baseline achieves an activity F1 score of 4.23 and entity F1 score of 2.72 compared to 4.87 and 2.99 respectively achieved by our method on the Ubuntu corpus.

The Tech Support dataset is not evaluated using the above metrics, since activity and entity information is not available for this dataset.

[3]https://github.com/julianser/Ubuntu-Multiresolution-Tools/blob/master/ActEntRepresentation/eval_file.sh

5.1.2 Embedding Metrics

Embedding metrics (Lowe et al., 2017) were proposed as an alternative to word by word comparison metrics such as BLEU. We use pre-trained Google news word embeddings[4] similar to Serban et al. (2017b), for easy reproducibility as these metrics are sensitive to the word embeddings used. The three metrics of interest utilize the word vectors in ground truth response and a predicted response and are discussed below:

- **Average**: Average word embedding vectors are computed for the candidate response and ground truth. The cosine similarity is computed between these averaged embeddings. High similarity gives as indication that ground truth and predicted response have similar words.

- **Greedy**: Greedy matching score finds the most similar word in predicted response to ground truth response using cosine similarity.

- **Extrema**: Vector extrema score computes the maximum or minimum value of each dimension of word vectors in candidate response and ground truth.

Of these, the embedding average metric is the most reflective of performance for our setup. The extrema representation, for instance, is very sensitive to text length and becomes ineffective beyond single length sentences(Forgues et al., 2014). We use the publicly available script[5] for all our computations. As the test outputs for HRED are not available for Technical Support dataset, we use our

[4]GoogleNews-vectors-negative300.bin from `https://code.google.com/archive/p/word2vec/`

[5]`https://github.com/julianser/hed-dlg-truncated/blob/master/Evaluation/embedding_metrics.py`

	Ubuntu			Tech Support		
Model	**Average**	**Greedy**	**Extrema**	**Average**	**Greedy**	**Extrema**
HRED	0.5131	0.4103	**0.3263**	0.6372	0.5100	0.4355
EED	**0.5528**	**0.4172**	0.3134	**0.6420**	**0.5125**	**0.4356**

Table 5: Embedding Metrics (Lowe et al., 2015) for Ubuntu and Technical Support Corpus.

	Ubuntu			Tech Support		
Model	**distinct-1**	**distinct-2**	**distinct-3**	**distinct-1**	**distinct-2**	**distinct-3**
HRED	1154	8668	22417	645	3510	6365
EED	**1173**	**9572**	**29447**	**732**	**4149**	**7877**

Table 6: The number of unique tokens, token-pairs and token-triplets for Ubuntu and Technical Support Corpus.

	Input Context (C) and exemplar response (R)	**Generated/Ground Truth Responses**
1	**C:** i want to check ntfs file systems. is fsck ok with ntfs. $\rightarrow$ no , you 'd need to use chkdsk from windows. $\rightarrow$ **R:** i already symlinked ntfsfix to fsck. **ntfs** and am scanning the unmounted devices	**HRED:** i know , but i dont know how to do that **VHRED:** how can i do that ? **EED:** i am trying to use **ntfs** 3g i have a windows partition. **GT:** i dont have windows installed
2	**C:** anyone help me ? $\rightarrow$ seems that its detecting grub , have you tried checking continue without installing grub ? **R:** try the live **cd**... you ca n't boot into linux unless **grub install**	**HRED:** yes i did , but i have no idea how to fix it **VHRED:** yes i did. i dont know how to do it **EED:** nope not yet. **grub install cd** is not working **GT:** no i did not.
3	**C:** any ideas on my mplayer problem ? $\rightarrow$ reinstall. apt-get purge mplayer then reinstall **R:** but half my system depends on this **package** ...	**HRED:** I did that , but it didn't work **VHRED:** I dont know how to do that **EED:** does n t work. **package** does n t work **GT:** that does n't seem as a solution
4	**C:** any of you guys in here know how to take a package on launchpad and recompile it and submit to your own ppa ? https : //launchpad.net/ mojocode/+archive /ppa ¡ ¡ i 'm trying to take the nginx package and install the rtmp module $\rightarrow$ could n't you download the source and just commit it to your own ? **R:** i **need** prop . drivers	**HRED:** I don't know how to do that , I just want to know if there is a way to do it **VHRED:** no , I just want to make it so I can use it **EED:** i **need** to get the source package. package manager is not available **GT:** so i figured how how to copy into my private ppa . any idea how to download those source files maybe and make modifications to resubmit and recompile ? https : //launchpad.net / blendedbyus/+archive/master/+packages
5	**C:** hi people , can you help me uninstall a package ? i did dpkg -i virtualbox-4.2_4.2.2-81494 ubuntu precise _amd64.deb. how do i uninstall it , i ca n't find the package from apt-get $\rightarrow$ sudo apt-get purge virtualbox should work fine $\rightarrow$ package virtualbox is not installed , so not removed $\rightarrow$ does n't work : (**R: sudo dpkg** – remove – force < package >	**HRED:** what version of ubuntu ? **VHRED:** did you try sudo apt-get install virtualbox-ose **EED: sudo dpkg** reconfigure package package manager **GT:** if you ca n't find the package ; sudo aptitude search virtualbox

Table 7: Contexts, exemplar responses and responses generated by HRED, VHRED and the proposed EED model. We use the published responses for HRED and VHRED. GT indicates the ground truth response. The change of turn is indicated by $\rightarrow$. The highlighted words in **bold** are common between the exemplar response and the response predicted by EED.

own implementation of HRED. Table 5 compares our model with HRED, and depicts that our model scores better on all metrics for Technical Support dataset, and on majority of the metrics for Ubuntu dataset.

We note that the improvement achieved by the

EED model on activity and entity metrics are much more significant than those on embedding metrics. This suggests that the EED model is better able to capture the specific information (objects and actions) present in the conversations.

Finally, we evaluate the diversity of the generated responses for EED against HRED by counting the number of unique tokens, token-pairs and token-triplets present in the generated responses on Ubuntu and Tech Support dataset. The results are shown in Table 6. As can be observed, the responses in EED have a larger number of distinct tokens, token-pairs and token-triplets than HRED, and hence, are arguably more diverse.

5.2 Qualitative Evaluation

Table 7 presents the responses generated by HRED, VHRED and the proposed EED for a few selected contexts along with the corresponding similar exemplar responses. As can be observed from the table, the responses generated by EED tend to be more specific to the input context as compared to the responses of HRED and VHRED. For example, in conversations 1 and 2 we find that both HRED and VHRED generate simple generic responses whereas EED generates responses with additional information such as the type of disk partition used or a command not working. This is also confirmed by the quantitative results obtained using activity and entity metrics in the previous section. We further observe that the exemplar responses contain informative words that are utilized by the EED model for generating the responses as highlighted in Table 7.

6 Conclusions

In this work, we propose a deep learning method, Exemplar Encoder Decoder (EED), that given a conversation context uses similar contexts and corresponding responses from training data for generating a response. We show that by utilizing this information the system is able to outperform state of the art generative models on publicly available Ubuntu dataset. We further show improvements achieved by the proposed method on a large collection of technical support conversations.

While in this work, we apply the exemplar encoder decoder network on conversational task, the method is generic and could be used with other tasks such as question answering and machine translation. In our future work we plan to extend the proposed method to these other applications.

Acknowledgements

We are grateful to the anonymous reviewers for their comments that helped in improving the paper.

References

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 182–187.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2017. Search engine guided non-parametric neural machine translation. *arXiv preprint arXiv:1705.07267*.

Matthew Henderson, Rami Al-Rfou', Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

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

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017a. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 314–324.

Yichao Lu, Phillip Keung, Shaonan Zhang, Jason Sun, and Vikas Bhardwaj. 2017b. A practical approach to dialogue response generation in closed domains. *arXiv preprint arXiv:1703.09439*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *ACL*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Yu Wu, Wei Wu, Chen Xing, Can Xu, Zhoujun Li, and Ming Zhou. 2017. A sequential matching framework for multi-turn response selection in retrieval-based chatbots. *CoRR*, abs/1710.11344.

Rui Yan, Yiping Song, and Hua Wu. 2016a. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR*.

Rui Yan, Yiping Song, Xiangyang Zhou, and Hua Wu. 2016b. "shall i be your chat companion?": Towards an online human-computer conversation system. In *CIKM*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 654–664.

DialSQL: Dialogue Based Structured Query Generation

Izzeddin Gur and **Semih Yavuz** and **Yu Su** and **Xifeng Yan**
Department of Computer Science, University of California, Santa Barbara
{izzeddingur,syavuz,ysu,xyan}@cs.ucsb.edu

Abstract

The recent advance in deep learning and semantic parsing has significantly improved the translation accuracy of natural language questions to structured queries. However, further improvement of the existing approaches turns out to be quite challenging. Rather than solely relying on algorithmic innovations, in this work, we introduce DialSQL, a dialogue-based structured query generation framework that leverages human intelligence to boost the performance of existing algorithms via user interaction. DialSQL is capable of identifying potential errors in a generated SQL query and asking users for validation via simple multi-choice questions. User feedback is then leveraged to revise the query. We design a generic simulator to bootstrap synthetic training dialogues and evaluate the performance of DialSQL on the WikiSQL dataset. Using SQLNet as a black box query generation tool, DialSQL improves its performance from 61.3% to 69.0% using only 2.4 validation questions per dialogue.

1 Introduction

Building natural language interfaces to databases (NLIDB) is a long-standing open problem and has significant implications for many application domains. It can enable users without SQL programming background to freely query the data they have. For this reason, generating SQL queries from natural language questions has gained a renewed interest due to the recent advance in deep learning and semantic parsing (Yaghmazadeh et al., 2017; Zhong et al., 2017; Xu et al., 2017; Iyer et al., 2017).

While new methods race to achieve the state-of-the-art performance on NLIDB datasets such as WikiSQL (Xu et al., 2017; Zhong et al., 2017), the accuracy is still not high enough for real use. For example, SQLNet (Xu et al., 2017) achieves 61.3% accuracy on WikiSQL. After analyzing the error cases of Seq2SQL (Zhong et al., 2017) and SQLNet, we recognized that many wrong translations cannot be easily corrected due to the lack of external knowledge and semantic understanding.

In this paper, we aim to alleviate the aforementioned problem by putting human users in the loop. Previous human-in-the-loop NLIDBs (Li and Jagadish, 2014; Yaghmazadeh et al., 2017) rely on users to carefully go through a generated SQL query and revise it accordingly, which is not feasible for users who do not know the SQL language. Instead, we resort to a different approach by introducing a goal-oriented dialogue model, DialSQL, that interacts with users to extract and correct potential errors in the generated queries.

Given a SQL query generated from a natural language question, we assume any segment, or span, of the generated query such as a `WHERE` clause can be potentially erroneous. The goal of DialSQL is to extract the erroneous spans and ask users multi-choice questions to validate and correct these errors. DialSQL is based on a hierarchical encoder-decoder architecture with attention and pointer mechanisms. The model first encodes each turn of interaction and runs a dialogue level RNN network on the dialogue history. The output of the network is then used to predict the error category, i.e., whether it is a selection, projection, or aggregation error. Conditioned on the error category, the output of a second RNN is used to predict the start and end positions of the error span by pointing to the query tokens. Finally, candidate choices are decoded from the error category and span representations. Following previous

Natural Language Question :	What are the countries that joined the NATO before 2004 ?
Ground Truth SQL Query :	SELECT country WHERE date of join < 2004
Initial SQL Query :	SELECT count (country) WHERE date of join = 2004

Turn Side	NL Expression using Rule-Based Templates	Response
System:	Could you please confirm if you are asking **total number of the results**?	validate_agg (1, 2)
User:	No, that's not true.	negate()
System:	Could you please select one of the following options : (i) **average number of results**, (ii) **all the results**?	offer_agg (average, no_agg)
User:	It should be **all the results**.	select(2)
System:	Could you please confirm if **date of join is 2004** ?	validate_where_change(6, 11)
User:	That's wrong.	negate()
System:	Could you please select one of the following options : (i) **date of join is before 2004**, (ii) **date of invitation is before 2004**?	offer_where (date of join < 2004, date of invitation < 2004)
User:	**Date of join is before 2004**.	select(1)

Table 1: DialSQL model running example. Initial SQL query is generated by running a black box model on the question. Natural language (NL) expressions are generated using a template based method. Substrings in red represent the error spans and substrings in blue represent the choices offered. Each response is accompanied with natural language utterances for clarity.

work (Zhong et al., 2017; Xu et al., 2017), we only use column names and do not utilize table values.

How to train and evaluate DialSQL become two challenging issues due to the lack of error data and interaction data. In this work, we construct a simulator to generate simulated dialogues, a general approach practiced by many dialogue studies. Inspired by the agenda-based methods for user simulation (Schatzmann et al., 2007), we keep an agenda of pending actions that are needed to induce the ground truth query. At the start of the dialogue, a new query is carefully synthesized by randomly altering the ground truth query and the agenda is populated by the sequence of altering actions. Each action consists of three sub-actions: (i) Pick an error category and extract a span; (ii) Raise a question; (iii) Update the query by randomly altering the span and remove the action from the agenda. Consider the example in Figure 1: Step-1 synthesizes the initial query by randomly altering the `WHERE` clause and `AGGREGATION`; Step-2 generates the simulated dialogue by validating the altered spans and offering the correct choice.

To evaluate our model, we first train DialSQL on the simulated dialogues. Initial queries for new questions are manufactured by running a black box SQL generation system on the new questions. When tested on the WikiSQL (Zhong et al., 2017) dataset, our model increases the query match accuracy of SQLNet (Xu et al., 2017) from 61.3% to 69.0% using on average 2.4 validation questions per query.

2 Related Work

Research on natural language interfaces to databases (NLIDBs), or semantic parsing, has spanned several decades. Early rule-based NLIDBs (Woods, 1973; Androutsopoulos et al., 1995; Popescu et al., 2003) employ carefully designed rules to map natural language questions to formal meaning representations like SQL queries. While having a high precision, rule-based systems are brittle when facing with language variations. The rise of statistical models (Zettlemoyer and Collins, 2005; Kate et al., 2005; Berant et al., 2013), especially the ongoing wave of neural network models (Yih et al., 2015; Dong and Lapata, 2016; Sun et al., 2016; Zhong et al., 2017; Xu et al., 2017; Guo and Gao, 2018; Yavuz et al., 2016), has enabled NLIDBs that are more robust to language variations. Such systems allow users to formulate questions with greater flexibility. However, although state-of-the-art systems have achieved a high accuracy of 80% to 90% (Dong and Lapata, 2016) on well-curated datasets like GEO (Zelle and Ray, 1996) and ATIS (Zettlemoyer and Collins, 2007), the best accuracies on datasets with questions formulated by real human users, e.g., WebQuestions (Berant et al., 2013), GraphQuestions (Su et al., 2016), and WikiSQL (Zhong et al., 2017), are still far from enough for real use, typically in the range of 20% to 60%.

Human-in-the-loop systems are a promising paradigm for building practical NLIDBs. A number of recent studies have explored this paradigm with two types of user interaction: coarse-grained and fine-grained. Iyer et al. (2017) and Li et

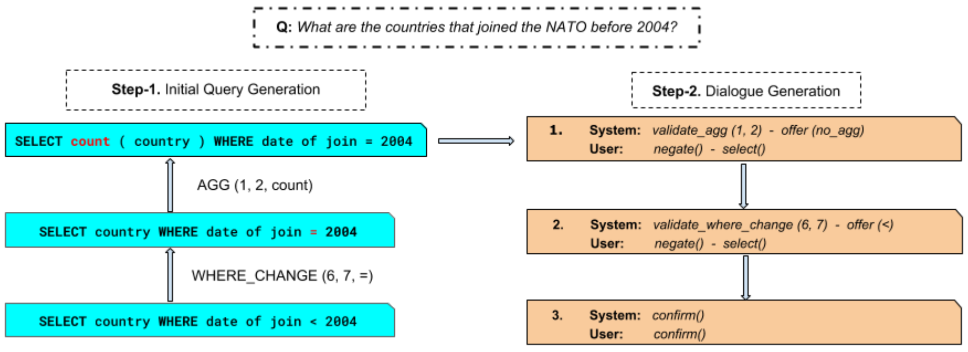

Q: What are the countries that joined the NATO before 2004?
Step-1. Initial Query Generation
Step-2. Dialogue Generation
SELECT count (country) WHERE date of join = 2004
AGG (1, 2, count)
SELECT country WHERE date of join = 2004
WHERE_CHANGE (6, 7, =)
SELECT country WHERE date of join < 2004
1. System: validate_agg (1, 2) - offer (no_agg)
User: negate() - select()
2. System: validate_where_change (6, 7) - offer (<)
User: negate() - select()
3. System: confirm()
User: confirm()

Figure 1: An instantiation of our dialogue simulation process. Step-1 synthesizes the initial query (top) by randomly altering the ground truth query (bottom). Step-2 generates the dialogue by validating the sequence of actions populated in Step-1 with the user. Each action is defined by the error category, start and end positions of the error span, and the random replacement, ex. `AGG (1, 2, count)`.

al. (2016) incorporate coarse-grained user interaction, i.e., asking the user to verify the correctness of the final results. However, for real-world questions, it may not always be possible for users to verify result correctness, especially in the absence of supporting evidence. Li and Jagadish (2014) and Yaghmazadeh et al. (2017) have shown that incorporating fine-grained user interaction can greatly improve the accuracy of NLIDBs. However, they require that the users have intimate knowledge of SQL, an assumption that does not hold for general users. Our method also enables fine-grained user interaction for NLIDBs, but we solicit user feedback via a dialogue between the user and the system.

Our model architecture is inspired by recent studies on hierarchical neural network models (Sordoni et al., 2015; Serban et al., 2015; Gur et al., 2017). Recently, Saha et al. (2018) propose a hierarchical encoder-decoder model augmented with key-value memory network for sequential question answering over knowledge graphs. Users ask a series of questions, and their system finds the answers by traversing a knowledge graph and resolves coreferences between questions. Our interactive query generation task significantly differs from their setup in that we aim to explicitly detect and correct the errors in the generated SQL query via a dialogue between our model and the user.

Agenda based user simulations have been investigated in goal-oriented dialogues for model training (Schatzmann et al., 2007). Recently, Seq2seq neural network models are proposed for user simulation (Asri et al., 2016) that utilize additional state tracking signals and encode dialogue turns in a more coarse way. We design a simulation method for the proposed task where we generate dialogues with annotated errors by altering queries and tracking the sequence of alteration steps.

3 Problem Setup and Datasets

We study the problem of building an interactive natural language interface to databases (INLIDB) for synthesizing SQL queries from natural language questions. In particular, our goal is to design a dialogue system to extract and validate potential errors in generated queries by asking users multi-choice questions over multiple turns. We will first define the problem formally and then explain our simulation strategy.

3.1 Interactive Query Generation

At the beginning of each dialogue, we are given a question $Q = \{q_1, q_2, \cdots, q_N\}$, a table with column names $T = \{T_1, T_2, \cdots, T_K\}$ where each name is a sequence of words, and an initial SQL query U generated using a black box SQL generation system. Each turn t is represented by a tuple of system and user responses, (S_t, R_t), and augmented with the dialogue history (list of previous turns), H_t. Each system response is a triplet of error category c, error span s, and a set of candidate choices C, i.e., $S_t = (c, s, C)$. An error category (Table 2) denotes the type of the error that we seek to correct and an error span is the segment of the current query that indicates the actual error. Candidate choices depend on the error category and range over the following possibilities: (i) a column name, (ii) an aggregation operator, or (iii) a where condition. User responses are represented by ei-

Error Category	Meaning in a dialogue
validate_sel	Validate the select clause
validate_agg	Validate the aggregation operator
validate_where_changed	Validate if a segment of a where clause is incorrect
validate_where_removed	Validate if a new where clause is needed
validate_where_added	Validate if an incorrect where clause exists
no_error	Validate if there is no remaining error

Table 2: The list of error categories and their explanations for our interactive query generation task.

ther an affirmation or a negation answer and an index c' to identify a choice. We define the *interactive query generation task* as a list of subtasks: at each turn t, (i) predict c, (ii) extract s from U, and (iii) decode C. The task is supervised and each subtask is annotated with labeled data.

Consider the example dialogue in Table 1. We first predict `validate_agg` as the error category and error span ($start = 1, end = 2$) is decoded by pointing to the *aggregation* segment of the query. Candidate choices, `(average, no_agg)`, are decoded using the predicted error category, predicted error span, and dialogue history. We use a template based natural language generation (NLG) component to convert system and user responses into natural language.

3.2 Dialogue Simulation for INLIDB

In our work, we evaluate our model on the WikiSQL task. Each example in WikiSQL consists of a natural language question and a table to query from. The task is to generate a SQL query that correctly maps the question to the given table. Unfortunately, the original WikiSQL lacks error data and user interaction data to train and evaluate DialSQL. We work around this problem by designing a simulator to bootstrap training dialogues and evaluate DialSQL on the test questions of WikiSQL.

Inspired by the agenda-based methods (Schatzmann et al., 2007), we keep an agenda of pending actions that are needed to induce the ground truth query. At the start of the dialogue, we synthesize a new query by randomly altering the ground truth query and populating the agenda by the sequence of altering actions. Each action launches a sequence of sub-actions: (i) Randomly select an error category and extract a related span from the current query, (ii) randomly generate a valid choice for the chosen span, and (iii) update the current query by replacing the span with the choice. The dialogue is initiated with the final query and a rule-based system interacts with a rule-based user simulator to populate the dialogue. The rule-based system follows the sequence of altering actions previously generated and asks the user simulator a single question at each turn. The user simulator has access to the ground truth query and answers each question by comparing the question (error span and the choice) with the ground truth.

Consider the example in Figure 1 where Step-1 synthesizes the initial query and Step-2 simulates a dialogue using the outputs of Step-1. Step-1 first randomly alters the `WHERE` clause; the operator is replaced with a random operator. The updated query is further altered and the final query is passed to Step-2. In Step-2, the system starts with validating the aggregation with the user simulator. In this motivating example, the aggregation is incorrect and the user simulator negates and selects the offered choice. During training, there is only a single choice offered and DialSQL trains to produce this choice; however, during testing, it can offer multiple choices. In the next step, the system validates the `WHERE` clause and generates a `no_error` action to issue the generated query. At the end of this process, we generate a set of labeled dialogues by executing Step-1 and Step-2 consecutively. DialSQL interacts with the same rule-based simulator during testing and the SQL queries obtained at the end of the dialogues are used to evaluate the model.

4 Dialogue Based SQL Generation

In this section, we present our DialSQL model and describe its operation in a fully supervised setting. DialSQL is composed of three layers linked in a hierarchical structure where each layer solves a different subtask : (i) Predicting error category, (ii) Decoding error span, and (iii) Decoding candidate choices (illustrated in Figure 2). Given a (Q, T, U) triplet, the model first encodes Q, each column name $T_i \in T$, and query U into vector representations in parallel using Recurrent Neural Networks (RNN). Next, the first layer of the model encodes the dialogue history with an RNN and predicts the error category from this encoding. The second layer is conditioned on the error category and decodes the start and end positions of the error span by attending over the outputs of query encoder. Finally, the last layer is conditioned on both error category and error span and decodes a list of choices to offer to the user.

4.1 Preliminaries and Notation

Each token w is associated with a vector e_w from rows of an embeddings matrix E. We aim at obtaining vector representations for question, table headers, and query, then generating error category, error span, and candidate choices.

For our purposes, we use GRU units (Cho et al., 2014) in our RNN encoders which are defined as

$$h_t = f(x_t; h_{t-1})$$

where h_t is the hidden state at time t. f is a nonlinear function operating on input vector x_t and previous state h_{t-1}. We refer to the last hidden state of an RNN encoder as the encoding of a sequence.

4.2 Encoding

The core of our model is a hierarchical encoder-decoder neural network that encodes dialogue history and decodes errors and candidate choices at the end of each user turn. The input to the model is the previous system turn and the current user turn and the output is the next system question.

Encoding Question, Column Names, and Query. Using decoupled RNNs (Enc), we encode natural language question, column names, and query sequences in parallel and produce outputs and hidden states. o^Q, o^{T_i}, and o^U denote the sequence of hidden states at each step and h^Q, h^{T_i}, and h^U denote the last hidden states of question, column name, and query encoders, respectively. Parameters of the encoders are decoupled and only the word embedding matrix E is shared.

Encoding System and User Turns Since there is only a single candidate choice during training, we ignore the index and encode user turn by doing an embedding lookup using the validation answer (affirmation or negation). Each element (error category, error span, and candidate choice) of the system response is encoded by doing an embedding lookup and different elements are used as input at different layers of our model.

Encoding Dialogue History At the end of each user turn, we first concatenate the previous error category and the current user turn encodings to generate the turn level input. We employ an RNN to encode dialogue history and current turn into a fixed length vector as

$$\begin{aligned} h_0^{D_1} &= h^Q \\ o_t^{D_1}, g_t^{D_1} &= Enc([E_c, E_a]) \\ h_t^{D_1} &= [Attn(g_t^{D_1}, H^T), o_t^D] \end{aligned}$$

where $[.]$ is vector concatenation, E_c is the error category encoding, E_a is the user turn encoding, $h_0^{D_1}$ is the initial hidden state, and $h_t^{D_1}$ is the current hidden state. $Attn$ is an attention layer with a bilinear product defined as in (Luong et al., 2015)

$$Attn(h, O) = \sum softmax(tanh(hWO)) * O$$

where W is attention parameter.

4.3 Predicting Error Category

We predict the error category by attending over query states using the output of the dialogue encoder as

$$\begin{aligned} c_t &= tanh(Lin([Attn(h_t^{D_1}, O^U), h_t^{D_1}])) \\ l_t &= softmax(c_t \cdot E(C)) \end{aligned}$$

where Lin is a linear transformation, $E(C)$ is a matrix with error category embeddings, and l_t is the probability distribution over categories.

4.4 Decoding Error Span

Consider the case in which there are more than one different `WHERE` clauses in the query and each clause has an error. In this case, the model needs to monitor previous error spans to avoid decoding the same error. DialSQL runs another RNN to generate a new dialogue encoding to solve the aforementioned problem as

$$\begin{aligned} h_0^{D_2} &= h^Q \\ o_t^{D_2}, g_t^{D_2} &= Enc(E_c) \\ h_t^{D_2} &= [Attn(g_t^{D_2}, H^T) o_t^{D_2}] \end{aligned}$$

where $h_0^{D_2}$ is the initial hidden state, and $h_t^{D_2}$ is the current hidden state. Start position i of the error span is decoded using the following probability distribution over query tokens

$$p_i = softmax(tanh(h_t^{D_2} L_1 H^U))$$

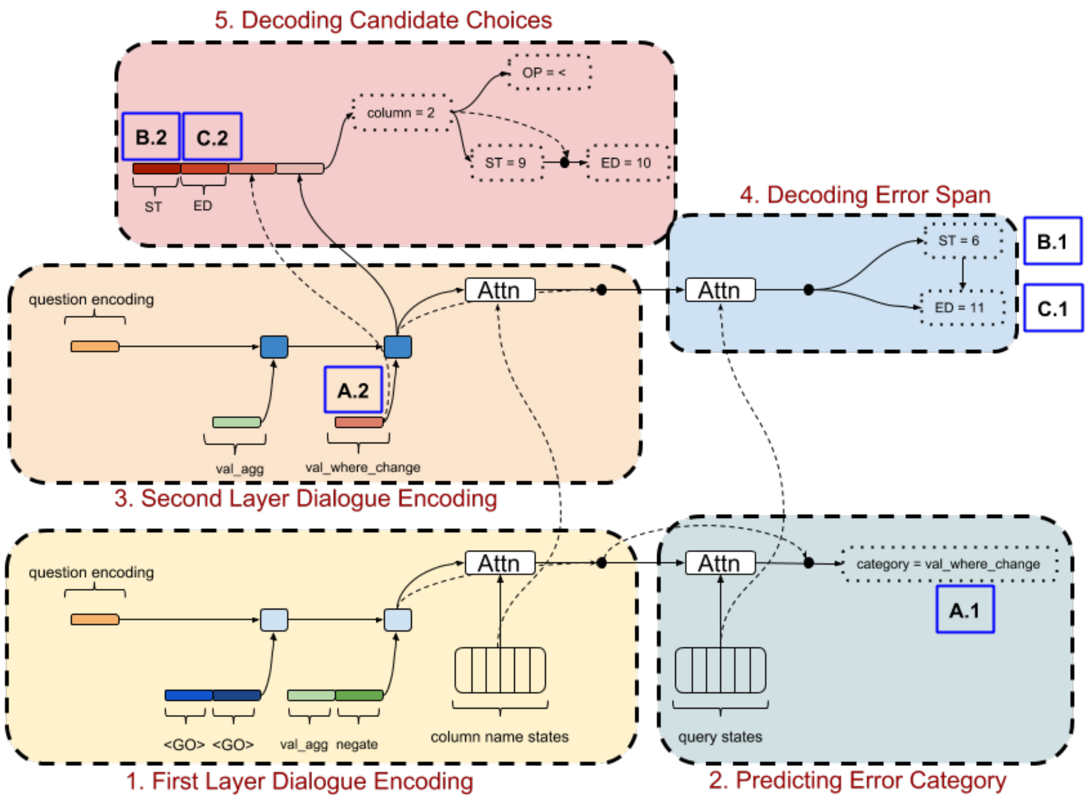


Figure 2: DialSQL model: Boxes are RNN cells, colors indicate parameter sharing. Dashed lines denote skip connections, dashed boxes denote classifications, and black circles denote vector concatenation. Blue boxes with capital letters and numbers (X.1, X.2) denote that the embeddings of predicted token at X.1 is passed as input to X.2. Each component in the pipeline is numbered according to execution order. `<GO>` is a special token to represent the start of a sequence and `ST` and `ED` denote the start and end indices of a span, respectively.

where p_i is the probability of start position over the ith query token. End position j of the error span is predicted by conditioning on the start position

$$c_i = \sum p_i * H^U$$
$$\hat{p}_j = softmax(tanh([h_t^{D_2}, c_i]L_2 H^U))$$

where $\hat{p}_j$ is the probability of end position over the jth query token. Conditioning on the error category will localize the span prediction problem as each category is defined by only a small segment of the query.

4.5 Decoding Candidate Choices

Given error category c and error span (i, j), DialSQL decodes a list of choices that will potentially replace the error span based on user feedback. Inspired by SQLNet (Xu et al., 2017), we describe our candidate choice decoding approach as follows.

Select column choice. We define the following scores over column names,

$$h = Attn(Lin([o_{i-1}^U, o_j^U, E_c]), H^T)$$
$$s_{sel} = u^T * tanh(Lin([H^T, h]))$$

where o_{i-1}^U is the output vector of the query encoder preceding the start position, and o_j^U is the output of query encoder at the end position.

Aggregation choice. Conditioned on the encoding e of the select column, we define the following scores over the set of aggregations (`MIN`, `MAX`, `COUNT`, `NO_AGGREGATION`)

$$s_{agg} = v^T * tanh(Lin(Attn(e, H^Q)))$$

Where condition choice. We first decode the condition column name similar to decoding select column. Given the encoding e of condition column, we define the following scores over the set of operators ($=, <, >$)

$$s_{op} = w^T * tanh(Lin(Attn(e, H^Q)))$$

Next, we define the following scores over question tokens for the start and end positions of the condition value

$$s_{st} = Attn(e, H^Q)$$
$$s_{ed} = Attn([e, h_{st}, H^Q])$$

where h_{st} is the context vector generated from the first attention. We denote the number of candidate choices to be decoded by k. We train DialSQL with $k = 1$. The list of $k > 1$ candidate choices is decoded similar to beam search during testing. As an example, we select k column names that have the highest scores as the candidate where column choices. For each column name, we first generate k different operators and from the set of $k * 2$ column name and operator pairs; select k operators that have the highest joint probability. Ideally, DialSQL should be able to learn the type of errors present in the generated query, extract precise error spans by pointing to query tokens, and using the location of the error spans, generate a set of related choices.

5 Experimental Results and Discussion

In this section, we evaluate DialSQL on WikiSQL using several evaluation metrics by comparing with previous literature.

5.1 Evaluation Setup and Metrics

We measure the query generation accuracy as well as the complexity of the questions and the length of the user interactions.

Query-match accuracy. We evaluate DialSQL on WikiSQL using query-match accuracy (Zhong et al., 2017; Xu et al., 2017). Query-match accuracy is the proportion of testing examples for which the generated query is exactly the same as the ground truth, except the ordering of the `WHERE` clauses.

Dialogue length. We count the number of turns to analyze whether DialSQL generates any redundant validation questions.

Question complexity. We use the average number of tokens in the generated validation questions to evaluate if DialSQL can generate simple questions without overwhelming users.

Since SQLNet and Seq2SQL are single-step models, we can not analyze DialSQL's performance by comparing against these on the last two metrics. We overcome this issue by generating simulated dialogues using an oracle system that has access to the ground truth query. The system compares `SELECT` and `AGGREGATION` clauses of the predicted query and the ground truth; asks a validation question if they differ. For each `WHERE` clause pairs of generated query and the ground truth, the system counts the number of matching segments namely `COLUMN, OP`, and `VALUE`. The system takes all the pairs with the highest matching scores and asks a validation question until one of the queries has no remaining `WHERE` clause. If both queries have no remaining clauses, the dialogue terminates. Otherwise, the system asks a `validate_where_added` (`validate_where_removed`) question when the generated query (ground truth query) has more remaining clauses. We call this strategy *Oracle-Matching (OM)*. OM ensures that the generated dialogues have the minimum number of turns possible.

5.2 Training Details

We implement DialSQL in TensorFlow (Abadi et al., 2016) using the Adam optimizer (Kingma and Ba, 2014) for the training with a learning rate of 1e−4. We use an embedding size of 300, RNN state size of 50, and a batch size of 64. The embeddings are initialized from pretrained GloVe embeddings (Pennington et al., 2014) and fine-tuned during training. We use bidirectional RNN encoders with two layers for questions, column names, and queries. Stanford CoreNLP tokenizer (Manning et al., 2014) is used to parse questions and column names. Parameters of each layer are decoupled from each other and only the embedding matrix is shared. The total number of turns is limited to 10 and 10 simulated dialogues are generated for each example in the WikiSQL training set. SQLNet and Seq2SQL models are trained on WikiSQL using the existing implemention provided by their authors. The code is available at `https://github.com/izzeddingur/DialSQL`.

5.3 Evaluation on the WikiSQL Dataset

Table 3 presents the results of query match accuracy. We observe that DialSQL model with a number of 5 choices improves the performance of both SQLNet and Seq2SQL by 7.7% and 9.4%, respectively. The higher gain on Seq2SQL model can be attributed that the single-step Seq2SQL makes more errors: DialSQL has more room for improvement. We also show the results of DialSQL where

Model	QM-Dev	QM-Test
Seq2SQL (Xu et al., 2017)	53.5%	51.6%
SQLNet (Xu et al., 2017)	63.2%	61.3%
BiAttn (Guo and Gao, 2018)	64.1%	62.5%
Seq2SQL - DialSQL	62.2%	61%
SQLNet - DialSQL	70.9%	69.0%
Seq2SQL - DialSQL$^+$	68.9%	67.8%
SQLNet - DialSQL$^+$	74.8%	73.9%
Seq2SQL - DialSQL*	84.4%	84%
SQLNet - DialSQL*	82.9%	83.7%

Table 3: Query-match accuracy on the WikiSQL development and test sets. The first two scores of our model are generated using 5 candidate choices, ($^+$) denotes a variant where users can revisit their previous answers, and (*) denotes a variant with more informative user responses.

users are allowed to revisit their previous answers and with more informative user responses; instead the model only validates the error span and the user directly gives the correct choice. In this scenario, the performance further improves on both development and test sets. It seems decoding candidate choices is a hard task and has room for improvement. For the rest of the evaluation, we present results with multi-choice questions.

5.4 Query Complexity and Dialogue Length

In Table 4, we compare DialSQL to the OM strategy on query complexity (QC) and dialogue length (DL) metrics. DialSQL and SQLNet-OM both have very similar query complexity scores showing that DialSQL produces simple questions. The number of questions DialSQL asks is around 3 for both query generation models. Even though SQLNet-OM dialogues have much smaller dialogue lengths, we attribute this to the fact that 61.3% of the dialogues have empty interactions since OM will match every segment in the generated query and the ground truth. The average number of turns in dialogues with non-empty interactions, on the other hand, is 3.10 which is close to DialSQL.

5.5 A Varying Number of Choices

In Figure 3, we plot the accuracy of DialSQL on WikiSQL with a varying number of choices at each turn. We train DialSQL once and generate a different number of choices at each turn by offering top-k candidates during testing. We observe that offering even a single candidate improves the performance of SQLNet remarkably, 1.9% and

Model	QC Dev	DL Dev	QC Test	DL Test
Seq2SQL - OM	3.47 (2.25)	0.84 (1.77)	3.51 (2.41)	0.88 (1.8)
SQLNet - OM	3.37 (2.63)	0.61 (1.45)	3.34 (2.51)	0.63 (1.49)
Seq2SQL - DialSQL	3.53 (1.79)	5.54 (2.32)	3.55 (1.81)	5.55 (2.34)
SQLNet - DialSQL	3.6 (1.86)	5.57 (2.34)	3.17 (1.55)	4.77 (1.57)

Table 4: Average query complexity and dialogue length on the WikiSQL datasets (values in paranthesis are standard deviations). Metrics for SQLNet and Seq2SQL models are generated by the OM strategy as described earlier.

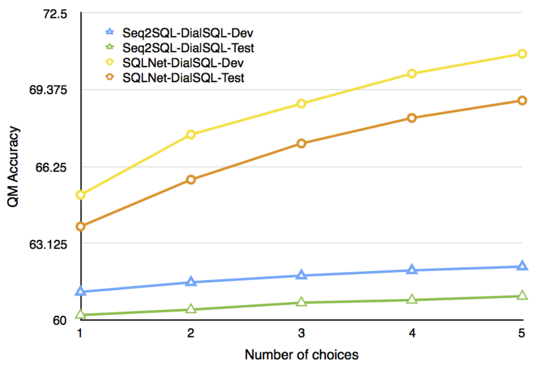


Figure 3: DialSQL performance on WikiSQL with a varying number of choices at each turn.

2.5% for development and test sets, respectively. As the number of choices increases, the performance of DialSQL improves in all the cases. Particularly, for the SQLNet-DialSQL model we observe more accuray gain. We increased the number of choices to 10 and observed no notable further improvement in the development set which suggests that 5 is a good value for the number of choices.

5.6 Error Distribution

We examine the error distribution of DialSQL and SQLNet. In DialSQL, almost all the errors are caused by `validate_sel` and `validate_where_change`, while in SQLNet `validate_where_change` is the major cause of error and other errors are distributed uniformly.

5.7 Human Evaluation

We extend our evaluation of DialSQL using human subject experiment so that real users interact with the system instead of our simulated user. We randomly pick 100 questions from WikiSQL development set and run SQLNet to generate initial candidate queries. Next, we run DialSQL using these candidate queries to generate 100 dialogues, each of which is evaluated

Model	Accuracy
SQLNet	58
DialSQL w/ User Simulation	75
DialSQL w/ Real Users	65 (1.4)

Table 5: QM accuracies of SQLNet, DialSQL with user simulation, and DialSQL with real users (value in paranthesis is standard deviation).

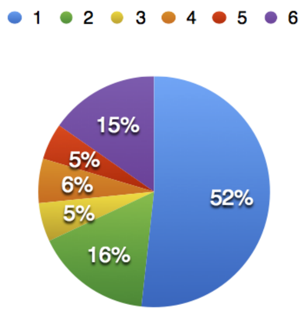


Figure 4: Distribution of user preference for DialSQL ranking (scaled to 1-6 with 6 is *None of the above.*).

by 3 different users. At each turn, we show users the headers of the corresponding table, original question, system response, and list of candidate choices for users to pick. For each error category, we generate 5 choices except for the `validate_where_added` category for which we only show 2 choices (YES or NO). Also, we add an additional choice of *None of the above* so that users can keep the previous prediction unchanged. At the end of each turn, we also ask users to give an *overall score* between 1 and 3 to evaluate whether they had a successful interaction with the DialSQL for the current turn. On average, the length of the generated dialogues is 5.6.

In Table 5, we compare the performance of SQLNet, DialSQL with user simulation, and DialSQL with real users using QM metric. We present the average performance across 3 different users with the standard deviation estimated over all dialogues. We observe that when real users interact with our system, the overall performance of the generated queries are better than SQLNet model showing that DialSQL can improve the performance of a strong NLIDB system in a real setting. However, there is still a large room for improvement between simulated dialogues and real users.

In Figure 4, we present the correlation between DialSQL ranking of the candidate choices and user preferences. We observe that, user answers and DialSQL rankings are positively correlated; most of the time users prefer the top-1 choice. Interestingly, 15% of the user answers is *None of the above*. This commonly happens in the scenario where DialSQL response asks to replace a correct condition and users prefer to keep the original prediction unchanged. Another scenario where users commonly select *None of the above* is when table headers without the content remain insufficient for users to correctly disambiguate condition values from questions. We also compute the Mean Reciprocal Rank (MMR) for each user to measure the correlation between real users and DialSQL. Average MMR is 0.69 with standard deviation of 0.004 which also shows that users generally prefer the choices ranked higher by DialSQL. The overall score of each turn also suggests that users had a reasonable conversation with DialSQL. The average score is 2.86 with standard deviation of 0.14, showing users can understand DialSQL responses and can pick a choice confidently.

6 Conclusion

We demonstrated the efficacy of the DialSQL, improving the state of the art accuracy from 62.5% to 69.0% on the WikiSQL dataset. DialSQL successfully extracts error spans from queries and offers several alternatives to users. It generates simple questions over a small number of turns without overwhelming users. The model learns from only simulated data which makes it easy to adapt to new domains. We further investigate the usability of DialSQL in a real life setting by conducting human evaluations. Our results suggest that the accuracy of the generated queries can be improved via real user feedback.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053 and NSF 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *CoRR*, abs/1607.00070.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

T. Guo and H. Gao. 2018. Bidirectional Attention for SQL Generation. *ArXiv e-prints*.

Izzeddin Gur, Daniel Hewlett, Llion Jones, and Alexandre Lacoste. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *CoRR*, abs/1704.08760.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

F. Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8(1):73–84.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016. Dialogue learning with human-in-the-loop. *CoRR*, abs/1611.09823.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

A. Saha, V. Pahuja, M. M. Khapra, K. Sankaranarayanan, and S. Chandar. 2018. Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph. *ArXiv e-prints*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152, Rochester, New York. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. *CoRR*, abs/1605.06069.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. *CoRR*, abs/1507.02221.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the International Conference on World Wide Web*.

William A Woods. 1973. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the American Federation of Information Processing Societies Conference*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *CoRR*, abs/1711.04436.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: Query synthesis from natural language. *Proc. ACM Program. Lang.*, 1(OOPSLA):63:1–63:26.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

John M Zelle and Mooney Ray. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Conversations Gone Awry: Detecting Early Signs of Conversational Failure

Justine Zhang and **Jonathan P. Chang** and **Cristian Danescu-Niculescu-Mizil**[*]
Cornell University
{jz727,jpc362}@cornell.edu, cristian@cs.cornell.edu

Lucas Dixon and **Nithum Thain**
Jigsaw
{ldixon,nthain}@google.com

Yiqing Hua
Cornell University
yh663@cornell.edu

Dario Taraborelli
Wikimedia Foundation
dario@wikimedia.org

Abstract

One of the main challenges online social systems face is the prevalence of antisocial behavior, such as harassment and personal attacks. In this work, we introduce the task of predicting from the very start of a conversation whether it will get out of hand. As opposed to detecting undesirable behavior after the fact, this task aims to enable early, actionable prediction at a time when the conversation might still be salvaged.

To this end, we develop a framework for capturing pragmatic devices—such as politeness strategies and rhetorical prompts—used to start a conversation, and analyze their relation to its future trajectory. Applying this framework in a controlled setting, we demonstrate the feasibility of detecting early warning signs of antisocial behavior in online discussions.

1 Introduction

"Or vedi l'anime di color cui vinse l'ira."[1]
– Dante Alighieri, Divina Commedia, Inferno

Online conversations have a reputation for going awry (Hinds and Mortensen, 2005; Gheitasy et al., 2015): antisocial behavior (Shepherd et al., 2015) or simple misunderstandings (Churchill and Bly, 2000; Yamashita and Ishida, 2006) hamper the efforts of even the best intentioned collaborators. Prior computational work has focused on characterizing and detecting content exhibiting antisocial online behavior: trolling (Cheng et al., 2015, 2017), hate speech (Warner and Hirschberg, 2012; Davidson et al., 2017), harassment (Yin et al., 2009), personal attacks (Wulczyn et al., 2017) or, more generally, toxicity (Chandrasekharan et al., 2017; Pavlopoulos et al., 2017b).

Our goal is crucially different: instead of identifying antisocial comments *after the fact*, we aim to detect *warning signs* indicating that a civil conversation is at risk of derailing into such undesirable behaviors. Such warning signs could provide potentially actionable knowledge at a time when the conversation is still salvageable.

As a motivating example, consider the pair of conversations in Figure 1. Both exchanges took place in the context of the Wikipedia discussion page for the article on the Dyatlov Pass Incident, and both show (ostensibly) civil disagreement between the participants. However, only one of these conversations will eventually turn awry and devolve into a personal attack ("Wow, you're coming off as a total d**k. [...] What the hell is wrong with you?"), while the other will remain civil.

As humans, we have some intuition about which conversation is more likely to derail.[2] We may note the repeated, direct questioning with which **A1** opens the exchange, and that **A2** replies with yet another question. In contrast, **B1**'s softer, hedged approach ("it seems", "I don't think") appears to invite an exchange of ideas, and **B2** actually addresses the question instead of stonewalling. Could we endow artificial systems with such intuitions about the future trajectory of conversations?

In this work we aim to computationally capture linguistic cues that predict a conversation's future health. Most existing conversation modeling approaches aim to detect characteristics of an observed discussion or predict the outcome after the discussion concludes—e.g., whether it involves a present dispute (Allen et al., 2014; Wang and Cardie, 2014) or contributes to the even-

[*] Corresponding senior author.
[1] "Now you see the souls of those whom anger overcame."

[2] In fact, humans achieve an accuracy of 72% on this balanced task, showing that it is feasible, but far from trivial.

A1: Why there's no mention of it here? Namely, an altercation with a foreign intelligence group? True, by the standards of sources some require it wouln't even come close, not to mention having some really weak points, but it doesn't mean that it doesn't exist.

A2: So what you're saying is we should put a bad source in the article because it exists?

B1: Is the St. Petersberg Times considered a reliable source by wikipedia? It seems that the bulk of this article is coming from that one article, which speculates about missile launches and UFOs. I'm going to go through and try and find corroborating sources and maybe do a rewrite of the article. I don't think this article should rely on one so-so source.

B2: I would assume that it's as reliable as any other mainstream news source.

Figure 1: Two examples of initial exchanges from conversations concerning disagreements between editors working on the Wikipedia article about the Dyatlov Pass Incident. Only one of the conversations will eventually turn awry, with an interlocutor launching into a personal attack.

tual solution of a problem (Niculae and Danescu-Niculescu-Mizil, 2016). In contrast, for this new task we need to discover interactional signals of the *future* trajectory of an *ongoing* conversation.

We make a first approach to this problem by analyzing the role of politeness (or lack thereof) in keeping conversations on track. Prior work has shown that politeness can help shape the course of offline (Clark, 1979; Clark and Schunk, 1980), as well as online interactions (Burke and Kraut, 2008), through mechanisms such as softening the perceived force of a message (Fraser, 1980), acting as a buffer between conflicting interlocutor goals (Brown and Levinson, 1987), and enabling all parties to save face (Goffman, 1955). This suggests the potential of politeness to serve as an indicator of whether a conversation will sustain its initial civility or eventually derail, and motivates its consideration in the present work.

Recent studies have computationally operationalized prior formulations of politeness by extracting linguistic cues that reflect politeness strategies (Danescu-Niculescu-Mizil et al., 2013; Aubakirova and Bansal, 2016). Such research has additionally tied politeness to social factors such as individual status (Danescu-Niculescu-Mizil et al., 2012; Krishnan and Eisenstein, 2015), and the success of requests (Althoff et al., 2014) or of collaborative projects (Ortu et al., 2015). However, to the best of our knowledge, this is the first computational investigation of the relation between politeness strategies and the future trajectory of the conversations in which they are deployed. Furthermore, we generalize beyond predefined politeness strategies by using an unsupervised method to discover additional rhetorical prompts used to initiate different types of conversations that may be specific to online collaborative settings, such as coordinating work (Kittur and Kraut, 2008) or conducting factual checks.

We explore the role of such pragmatic and rhetorical devices in foretelling a particularly perplexing type of conversational failure: when participants engaged in previously civil discussion start to attack each other. This type of derailment "from within" is arguably more disruptive than other forms of antisocial behavior, such as vandalism or trolling, which the interlocutors have less control over or can choose to ignore.

We study this phenomenon in a new dataset of Wikipedia talk page discussions, which we compile through a combination of machine learning and crowdsourced filtering. The dataset consists of conversations which begin with ostensibly civil comments, and either remain healthy or derail into personal attacks. Starting from this data, we construct a setting that mitigates effects which may trivialize the task. In particular, some topical contexts (such as politics and religion) are naturally more susceptible to antisocial behavior (Kittur et al., 2009; Cheng et al., 2015). We employ techniques from causal inference (Rosenbaum, 2010) to establish a controlled framework that focuses our study on topic-agnostic linguistic cues.

In this controlled setting, we find that pragmatic cues extracted from the very first exchange in a conversation (i.e., the first comment-reply pair) can indeed provide some signal of whether the conversation will subsequently go awry. For example, conversations prompted by hedged remarks sustain their initial civility more so than those prompted by forceful questions, or by direct language addressing the other interlocutor.

In summary, our main contributions are:

- We articulate the new task of detecting early on whether a conversation will derail into personal attacks;
- We devise a controlled setting and build a labeled dataset to study this phenomenon;

- We investigate how politeness strategies and other rhetorical devices are tied to the future trajectory of a conversation.

More broadly, we show the feasibility of automatically detecting warning signs of future misbehavior in collaborative interactions. By providing a labeled dataset together with basic methodology and several baselines, we open the door to further work on understanding factors which may derail or sustain healthy online conversations. To facilitate such future explorations, we distrubute the data and code as part of the Cornell Conversational Analysis Toolkit.[3]

2 Further Related Work

Antisocial behavior. Prior work has studied a wide range of disruptive interactions in various online platforms like Reddit and Wikipedia, examining behaviors like aggression (Kayany, 1998), harassment (Chatzakou et al., 2017; Vitak et al., 2017), and bullying (Akbulut et al., 2010; Kwak et al., 2015; Singh et al., 2017), as well as their impact on aspects of engagement like user retention (Collier and Bear, 2012; Wikimedia Support and Safety Team, 2015) or discussion quality (Arazy et al., 2013). Several studies have sought to develop machine learning techniques to detect signatures of online toxicity, such as personal insults (Yin et al., 2009), harassment (Sood et al., 2012) and abusive language (Nobata et al., 2016; Gambäck and Sikdar, 2017; Pavlopoulos et al., 2017a; Wulczyn et al., 2017). These works focus on detecting toxic behavior after it has already occurred; a notable exception is Cheng et al. (2017), which predicts future community enforcement against users in news-based discussions. Our work similarly aims to understand *future* antisocial behavior; however, our focus is on studying the trajectory of a conversation rather than the behavior of individuals across disparate discussions.

Discourse analysis. Our present study builds on a large body of prior work in computationally modeling discourse. Both unsupervised (Ritter et al., 2010) and supervised (Zhang et al., 2017a) approaches have been used to categorize behavioral patterns on the basis of the language that ensues in a conversation, in the particular realm of online discussions. Models of conversational behavior have also been used to predict conversation outcomes, such as betrayal in games (Niculae et al., 2015), and success in team problem solving settings (Fu et al., 2017) or in persuading others (Tan et al., 2016; Zhang et al., 2016).

While we are inspired by the techniques employed in these approaches, our work is concerned with predicting the future trajectory of an ongoing conversation as opposed to a post-hoc outcome. In this sense, we build on prior work in modeling conversation trajectory, which has largely considered *structural* aspects of the conversation (Kumar et al., 2010; Backstrom et al., 2013). We complement these structural models by seeking to extract potential signals of future outcomes from the *linguistic discourse* within the conversation.

3 Finding Conversations Gone Awry

We develop our framework for understanding linguistic markers of conversational trajectories in the context of Wikipedia's *talk page* discussions—public forums in which contributors convene to deliberate on editing matters such as evaluating the quality of an article and reviewing the compliance of contributions with community guidelines. The dynamic of conversational derailment is particularly intriguing and consequential in this setting by virtue of its collaborative, goal-oriented nature. In contrast to unstructured commenting forums, cases where one *collaborator* turns on another over the course of an initially civil exchange constitute perplexing pathologies. In turn, these toxic attacks are especially disruptive in Wikipedia since they undermine the social fabric of the community as well as the ability of editors to contribute (Henner and Sefidari, 2016).

To approach this domain we reconstruct a complete view of the conversational process in the edit history of English Wikipedia by translating sequences of revisions of each talk page into structured conversations. This yields roughly 50 million conversations across 16 million talk pages.

Roughly one percent of Wikipedia comments are estimated to exhibit antisocial behavior (Wulczyn et al., 2017). This illustrates a challenge for studying conversational failure: one has to sift through many conversations in order to find even a small set of examples. To avoid such a prohibitively exhaustive analysis, we first use a machine learning classifier to identify candidate conversations that are likely to contain a toxic contribution, and then use crowdsourcing to vet the resulting labels and construct our controlled dataset.

[3] `http://convokit.infosci.cornell.edu`

Job 1: Ends in personal attack. We show three annotators a conversation and ask them to determine if its last comment is a personal attack toward someone else in the conversation.			**Job 2: Civil start.** We split conversations into snippets of three consecutive comments. We ask three annotators to determine whether any of the comments in a snippet is toxic.			
Annotators	**Conversations**	**Agreement**	**Annotators**	**Conversations**	**Snippets**	**Agreement**
367	4,022	67.8%	247	1,252	2,181	87.5%

Table 1: Descriptions of crowdsourcing jobs, with relevant statistics. More details in Appendix A.

Candidate selection. Our goal is to analyze how the start of a *civil* conversation is tied to its potential future derailment into personal attacks. Thus, we only consider conversations that start out as ostensibly civil, i.e., where at least the first exchange does not exhibit any toxic behavior,[4] and that continue beyond this first exchange. To focus on the especially perplexing cases when the attacks come *from within*, we seek examples where the attack is initiated by one of the two participants in the initial exchange.

To select candidate conversations to include in our collection, we use the toxicity classifier provided by the Perspective API,[5] which is trained on Wikipedia talk page comments that have been annotated by crowdworkers (Wulczyn et al., 2016). This provides a toxicity score t for all comments in our dataset, which we use to preselect two sets of conversations: (a) candidate conversations that are civil throughout, i.e., conversations in which all comments (including the initial exchange) are not labeled as toxic ($t < 0.4$); and (b) candidate conversations that turn toxic after the first (civil) exchange, i.e., conversations in which the N-th comment ($N > 2$) is labeled toxic ($t \geq 0.6$), but all the preceding comments are not ($t < 0.4$).

Crowdsourced filtering. Starting from these candidate sets, we use crowdsourcing to vet each conversation and select a subset that are perceived by humans to either stay civil throughout ("on-track" conversations), or start civil but end with a *personal attack* ("awry-turning" conversations). To inform the design of this human-filtering process and to check its effectiveness, we start from a seed set of 232 conversations manually verified by the authors to end in personal attacks (more details about the selection of the seed set and its role in the crowd-sourcing process can be found in Appendix A). We take particular care to not over-constrain crowdworker interpretations of what personal attacks may be, and to separate toxicity from civil disagreement, which is recognized as a key aspect of effective collaborations (Coser, 1956; De Dreu and Weingart, 2003).

We design and deploy two filtering jobs using the CrowdFlower platform, summarized in Table 1 and detailed in Appendix A. **Job 1** is designed to select conversations that contain a "rude, insulting, or disrespectful" comment towards another user in the conversation—i.e., a personal attack. In contrast to prior work labeling antisocial comments in isolation (Sood et al., 2012; Wulczyn et al., 2017), annotators are asked to label personal attacks in the *context* of the conversations in which they occur, since antisocial behavior can often be context-dependent (Cheng et al., 2017). In fact, in order to ensure that the crowdworkers read the entire conversation, we also ask them to indicate who is the target of the attack. We apply this task to the set of candidate awry-turning conversations, selecting the 14% which all three annotators perceived as ending in a personal attack.[6]

Job 2 is designed to filter out conversations that do not actually start out as civil. We run this job to ensure that the *awry-turning* conversations are civil up to the point of the attack—i.e., they *turn* awry—discarding 5% of the candidates that passed Job 1. We also use it to verify that the candidate *on-track* conversations are indeed civil throughout, discarding 1% of the respective candidates. In both cases we filter out conversations in which three annotators could identify at least one comment that is "rude, insulting, or disrespectful".

Controlled setting. Finally, we need to construct a setting that affords for meaningful comparison between conversations that derail and those that stay on track, and that accounts for trivial topical confounds (Kittur et al., 2009; Cheng et al., 2015). We mitigate topical confounds using matching, a technique developed for causal inference in observational studies (Rubin, 2007). Specifically, start-

[4]For the sake of generality, in this work we focus on this most basic conversational unit: the first comment-reply pair starting a conversation.

[5]https://www.perspectiveapi.com/

[6]We opted to use unanimity in this task to account for the highly subjective nature of the phenomenon.

ing from our human-vetted collection of conversations, we pair each *awry-turning* conversation, with an *on-track* conversation, such that both took place on the same talk page. If we find multiple such pairs, we only keep the one in which the paired conversations take place closest in time, to tighten the control for topic. Conversations that cannot be paired are discarded.

This procedure yields a total of 1,270 paired awry-turning and on-track conversations (including our initial seed set), spanning 582 distinct talk pages (averaging 1.1 pairs per page, maximum 8) and 1,876 (overlapping) topical categories. The average length of a conversation is 4.6 comments.

4 Capturing Pragmatic Devices

We now describe our framework for capturing linguistic cues that might inform a conversation's future trajectory. Crucially, given our focus on conversations that start seemingly civil, we do not expect overtly hostile language—such as insults (Yin et al., 2009)—to be informative. Instead, we seek to identify pragmatic markers within the initial exchange of a conversation that might serve to reveal or exacerbate underlying tensions that eventually come to the fore, or conversely suggest sustainable civility. In particular, in this work we explore how politeness strategies and rhetorical prompts reflect the future health of a conversation.

Politeness strategies. Politeness can reflect a-priori good will and help navigate potentially face-threatening acts (Goffman, 1955; Lakoff, 1973), and also offers hints to the underlying intentions of the interlocutors (Fraser, 1980). Hence, we may naturally expect certain politeness strategies to signal that a conversation is likely to stay on track, while others might signal derailment.

In particular, we consider a set of pragmatic devices signaling politeness drawn from Brown and Levinson (1987). These linguistic features reflect two overarching types of politeness. *Positive* politeness strategies encourage social connection and rapport, perhaps serving to maintain cohesion throughout a conversation; such strategies include gratitude ("*thanks* for your help"), greetings ("*hey*, how is your day so far") and use of "please", both at the start ("*Please* find sources for your edit...") and in the middle ("Could you *please* help with...?") of a sentence. *Negative* politeness strategies serve to dampen an interlocutor's imposition on an addressee, often through conveying indirectness or uncertainty on the part of the commenter. Both commenters in example **B** (Fig. 1) employ one such strategy, hedging, perhaps seeking to soften an impending disagreement about a source's reliability ("I *don't think...*", "I would *assume...*"). We also consider markers of *impolite* behavior, such as the use of direct questions ("*Why*'s there no mention of it?') and sentence-initial second person pronouns ("*Your* sources don't matter..."), which may serve as forceful-sounding contrasts to negative politeness markers. Following Danescu-Niculescu-Mizil et al. (2013), we extract such strategies by pattern matching on the dependency parses of comments.

Types of conversation prompts. To complement our pre-defined set of politeness strategies, we seek to capture domain-specific rhetorical patterns used to initiate conversations. For instance, in a collaborative setting, we may expect conversations that start with an invitation for working together to signal less tension between the participants than those that start with statements of dispute. We discover types of such *conversation prompts* in an unsupervised fashion by extending a framework used to infer the rhetorical role of questions in (offline) political debates (Zhang et al., 2017b) to more generally extract the rhetorical functions of comments. The procedure follows the intuition that the rhetorical role of a comment is reflected in the type of replies it is likely to elicit. As such, comments which tend to trigger similar replies constitute a particular type of prompt.

To implement this intuition, we derive two different low-rank representations of the common lexical phrasings contained in comments (agnostic to the particular topical content discussed), automatically extracted as recurring sets of arcs in the dependency parses of comments. First, we derive *reply-vectors* of phrasings, which reflect their propensities to *co-occur*. In particular, we perform singular value decomposition on a term-document matrix $\mathcal{R}$ of phrasings and replies as $\mathcal{R} \approx \hat{\mathcal{R}} = U_R S V_R^T$, where rows of U_R are low-rank reply-vectors for each phrasing.

Next, we derive *prompt-vectors* for the phrasings, which reflect similarities in the subsequent replies that a phrasing *prompts*. We construct a prompt-reply matrix $\mathcal{P} = (p_{ij})$ where $p_{ij} = 1$ if phrasing j occurred in a reply to a comment containing phrasing i. We project $\mathcal{P}$ into the same space as U_R by solving for $\hat{\mathcal{P}}$ in $\mathcal{P} = \hat{\mathcal{P}} S V_R^T$ as

Prompt Type	Description	Examples
`Factual check`	Statements about article content, pertaining to or contending issues like factual accuracy.	The terms **are used** interchangeably in the US. The census **is not talking about** families here.
`Moderation`	Rebukes or disputes concerning moderation decisions such as blocks and reversions.	**If** you continue, you may **be blocked** from editing. He's **accused** me **of** being a troll.
`Coordination`	Requests, questions, and statements of intent pertaining to collaboratively editing an article.	It's a long list so I **could do with** your **help**. **Let me know** if you agree with this and I'll go ahead [...]
`Casual remark`	Casual, highly conversational aside-remarks.	**What's with** this flag image? **I'm surprised** there wasn't an article before.
`Action statement`	Requests, statements, and explanations about various editing actions.	**Please consider improving** the article to address the issues [...] The page **was deleted as** self-promotion.
`Opinion`	Statements seeking or expressing opinions about editing challenges and decisions.	I **think** that it **should be** the other way around. This article **seems to have** a lot of bias.

Table 2: Prompt types automatically extracted from talk page conversations, with interpretations and examples from the data. Bolded text indicate common prompt phrasings extracted by the framework. Further examples are shown in Appendix B, Table 4.

$\hat{\mathcal{P}} = \mathcal{P}V_R S^{-1}$. Each row of $\hat{\mathcal{P}}$ is then a prompt-vector of a phrasing, such that the prompt-vector for phrasing i is close to the reply-vector for phrasing j if comments with phrasing i tend to prompt replies with phrasing j. Clustering the rows of $\hat{\mathcal{P}}$ then yields k conversational *prompt types* that are unified by their similarity in the space of replies. To infer the prompt type of a new comment, we represent the comment as an average of the representations of its constituent phrasings (i.e., rows of $\hat{\mathcal{P}}$) and assign the resultant vector to a cluster.[7]

To determine the prompt types of comments in our dataset, we first apply the above procedure to derive a set of prompt types from a *disjoint* (unlabeled) corpus of Wikipedia talk page conversations (Danescu-Niculescu-Mizil et al., 2012). After initial examination of the framework's output on this external data, we chose to extract $k = 6$ prompt types, shown in Table 2 along with our interpretations.[8] These prompts represent signatures of conversation-starters spanning a wide range of topics and contexts which reflect core elements of Wikipedia, such as moderation disputes and coordination (Kittur et al., 2007; Kittur and Kraut, 2008). We assign each comment in our present dataset to one of these types.[9]

5 Analysis

We are now equipped to computationally explore how the pragmatic devices used to start a conversation can signal its future health. Concretely, to quantify the relative propensity of a linguistic marker to occur at the start of awry-turning versus on-track conversations, we compute the log-odds ratio of the marker occurring in the initial exchange—i.e., in the first or second comments—of awry-turning conversations, compared to initial exchanges in the on-track setting. These quantities are depicted in Figure 2A.[10]

Focusing on the **first** comment (represented as ◇s), we find a rough correspondence between linguistic *directness* and the likelihood of future personal attacks. In particular, comments which contain *direct questions*, or exhibit *sentence-initial you* (i.e., "2nd person start"), tend to start awry-turning conversations significantly more often than ones that stay on track (both $p < 0.001$).[11] This effect coheres with our intuition that directness signals some latent hostility from the conversation's initiator, and perhaps reinforces the forcefulness of contentious impositions (Brown and Levinson, 1987). This interpretation is also sug-

[7] We scale rows of U_R and $\hat{\mathcal{P}}$ to unit norm. We assign comments whose vector representation has (ℓ_2) distance ≥ 1 to all cluster centroids to an extra, infrequently-occurring null type which we ignore in subsequent analyses.

[8] We experimented with more prompt types as well, finding that while the methodology recovered finer-grained types, and obtained qualitatively similar results and prediction accuracies as described in Sections 5 and 6, the assignment of comments to types was relatively sparse due to the small data size, resulting in a loss of statistical power.

[9] While the particular prompt types we discover are specific to Wikipedia, the methodology for inferring them is unsupervised and is applicable in other conversational settings.

[10] To reduce clutter we only depict features which occur a minimum of 50 times and have absolute log-odds ≥ 0.2 in at least one of the data subsets. The markers indicated as statistically significant for Figure 2A remain so after a Bonferroni correction, with the exception of factual checks, hedges (lexicon, ◇), gratitude (◇), and opinion.

[11] All p values in this section are computed as two-tailed binomial tests, comparing the proportion of awry-turning conversations exhibiting a particular device to the proportion of on-track conversations.

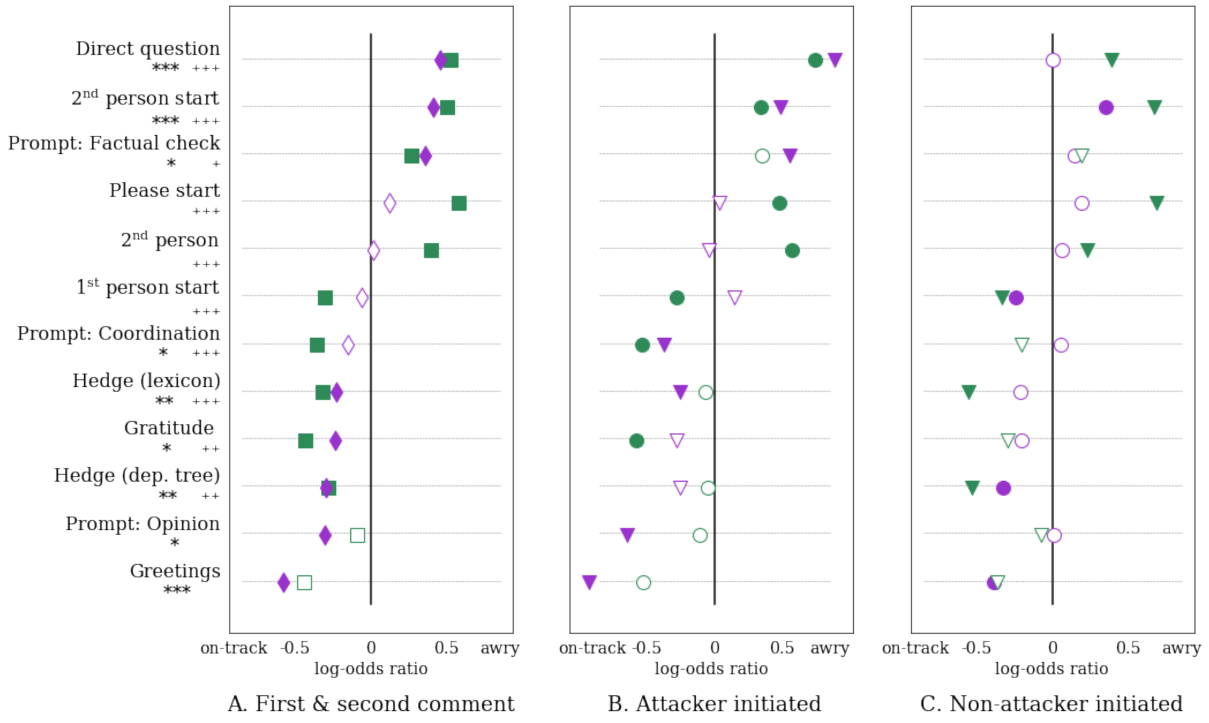


Figure 2: Log-odds ratios of politeness strategies and prompt types exhibited in the first and second comments of conversations that turn awry, versus those that stay on-track. **All**: Purple and green markers denote log-odds ratios in the first and second comments, respectively; points are solid if they reflect significant ($p < 0.05$) log-odds ratios with an effect size of at least 0.2. **A**: ◇s and □s denote **first** and **second** comment log-odds ratios, respectively; * denotes statistically significant differences at the $p < 0.05$ (*), $p < 0.01$ (**) and $p < 0.001$ (***) levels for the first comment (two-tailed binomial test); + denotes corresponding statistical significance for the second comment. **B** and **C**: ▽s and ◯s correspond to effect sizes in the comments authored by the **attacker** and **non-attacker**, respectively, in **attacker initiated** (**B**) and **non-attacker initiated** (**C**) conversations.

gested by the relative propensity of the `factual check` prompt, which tends to cue disputes regarding an article's factual content ($p < 0.05$).

In contrast, comments which initiate on-track conversations tend to contain *gratitude* ($p < 0.05$) and *greetings* ($p < 0.001$), both positive politeness strategies. Such conversations are also more likely to begin with `coordination` prompts ($p < 0.05$), signaling active efforts to foster constructive teamwork. Negative politeness strategies are salient in on-track conversations as well, reflected by the use of *hedges* ($p < 0.01$) and `opinion` prompts ($p < 0.05$), which may serve to soften impositions or factual contentions (Hübler, 1983).

These effects are echoed in the **second** comment—i.e., the **first reply** (represented as □s). Interestingly, in this case we note that the difference in pronoun use is especially marked. First replies in conversations that eventually derail tend to contain more *second person pronouns* ($p < 0.001$), perhaps signifying a replier pushing back to contest the initiator; in contrast, on-track conversations have more *sentence-initial I/We* (i.e., "1st person start", $p < 0.001$), potentially indicating the replier's willingness to step into the conversation and work with—rather than argue against—the initiator (Tausczik and Pennebaker, 2010).

Distinguishing interlocutor behaviors. Are the linguistic signals we observe solely driven by the eventual attacker, or do they reflect the behavior of both actors? To disentangle the attacker and non-attackers' roles in the initial exchange, we examine their language use in these two possible cases: when the *future* attacker initiates the conversation, or is the first to reply. In **attacker-initiated** conversations (Figure 2B, 608 conversations), we see that both actors exhibit a propensity for the linguistically direct markers (e.g., *direct questions*) that tend to signal future attacks. Some of these markers are used particularly often by the **non-attacking replier** in awry-turning conversations (e.g., *second person pronouns*, $p < 0.001$, ○s), further suggesting the dynamic of the replier pushing back at—and perhaps even escalating—the attacker's initial hint of aggression. Among conversations initiated instead by the **non-attacker** (Figure 2C, 662 conversations), the non-attacker's linguistic behavior in the first comment (○s) is less distinctive from that of initiators in the on-track setting (i.e., log-odds ratios closer to 0); markers of future derailment are (unsurprisingly) more pronounced once the eventual attacker (▽s) joins the conversation in the second comment.[12]

More broadly, these results reveal how different politeness strategies and rhetorical prompts deployed in the initial stages of a conversation are tied to its future trajectory.

6 Predicting Future Attacks

We now show that it is indeed feasible to predict whether a conversation will turn awry based on linguistic properties of its very first exchange, providing several baselines for this new task. In doing so, we demonstrate that the pragmatic devices examined above encode signals about the future trajectory of conversations, capturing some of the intuition humans are shown to have.

We consider the following balanced prediction task: given a pair of conversations, which one will eventually lead to a personal attack? We extract all features from the very first exchange in a conversation—i.e., a comment-reply pair, like those illustrated in our introductory example (Figure 1). We use logistic regression and report accuracies on a leave-one-page-out cross validation, such that in each fold, all conversation pairs from a given talk page are held out as test data and pairs from all other pages are used as training data (thus preventing the use of page-specific information). Prediction results are summarized in Table 3.

Language baselines. As baselines, we consider several straightforward features: word count (which performs at chance level), sentiment lexicon (Liu et al., 2005) and bag of words.

Pragmatic features. Next, we test the predictive power of the **prompt types** and **politeness strategies** features introduced in Section 4. The 12 prompt type features (6 features for each comment in the initial exchange) achieve 59.2% accuracy, and the 38 politeness strategies features (19 per comment) achieve 60.5% accuracy. The **pragmatic** features combine to reach 61.6% accuracy.

[12]As an interesting avenue for future work, we note that some markers used by non-attacking initiators potentially still anticipate later attacks, suggested by, e.g., the relative prevalence of *sentence-initial you* ($p < 0.05$, ○s).

Feature set	**# features**	**Accuracy**
Bag of words	5,000	56.7%
Sentiment lexicon	4	55.4%
Politeness strategies	38	60.5%
Prompt types	12	59.2%
Pragmatic (all)	50	61.6%
Interlocutor features	5	51.2%
Trained toxicity	2	60.5%
Toxicity + ***Pragmatic***	52	64.9%
Humans		72.0%

Table 3: Accuracies for the balanced future-prediction task. Features based on pragmatic devices are **bolded**, reference points are *italicized*.

Reference points. To better contextualize the performance of our features, we compare their predictive accuracy to the following reference points:

Interlocutor features: Certain kinds of interlocutors are potentially more likely to be involved in awry-turning conversations. For example, perhaps newcomers or anonymous participants are more likely to derail interactions than more experienced editors. We consider a set of features representing participants' experience on Wikipedia (i.e., number of edits) and whether the comment authors are anonymous. In our task, these features perform at the level of random chance.

Trained toxicity: We also compare with the toxicity score of the exchange from the Perspective API classifier—a perhaps unfair reference point, since this supervised system was trained on additional human-labeled training examples from the same domain and since it was used to create the very data on which we evaluate. This results in an accuracy of 60.5%; combining trained toxicity with our pragmatic features achieves 64.9%.

Humans: A sample of 100 pairs were labeled by (non-author) volunteer human annotators. They were asked to guess, from the initial exchange, which conversation in a pair will lead to a personal attack. Majority vote across three annotators was used to determine the human labels, resulting in an accuracy of 72%. This confirms that humans have some intuition about whether a conversation might be heading in a bad direction, which our features can partially capture. In fact, the classifier using pragmatic features is accurate on 80% of the examples that humans also got right.

Attacks on the horizon. Finally, we seek to understand whether cues extracted from the first exchange can predict future discussion trajectory beyond the immediate next couple of comments. We thus repeat the prediction experiments on the subset of conversations in which the first personal attack happens after the fourth comment (282 pairs), and find that the pragmatic devices used in the first exchange maintain their predictive power (67.4% accuracy), while the sentiment and bag of words baselines drop to the level of random chance.

Overall, these initial results show the feasibility of reconstructing some of the human intuition about the future trajectory of an ostensibly civil conversation in order to predict whether it will eventually turn awry.

7 Conclusions and Future Work

In this work, we started to examine the intriguing phenomenon of conversational derailment, studying how the use of pragmatic and rhetorical devices relates to future conversational failure. Our investigation centers on the particularly perplexing scenario in which one participant of a civil discussion later attacks another, and explores the new task of predicting whether an initially healthy conversation will derail into such an attack. To this end, we develop a computational framework for analyzing how general politeness strategies and domain-specific rhetorical prompts deployed in the initial stages of a conversation are tied to its future trajectory.

Making use of machine learning and crowdsourcing tools, we formulate a tightly-controlled setting that enables us to meaningfully compare conversations that stay on track with those that go awry. The human accuracy on predicting future attacks in this setting (72%) suggests it is feasible at least at the level of human intuition. We show that our computational framework can recover some of that intuition, hinting at the potential of automated methods to identify signals of the future trajectories of online conversations.

Our approach has several limitations which open avenues for future work. Our correlational analyses do not provide any insights into *causal* mechanisms of derailment, which randomized experiments could address. Additionally, since our procedure for collecting and vetting data focused on precision rather than recall, it might miss more subtle attacks that are overlooked by the toxicity classifier. Supplementing our investigation with other indicators of antisocial behavior, such as editors blocking one another, could enrich the range of attacks we study. Noting that our framework is not specifically tied to Wikipedia, it would also be valuable to explore the varied ways in which this phenomenon arises in other (possibly non-collaborative) public discussion venues, such as Reddit and Facebook Pages.

While our analysis focused on the very first exchange in a conversation for the sake of generality, more complex modeling could extend its scope to account for conversational features that more comprehensively span the interaction. Beyond the present binary classification task, one could explore a sequential formulation predicting whether the next turn is likely to be an attack as a discussion unfolds, capturing conversational dynamics such as sustained escalation.

Finally, our study of derailment offers only one glimpse into the space of possible conversational trajectories. Indeed, a manual investigation of conversations whose eventual trajectories were misclassified by our models—as well as by the human annotators—suggests that interactions which initially seem prone to attacks can nonetheless maintain civility, by way of level-headed interlocutors, as well as explicit acts of reparation. A promising line of future work could consider the complementary problem of identifying pragmatic strategies that can help bring uncivil conversations back on track.

Acknowledgements. We are grateful to the anonymous reviewers for their thoughtful comments and suggestions, and to Maria Antoniak, Valts Blukis, Liye Fu, Sam Havron, Jack Hessel, Ishaan Jhaveri, Lillian Lee, Alex Niculescu-Mizil, Alexandra Schofield, Laure Thompson, Andrew Wang, Leila Zia and the members of the Wikimedia Foundation anti-harassment program for extremely insightful (on-track) conversations and for assisting with data annotation efforts. This work is supported in part by NSF CAREER Award IIS-1750615, NSF Grant SES-1741441, a Google Faculty Award, a WMF gift and a CrowdFlower AI for Everyone Award.

References

Yavuz Akbulut, Yusuf Levent Sahin, and Bahadir Eristi. 2010. Cyberbullying victimization among Turkish online social utility members. *Journal of Educational Technology & Society*.

Kelsey Allen, Giuseppe Carenini, and Raymond T Ng. 2014. Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *Proceedings of EMNLP*.

Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2014. How to ask for a favor: A case study on the success of altruistic requests. In *Proceedings of ICWSM*.

Ofer Arazy, Lisa Yeo, and Oded Nov. 2013. Stay on the Wikipedia task: When task-related disagreements slip into personal and procedural conflicts. *Journal of the Association for Information Science and Technology*.

Malika Aubakirova and Mohit Bansal. 2016. Interpreting neural networks to improve politeness comprehension. In *Proceedings of EMNLP*.

Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and curating conversation threads: Expansion, focus, volume, re-entry. In *Proceedings of WSDM*.

Penelope Brown and Stephen Levinson. 1987. *Politeness: Some universals in language usage*. Cambridge University Press.

Moira Burke and Robert Kraut. 2008. Mind your Ps and Qs: The impact of politeness and rudeness in online communities. In *Proceedings of CSCW*.

Eshwar Chandrasekharan, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein, and Eric Gilbert. 2017. You can't stay here: The efficacy of Reddit's 2015 ban examined through hate speech. In *Proceedings of CSCW*.

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Measuring #GamerGate: A tale of hate, sexism, and bullying. In *Proceedings of WWW*.

Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of CSCW*.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. In *Proceedings of ICWSM*.

Elizabeth F Churchill and Sara Bly. 2000. Culture vultures: Considering culture and communication in virtual environments. *SIGGroup Bulletin*.

Herbert H Clark. 1979. Responding to indirect speech acts. *Cognitive psychology*.

Herbert H Clark and Dale H Schunk. 1980. Polite responses to polite requests. *Cognition*.

Benjamin Collier and Julia Bear. 2012. Conflict, criticism, or confidence: An empirical examination of the gender gap in Wikipedia contributions. In *Proceedings of CSCW*.

Lewis A Coser. 1956. *The Functions of Social Conflict*. Routledge.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of ACL*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of ICWSM*.

Carsten K De Dreu and Laurie R Weingart. 2003. Task versus relationship conflict, team performance, and team member satisfaction: A meta-analysis. *Journal of Applied Psychology*.

Bruce Fraser. 1980. Conversational mitigation. *Journal of Pragmatics*.

Liye Fu, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2017. When confidence and competence collide: Effects on online decision-making discussions. In *Proceedings of WWW*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the Workshop on Abusive Language Online*.

Ali Gheitasy, José Abdelnour-Nocera, and Bonnie Nardi. 2015. Socio-technical gaps in online collaborative consumption (OCC): An example of the Etsy community. In *Proceedings of ICDC*.

Erving Goffman. 1955. On face-work: An analysis of ritual elements in social interaction. *Psychiatry*.

Christophe Henner and Maria Sefidari. 2016. Wikimedia Foundation Board on healthy Wikimedia community culture, inclusivity, and safe spaces. Wikimedia Blog.

Pamela J Hinds and Mark Mortensen. 2005. Understanding conflict in geographically distributed teams: The moderating effects of shared identity, shared context, and spontaneous communication. *Organization Science*.

Axel Hübler. 1983. *Understatements and Hedges in English*. John Benjamins Publishing.

Joseph M Kayany. 1998. Contexts of uninhibited online behavior: Flaming in social newsgroups on usenet. *Journal of the Association for Information Science and Technology*.

Aniket Kittur, Ed H Chi, and Bongwon Suh. 2009. What's in Wikipedia?: Mapping topics and conflict using socially annotated category structure. In *Proceedings of CHI*.

Aniket Kittur and Robert E Kraut. 2008. Harnessing the wisdom of crowds in Wikipedia: Quality through coordination. In *Proceedings of CSCW*.

Aniket Kittur, Bongwon Suh, Bryan A Pendleton, and Ed H Chi. 2007. He says, she says: Conflict and coordination in Wikipedia. In *Proceedings of CHI*.

Vinodh Krishnan and Jacob Eisenstein. 2015. "You're Mr. Lebowski, I'm the Dude": Inducing address term formality in signed social networks. In *Proceedings of NAACL*.

Ravi Kumar, Mohammad Mahdian, and Mary McGlohon. 2010. Dynamics of conversations. In *Proceedings of KDD*.

Haewoon Kwak, Jeremy Blackburn, and Seungyeop Han. 2015. Exploring cyberbullying and other toxic behavior in team competition online games. In *Proceedings of CHI*.

Robin T Lakoff. 1973. The logic of politeness: Minding your P's and Q's. In *Proceedings of the Chicago Linguistic Society*.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW*.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2016. Conversational markers of constructive discussions. In *Proceedings of NAACL*.

Vlad Niculae, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. 2015. Linguistic harbingers of betrayal: A case study on an online strategy game. In *Proceedings of ACL*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of WWW*.

Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In *Proceedings of MSR*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017a. Deep learning for user comment moderation. In *Proceedings of the Workshop on Abusive Language Online*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017b. Deeper attention to abusive user content moderation. In *Proceedings of EMNLP*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. In *Proceedings of NAACL*.

Paul R Rosenbaum. 2010. *Design of observational studies*. Springer.

Donald B Rubin. 2007. The design versus the analysis of observational studies for causal effects: Parallels with the design of randomized trials. *Statistics in Medicine*.

Tamara Shepherd, Alison Harvey, Tim Jordan, Sam Srauy, and Kate Miltner. 2015. Histories of hating. *Social Media + Society*.

Vivek K Singh, Marie L Radford, Qianjia Huang, and Susan Furrer. 2017. "They basically like destroyed the school one day": On newer app features and cyberbullying in schools. In *Proceedings of CSCW*.

Sara Owsley Sood, Elizabeth F Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of WWW*.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*.

Kal Turnbull. 2018. "Thats Bullshit" – Rude Enough for Removal? A Multi-Mod Perspective. Change My View Blog.

Jessica Vitak, Kalyani Chadha, Linda Steiner, and Zahra Ashktorab. 2017. Identifying women's experiences with and strategies for mitigating negative effects of online harassment. In *Proceedings of CSCW*.

Lu Wang and Claire Cardie. 2014. A piece of my mind: A sentiment analysis approach for online dispute detection. In *Proceedings of ACL*.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the World Wide Web. In *Proceedings of the Workshop on Language in Social Media*.

Wikimedia Support and Safety Team. 2015. Harassment survey. Wikimedia Foundation.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2016. Wikipedia talk labels: Toxicity.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of WWW*.

Naomi Yamashita and Toru Ishida. 2006. Automatic prediction of misconceptions in multilingual computer-mediated communication. In *Proceedings of IUI*.

Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian D Davison, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on Web 2.0. In *Proceedings of the Workshop on Content Analysis in the Web 2.0*.

Amy X Zhang, Bryan Culbertson, and Praveen Paritosh. 2017a. Characterizing online discussion using coarse discourse sequences. In *Proceedings of ICWSM*.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in Oxford-style debates. In *Proceedings of NAACL*.

Justine Zhang, Arthur Spirling, and Cristian Danescu-Niculescu-Mizil. 2017b. Asking too much? The rhetorical role of questions in political discourse. In *Proceedings of EMNLP*.

Are BLEU and Meaning Representation in Opposition?

Ondřej Cífka and **Ondřej Bojar**
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{cifka,bojar}@ufal.mff.cuni.cz

Abstract

One of possible ways of obtaining continuous-space sentence representations is by training neural machine translation (NMT) systems. The recent attention mechanism however removes the single point in the neural network from which the source sentence representation can be extracted. We propose several variations of the attentive NMT architecture bringing this meeting point back. Empirical evaluation suggests that the better the translation quality, the worse the learned sentence representations serve in a wide range of classification and similarity tasks.

1 Introduction

Deep learning has brought the possibility of automatically learning continuous representations of sentences. On the one hand, such representations can be geared towards particular tasks such as classifying the sentence in various aspects (e.g. sentiment, register, question type) or relating the sentence to other sentences (e.g. semantic similarity, paraphrasing, entailment). On the other hand, we can aim at "universal" sentence representations, that is representations performing reasonably well in a range of such tasks.

Regardless the evaluation criterion, the representations can be learned either in an unsupervised way (from simple, unannotated texts) or supervised, relying on manually constructed training sets of sentences equipped with annotations of the appropriate type. A different approach is to obtain sentence representations from training neural machine translation models (Hill et al., 2016).

Since Hill et al. (2016), NMT has seen substantial advances in translation quality and it is thus natural to ask how these improvements affect the learned representations.

One of the key technological changes was the introduction of "attention" (Bahdanau et al., 2014), making it even the very central component in the network (Vaswani et al., 2017). Attention allows the NMT system to dynamically choose which parts of the source are most important when deciding on the current output token. As a consequence, there is no longer a static vector representation of the sentence available in the system.

In this paper, we remove this limitation by proposing a novel encoder-decoder architecture with a structured fixed-size representation of the input that still allows the decoder to explicitly focus on different parts of the input. In other words, our NMT system has both the capacity to attend to various parts of the input and to produce static representations of input sentences.

We train this architecture on English-to-German and English-to-Czech translation and evaluate the learned representations of English on a wide range of tasks in order to assess its performance in learning "universal" meaning representations.

In Section 2, we briefly review recent efforts in obtaining sentence representations. In Section 3, we introduce a number of variants of our novel architecture. Section 4 describes some standard and our own methods for evaluating sentence representations. Section 5 then provides experimental results: translation and representation quality. The relation between the two is discussed in Section 6.

2 Related Work

The properties of continuous sentence representations have always been of interest to researchers working on neural machine translation. In the first works on RNN sequence-to-sequence models, Cho et al. (2014) and Sutskever et al. (2014)

	Bahdanau et al. ATTN	Sutskever et al. FINAL	Cho et al. FINAL-CTX	*POOL	*POOL-CTX	ATTN-CTX	Compound attention ATTN-ATTN
Encoder states used	all	final	final	all	all	all	all
Combined using …	—	—	—	pooling	pooling	inner att.	inner att.
Sent. emb. available	✗	✓	✓	✓	✓	✓	✓
Dec. attends to enc. states	✓	✗	✗	✗	✗	✗	✗
Dec. attends to sent. emb.	✗	✗	✗	✗	✗	✗	✓
Sent. emb. used in …	—	init	init+ctx	init	init+ctx	init+ctx	input for att.

Table 1: Different RNN-based architectures and their properties. Legend: "pooling" – vectors combined by mean or max (AVGPOOL, MAXPOOL); "sent. emb." – sentence embedding, i.e. the fixed-size sentence representation computed by the encoder. "init" – initial decoder state. "ctx" – context vector, i.e. input for the decoder cell. "input for att." – input for the decoder attention.

provided visualizations of the phrase and sentence embedding spaces and observed that they reflect semantic and syntactic structure to some extent.

Hill et al. (2016) perform a systematic evaluation of sentence representation in different models, including NMT, by applying them to various sentence classification tasks and by relating semantic similarity to closeness in the representation space.

Shi et al. (2016) investigate the syntactic properties of representations learned by NMT systems by predicting sentence- and word-level syntactic labels (e.g. tense, part of speech) and by generating syntax trees from these representations.

Schwenk and Douze (2017) aim to learn language-independent sentence representations using NMT systems with multiple source and target languages. They do not consider the attention mechanism and evaluate primarily by similarity scores of the learned representations for similar sentences (within or across languages).

3 Model Architectures

Our proposed model architectures differ in (a) which encoder states are considered in subsequent processing, (b) how they are combined, and (c) how they are used in the decoder.

Table 1 summarizes all the examined configurations of RNN-based models. The first three (ATTN, FINAL, FINAL-CTX) correspond roughly to the standard sequence-to-sequence models, Bahdanau et al. (2014), Sutskever et al. (2014) and Cho et al. (2014), resp. The last column (ATTN-ATTN) is our main proposed architecture: compound attention, described here in Section 3.1.

In addition to RNN-based models, we experiment with the Transformer model, see Section 3.3.

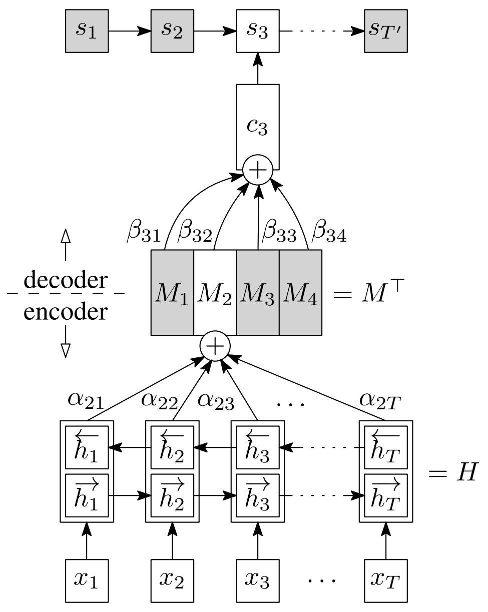


Figure 1: An illustration of compound attention with 4 attention heads. The figure shows the computations that result in the decoder state s_3 (in addition, each state s_i depends on the previous target token y_{i-1}). Note that the matrix M is the same for all positions in the output sentence and it can thus serve as the source sentence representation.

3.1 Compound Attention

Our compound attention model incorporates attention in both the encoder and the decoder, Fig. 1.

Encoder with inner attention. First, we process the input sequence $x_1, x_2, \ldots, x_T$ using a bi-directional recurrent network with gated recurrent units (GRU, Cho et al., 2014):

$$\begin{aligned}\overrightarrow{h_t} &= \overrightarrow{\text{GRU}}(x_t, \overrightarrow{h_{t-1}}),\\ \overleftarrow{h_t} &= \overleftarrow{\text{GRU}}(x_t, \overleftarrow{h_{t+1}}),\\ h_t &= [\overrightarrow{h_t}, \overleftarrow{h_t}].\end{aligned}$$

We denote by u the combined number of units in the two RNNs, i.e. the dimensionality of h_t.

Next, our goal is to combine the states $(h_1, h_2, \dots, h_T) = H$ of the encoder into a vector of fixed dimensionality that represents the entire sentence. Traditional seq2seq models concatenate the final states of both encoder RNNs ($\overrightarrow{h_T}$ and $\overleftarrow{h_1}$) to obtain the sentence representation (denoted as FINAL in Table 1). Another option is to combine all encoder states using the average or maximum over time (Collobert and Weston, 2008; Schwenk and Douze, 2017) (AVGPOOL and MAXPOOL in Table 1 and following).

We adopt an alternative approach, which is to use *inner attention*[1] (Liu et al., 2016; Li et al., 2016) to compute several weighted averages of the encoder states (Lin et al., 2017). The main motivation for incorporating these multiple "views" of the state sequence is that it removes the need for the RNN cell to accumulate the representation of the whole sentence as it processes the input, and therefore it should have more capacity for modeling local dependencies.

Specifically, we fix a number r, the number of *attention heads*, and compute an $r \times T$ matrix A of attention weights α_{jt}, representing the importance of position t in the input for the j^{th} attention head. We then use this matrix to compute r weighted sums of the encoder states, which become the rows of a new matrix M:

$$M = AH. \qquad (1)$$

A vector representation of the source sentence (the "sentence embedding") can be obtained by flattening the matrix M. In our experiments, we project the encoder states $h_1, h_2, \dots, h_T$ down to a given dimensionality before applying Eq. (1), so that we can control the size of the representation.

Following Lin et al. (2017), we compute the attention matrix by feeding the encoder states to a two-layer feed-forward network:

$$A = \operatorname{softmax}(U \tanh(W H^\top)), \qquad (2)$$

where W and U are weight matrices of dimensions $d \times u$ and $r \times d$, respectively (d is the number of hidden units); the softmax function is applied along the second dimension, i.e. across the encoder states.

[1]Some papers call the same or similar approach *self-attention* or *single-time attention*.

Attentive decoder. In vanilla seq2seq models with a fixed-size sentence representation, the decoder is usually conditioned on this representation via the initial RNN state. We propose to instead leverage the structured sentence embedding by applying attention to its components. This is no different from the classical attention mechanism used in NMT (Bahdanau et al., 2014), except that it acts on this fixed-size representation instead of the sequence of encoder states.

In the i^{th} decoding step, the attention mechanism computes a distribution $\{\beta_{ij}\}_{j=1}^{r}$ over the r components of the structured representation. This is then used to weight these components to obtain the context vector c_i, which in turn is used to update the decoder state. Again, we can write this in matrix form as

$$C = BM, \qquad (3)$$

where $B = (\beta_{ij})_{i=1,j=1}^{T',r}$ is the attention matrix and $C = (c_i, c_2, \dots, c_{T'})$ are the context vectors.

Note that by combining Eqs. (1) and (3), we get

$$C = (BA)H. \qquad (4)$$

Hence, the composition of the encoder and decoder attentions (the "compound attention") defines an implicit alignment between the source and the target sequence. From this viewpoint, our model can be regarded as a restriction of the conventional attention model.

The decoder uses a conditional GRU cell (cGRU$_{\text{att}}$; Sennrich et al., 2017), which consists of two consecutively applied GRU blocks. The first block processes the previous target token y_{i-1}, while the second block receives the context vector c_i and predicts the next target token y_i.

3.2 Constant Context

Compared to the FINAL model, the compound attention architecture described in the previous section undoubtedly benefits from the fact that the decoder is presented with information from the encoder (i.e. the context vectors c_i) in every decoding step. To investigate this effect, we include baseline models where we replace all context vectors c_i with the entire sentence embedding (indicated by the suffix "-CTX" in Table 1). Specifically, we provide either the flattened matrix M (for models with inner attention; ATTN or ATTN-CTX), the final state of the encoder (FINAL-CTX), or the result of mean- or max-pooling (*POOL-CTX) as a constant input to the decoder cell.

Name	Cl.	Train	Test	Task	Example Input and Label
MR	2	11k	—	sentiment (movies)	*an idealistic love story that brings out the latent 15-year-old romantic in everyone.* (+)
CR	2	4k	—	product review polarity	*no way to contact their customer service.* (−)
SUBJ	2	10k	—	subjectivity	*a little weak – and it isn't that funny.* (subjective)
MPQA	2	11k	—	opinion polarity	*was hoping* (+), *breach of the very constitution* (−)
SST2	2	68k	2k	sentiment (movies)	*contains very few laughs and even less surprises* (−)
SST5	5	10k	2k	sentiment (movies)	*it's worth taking the kids to.* (4)
TREC	6	5k	500	question type	*What was Einstein s IQ?* (NUM)
MRPC	2	4k	2k	semantic equivalence	*Lawtey is not the first faith-based program in Florida's prison system. / But Lawtey is the first entire prison to take that path.* (−)
SNLI	3	559k	10k	natural language inference	*Two doctors perform surgery on patient. / Two surgeons are having lunch.* (contradiction)
SICK-E	3	5k	5k	natural language inference	*A group of people is near the ocean / A crowd of people is near the water* (entailment)

Table 2: SentEval classification tasks. Tasks without a test set use 10-fold cross-validation.

Name	Train	Test	Method
SICK-R	5k	5k	regression
STSB	7k	1k	regression
STS12	—	3k	cosine similarity
STS13	—	2k	cosine similarity
STS14	—	4k	cosine similarity
STS15	—	9k	cosine similarity
STS16	—	9k	cosine similarity

Table 3: SentEval semantic relatedness tasks.

3.3 Transformer with Inner Attention

The Transformer (Vaswani et al., 2017) is a recently proposed model based entirely on feed-forward layers and attention. It consists of an encoder and a decoder, each with 6 layers, consisting of multi-head attention on the previous layer and a position-wise feed-forward network.

In order to introduce a fixed-size sentence representation into the model, we modify it by adding inner attention after the last encoder layer. The attention in the decoder then operates on the components of this representation (i.e. the rows of the matrix M). This variation on the Transformer model corresponds to the ATTN-ATTN column in Table 1 and is therefore denoted TRF-ATTN-ATTN.

4 Representation Evaluation

Continuous sentence representations can be evaluated in many ways, see e.g. Kiros et al. (2015), Conneau et al. (2017) or the RepEval workshops.[2]

We evaluate our learned representations with classification and similarity tasks from SentEval (Section 4.1) and by examining clusters of sentence paraphrase representations (Section 4.2).

4.1 SentEval

We perform evaluation on 10 classification and 7 similarity tasks using the SentEval[3] (Conneau et al., 2017) evaluation tool. This is a superset of the tasks from Kiros et al. (2015).

Table 2 describes the classification tasks (number of classes, data size, task type and an example) and Table 3 lists the similarity tasks. The similarity (relatedness) datasets contain pairs of sentences labeled with a real-valued similarity score. A given sentence representation model is evaluated either by learning to directly predict this score given the respective sentence embeddings ("regression"), or by computing the cosine similarity of the embeddings ("similarity") without the need of any training. In both cases, Pearson and Spearman correlation of the predictions with the gold ratings is reported.

See Dolan et al. (2004) for details on MRPC and Hill et al. (2016) for the remaining tasks.

4.2 Paraphrases

We also evaluate the representation of paraphrases. We use two paraphrase sources for this purpose: COCO and HyTER Networks.

COCO (Common Objects in Context; Lin et al., 2014) is an object recognition and image captioning dataset, containing 5 captions for each image. We extracted the captions from its validation set to form a set of $5 \times 5\text{k} = 25\text{k}$ sentences grouped by the source image. The average sentence length is 11 tokens and the vocabulary size is 9k types.

HyTER Networks (Dreyer and Marcu, 2014) are large finite-state networks representing a sub-

[2]https://repeval2017.github.io/

[3]https://github.com/facebookresearch/SentEval/

set of all possible English translations of 102 Arabic and 102 Chinese sentences. The networks were built by manually based on reference sentences in Arabic, Chinese and English. Each network contains up to hundreds of thousands of possible translations of the given source sentence. We randomly generated 500 translations for each source sentence, obtaining a corpus of 102k sentences grouped into 204 clusters, each containing 500 paraphrases. The average length of the 102k English sentences is 28 tokens and the vocabulary size is 11k token types.

For every model, we encode each dataset to obtain a set of sentence embeddings with cluster labels. We then compute the following metrics:

Cluster classification accuracy (denoted "Cl"). We remove 1 point (COCO) or half of the points (HyTER) from each cluster, and fit an LDA classifier on the rest. We then compute the accuracy of the classifier on the removed points.

Nearest-neighbor paraphrase retrieval accuracy (NN). For each point, we find its nearest neighbor according to cosine or L_2 distance, and count how often the neighbor lies in the same cluster as the original point.

Inverse Davies-Bouldin index (iDB). The Davies-Bouldin index (Davies and Bouldin, 1979) measures cluster separation. For every pair of clusters, we compute the ratio R_{ij} of their combined scatter (average L_2 distance to the centroid) $S_i + S_j$ and the L_2 distance of their centroids d_{ij}, then average the maximum values for all clusters:

$$R_{ij} = \frac{S_i + S_j}{d_{ij}} \tag{5}$$

$$\mathrm{DB} = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} R_{ij} \tag{6}$$

The lower the DB index, the better the separation. To match with the rest of our metrics, we take its inverse: $\mathrm{iDB} = \frac{1}{\mathrm{DB}}$.

5 Experimental Results

We trained English-to-German and English-to-Czech NMT models using Neural Monkey[4] (Helcl and Libovický, 2017a). In the following, we distinguish these models using the code of the target language, i.e. *de* or *cs*.

The *de* models were trained on the Multi30K multilingual image caption dataset (Elliott et al., 2016), extended by Helcl and Libovický (2017b), who acquired additional parallel data using back-translation (Sennrich et al., 2016) and perplexity-based selection (Yasuda et al., 2008). This extended dataset contains 410k sentence pairs, with the average sentence length of 12 ± 4 tokens in English. We train each model for 20 epochs with the batch size of 32. We truecased the training data as well as all data we evaluate on. For German, we employed Neural Monkey's reversible pre-processing scheme, which expands contractions and performs morphological segmentation of determiners. We used a vocabulary of at most 30k tokens for each language (no subword units).

The *cs* models were trained on CzEng 1.7 (Bojar et al., 2016).[5] We used byte-pair encoding (BPE) with a vocabulary of 30k sub-word units, shared for both languages. For English, the average sentence length is 15 ± 19 BPE tokens and the original vocabulary size is 1.9M. We performed 1 training epoch with the batch size of 128 on the entire training section (57M sentence pairs).

The datasets for both *de* and *cs* models come with their respective development and test sets of sentence pairs, which we use for the evaluation of translation quality. (We use 1k randomly selected sentence pairs from CzEng 1.7 dtest as a development set. For evaluation, we use the entire etest.)

We also evaluate the InferSent model[6] (Conneau et al., 2017) as pre-trained on the natural language inference (NLI) task. InferSent has been shown to achieve state-of-the-art results on the SentEval tasks. We also include a bag-of-words baseline (GloVe-BOW) obtained by averaging GloVe[7] word vectors (Pennington et al., 2014).

5.1 Translation Quality

We estimate translation quality of the various models using single-reference case-sensitive BLEU (Papineni et al., 2002) as implemented in Neural Monkey (the reference implementation is `mteval-v13a.pl` from NIST or Moses).

Tables 4 and 5 provide the results on the two datasets. The *cs* dataset is much larger and the training takes much longer. We were thus able to experiment with only a subset of the possible model configurations.

[4]`https://github.com/ufal/neuralmonkey`

[5]`http://ufal.mff.cuni.cz/czeng/czeng17`

[6]`https://github.com/facebookresearch/InferSent`

[7]`https://nlp.stanford.edu/projects/glove/`

Model	Size	Heads	BLEU dev	BLEU test
de-ATTN	—	—	37.6	36.2
de-TRF	—	—	38.2	36.1
de-ATTN-ATTN	2400	12	36.2	34.8
de-ATTN-ATTN	1200	12	35.6	34.3
de-ATTN-ATTN	600	8	35.4	33.7
de-ATTN-ATTN	600	12	35.3	33.4
de-ATTN-ATTN	1200	6	35.0	33.2
de-ATTN-ATTN	600	6	35.1	33.2
de-TRF-ATTN-ATTN	600	3	32.3	30.1
de-ATTN-ATTN	600	3	31.4	29.4
de-ATTN-CTX	1200	12	30.6	29.2
de-ATTN-CTX	600	12	29.8	29.1
de-ATTN-CTX	600	8	29.8	28.9
de-ATTN-CTX	600	6	29.5	28.8
de-TRF-ATTN-ATTN	2400	12	30.6	28.5
de-MAXPOOL-CTX	600	—	27.8	28.1
de-FINAL-CTX	600	—	28.1	26.9
de-ATTN-CTX	600	3	27.8	26.9
de-AVGPOOL-CTX	600	—	27.1	26.5
de-ATTN-ATTN	600	1	27.2	26.0
de-TRF-ATTN-ATTN	600	6	26.5	25.8
de-TRF-ATTN-ATTN	1200	12	26.6	25.3
de-FINAL	600	—	23.9	23.8

Table 4: Translation quality of *de* models.

Model	Size	Heads	BLEU dev	BLEU test	Manual > others
cs-ATTN	—	—	22.8	22.2	89.1
cs-ATTN-ATTN	1000	8	19.1	18.4	78.8
cs-ATTN-ATTN	4000	4	18.4	17.9	—
cs-ATTN-ATTN	1000	4	17.5	17.1	—
cs-ATTN-CTX	1000	4	16.6	16.1	58.8
cs-FINAL-CTX	1000	—	16.1	15.5	—
cs-ATTN-ATTN	1000	1	15.3	14.8	49.1
cs-FINAL	1000	—	11.2	10.8	—
cs-AVGPOOL	1000	—	11.1	10.6	—
cs-MAXPOOL	1000	—	5.4	5.4	3.0

Table 5: Translation quality of *cs* models.

The columns "Size" and "Heads" specify the total size of sentence representation and the number of heads of encoder inner attention.

In both cases, the best performing is the ATTN Bahdanau et al. model, followed by Transformer (*de* only) and our ATTN-ATTN (compound attention). The non-attentive FINAL Cho et al. is the worst, except cs-MAXPOOL.

For 5 selected *cs* models, we also performed the WMT-style 5-way manual ranking on 200 sentence pairs. The annotations are interpreted as simulated pairwise comparisons. For each model, the final score is the number of times the model was judged better than the other model in the pair. Tied pairs are excluded. The results, shown in Table 5, confirm the automatic evaluation results.

We also checked the relation between BLEU and the number of heads and representation size. While there are many exceptions, the general tendency is that the larger the representation or the more heads, the higher the BLEU score. The Pearson correlation between BLEU and the number of heads is 0.87 for *cs* and 0.31 for *de*.

5.2 SentEval

Due to the large number of SentEval tasks, we present the results abridged in two different ways: by reporting averages (Table 6) and by showing only the best models in comparison with other methods (Table 7). The full results can be found in the supplementary material.

Table 6 provides averages of the classification and similarity results, along with the results of selected tasks (SNLI, SICK-E). As the baseline for classifications tasks, we assign the most frequent class to all test examples.[8] The *de* models are generally worse, most likely due to the higher OOV rate and overall simplicity of the training sentences. On *cs*, we see a clear pattern that more heads hurt the performance. The *de* set has more variations to consider but the results are less conclusive.

For the similarity results, it is worth noting that cs-ATTN-ATTN performs very well with 1 attention head but fails miserably with more heads. Otherwise, the relation to the number of heads is less clear.

Table 7 compares our strongest models with other approaches on all tasks. Besides InferSent and GloVe-BOW, we include SkipThought as evaluated by Conneau et al. (2017), and the NMT-based embeddings by Hill et al. (2016) trained on the English-French WMT15 dataset (this is the best result reported by Hill et al. for NMT).

We see that the supervised InferSent clearly outperforms all other models in all tasks except for MRPC and TREC. Results by Hill et al. are always lower than our best setups, except MRPC and TREC again. On classification tasks, our models are outperformed even by GloVe-BOW, except for the NLI tasks (SICK-E and SNLI) where cs-FINAL-CTX is better.

5.3 Paraphrase Scores

Table 6 also provides our measurements based on sentence paraphrases. For paraphrase retrieval (NN), we found cosine distance to work better

[8]For MR, CR, SUBJ, and MPQA, where there is no distinct test set, the class is established on the whole collection. For other tasks, the class is learned from the training set.

Name	Size	H.	SNLI	SICK-E	AvgAcc	AvgSim	Hy-Cl	Hy-NN	Hy-iDB	CO-Cl	CO-NN	CO-iDB
InferSent	4096	—	83.7	86.4	81.7	.70	99.99	100.00	0.579	31.58	26.21	0.367
GloVe-BOW	300	—	66.0	78.2	75.8	.59	99.94	100.00	0.654	34.28	19.72	0.352
cs-FINAL-CTX	1000	—	70.2	82.1	74.4	.60	99.92	100.00	0.406	23.20	16.07	0.346
cs-ATTN-ATTN	1000	1	69.3	80.8	73.4	.54	99.88	99.91	0.347	21.54	11.50	0.331
cs-FINAL	1000	—	69.2	81.1	73.2	.60	99.91	100.00	0.439	22.40	14.63	0.340
cs-MAXPOOL	1000	—	68.5	81.7	73.0	.60	99.86	100.00	0.447	21.76	16.34	0.348
cs-AVGPOOL	1000	—	67.8	79.7	72.4	.50	99.80	99.99	0.387	17.90	8.61	0.311
cs-ATTN-CTX	1000	4	66.0	79.5	72.2	.45	99.75	99.74	0.287	14.60	7.54	0.318
cs-ATTN-ATTN	4000	4	65.2	78.0	71.2	.39	99.54	98.98	0.252	11.52	5.51	0.303
cs-ATTN-ATTN	1000	4	64.6	78.0	70.8	.39	99.26	98.93	0.253	10.84	5.20	0.299
cs-ATTN-ATTN	1000	8	63.2	76.6	70.0	.36	99.41	98.09	0.243	10.24	4.64	0.287
de-MAXPOOL-CTX	600	—	68.0	78.8	67.1	.50	98.42	99.90	0.343	21.54	15.62	0.341
de-ATTN-CTX	1200	12	65.0	77.4	66.7	.52	98.88	99.91	0.347	20.06	16.68	0.348
de-ATTN-CTX	600	8	64.0	75.7	65.8	.51	98.11	99.90	0.348	21.64	17.32	0.349
de-AVGPOOL-CTX	600	—	65.2	77.5	65.6	.48	97.72	99.60	0.312	20.04	14.27	0.337
de-ATTN-CTX	600	12	61.9	76.0	65.5	.50	97.79	99.89	0.360	20.22	16.10	0.344
de-FINAL	600	—	64.7	77.0	65.3	.47	97.01	99.30	0.305	19.88	12.40	0.328
de-ATTN-CTX	600	3	63.3	76.0	65.3	.50	97.81	99.87	0.328	19.74	16.43	0.343
de-ATTN-ATTN	600	1	63.8	76.9	64.8	.50	97.70	99.73	0.352	19.74	16.26	0.340
de-ATTN-ATTN	600	3	61.5	74.7	64.5	.47	97.42	99.75	0.314	17.36	14.35	0.333
de-FINAL-CTX	600	—	62.6	76.2	64.5	.48	96.65	99.70	0.323	17.22	12.84	0.333
de-ATTN-ATTN	1200	6	59.6	72.3	64.3	.41	98.05	99.80	0.289	11.90	10.69	0.327
de-TRF-ATTN-ATTN	600	3	61.4	72.5	63.9	.49	95.79	99.64	0.315	15.76	14.04	0.340
de-ATTN-ATTN	1200	12	58.2	72.5	63.4	.43	97.15	99.65	0.283	12.18	11.97	0.330
de-ATTN-ATTN	2400	12	59.8	73.9	63.2	.41	98.69	99.77	0.287	10.26	10.94	0.326
de-TRF-ATTN-ATTN	2400	12	59.0	71.2	63.0	.46	95.82	99.03	0.307	5.66	14.53	0.339
de-ATTN-ATTN	600	6	57.5	70.9	62.6	.40	96.03	99.71	0.287	12.22	10.59	0.323
de-ATTN-ATTN	600	8	55.6	68.6	62.1	.39	95.32	99.73	0.275	10.22	10.58	0.325
de-TRF-ATTN-ATTN	600	6	59.5	71.0	61.9	.45	90.24	98.44	0.313	9.06	13.64	0.332
de-ATTN-ATTN	600	12	55.2	70.5	61.5	.40	95.16	99.64	0.278	9.62	10.47	0.323
de-TRF-ATTN-ATTN	1200	12	58.2	68.8	61.1	.46	90.71	98.22	0.301	7.06	13.70	0.333
de-ATTN-CTX	600	6	62.9	68.7	61.0	.43	98.11	99.86	0.358	20.44	15.57	0.342
LM perplexity (*cs*)			190.6	299.4	1150.2	1224.2		668.5			238.5	
% OOV (*cs*)			0.3	0.2	2.3	2.6		1.2			0.1	
LM perplexity (*de*)			38.8	65.0	3578.2	2010.6		3354.8			86.3	
% OOV (*de*)			1.5	1.7	17.8	16.2		19.3			1.9	

Table 6: Abridged SentEval and paraphrase evaluation results. Full results in supplementary material. AvgAcc is the average of all 10 SentEval classification tasks (see Table S1 in supplementary material), AvgSim averages all 7 similarity tasks (see Table S2). Hy- and CO- stand for HyTER and COCO, respectively. "H." is the number of attention heads. We give the out-of-vocabulary (OOV) rate and the perplexity of a 4-gram language model (LM) trained on the English side of the respective parallel corpus and evaluated on all available data for a given task.

Name	Size	H.	MR	CR	SUBJ	MPQA	SST2	SST5	TREC	MRPC	SICK-E	SNLI	AvgAcc
Most frequent baseline			50.0	63.8	50.0	68.8	49.9	23.1	18.8	66.5	56.7	34.3	48.19
InferSent	4096	—	**81.5**	**86.7**	92.7	**90.6**	**85.0**	**45.8**	88.2	76.6	**86.4**	(83.7)	**81.7**
Hill et al. en→fr[†]	2400	—	64.7	70.1	84.9	81.5	—	—	82.8	**96.1**	—	—	—
SkipThought-LN[†]	—	—	79.4	83.1	**93.7**	89.3	82.9	—	88.4	—	79.5	—	—
GloVe-BOW	300	—	77.0	78.2	91.1	87.9	81.0	44.4	82.0	72.3	78.2	66.0	75.8
cs-FINAL-CTX	1000	—	68.7	77.4	88.5	85.5	73.0	38.2	88.6	71.8	82.1	70.2	74.4
cs-ATTN-ATTN	1000	1	68.2	76.0	86.9	84.9	72.0	35.7	**89.0**	70.7	80.8	69.3	73.4

Name	Size	H.	SICK-R	STSB	STS12	STS13	STS14	STS15	STS16	AvgSim
InferSent	4096	—	**.88/.83**	**.76/.75**	**.59/.60**	**.59/.59**	**.70/.67**	**.71/.72**	**.71/.73**	**.70**
SkipThought-LN[†]	—	—	.85/ —	—	—	—	.44/.45	—	—	—
GloVe-BOW	300	—	.80/.72	.64/.62	.52/.53	.50/.51	.55/.56	.56/.59	.51/.58	.59
cs-FINAL-CTX	1000	—	.82/.76	.74/.74	.51/.53	.44/.44	.52/.50	.62/.61	.57/.58	.60
cs-ATTN-ATTN	1000	1	.81/.76	.73/.73	.46/.49	.32/.33	.45/.44	.53/.52	.47/.48	.54

Table 7: Comparison of state-of-the-art SentEval results with our best models and the Glove-BOW baseline. "H." is the number of attention heads. Reprinted results are marked with †, others are our measurements.

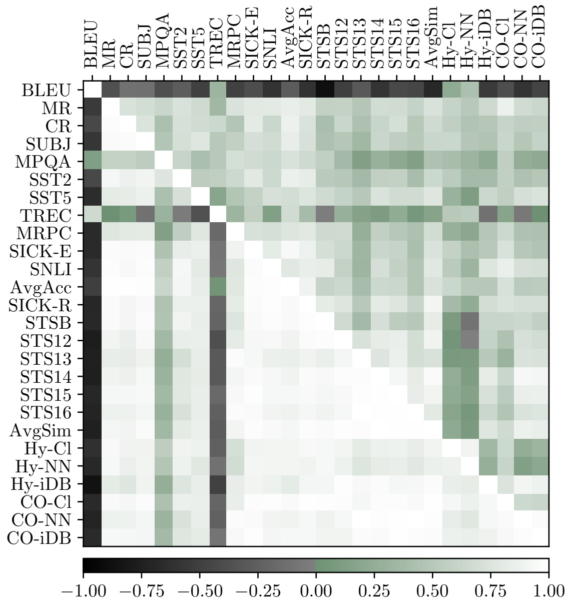


Figure 2: Pearson correlations. Upper triangle: *de* models, lower triangle: *cs* models. Positive values shown in shades of green. For similarity tasks, only the Pearson (not Spearman) coefficient is represented.

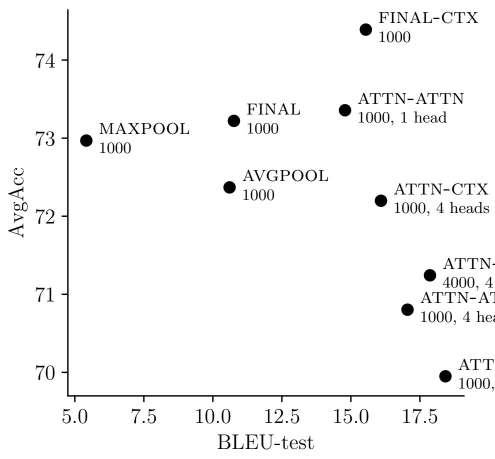


Figure 3: BLEU vs. AvgAcc for *cs* models.

than L_2 distance. We therefore do not list L_2-based results (except in the supplementary material).

This evaluation seems less stable and discerning than the previous two, but we can again confirm the victory of InferSent followed by our non-attentive *cs* models. *cs* and *de* models are no longer clearly separated.

6 Discussion

To assess the relation between the various measures of sentence representations and translation quality as estimated by BLEU, we plot a heatmap of Pearson correlations in Fig. 2. As one example, Fig. 3 details the *cs* models' BLEU scores and AvgAcc.

A good sign is that on the *cs* dataset, most metrics of representation are positively correlated (the pairwise Pearson correlation is 0.78 ± 0.32 on average), the outlier being TREC (-0.16 ± 0.16 correlation with the other metrics on average)

On the other hand, most representation metrics correlate with BLEU negatively (-0.57 ± 0.31) on *cs*. The pattern is less pronounced but still clear also on the *de* dataset.

A detailed understanding of what the learned representations contain is difficult. We can only speculate that if the NMT model has some capability for following the source sentence superficially, it will use it and spend its capacity on closely matching the target sentences rather than on deriving some representation of meaning which would reflect e.g. semantic similarity. We assume that this can be a direct consequence of NMT being trained for cross entropy: putting the exact word forms in exact positions as the target sentence requires. Performing well in single-reference BLEU is not an indication that the system understands the meaning but rather that it can maximize the chance of producing the n-grams required by the reference.

The negative correlation between the number of attention heads and the representation metrics from Fig. 3 (-0.81 ± 0.12 for *cs* and -0.18 ± 0.19 for *de*, on average) can be partly explained by the following observation. We plotted the induced alignments (e.g. Fig. 4) and noticed that the heads tend to "divide" the sentence into segments. While one would hope that the segments correspond to some meaningful units of the sentence (e.g. subject, predicate, object), we failed to find any such interpretation for ATTN-ATTN and for *cs* models in general. Instead, the heads divide the source sentence more or less equidistantly, as documented by Fig. 5. Such a multi-headed sentence representation is then *less* fit for representing e.g. paraphrases where the subject and object swap their position due to passivization, because their representations are then accessed by different heads, and thus end up in different parts of the sentence embedding vector.

For de-ATTN-CTX models, we observed a much

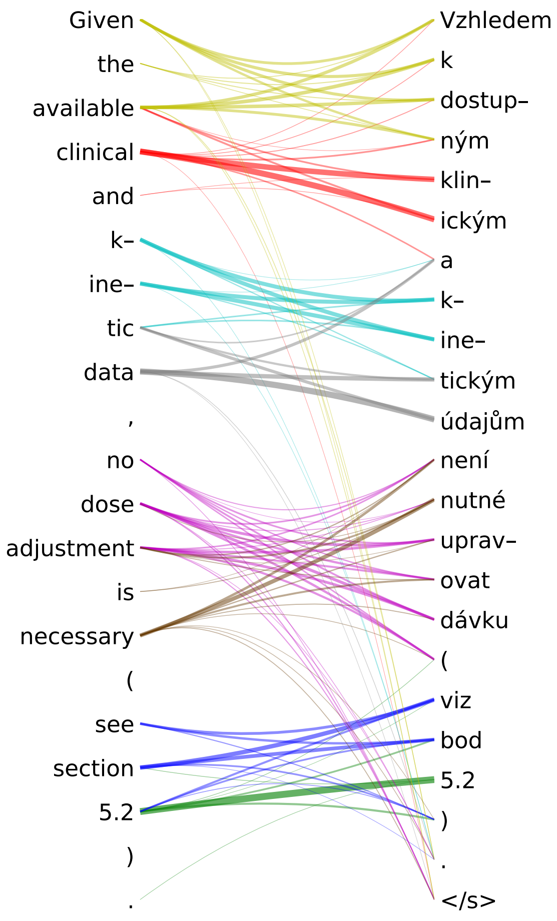


Figure 4: Alignment between a source sentence (left) and the output (right) as represented in the ATTN-ATTN model with 8 heads and size of 1000. Each color represents a different head; the stroke width indicates the alignment weight; weights ≤ 0.01 pruned out. (Best viewed in color.)

flatter distribution of attention weights for each head and, unlike in the other models, we were often able to identify a head focusing on the main verb. This difference between ATTN-ATTN and some ATTN-CTX models could be explained by the fact that in the former, the decoder is oblivious to the ordering of the heads (because of decoder attention), and hence it may not be useful for a given head to look for a specific syntactic or semantic role.

7 Conclusion

We presented a novel variation of attentive NMT models (Bahdanau et al., 2014; Vaswani et al., 2017) that again provides a single meeting point with a continuous representation of the source sentence. We evaluated these representations with a

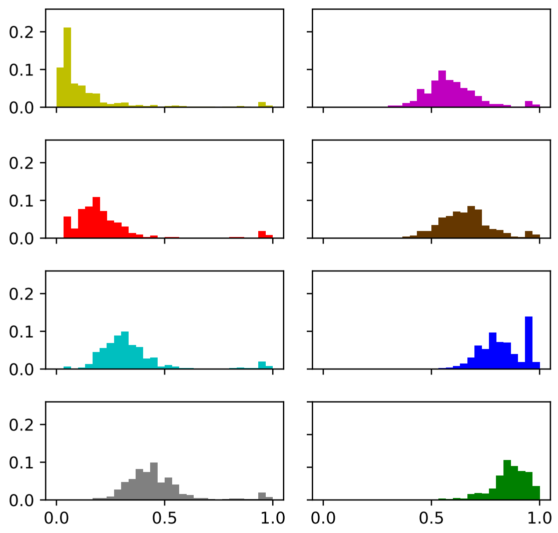


Figure 5: Attention weight by relative position in the source sentence (average over dev set excluding sentences shorter than 8 tokens). Same model as in Fig. 4. Each plot corresponds to one head.

number of measures reflecting how well the meaning of the source sentence is captured.

While our proposed "compound attention" leads to translation quality not much worse than the fully attentive model, it generally does not perform well in the meaning representation. Quite on the contrary, the better the BLEU score, the worse the meaning representation.

We believe that this observation is important for representation learning where bilingual MT now seems less likely to provide useful data, but perhaps more so for MT itself, where the struggle towards a high single-reference BLEU score (or even worse, cross entropy) leads to systems that refuse to consider the meaning of the sentence.

Acknowledgement

This work has been supported by the grants 18-24210S of the Czech Science Foundation, SVV 260 453 and "Progress" Q18+Q48 of Charles University, and using language resources distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (projects LM2015071 and OP VVV VI CZ.02.1.01/0.0/0.0/16 013/0001781).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ondřej Bojar et al. 2016. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *Text, Speech, and Dialogue (TSD)*, number 9924 in LNAI, pages 231–238.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

David L. Davies and Donald W. Bouldin. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1:224–227.

William B. Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*.

Markus Dreyer and Daniel Marcu. 2014. HyTER networks of selected OpenMT08/09 sentences. Linguistic Data Consortium. LDC2014T09.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. *CoRR*, abs/1605.00459.

Jindřich Helcl and Jindřich Libovický. 2017a. Neural Monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, 107(1):5–17.

Jindřich Helcl and Jindřich Libovický. 2017b. CUNI System for the WMT17 Multimodal Traslation Task.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS Vol. 2*, NIPS'15, pages 3294–3302.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *CoRR*, abs/1607.06275.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. volume abs/1704.04154.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Rico Sennrich et al. 2017. Nematus: a toolkit for neural machine translation. In *EACL*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Keiji Yasuda, Ruiqiang Zhang, Hirofumi Yamamoto, and Eiichiro Sumita. 2008. Method of selecting training data to build a compact and efficient translation model. In *IJCNLP*.

Automatic Metric Validation for Grammatical Error Correction

Leshem Choshen[1] and Omri Abend[1,2]
[1]School of Computer Science and Engineering, [2] Department of Cognitive Sciences
The Hebrew University of Jerusalem
`leshem.choshen@mail.huji.ac.il, oabend@cs.huji.ac.il`

Abstract

Metric validation in Grammatical Error Correction (GEC) is currently done by observing the correlation between human and metric-induced rankings. However, such correlation studies are costly, methodologically troublesome, and suffer from low inter-rater agreement. We propose MAEGE, an automatic methodology for GEC metric validation, that overcomes many of the difficulties with existing practices. Experiments with MAEGE shed a new light on metric quality, showing for example that the standard M^2 metric fares poorly on corpus-level ranking. Moreover, we use MAEGE to perform a detailed analysis of metric behavior, showing that correcting some types of errors is consistently penalized by existing metrics.

1 Introduction

Much recent effort has been devoted to automatic evaluation, both within GEC (Napoles et al., 2015; Felice and Briscoe, 2015; Ng et al., 2014; Dahlmeier and Ng, 2012, see §2), and more generally in text-to-text generation tasks. Within Machine Translation (MT), an annual shared task is devoted to automatic metric development, accompanied by an extensive analysis of metric behavior (Bojar et al., 2017). Metric validation is also raising interest in GEC, with several recent works on the subject (Grundkiewicz et al., 2015; Napoles et al., 2015, 2016b; Sakaguchi et al., 2016), all using correlation with human rankings (henceforth, *CHR*) as their methodology.

Human rankings are often considered as ground truth in text-to-text generation, but using them reliably can be challenging. Other than the costs of compiling a sizable validation set, human rankings are known to yield poor inter-rater agreement in MT (Bojar et al., 2011; Lopez, 2012; Graham et al., 2012), and to introduce a number of methodological problems that are difficult to overcome, notably the treatment of ties in the rankings and uncomparable sentences (see §3). These difficulties have motivated several proposals to alter the MT metric validation protocol (Koehn, 2012; Dras, 2015), leading to a recent abandoning of evaluation by human rankings due to its unreliability (Graham et al., 2015; Bojar et al., 2016). These conclusions have not yet been implemented in GEC, despite their relevance. In §3 we show that human rankings in GEC also suffer from low inter-rater agreement, motivating the development of alternative methodologies.

The main contribution of this paper is an automatic methodology for metric validation in GEC called MAEGE (Methodology for Automatic Evaluation of GEC Evaluation), which addresses these difficulties. MAEGE requires no human rankings, and instead uses a corpus with gold standard GEC annotation to generate lattices of corrections with similar meanings but varying degrees of grammaticality. For each such lattice, MAEGE generates a partial order of correction quality, a quality score for each correction, and the number and types of edits required to fully correct each. It then computes the correlation of the induced partial order with the metric-induced rankings.

MAEGE addresses many of the problems with existing methodology:

- Human rankings yield low inter-rater and intra-rater agreement (§3). Indeed, Choshen and Abend (2018a) show that while annotators often generate different corrections given a sentence, they generally agree on whether a correction is valid or not. Unlike CHR, MAEGE bases its scores on human corrections, rather than on rankings.

- CHR uses system outputs to obtain human rankings, which may be misleading, as systems may share similar biases, thus neglecting to evaluate some types of valid corrections (§7). MAEGE addresses this issue by systematically traversing an inclusive space of corrections.

- The difficulty in handling ties is addressed by only evaluating correction pairs where one contains a sub-set of the errors of the other, and is therefore clearly better.

- MAEGE uses established statistical tests for determining the significance of its results, thereby avoiding ad-hoc methodologies used in CHR to tackle potential biases in human rankings (§5, §6).

In experiments on the standard NUCLE test set (Dahlmeier et al., 2013), we find that MAEGE often disagrees with CHR as to the quality of existing metrics. For example, we find that the standard GEC metric, M^2, is a poor predictor of corpus-level ranking, but a good predictor of sentence-level pair-wise rankings. The best predictor of corpus-level quality by MAEGE is the reference-less LT metric (Miłkowski, 2010; Napoles et al., 2016b), while of the reference-based metrics, GLEU (Napoles et al., 2015) fares best.

In addition to measuring metric reliability, MAEGE can also be used to analyze the sensitivities of the metrics to corrections of different types, which to our knowledge is a novel contribution of this work. Specifically, we find that not only are valid edits of some error types better rewarded than others, but that correcting certain error types is consistently penalized by existing metrics (Section 7). The importance of interpretability and detail in evaluation practices (as opposed to just providing bottom-line figures), has also been stressed in MT evaluation (e.g., Birch et al., 2016).

2 Examined Metrics

We turn to presenting the metrics we experiment with. The standard practice in GEC evaluation is to define differences between the source and a correction (or a reference) as a set of edits (Dale et al., 2012). An edit is a contiguous span of tokens to be edited, a substitute string, and the corrected error type. For example: "I want book" might have an edit (2-3, "a book", ArtOrDet); applying the edit results in "I want a book". Edits are defined (by the annotation guidelines) to be maximally independent, so that each edit can be applied independently of the others. We denote the examined set of metrics with METRICS.

BLEU. BLEU (Papineni et al., 2002) is a reference-based metric that averages the output-reference n-gram overlap precision values over different ns. While commonly used in MT and other text generation tasks (Sennrich et al., 2017; Krishna et al., 2017; Yu et al., 2017), BLEU was shown to be a problematic metric in monolingual translation tasks, in which much of the source sentence should remain unchanged (Xu et al., 2016). We use the NLTK implementation of BLEU, using smoothing method 3 by Chen and Cherry (2014).

GLEU. GLEU (Napoles et al., 2015) is a reference-based GEC metric inspired by BLEU. Recently, it was updated to better address multiple references (Napoles et al., 2016a). GLEU rewards n-gram overlap of the correction with the reference and penalizes unchanged n-grams in the correction that are changed in the reference.

iBLEU. iBLEU (Sun and Zhou, 2012) was introduced to monolingual translation in order to balance BLEU, by averaging it with the BLEU score of the source and the output. This yields a metric that rewards similarity to the source, and not only overlap with the reference:

$$iBLEU(S,R,O) = \alpha BLEU(O,R) - (1-\alpha) BLEU(O,S)$$

We set $\alpha = 0.8$ as suggested by Sun and Zhou.

F-Score computes the overlap of edits to the source in the reference, and in the output. As system edits can be constructed in multiple ways, the standard M^2 scorer (Dahlmeier and Ng, 2012) computes the set of edits that yields the maximum F-score. As M^2 requires edits from the source to the reference, and as MAEGE generates new source sentences, we use an established protocol to automatically construct edits from pairs of strings (Felice et al., 2016; Bryant et al., 2017). The protocol was shown to produce similar M^2 scores to those produced with manual edits. Following common practice, we use the Precision-oriented $F_{0.5}$.

SARI. SARI (Xu et al., 2016) is a reference-based metric proposed for sentence simplification.

SARI averages three scores, measuring the extent to which n-grams are correctly added to the source, deleted from it and retained in it. Where multiple references are present, SARI's score is determined not as the maximum single-reference score, but some averaging over them. As this may lead to an unintuitive case, where a correction which is identical to the output gets a score of less than 1, we experiment with an additional metric, MAX-SARI, which coincides with SARI for a single reference, and computes the maximum single-reference SARI score for multiple-references.

Levenshtein Distance. We use the Levenshtein distance (Kruskal and Sankoff, 1983), i.e., the number of character edits needed to convert one string to another, between the correction and its closest reference ($MinLD_{O\to R}$). To enrich the discussion, we also report results with a measure of conservatism, $LD_{S\to O}$, i.e., the Levenshtein distance between the correction and the source. Both distances are normalized by the number of characters in the second string (R, O respectively). In order to convert these distance measures into measures of similarity, we report $1 - \frac{LD(c1,c2)}{len(c1)}$.

Grammaticality is a reference-less metric, which uses grammatical error detection tools to assess the grammaticality of GEC system outputs. We use LT (Miłkowski, 2010), the best performing non-proprietary grammaticality metric (Napoles et al., 2016b). The detection tool at the base of LT can be much improved. Indeed, Napoles et al. (2016b) reported that the proprietary tool they used detected 15 times more errors than LT. A sentence's score is defined to be $1 - \frac{\#errors}{\#tokens}$. See (Asano et al., 2017; Choshen and Abend, 2018b) for additional reference-less measures, published concurrently with this work.

I-Measure. I-Measure (Felice and Briscoe, 2015) is a weighted accuracy metric over tokens. I-measure rank determines whether a correction is better than the source and to what extent. Unlike in this paper, I-measure assumes that every pair of intersecting edits (i.e., edits whose spans of tokens overlap) are alternating, and that non-intersecting edits are independent. Consequently, where multiple references are present, it extends the set of references, by generating every possible combination of independent edits. As the number of combinations is generally exponential in the number of references, the procedure can be severely inefficient.

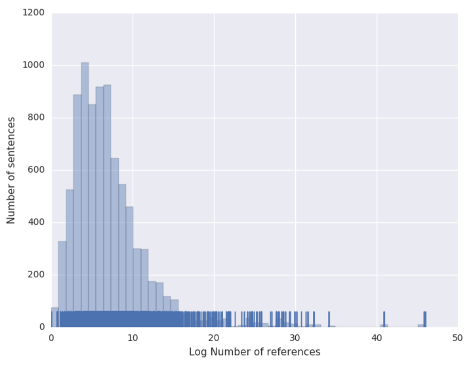


Figure 1: Histogram and rug plot of the log number of references under I-measure assumptions, i.e. overlapping edits alternate as valid corrections of the same error. There are billions of ways to combine 8 references on average.

Indeed, a sentence in the test set has 3.5 billion references on average, where the median is 512 (See Figure 1). I-measure can also be run without generating new references, but despite parallelization efforts, this version did not terminate after 140 CPU days, while the cumulative CPU time of the rest of the metrics was less than 1.5 days.

3 Human Ranking Experiments

Correlation with human rankings (CHR) is the standard methodology for assessing the validity of GEC metrics. While informative, human rankings are costly to produce, present low inter-rater agreement (shown for MT evaluation in (Bojar et al., 2011; Dras, 2015)), and introduce methodological difficulties that are hard to overcome. We begin by showing that existing sets of human rankings produce inconsistent results with respect to the quality of different metrics, and proceed by proposing an improved protocol for computing this correlation in the future.

There are two existing sets of human rankings for GEC that were compiled concurrently: GJG15 by Grundkiewicz et al. (2015), and NSPT15 by Napoles et al. (2015). Both sets are based on system outputs from the CoNLL 2014 (Ng et al., 2014) shared task, using sentences from the NUCLE test set. We compute CHR against each. System-level correlations are computed by TrueSkill (Sakaguchi et al., 2014), which adopts its methodology from MT.[1]

[1]There's a minor problem in the output of the NTHU system: a part of the input is given as sentence 39 and sentence 43 is missing. We corrected it to avoid unduly penalizing NTHU for all the sentences in this range.

Table 1 shows CHR with Spearman ρ (Pearson r shows similar trends). Results on the two datasets diverge considerably, despite their use of the same systems and corpus (albeit a different sub-set thereof). For example, BLEU receives a high positive correlation on GJG15, but a negative one on NSPT15; GLEU receives a correlation of 0.51 against GJG15 and 0.76 against NSPT15; and M^2 ranges between 0.4 (GJG15) and 0.7 (NSPT15). In fact, this variance is already apparent in the *published* correlations of GLEU, e.g., Napoles et al. (2015) reported a ρ of 0.56 against NSPT15 and Napoles et al. (2016b) reported a ρ of 0.85 against GJG15.[2] This variance in the metrics' scores is an example of the low agreement between human rankings, echoing similar findings in MT (Bojar et al., 2011; Lopez, 2012; Dras, 2015).

Another source of inconsistency in CHR is that the rankings are relative and sampled, so datasets rank different sets of outputs (Lopez, 2012). For example, if a system is judged against the best systems more often then others, it may unjustly receive a lower score. TrueSkill is the best known practice to tackle such issues (Bojar et al., 2014), but it produces a probabilistic corpus-level score, which can vary between runs (Sakaguchi et al., 2016).[3] This makes CHR more difficult to interpret, compared to classic correlation coefficients.

We conclude by proposing a practice for reporting CHR in future work. First, we combine both sets of human judgments to arrive at the statistically most powerful test. Second, we compute the metrics' corpus-level rankings according to the same subset of sentences used for human rankings. The current practice of allowing metrics to rank systems based on their output on the entire CoNLL test set (while human rankings are only collected for a sub-set thereof), may bias the results due to potential non-uniform system performance on the test set. We report CHR according to the proposed protocol in Table 1 (left column).

	Combined		GJG15		NSPT15	
	ρ	P-val	ρ	Rank	ρ	Rank
GLEU	0.771	0.001	0.512	1	0.758	1
LT	0.692	0.006	0.358	4	0.615	3
M^2	0.626	0.017	0.398	3	0.703	2
SARI	0.596	0.025	0.323	6	0.599	4
MAX-SARI	0.552	0.041	0.292	7	0.577	5
$MinLD_{O\rightarrow R}$	0.191	0.513	0.350	5	-0.187	7
BLEU	0.143	0.626	0.455	2	-0.126	6
iBLEU	-0.059	0.840	0.226	8	-0.462	8
$LD_{S\rightarrow O}$	-0.481	0.081	-0.178		-0.505	

Table 1: Metrics correlation with human judgments. The *Combined* column presents the Spearman correlation coefficient (ρ) according to the combined set of human rankings, with its associated P-value. The GJG15 and NSPT15 columns present the Spearman correlation according to the two sets of human rankings, as well as the rank of the metric according to this correlation. Measures are ordered by their rank in the combined human judgments. The discrepancy between the ρ values obtained against GJG15 and NSPT15 demonstrate low inter-rater agreement in human rankings.

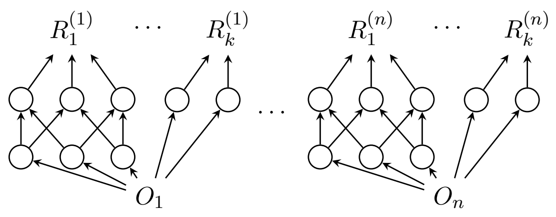


Figure 2: An illustration of the generated corrections lattices. The O_is are the original sentences, directed edges represent an application of an edit and $R_j^{(i)}$ is the j-th perfect correction of O_i (i.e., the perfect correction that result from applying all the edits of the j-th annotation of O_i).

4 Constructing Lattices of Corrections

In the following sections we present MAEGE an alternative methodology to CHR, which uses human corrections to induce more reliable and scalable rankings to compare metrics against. We begin our presentation by detailing the method MAEGE uses to generate source-correction pairs and a partial order between them. MAEGE operates by using a corpus with gold annotation, given as edits, to generate lattices of corrections, each defined by a sub-set of the edits. Within the lattice, every pair of sentences can be regarded as a potential source and a potential output. We create sentence chains, in an increasing order of quality, taking a source sentence and applying edits in some order one after the other (see Figure 2 and 3).

Formally, for each sentence s in the corpus and each annotation a, we have a set of typed edits $edits(s,a) = \{e_{s,a}^{(1)}, \ldots, e_{s,a}^{(n_{s,a})}\}$ of size $n_{s,a}$. We call $2^{edits(s,a)}$ the *corrections lattice*, and denote it with $E_{s,a}$. We call, s, the correction corresponding to $\emptyset$ the *original*. We define a partial order relation between $x, y \in E_{s,a}$ such that $x < y$ if $x \subset y$. This order relation is assumed to be the gold standard ranking between the corrections.

For our experiments, we use the NUCLE test data (Ng et al., 2014). Each sentence is paired with two annotations. The other eight available

[2]The difference between our results and previously reported ones is probably due to a recent update in GLEU to better tackles multiple references (Napoles et al., 2016a).

[3]The standard deviation of the results is about 0.02.

Social media make our **pace of life** so fast
and leave us less time to think about our life.

↑ ~~life patten~~ pace of life

Social media **make** our life patten so fast
and leave us less time to think about our life.

↑ ~~makes~~ make

Social media makes our life patten so fast
and **leave** us less time to think about our life.

↑ ~~left~~ leave

Social media makes our life patten so fast
and left us less time to think about our life.

Figure 3: An example chain from a corrections lattice – each sentence is the result of applying a single edit to the sentence below it. The top sentence is a perfect correction, while the bottom is the original.

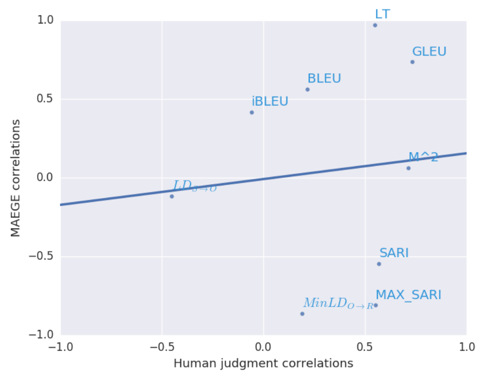


Figure 4: A scatter plot of the corpus-level correlation of metrics according to the different methodologies. The x-axis corresponds to the correlation according to human rankings (*Combined* setting), and the y-axis corresponds to the correlation according to MAEGE. While some get similar correlation (e.g., GLEU), other metrics change drastically (e.g., SARI).

references, produced by Bryant and Ng (2015), are used as references for the reference-based metrics. Denote the set of references for s with R_s.

Sentences which require no correction according to at least one of the two annotations are discarded. In 26 cases where two edit spans intersect in the same annotation (out of a total of about 40K edits), the edits are manually merged or split.

5 Corpus-level Analysis

We conduct a corpus-level analysis, namely testing the ability of metrics to determine which corpus of corrections is of better quality. In practice, this procedure is used to rank systems based on their outputs on the test corpus.

In order to compile corpora corresponding to systems of different quality levels, we define several corpus models, each applying a different expected number of edits to the original. Models are denoted with the expected number of edits they apply to the original which is a positive number $M \in \mathbb{R}^+$. Given a corpus model M, we generate a corpus of corrections by traversing the original sentences, and for each sentence s uniformly sample an annotation a (i.e., a set of edits that results in a perfect correction), and the number of edits applied n_{edits}, which is sampled from a clipped binomial probability with mean M and variance 0.9. Given n_{edits}, we uniformly sample from the lattice $E_{s,a}$ a sub-set of edits of size n_{edits}, and apply this set of edits to s. The corpus of $M = 0$ is the set of originals.

The corpus of source sentences, against which all other corpora are compared, is sampled by traversing the original sentences, and for each sentence s, uniformly sample an annotation a, and given s, a, uniformly sample a sentence from $E_{s,a}$.

Given a metric $m \in$ METRICS, we compute its score for each sampled corpus. Where corpus-level scores are not defined by the metrics themselves, we use the average sentence score instead. We compare the rankings induced by the scores of m and the ranking of systems according to their corpus model (i.e., systems that have a higher M should be ranked higher), and report the correlation between these rankings.

5.1 Experiments

Setup. For each model, we sample one correction per NUCLE sentence, noting that it is possible to reduce the variance of the metrics' corpus-level scores by sampling more. Corpus models of integer values between 0 and 10 are taken. We report Spearman ρ, commonly used for system-level rankings (Bojar et al., 2017).[4]

Results. Results, presented in Table 2 (left part), shows that LT correlates best with the rankings induced by MAEGE, where GLEU is second. M^2's correlation is only 0.06. We note that the LT requires a complementary metric to penalize grammatical outputs that diverge in meaning from the source (Napoles et al., 2016b). See §8.

Comparing the metrics' quality in corpus-level evaluation with their quality according to CHR (§3), we find they are often at odds. Figure 4 plots the Spearman correlation of the different metrics according to the two validation methodologies,

[4]Using Pearson correlation shows similar trends.

	Corpus-level		Sentence-level			
	ρ	P-val	r	P-val	τ	P-val
iBLEU	0.418	0.200	**0.230**	†	0.050	†
M^2	0.060	0.853	-0.025	0.024	0.213	†
LT	**0.973**	†	0.167	†	**0.222**	†
BLEU	0.564	0.071	0.214	†	0.111	†
$MinLD_{O\to R}$	-0.867	†	0.011	0.327	-0.183	†
GLEU	0.736	0.001	0.189	†	-0.028	†
MAX-SARI	-0.809	0.003	0.027	0.015	-0.070	†
SARI	-0.545	0.080	0.061	†	-0.039	†
$LD_{S\to O}$	-0.118	0.729	0.109	†	0.094	†

Table 2: Corpus-level Spearman ρ, sentence-level Pearson r and Kendall τ with the metrics (left). † represents P-value < 0.001. LT correlates best at the corpus level and has the highest sentence-level τ, while iBLEU has the highest sentence-level r.

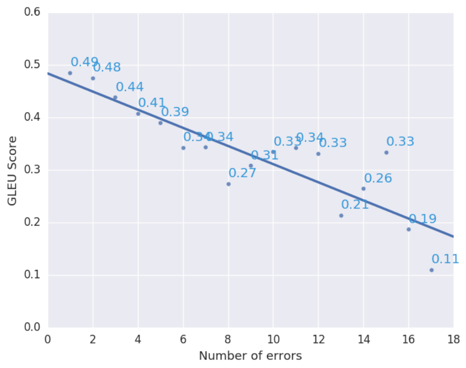


Figure 5: Average GLEU score of originals (y-axis), plotted against the number of errors they contain (x-axis). Their substantial correlation indicates that GLEU is globally reliable.

showing correlations are slightly correlated, but disagreements as to metric quality are frequent and substantial (e.g., with iBLEU or SARI).

6 Sentence-level Analysis

We proceed by presenting a method for assessing the correlation between metric-induced scores of corrections of the same sentence, and the scores given to these corrections by MAEGE. Given a sentence s and an annotation a, we sample a random permutation over the edits in $edits(s,a)$. We denote the permutation with $\sigma \in S_{n_{s,a}}$, where $S_{n_{s,a}}$ is the permutation group over $\{1,\cdots,n_{s,a}\}$. Given σ, we define a monotonic chain in $E_{i,j}$ as:

$$chain(s,a,\sigma) = \Big(\emptyset < \{e_{s,a}^{(\sigma(1))}\} < \{e_{s,a}^{(\sigma(1))}, e_{s,a}^{(\sigma(2))}\} < \dots < edits(s,a)\Big)$$

For each chain, we uniformly sample one of its elements, mark it as the source, and denote it with src. In order to generate a set of chains, MAEGE traverses the original sentences and annotations, and for each sentence-annotation pair, uniformly samples n_{ch} chains without repetition. It then uniformly samples a source sentence from each chain. If the number of chains in $E_{s,a}$ is smaller than n_{ch}, MAEGE selects all the chains.

Given a metric $m \in$ METRICS, we compute its score for every correction in each sampled chain against the sampled source and available references. We compute the sentence-level correlation of the rankings induced by the scores of m and the rankings induced by $<$. For computing rank correlation (such as Spearman ρ or Kendall τ), such a relative ranking is sufficient.

We report Kendall τ, which is only sensitive to the relative ranking of correction pairs within the same chain. Kendall is minimalistic in its assumptions, as it does not require numerical scores, but only assuming that $<$ is well-motivated, i.e., that applying a set of valid edits is better in quality than applying only a subset of it.

As $<$ is a partial order, and as Kendall τ is standardly defined over total orders, some modification is required. τ is a function of the number of compared pairs and of discongruent pairs (ordered differently in the compared rankings):

$$\tau = 1 - \frac{2\,|\text{discongruent pairs}|}{|\text{all pairs}|}.$$

To compute these quantities, we extract all unique pairs of corrections that can be compared with $<$ (i.e., one applies a sub-set of the edits of the other), and count the number of discongruent ones between the metric's ranking and $<$. Significance is modified accordingly.[5] Spearman ρ is

[5]Code can be found in https://github.com/borgr/EoE

less applicable in this setting, as it compares total orders whereas here we compare partial orders.

To compute linear correlation with Pearson r, we make the simplifying assumption that all edits contribute equally to the overall quality. Specifically, we assume that a perfect correction (i.e., the top of a chain) receives a score of 1. Each original sentence s (the bottom of a chain), for which there exists annotations $a_1, \ldots, a_n$, receives a score of

$$1 - \min_i \frac{|edits(s, a_i)|}{|tokens(s)|}.$$

The scores of partial (non-perfect) corrections in each chain are linearly spaced between the score of the perfect correction and that of the original. This scoring system is well-defined, as a partial correction receives the same score according to all chains it is in, as all paths between a partial correction and the original have the same length.

6.1 Experiments

Setup. We experiment with $n_{ch} = 1$, yielding 7936 sentences in 1312 chains (same as the number of original sentences in the NUCLE test set). We report the Pearson correlation over the scores of all sentences in all chains (r), and Kendall τ over all pairs of corrections within the same chain.

Results. Results are presented in Table 2 (right part). No metric scores very high, neither according to Pearson r nor according to Kendall τ. iBLEU correlates best with $<$ according to r, obtaining a correlation of 0.23, whereas LT fares best according to τ, obtaining 0.222.

Results show a discrepancy between the low corpus-level and sentence-level r correlations of M^2 and its high sentence-level τ. It seems that although M^2 orders pairs of corrections well, its scores are not a linear function of MAEGE's scores. This may be due to M^2's assignment of the minimal possible score to the source, regardless of its quality. M^2 thus seems to predict well the relative quality of corrections of the same sentence, but to be less effective in yielding a globally coherent score (cf. Felice and Briscoe (2015)).

GLEU shows the inverse behaviour, failing to correctly order pairs of corrections of the same sentence, while managing to produce globally coherent scores. We test this hypothesis by computing the average difference in GLEU score between all pairs in the sampled chains, and find it to be slightly negative (-0.00025), which is in line with GLEU's small negative τ. On the other hand, plotting the GLEU scores of the originals grouped by the number of errors they contain, we find they correlate well (Figure 5), indicating that GLEU performs well in comparing the quality of corrections of different sentences. Four sentences with considerably more errors than the others were considered outliers and removed.

7 Metric Sensitivity by Error Type

MAEGE's lattice can be used to analyze how the examined metrics reward corrections of errors of different types. For each edit type t, we denote with S_t the set of correction pairs from the lattice that only differ in an edit of type t. For each such pair (c, c') and for each metric m, we compute the difference in the score assigned by m to c and c'. The average difference is denoted with $\Delta_{m,t}$.

$$\Delta_{m,t} = \frac{1}{|S_t|} \sum_{(c,c') \in S_t} \left[m(src, c, R) - m(src, c', R) \right]$$

R is the corresponding reference set. A negative (positive) $\Delta_{m,t}$ indicates that m penalizes (awards) valid corrections of type t.

7.1 Experiments

Setup. We sample chains using the same sampling method as in §6, and uniformly sample a source from each chain. For each edit type t, we detect all pairs of corrections in the sampled chains that only differ in an edit of type t, and use them to compute $\Delta_{m,t}$. We use the set of 27 edit types given in the NUCLE corpus.

Results. Table 3 presents the results, showing that under all metrics, some edits types are penalized and others rewarded. iBLEU and LT penalize the least edit types, and GLEU penalizes the most, providing another perspective on GLEU's negative Kendall τ (§6). Certain types are penalized by almost all metrics. One such type is Vm, wrong verb modality (e.g., "as they [$\emptyset \rightsquigarrow$ may] not want to know"). Another such type is Npos, a problem in noun possessive (e.g., "their [facebook's $\rightsquigarrow$ Facebook] page"). Other types, such as Mec, mechanical (e.g., "[real-life $\rightsquigarrow$ real life]"), and V0, missing verb (e.g., "'Privacy', this is the word that [$\emptyset \rightsquigarrow$ is] popular"), are often rewarded by the metrics.

In general, the tendency of reference-based metrics (the vast majority of GEC metrics) to penalize edits of various types suggests that many edit

Type	iBLEU	M^2	LT	BLEU	$MinLD_{O\to R}$	GLEU	MAX-SARI	SARI	$LD_{S\to O}$
WOinc	0.016	−0.000	−0.002	−0.005	−0.026	−0.051	−0.075	−0.046	0.063
Nn	0.033	−0.001	0.004	0.029	−0.007	0.025	0.043	0.037	0.017
Npos	−0.001	0.001	−0.004	−0.011	−0.007	−0.030	−0.023	−0.009	0.014
Sfrag	−0.025	−0.003	−0.000	−0.067	−0.068	−0.143	−0.177	−0.142	0.076
Wtone	−0.013	−0.002	−0.008	−0.024	−0.021	−0.026	−0.086	−0.055	0.018
Srun	−0.027	−0.004	−0.004	−0.048	−0.014	−0.078	−0.039	−0.030	0.020
ArtOrDet	0.028	−0.001	0.001	0.019	−0.006	−0.003	−0.022	−0.003	0.024
Vt	0.054	−0.001	0.005	0.046	−0.002	0.011	0.003	0.018	0.025
Wa	0.041	−0.002	−0.002	−0.013	0.006	−0.028	−0.073	−0.090	0.071
Wform	0.049	−0.001	0.002	0.044	−0.003	0.010	0.004	0.020	0.022
WOadv	0.007	0.000	0.009	0.011	0.012	0.006	0.088	0.054	−0.014
V0	0.015	−0.001	0.019	0.005	−0.003	−0.006	−0.010	−0.004	0.015
Trans	−0.011	0.000	0.005	−0.022	−0.029	−0.031	−0.019	−0.004	0.013
Pform	0.021	−0.001	0.003	0.011	−0.012	−0.019	−0.003	0.005	0.030
Smod	−0.052	0.001	0.004	−0.093	−0.072	−0.126	−0.062	−0.043	0.055
Ssub	−0.005	0.000	−0.011	−0.024	−0.027	−0.052	−0.072	−0.038	0.026
Wci	−0.008	−0.001	0.004	−0.022	−0.029	−0.045	−0.049	−0.032	0.017
Vm	−0.007	−0.001	−0.001	−0.029	−0.027	−0.075	−0.070	−0.059	0.030
Pref	−0.003	−0.001	0.002	−0.015	−0.022	−0.045	−0.048	−0.035	0.018
Mec	0.012	0.001	0.014	0.004	−0.013	−0.014	0.000	0.002	0.018
Vform	0.043	−0.001	0.006	0.044	0.000	0.030	0.033	0.043	0.013
Prep	0.018	−0.000	0.004	0.011	−0.008	−0.001	−0.010	0.005	0.014
Um	−0.038	−0.001	−0.007	−0.043	−0.100	−0.037	−0.046	−0.032	0.009
Others	−0.048	−0.000	0.007	−0.063	−0.054	−0.060	−0.040	−0.024	−0.000
Rloc-	0.004	−0.001	−0.004	−0.006	−0.027	−0.023	−0.028	−0.019	0.022
Spar	0.041	0.001	0.003	0.035	−0.012	−0.003	0.008	0.026	0.024
SVA	0.045	−0.001	−0.001	0.037	−0.005	−0.002	0.012	0.015	0.021

Table 3: Average change in metric score by metric and edit types ($\Delta_{m,t}$; see text). Rows correspond to edit types (abbreviations in Dahlmeier et al. (2013)); columns correspond to metrics. Some edit types are consistently penalized.

types are under-represented in available reference sets. Automatic evaluation of systems that perform these edit types may, therefore, be unreliable. Moreover, not addressing these biases in the metrics may hinder progress in GEC. Indeed, M^2 and GLEU, two of the most commonly used metrics, only award a small sub-set of edit types, thus offering no incentive for systems to improve performance on such types.[6]

8 Discussion

We revisit the argument that using system outputs to perform metric validation poses a methodological difficulty. Indeed, as GEC systems are developed, trained and tested using available metrics, and as metrics tend to reward some correction types and penalize others (§7), it is possible that GEC development adjusts to the metrics, and neglects some error types. Resulting tendencies in GEC systems would then yield biased sets of outputs for human rankings, which in turn would result in biases in the validation process.

To make this concrete, GEC systems are often precision-oriented: trained to prefer not to correct than to invalidly correct. Indeed, Choshen and Abend (2018a) show that modern systems tend to be highly conservative, often performing an order of magnitude fewer changes to the source than references do. Validating metrics on their ability to rank conservative system outputs (as is de facto the common practice) may produce a different picture of metric quality than when considering a more inclusive set of corrections.

We use MAEGE to mimic a setting of ranking against precision-oriented outputs. To do so, we perform corpus-level and sentence-level analyses, but instead of randomly sampling a source, we invariably take the original sentence as the source. We thereby create a setting where all edits applied are valid (but not all valid edits are applied).

Comparing the results to the regular MAEGE correlation (Table 4), we find that LT remains reliable, while M^2, that assumes the source receives the worst possible score, gains from this unbalanced setting. iBLEU drops, suggesting it may need to be retuned to this setting and give less weight to $BLEU(O, S)$, thus becoming more like BLEU and GLEU. The most drastic change we see is in SARI and MAX-SARI, which flip their sign and present strong performance. Interestingly, the metrics that benefit from this precision-oriented setting in the corpus-level are the same metrics that perform better according to CHR than to MAEGE (Figure 4). This indicates the different trends produced by MAEGE and CHR, may result

[6] $LD_{S\to O}$ tends to award valid corrections of almost all types. As source sentences are randomized across chains, this indicates that on average, corrections with more applied edits tend to be more similar to comparable corrections on the lattice. This is also reflected by the slightly positive sentence-level correlation of $LD_{S\to O}$ (§6).

	Corpus-level		Sentence-level			
	ρ	P-val	r	P-val	τ	P-val
iBLEU	-0.872 (0.418)	†	0.235 (0.230)	†	0.053 (0.050)	†
M^2	0.882 (0.060)	†	-0.014 (-0.025)	0.223	0.223 (0.213)	†
LT	0.836 (0.973)	0.001	0.175 (0.167)	0.019	0.184 (0.222)	†
BLEU	0.845 (0.564)	0.001	0.217 (0.214)	†	0.115 (0.111)	†
$MinLD_{O \to R}$	-0.909 (-0.867)	†	0.022 (0.011)	†	-0.180 (-0.183)	†
GLEU	0.945 (0.736)	†	0.208 (0.189)	†	0.003 (-0.028)	†
MAX-SARI	0.772 (-0.809)	0.005	0.053 (0.027)	†	0.004 (-0.070)	0.6
SARI	0.800 (-0.545)	0.003	0.084 (0.061)	†	0.022 (-0.039)	0.001
$LD_{S \to O}$	-0.972 (-0.118)	†	0.025 (0.109)	0.027	0.070 (0.094)	†

Table 4: Corpus-level Spearman ρ, sentence-level Pearson r and Kendall τ correlations using origin as the source with the various metrics (left). Correlations using a random source are found in parenthesis. † represents $P - value < 0.001$. LT is the best corpus correlated, and has the best τ while iBLEU has the best r

from the latter's use of precision-oriented outputs.

Drawbacks. Like any methodology MAEGE has its simplifying assumptions and drawbacks; we wish to make them explicit. First, any biases introduced in the generation of the test corpus are inherited by MAEGE (e.g., that edits are contiguous and independent of each other). Second, MAEGE does not include errors that a human will not perform but machines might, e.g., significantly altering the meaning of the source. This partially explains why LT, which measures grammaticality but not meaning preservation, excels in our experiments. Third, MAEGE's scoring system (§6) assumes that all errors damage the score equally. While this assumption is made by GEC metrics, we believe it should be refined in future work by collecting user information.

9 Conclusion

In this paper, we show how to leverage existing annotation in GEC for performing validation reliably. We propose a new automatic methodology, MAEGE, which overcomes many of the shortcomings of the existing methodology. Experiments with MAEGE reveal a different picture of metric quality than previously reported. Our analysis suggests that differences in observed metric quality are partly due to system outputs sharing consistent tendencies, notably their tendency to under-predict corrections. As existing methodology ranks system outputs, these shared tendencies bias the validation process. The difficulties in basing validation on system outputs may be applicable to other text-to-text generation tasks, a question we will explore in future work.

Acknowledgments

This work was supported by the Israel Science Foundation (grant No. 929/17), and by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. We thank Joel Tetreault and Courtney Napoles for helpful feedback and inspiring conversations.

References

Hiroki Asano, Tomoya Mizumoto, and Kentaro Inui. 2017. Reference-based metrics can be replaced with reference-less metrics in evaluating grammatical error correction systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 343–348.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. Hume: Human ucca-based evaluation of machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1264–1274.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Ondřej Bojar, Miloš Ercegovčević, Martin Popel, and Omar F Zaidan. 2011. A grain of salt for the wmt manual evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 1–11. Association for Computational Linguistics.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. 2016. Results of the wmt16 metrics shared task. In *Proceedings of the First Conference on Machine Translation*, pages 199–231, Berlin, Germany. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *ACL (1)*, pages 697–707.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367.

Leshem Choshen and Omri Abend. 2018a. Inherent biases in reference-based evaluation for grammatical error correction and text simplification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Leshem Choshen and Omri Abend. 2018b. Reference-less measure of faithfulness for grammatical error correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62. Association for Computational Linguistics.

Mark Dras. 2015. Evaluating human pairwise preference judgments. *Computational Linguistics*, 41(2):337–345.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *HLT-NAACL*, pages 578–587.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in esl sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan. The COLING 2016 Organizing Committee.

Yvette Graham, Timothy Baldwin, Aaron Harwood, Alistair Moffat, and Justin Zobel. 2012. Measurement of progress in machine translation. In *Proceedings of the Australasian Language Technology Association Workshop 2012*, pages 70–78.

Yvette Graham, Timothy Baldwin, and Nitika Mathur. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1183–1191.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, Edward Gillian, et al. 2015. Human evaluation of grammatical error correction systems. In *EMNLP*, pages 461–470.

Philipp Koehn. 2012. Simulating human judgment in machine translation evaluation campaigns. In *International Workshop on Spoken Language Translation (IWSLT) 2012*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 1(123):32–73.

Joseph B Kruskal and David Sankoff. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley.

Adam Lopez. 2012. Putting human assessments of machine translation systems in order. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 1–9. Association for Computational Linguistics.

Marcin Miłkowski. 2010. Developing an open-source, rule-based proofreading tool. *Software: Practice and Experience*, 40(7):543–566.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016a. GLEU without tuning. *eprint arXiv:1605.02592 [cs.CL]*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016b. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2109–2115. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *CoNLL Shared Task*, pages 1–14.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357*.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual smt system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 38–42. Association for Computational Linguistics.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing

Rotem Dror **Gili Baumer** **Segev Shlomov** **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
{rtmdrr@campus|sgbaumer@campus|segevs@campus|roiri}.technion.ac.il

Abstract

Statistical significance testing is a standard statistical tool designed to ensure that experimental results are not coincidental. In this opinion/theoretical paper we discuss the role of statistical significance testing in Natural Language Processing (NLP) research. We establish the fundamental concepts of significance testing and discuss the specific aspects of NLP tasks, experimental setups and evaluation measures that affect the choice of significance tests in NLP research. Based on this discussion, we propose a simple practical protocol for statistical significance test selection in NLP setups and accompany this protocol with a brief survey of the most relevant tests. We then survey recent empirical papers published in ACL and TACL during 2017 and show that while our community assigns great value to experimental results, statistical significance testing is often ignored or misused. We conclude with a brief discussion of open issues that should be properly addressed so that this important tool can be applied in NLP research in a statistically sound manner[1].

1 Introduction

The field of Natural Language Processing (NLP) has recently made great progress due to the data revolution that has made abundant amounts of textual data from a variety of languages and linguistic domains (newspapers, scientific journals, social media etc.) available. This, together with the emergence of a new generation of computing resources and the related development of Deep Neural Network models, have resulted in dramatic improvements in the capabilities of NLP algorithms.

[1]The code for all statistical tests detailed in this paper is found on: https://github.com/rtmdrr/testSignificanceNLP.git

The extended reach of NLP algorithms has also resulted in NLP papers giving much more emphasis to the experiment and result sections by showing comparisons between multiple algorithms on various datasets from different languages and domains. This emphasis on empirical results highlights the role of statistical significance testing in NLP research: if we rely on empirical evaluation to validate our hypotheses and reveal the correct language processing mechanisms, we better be sure that our results are not coincidental.

This paper aims to discuss the various aspects of proper statistical significance testing in NLP and to provide a simple and sound guide to the way this important tool should be used. We also discuss the particular challenges of statistical significance in the context of language processing tasks.

To facilitate a clear and coherent presentation, our (somewhat simplified) model of an NLP paper is one that presents a new algorithm and makes the hypothesis that this algorithm is better than a previous strong algorithm, which serves as the baseline. This hypothesis is verified in experiments where the two algorithms are applied to the same datasets (test sets), reasoning that if one algorithm is consistently better than the other, hopefully with a sufficiently large margin, then it should also be better on future, currently unknown, datasets. Yet, the experimental differences might be coincidental. Here comes statistical significance testing into the picture: we have to make sure that the probability of falsely concluding that one algorithm is better than the other is very small.

We note that in this paper we do not deal with the problem of drawing valid conclusions from multiple comparisons between algorithms across a large number of datasets , a.k.a. replicability analysis (see (Dror et al., 2017)). Instead, our focus is on a single comparison: how can we make sure that the difference between the two algorithms, as observed in an individual comparison, is not coincidental. Statistical significance testing of each individual comparison is the basic building block of replicability analysis – its accurate performance is a pre-condition for any multiple dataset analysis.

Statistical significance testing (§ 2) is a well researched problem in the statistical literature. However, the unique structured nature of natural language data is reflected in specialized evaluation measures such as BLEU (machine translation, (Papineni et al., 2002)), ROUGE (extractive summarization, (Lin, 2004)), UAS and LAS (dependency parsing, (Kübler et al., 2009)). The distribution of these measures is of great importance to statistical significance testing. Moreover, certain properties of NLP datasets and the community's evaluation standards also affect the way significance testing should be performed. An NLP-specific discussion of significance testing is hence in need.

In § 3 we discuss the considerations to be made in order to select the proper statistical significance test in NLP setups. We propose a simple decision tree algorithm for this purpose, and survey the prominent significance tests – parametric and non-parametric – for NLP tasks and data.

In § 4 we survey the current evaluation and significance testing practices of the community. We provide statistics collected from the long papers of the latest ACL proceedings (Barzilay and Kan, 2017) as well as from the papers published in the TACL journal during 2017. Our analysis reveals that there is still a room for improvement in the way statistical significance is used in papers published in our top-tier publication venues. Particularly, a large portion of the surveyed papers do not test the significance of their results, or use incorrect tests for this purpose.

Finally, in § 5 we discuss open issues. A particularly challenging problem is that while most significance tests assume the test set consists of independent observations, most NLP datasets consist of dependent data points. For example, many NLP standard evaluation sets consist of sentences coming from the same source (e.g. newspaper) or document (e.g. newspaper article) or written by the same author. Unfortunately, the nature of these dependencies is hard to characterize, let alone to quantify. Another important problem is how to test significance when cross-validation, a popular evaluation methodology in NLP papers, is performed.

Besides its practical value, we hope this paper will encourage further research into the role of statistical significance testing in NLP and on the questions that still remain open.

2 Preliminaries

In this section we provide the required preliminaries for our discussion. We start with a formal definition of statistical significance testing and proceed with an overview of the prominent evaluation measures in NLP.

2.1 Statistical Significance Testing

In this paper we focus on the setup where the performance of two algorithms, A and B, on a dataset X, is compared using an evaluation measure $\mathcal{M}$. Let us denote $\mathcal{M}(ALG, X)$ as the value of the evaluation measure $\mathcal{M}$ when algorithm ALG is applied to the dataset X. Without loss of generality, we assume that higher values of the measure are better. We define the difference in performance between the two algorithms according to the measure $\mathcal{M}$ on the dataset X as:

$$\delta(X) = \mathcal{M}(A, X) - \mathcal{M}(B, X). \quad (1)$$

In this paper we will refer to $\delta(X)$ as our test statistic. Using this notation we formulate the following statistical hypothesis testing problem:[2]

$$\begin{aligned} H_0 &: \delta(X) \leq 0 \\ H_1 &: \delta(X) > 0. \end{aligned}$$

In order to decide whether or not to reject the null hypothesis, that is reaching the conclusion that $\delta(X)$ is indeed greater than 0, we usually compute a $p-$value for the test. The $p-$value is defined as the probability, under the null hypothesis H_0, of obtaining a result equal to or more extreme than what was actually observed. For the one-sided hypothesis testing defined here, the $p-$value is defined as:

$$Pr(\delta(X) \geq \delta_{observed} | H_0).$$

Where $\delta_{observed}$ is the performance difference between the algorithms (according to $\mathcal{M}$) when applied to X. The smaller the p-value, the higher the significance, or, in other words, the stronger

[2]For simplicity we consider a one-sided hypothesis, it can be easily re-formulated as a double-sided hypothesis.

the indication provided by the data that the null-hypothesis, H_0, does not hold. In order to decide whether H_0 should be rejected, the researcher should pre-define an arbitrary, fixed threshold value α, a.k.a *the significance level*. Only if $p-$value $< \alpha$ then the null hypothesis is rejected.

In significance (or hypothesis) testing we consider two error types. *Type* I *error* refers to the case where the null hypothesis is rejected when it is actually true. *Type* II *error* refers to the case where the null hypothesis is not rejected although it should be. A common approach in hypothesis testing is to choose a test that guarantees that the probability of making a type I error is upper bounded by the test significance level α, mentioned above, while achieving the highest possible *power*: i.e. the lowest possible probability of making a type II error.

2.2 Evaluation Measures in NLP

Evaluation Measure	ACL 17	TACL 17
F-scores	78 (39.8%)	9 (25.71%)
Accuracy	67 (34.18%)	13 (37.14%)
Precision/ Recall	44 (22.45%)	6 (17.14%)
BLEU	26 (13.27%)	4 (11.43%)
ROUGE	12 (6.12%)	0 (0%)
Pearson/ Spearman correlations	4 (2.04%)	6 (17.14%)
Perplexity	7 (3.57%)	2 (5.71%)
METEOR	6 (3.06%)	1 (2.86%)
UAS+LAS	1 (0.51%)	3 (8.57%)

Table 1: The most common evaluation measures in (long) ACL and TACL 2017 papers, ordered by ACL frequency. For each measure we present the total number of papers where it is used and the fraction of papers in the corresponding venue.

In order to draw valid conclusions from the experiments formulated in § 2.1 it is crucial to apply the correct statistical significance test. In § 3 we explain that the choice of the significance test is based, among other considerations, on the distribution of the test statistics, $\delta(X)$. From equation 1 it is clear that $\delta(X)$ depends on the evaluation measure $\mathcal{M}$. We hence turn to discuss the evaluation measures employed in NLP.

In § 4 we analyze the (long) ACL and TACL 2017 papers, and observe that the most commonly used evaluation measures are the 12 measures that appear in Table 1. Notice that seven of these measures: Accuracy, Precision, Recall, F-score, Pearson and Spearman correlations and Perplexity, are not specific to NLP. The other five measures: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), UAS and LAS (Kübler et al., 2009), are unique measures that were developed for NLP applications. BLEU and METEOR are standard evaluation measures for machine translation, ROUGE for extractive summarization, and UAS and LAS for dependency parsing. While UAS and LAS are in fact accuracy measures, BLEU, ROUGE and METEOR are designed for tasks where there are several possible outputs - a characteristic property of several NLP tasks. In machine translation, for example, a sentence in one language can be translated in multiple ways to another language. Consequently, BLEU takes an n-gram based approach on the surface forms, while METEOR considers only unigram matches but uses stemming and controls for synonyms.

All 12 measures return a real number, either in $[0, 1]$ or in $\mathbb{R}$. Notice though that accuracy may reflect an average over a set of categorical scores (observations), e.g., in document-level binary sentiment analysis where every document is tagged as either positive or negative. In other cases, the individual observations are also continuous. For example, when comparing two dependency parsers, we may want to understand how likely it is, given our results, that one parser will do better than the other on a new sentence. In such a case we will consider the sentence-level UAS or LAS differences between the two parsers on all the sentences in the test set. Such sentence level UAS or LAS scores - the individual observations to be considered in the significance test - are real-valued.

With the basic concepts clarified, we are ready to discuss the considerations to be made when choosing a statistical significance test.

3 Statistical Significance in NLP

The goal of this section is to detail the considerations involved in the selection of a statistical significance test for an NLP application. Based on these considerations we provide a practical recipe that can be applied in order to make a good choice. In order to make this paper a practical guide for the community, we also provide a short description of the significance tests that are most relevant for NLP setups.

3.1 Parametric vs. Non-parametric Tests

As noted above, a major consideration in the selection of a statistical significance test is the distribution of the test statistic, $\delta(X)$, under the null hypothesis. If the distribution is known, then the suitable test will come from the family of *parametric tests*, that uses this distribution in order to achieve powerful results (i.e., low probability of making a type II error, see § 2). If the distribution is unknown then any assumption made by a test may lead to erroneous conclusions and hence we should rely on *non-parametric tests* that do not make any such assumption. While non-parametric tests may be less powerful than their parametric counterparts, they do not make unjustified assumptions and are hence statistically sound even when the test statistic distribution is unknown.

But how can one know the test statistic distribution? One possibility is to apply tests designed to evaluate the distribution of a sample of observations. For example, the Shapiro-Wilk test (Shapiro and Wilk, 1965) tests the null hypothesis that a sample comes from a normally distributed population, the Kolmogorov-Smirnov test quantifies the distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, and the Anderson-Darling test (Anderson and Darling, 1954) tests whether a given sample of data is drawn from a given probability distribution. As discussed below, there seems to be other heuristics that are used in practice but are not often mentioned in research papers.

In what follows we discuss the prominent parametric and non-parametric tests for NLP setups. Based on this discussion we end this section with a simple decision tree that aims to properly guide the significance test choice process.

3.2 Prominent Significance Tests

3.2.1 Parametric Tests

Parametric significance tests assume that the test statistic is distributed according to a known distribution with defined parameters, typically the normal distribution. While this assumption may be hard to verify (see discussion above), when it holds, these parametric tests have stronger statistical power compared to non-parametric tests that do not make this assumption (Fisher, 1937).

Here we discuss the prominent parametric test for NLP setups - the paired student's t-test.

Paired Student's t-test This test assesses whether the population means of two sets of measurements differ from each other, and is based on the assumption that both samples come from a normal distribution (Fisher, 1937).

In practice, t-test is often applied with evaluation measures such as accuracy, UAS and LAS, that compute the mean number of correct predictions per input example. When comparing two dependency parsers, for example, we can apply the test to check if the averaged difference of their UAS scores is significantly larger than zero, which can serve as an indication that one parser is better than the other.

Although we have not seen this discussed in NLP papers, we believe that the decision to use the t-test with these measures is based on the Central Limit Theorem (CLT). CLT establishes that, in most situations, when independent random variables are added, their properly normalized sum tends toward a normal distribution even if the original variables themselves are not normally distributed. That is, accuracy measures in structured tasks tend to be normally distributed when the number individual predictions (e.g. number of words in a sentence when considering sentence-level UAS) is large enough.

One case where it is theoretically justified to employ the t-test is described in (Sethuraman, 1963). The authors prove that for large enough data, the sampling distribution of a certain function of the Pearson's correlation coefficient follows the Student's t-distribution with $n - 2$ degrees of freedom. With the recent surge in word similarity research with word embedding models, this result is of importance to our community.

For other evaluation measures, such as F-score, BLEU, METEOR and ROUGE that do not compute means, the common practice is to assume that they are not normally distributed (Yeh, 2000; Berg-Kirkpatrick et al., 2012). We believe this issue requires a further investigation and suggest that it may be best to rely on the normality tests discussed in § 3.1 when deciding whether or not to employ the t-test.

3.2.2 Non-parametric Tests

When the test statistic distribution is unknown, non-parametric significance testing should be used. The non-parametric tests that are commonly used in NLP setups can be divided into two families that differ with respect to their statistical power and computational complexity.

The first family consists of tests that do not consider the actual values of the evaluation measures. The second family do consider the values of the measures: it tests repeatedly sample from the test data, and estimates the p-value based on the test statistic values in the samples. We refer to the first family as the family of *sampling-free* tests and to the second as the family of *sampling-based* tests.

The two families of tests reflect different preferences with respect to the balance between statistical power and computational efficiency. Sampling-free tests do not consider the evaluation measure values, only higher level statistics of the results such as the number of cases in which each of the algorithms performs better than the other. Consequently, their statistical power is lower than that of sampling-based tests that do consider the evaluation measure values. Sampling-based tests, however, compensate for the lack of distributional assumptions over the data with re-sampling – a computationally intensive procedure. Sampling-based methods are hence not the optimal candidates for very large datasets.

We consider here four commonly used sampling-free tests: the sign test and two of its variants, and the wilcoxon signed-rank test.

Sign test This test tests whether matched pair samples are drawn from distributions with equal medians. The test statistic is the number of examples for which algorithm A is better than algorithm B, and the null hypothesis states that given a new pair of measurements (e.g. evaluations (a_i, b_i) of the two algorithms on a new test example), then a_i and b_i are equally likely to be larger than the other (Gibbons and Chakraborti, 2011).

The sign test has limited practical implications since it only checks if algorithm A is better than B and ignores the extent of the difference. Yet, it has been used in a variety of NLP papers (e.g. (Collins et al., 2005; Chan et al., 2007; Rush et al., 2012)). The assumptions of this test is that the data samples are i.i.d, the differences come from a continuous distribution (not necessarily normal) and that the values are ordered.

The next test is a special case of the sign test for binary classification (or a two-tailed sign test).

McNemar's test (McNemar, 1947) This test is designed for paired nominal observations (binary labels). The test is applied to a 2×2 contingency table, which tabulates the outcomes of two algorithms on a sample of n examples. The null hypothesis for this test states that the marginal probability for each outcome (label one or label two) is the same for both algorithms. That is, when applying both algorithms on the same data we would expect them to be correct/incorrect on the same proportion of items. Under the null hypothesis, with a sufficiently large number of disagreements between the algorithms, the test statistic has a distribution of χ^2 with one degree of freedom. This test is appropriate for binary classification tasks, and has been indeed used in such NLP works (e.g. sentiment classificaiton, (Blitzer et al., 2006; Ziser and Reichart, 2017)). The **Cochran's Q test** (Cochran, 1950) generalizes the McNemar's test for multi-class classification setups.

The sign test and its variants consider only pairwise ranks: which algorithm performs better on each test example. In NLP setups, however, we also have access to the evaluation measure values, and this allows us to rank the differences between the algorithms. The Wilcoxon signed-rank test makes use of such a rank and hence, while it does not consider the evaluation measure values, it is more powerful than the sign test and its variants.

Wilcoxon signed-rank test (Wilcoxon, 1945) Like the sign test variants, this test is used when comparing two matched samples (e.g. UAS values of two dependency parsers on a set of sentences). Its null hypothesis is that the differences follow a symmetric distribution around zero. First, the absolute values of the differences are ranked. Then, each rank gets a sign according to the sign of the difference. The Wilcoxon test statistic sums these signed ranks. The test is actually applicable for most NLP setups and it has been used widely (e.g. (Søgaard et al., 2014; Søgaard, 2013; Yang and Mitchell, 2017)) due to its improved power compared to the sign test variants.

As noted above, sampling-free tests trade statistical power for efficiency. Sampling-based methods take the opposite approach. This family includes two main methods: permutation/randomization tests (Noreen, 1989) and the paired bootstrap (Efron and Tibshirani, 1994).

Pitman's permutation test This test estimates the test statistic distribution under the null hypothesis by calculating the values of this statistic under all possible labellings (permutations) of the test set. The (two-sided) p-value of the test is calculated as the proportion of these permutations where the absolute difference was greater than or equal to the absolute value of the difference in the output of the algorithm.

Obviously, permutation tests are computationally intensive due to the exponentially large number of possible permutations. In practice, approximate randomization tests are used where a pre-defined limited number of permutations are drawn from the space of all possible permutations, without replacements (see, e.g. (Riezler and Maxwell, 2005) in the context of machine translation). The bootstrap test (Efron and Tibshirani, 1994) is based on a closely related idea.

Paired bootstrap test This test is very similar to approximate randomization of the permutation test, with the difference that the sampling is done with replacements (i.e., an example from the original test data can appear more than once in a sample). The idea of bootstrap is to use the samples as surrogate populations, for the purpose of approximating the sampling distribution of the statistic. The p-value is calculated in a similar manner to the permutation test.

Bootstrap was used with a variety of NLP tasks, including machine translation, text summarization and semantic parsing (e.g. (Koehn, 2004; Li et al., 2017; Wu et al., 2017; Ouchi et al., 2017)). The test is less effective for small test sets, as it assumes that the test set distribution does not deviate too much from the population distribution.

Clearly, Sampling-based methods are computationally intensive and can be intractable for large datasets, even with modern computing power. In such cases, sampling-free methods form an available alternative.

3.3 Significance Test Selection

With the discussion of significance test families - parametric vs. non-parametric (§ 3.1), and the properties of the actual significance tests (§ 3.2) we are now ready to provide a simple recipe for significance test selection in NLP setups. The decision tree in Figure 1 provides an illustration.

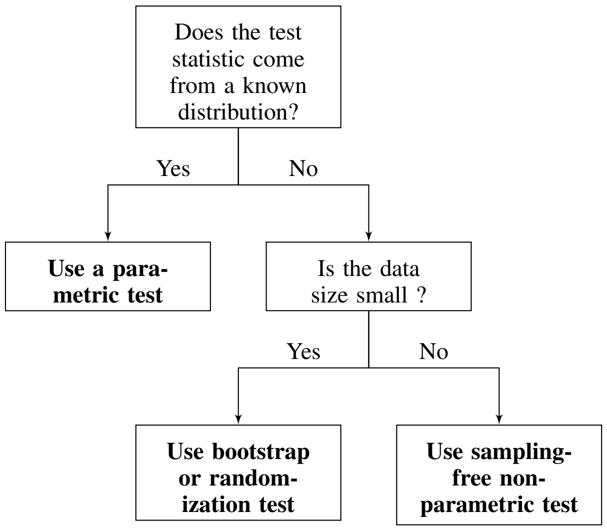

Does the test statistic come from a known distribution?
Yes
No
Use a parametric test
Is the data size small ?
Yes
No
Use bootstrap or randomization test
Use sampling-free non-parametric test

Figure 1: Decision tree for statistical significance test selection.

If the distribution of the test statistic is known, then parametric tests are most appropriate. These tests are more statistically powerful and less computationally intensive compared to their non-parametric counterparts. The stronger statistical power of parametric tests stems from the stronger, parametric assumptions they make, while the higher computational demand of some non-parametric tests is the result of their sampling process.

When the distribution of the test statistic is unknown, the first non-parametric family of choice is that of sampling-based tests. These tests consider the actual values of the evaluation measures and are not restricted to higher order properties (e.g. ranks) of the observed values – their statistical power is hence higher. As noted in (Riezler and Maxwell, 2005), in the case where the distributional assumptions of the parametric tests are violated, sampling-based tests have more statistical power than parametric tests.

Nonetheless, sampling-based tests are computationally intensive – the exact permutation test, for example, requires the generation of all 2^n data permutations (where n is the number of points in the dataset). To overcome this, approximate randomization can be used, as was done, e.g., by Yeh (2000) for test sets of more than 20 points. The other alternative for very large datasets are sampling-free tests that are less powerful but are computationally feasible.

In what follows we check whether recent ACL and TACL papers follow these guidelines.

4 Survey of ACL and TACL papers

General Statistics	ACL '17	TACL '17
Total number of papers	196	37
# relevant (experimental) papers	180	33
# different tasks	36	15
# different evaluation measures	24	19
Average number of measures per paper	2.34	2.1
# papers that **do not** report significance	117	15
# papers that report significance	63	18
# papers that report significance but use the **wrong** statistical test	6	0
# papers that report significance but do not mention the test name	21	3
# papers that have to report replicability	110	19
# papers that report replicability	3	4
# papers that perform cross validation	23	5

Table 2: Statistical significance statistics for empirical ACL and TACL 2017 papers.

We analyzed the long papers from the proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL17, (Barzilay and Kan, 2017)), a total of 196 papers, and the papers from the Transactions of the Association of Computational Linguistics journal (TACL17), Volume 5, Issue 1, a total of 37 papers. We have focused on empirical papers where at least one comparison between methods was performed.

Table 2 presents the main results from our survey. The top part of the table presents general statistics of our dataset. In both conference and journal papers, the variety of different NLP tasks is quite large: 36 tasks in ACL 2017 and 15 tasks in TACL. Interestingly, in almost every paper in our survey the researchers chose to analyze their results using more than one evaluation measure, with an average of 2.34 (ACL) and 2.1 (TACL). Table 1 presents the most common of these evaluation measures.

Statistical Test	ACL '17	TACL '17
Bootstrap	6	1
t-test	17	2
Wilcoxon	3	0
Chi square	3	1
Randomization	3	1
McNemar	2	3
Sign	2	3
Permutation	1	4

Table 3: Number of times each of the prominent statistical significance tests in ACL and TACL 2017 papers was used. 42 ACL and 15 TACL papers reported the significance test name. 5 ACL papers mentioned an unrecognized test name.

The lower part of Table 2 depicts the disturbing reality of statistical significance testing in our research community. Out of the 180 experimental long papers of ACL 2017, only 63 papers included a statistical significance test. Moreover, out of these 63 papers 21 did not mention the name of the significance test they employed. Of the 42 papers that did mention the name of the significance test, 6 used the wrong test according to the considerations discussed in § 3.[3] In TACL, where the review process is presumably more strict and of higher quality, out of 33 experimental papers, 15 did not include statistical significance testing, and all the papers that report significance and mentioned the name of the test used a valid test.

While this paper focuses on the correct choice of a significance test, we also checked whether the papers in our sample account for the effect of multiple hypothesis testing when testing statistical significance (see (Dror et al., 2017)). When testing multiple hypotheses, as in the case of comparing the participating algorithms across a large number of datasets, the probability of making one or more false claims may be very high, even if the probability of drawing an erroneous conclusion in each individual comparison is small. In ACL 2017, out

[3]We considered the significance test to be inappropriate in three cases: 1. Using the t-test when the evaluation measure is not an average measure; 2. Using the t-test for a classification task (i.e. when the observations are categorical rather then continuous), even if the evaluation measure is an average measure; and 3. Using a Boostrap test with a small test set size.

of 110 papers that used multiple datasets only 3 corrected for multiplicity (all using the Bonferroni correction). In TACL, the situation is slightly better with 4 papers correcting for multiplicity out of 19 that should have done that.

Regarding the statistical tests that were used in the papers that did report significance (Table 3), in ACL 2017 most of the papers used the Student's t-test that assumes the data is i.i.d and that the test statistics are normally distributed. As discussed in § 3 this is not the case in many NLP applications. Gladly, in TACL, t-test is not as prominent.

One final note is about the misuse of the word **significant**. We noticed that in a considerable number of papers this word was used as a synonym for words such as important, considerable, meaningful, substantial, major, notable etc. We believe that we should be more careful when using this word, ideally keeping its statistical sense and using other, more general words to indicate a substantial impact.

We close this discussion with two important open issues.

5 Open Questions

In this section we would like to point on two issues that remain open even after our investigation. We hope that bringing these issues to the attention of the research community will encourage our fellow researchers to come up with appropriate solutions.

The first open issue is that of *dependent observations*. An assumption shared by the statistical significance tests described in § 3, that are commonly used in NLP setups, is that the data samples are independent and identically distributed. This assumption, however, is rarely true in NLP setups.

For example, the popular WSJ Penn Treebank corpus (Marcus et al., 1993) consists of 2,499 articles from a three year Wall Street Journal (WSJ) collection of 98,732 stories. Obviously, some of the sentences included in the corpus come from the same article, were written by the same author or were reviewed before publication by the same editor. As another example, many sentences in the Europarl parallel corpus (Koehn, 2005) that is very popular in the machine translation literature are taken from the same parliament discussion. An independence assumption between the sentences in these corpora is not likely to hold.

This dependence between test examples violates the conditions under which the theoretical guarantees of the various tests were developed. The impact of this phenomenon on our results is hard to quantify, partly because it is hard to quantify the nature of the dependence between test set examples in NLP datasets. Some papers are even talking about abandoning the null hypothesis statistical significance test approach due to this hard-to-meet assumption (Koplenig, 2017; McShane et al., 2017; Carver, 1978; Leek et al., 2017). In our opinion, this calls for a future collaboration with statisticians in order to better understand the extent to which existing popular significance tests are relevant for NLP, and to develop alternative tests if necessary.

Another issue that deserves some thought is that of cross-validation. To increase the validity of reported results, it is customary in NLP papers to create a number of random splits of the experimental corpus into train, development and test portions (see Table 2). For each such split (fold), the tested algorithms are trained and tuned on the training and development datasets, respectively, and their results on the test data are recorded. The final reported result is typically the average of the test set results across the splits. Some papers also report the fraction of the folds for which one algorithm was better than the others. While cross-validation is surely a desired practice, it is challenging to report statistical significance when it is employed. Particularly, the test sets of the different folds are obviously not independent – their content is even likely to overlap.

One solution we would like to propose here is based on replicability analysis (Dror et al., 2017). This paper proposes a statistical significance framework for multiple comparisons performed with dependent test sets, using the $K_{Bonferroni}$ estimator for the number of datasets with significant effect. One statistically sound way to test for significance when a cross-validation protocol is employed is hence to calculate the p-value for each fold separately, and then to perform replicability analysis for dependent datasets with $K_{Bonferroni}$. Only if this analysis rejects the null hypothesis in all folds (or in more than a predefined threshold number of folds), the results should be declared significant. Here again, further statistical investigation may lead to additional, potentially better, solutions.

6 Conclusions

We discussed the use of significance testing in NLP. We provided the main considerations for significance test selection, and proposed a simple test selection protocol. We then surveyed the state of significance testing in recent top venue papers and concluded with open issues. We hope this paper will serve as a guide for NLP researchers and, not less importantly, that it will encourage discussions and collaborations that will contribute to the soundness and correctness of our research.

References

Theodore W Anderson and Donald A Darling. 1954. A test of goodness of fit. *Journal of the American statistical association* 49(268):765–769.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Regina Barzilay and Min-Yen Kan. 2017. Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers). In *Proceedings of ACL*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of EMNLP-CoNLL*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*.

Ronald Carver. 1978. The case against statistical significance testing. *Harvard Educational Review* 48(3):378–399.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of ACL*.

William G Cochran. 1950. The comparison of percentages in matched samples. *Biometrika* 37(3/4):256–266.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*.

Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. Replicability analysis for natural language processing: Testing significance with multiple datasets. *Transactions of the Association for Computational Linguistics* 5:471–486.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Ronald Aylmer Fisher. 1937. *The design of experiments*. Oliver And Boyd; Edinburgh; London.

Jean Dickinson Gibbons and Subhabrata Chakraborti. 2011. Nonparametric statistical inference. In *International encyclopedia of statistical science*, Springer, pages 977–979.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT summit*.

Alexander Koplenig. 2017. Against statistical significance testing in corpus linguistics. *Corpus Linguistics and Linguistic Theory* .

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies* 1(1):1–127.

Jeff Leek, Blakeley B McShane, Andrew Gelman, David Colquhoun, Michèle B Nuijten, and Steven N Goodman. 2017. Five ways to fix statistics. *Nature* 551(7682):557–559.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12(2):153–157.

Blakeley B McShane, David Gal, Andrew Gelman, Christian Robert, and Jennifer L Tackett. 2017. Abandon statistical significance. *arXiv preprint arXiv:1709.07588* .

Eric W Noreen. 1989. *Computer intensive methods for hypothesis testing: An introduction*. Wiley, New York.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Neural modeling of multi-predicate interactions for japanese predicate argument structure analysis. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Stefan Riezler and John T Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Alexander Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and pos tagging using inter-sentence consistency constraints. In *Proceedings of EMNLP-CoNLL*.

J Sethuraman. 1963. *The Advanced Theory of Statistics, Volume 2: Inference and Relationship*. JSTOR.

Samuel Sanford Shapiro and Martin B Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52(3/4):591–611.

Anders Søgaard. 2013. Estimating effect size across datasets. In *Proceedings of NAACL-HLT*.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. 2014. What's in a p-value in nlp? In *Proceedings of CoNLL*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1(6):80–83.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of ACL*.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of ACL*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING*.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proceedings of CoNLL 2017*.

Distilling Knowledge for Search-based Structured Prediction

Yijia Liu, Wanxiang Che[*], Huaipeng Zhao, Bing Qin, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
`{yjliu,car,hpzhao,qinb,tliu}@ir.hit.edu.cn`

Abstract

Many natural language processing tasks can be modeled into structured prediction and solved as a search problem. In this paper, we distill an ensemble of multiple models trained with different initialization into a single model. In addition to learning to match the ensemble's probability output on the reference states, we also use the ensemble to explore the search space and learn from the encountered states in the exploration. Experimental results on two typical search-based structured prediction tasks – transition-based dependency parsing and neural machine translation show that distillation can effectively improve the single model's performance and the final model achieves improvements of 1.32 in LAS and 2.65 in BLEU score on these two tasks respectively over strong baselines and it outperforms the greedy structured prediction models in previous literatures.

1 Introduction

Search-based structured prediction models the generation of natural language structure (part-of-speech tags, syntax tree, translations, semantic graphs, etc.) as a search problem (Collins and Roark, 2004; Liang et al., 2006; Zhang and Clark, 2008; Huang et al., 2012; Sutskever et al., 2014; Goodman et al., 2016). It has drawn a lot of research attention in recent years thanks to its competitive performance on both accuracy and running time. A stochastic policy that controls the whole search process is usually learned by imitating a *reference policy*. The imitation is usually addressed as training a classifier to predict the reference policy's search action on the encountered states when performing the reference policy. Such imitation process can sometimes be problematic. One problem is the ambiguities of the reference policy, in which multiple actions lead to the optimal structure but usually, only one is chosen as training instance (Goldberg and Nivre, 2012). Another problem is the discrepancy between training and testing, in which during the test phase, the learned policy enters non-optimal states whose search action is never learned (Ross and Bagnell, 2010; Ross et al., 2011). All these problems harm the generalization ability of search-based structured prediction and lead to poor performance.

Previous works tackle these problems from two directions. To overcome the ambiguities in data, techniques like *ensemble* are often adopted (Di-

* Email corresponding.

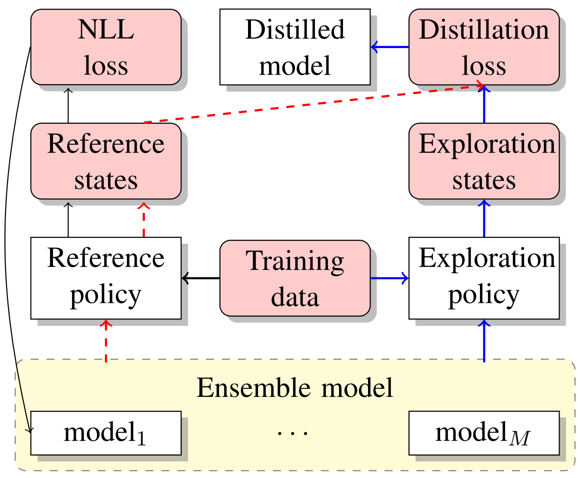


Figure 1: Workflow of our knowledge distillation for search-based structured prediction. The yellow bracket represents the ensemble of multiple models trained with different initialization. The dashed red line shows our *distillation from reference* (§3.2). The solid blue line shows our *distillation from exploration* (§3.3).

	Dependency parsing	Neural machine translation
s_t	(σ, β, A), where σ is a stack, β is a buffer, and A is the partially generated tree	$(\$, y_1, y_2, ..., y_t)$, where \$ is the start symbol.
$\mathcal{A}$	{SHIFT, LEFT, RIGHT}	pick one word w from the target side vocabulary $\mathcal{W}$.
$\mathcal{S}_0$	$\{([\,], [1, .., n], \emptyset)\}$	$\{(\$)\}$
$\mathcal{S}_T$	$\{([\text{ROOT}], [\,], A)\}$	$\{(\$, y_1, y_2, ..., y_m)\}$
$\mathcal{T}(s, a)$	• SHIFT: $(\sigma, j\|\beta) \rightarrow (\sigma\|j, \beta)$ • LEFT: $(\sigma\|i\ j, \beta) \rightarrow (\sigma\|j, \beta) \quad A \leftarrow A \cup \{i \leftarrow j\}$ • RIGHT: $(\sigma\|i\ j, \beta) \rightarrow (\sigma\|i, \beta) \quad A \leftarrow A \cup \{i \rightarrow j\}$	$(\$, y_1, y_2, ..., y_t) \rightarrow (\$, y_1, y_2, ..., y_t, y_{t+1} = w)$

Table 1: The search-based structured prediction view of transition-based dependency parsing (Nivre, 2008) and neural machine translation (Sutskever et al., 2014).

etterich, 2000). To mitigate the discrepancy, exploration is encouraged during the training process (Ross and Bagnell, 2010; Ross et al., 2011; Goldberg and Nivre, 2012; Bengio et al., 2015; Goodman et al., 2016). In this paper, we propose to consider these two problems in an integrated *knowledge distillation* manner (Hinton et al., 2015). We distill a single model from the ensemble of several baselines trained with different initialization by matching the ensemble's output distribution on the reference states. We also let the ensemble randomly explore the search space and learn the single model to mimic ensemble's distribution on the encountered exploration states. Combing the distillation from reference and exploration further improves our single model's performance. The workflow of our method is shown in Figure 1.

We conduct experiments on two typical search-based structured prediction tasks: transition-based dependency parsing and neural machine translation. The results of both these two experiments show the effectiveness of our knowledge distillation method by outperforming strong baselines. In the parsing experiments, an improvement of 1.32 in LAS is achieved and in the machine translation experiments, such improvement is 2.65 in BLEU. Our model also outperforms the greedy models in previous works.

Major contributions of this paper include:

- We study the knowledge distillation in search-based structured prediction and propose to distill the knowledge of an ensemble into a single model by learning to match its distribution on both the reference states (§3.2) and exploration states encountered when using the ensemble to explore the search space (§3.3). A further combination of these two methods is also proposed to improve the performance (§3.4).
- We conduct experiments on two search-based structured prediction problems: transition-based dependency parsing and neural machine translation. In both these two problems, the distilled model significantly improves over strong baselines and outperforms other greedy structured prediction (§4.2). Comprehensive analysis empirically shows the feasibility of our distillation method (§4.3).

2 Background

2.1 Search-based Structured Prediction

Structured prediction maps an input $\mathbf{x} = (x_1, x_2, ..., x_n)$ to its structural output $\mathbf{y} = (y_1, y_2, ..., y_m)$, where each component of $\mathbf{y}$ has some internal dependencies. Search-based structured prediction (Collins and Roark, 2004; Daumé III et al., 2005; Daumé III et al., 2009; Ross and Bagnell, 2010; Ross et al., 2011; Doppa et al., 2014; Vlachos and Clark, 2014; Chang et al., 2015) models the generation of the structure as a search problem and it can be formalized as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}(s, a), \mathcal{S}_0, \mathcal{S}_T)$, in which $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T}$ is a function that maps $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, $\mathcal{S}_0$ is a set of initial states, and $\mathcal{S}_T$ is a set of terminal states. Starting from an initial state $s_0 \in \mathcal{S}_0$, the structured prediction model repeatably chooses an action $a_t \in \mathcal{A}$ by following a *policy* $\pi(s)$ and applies a_t to s_t and enter a new state s_{t+1} as $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$, until a final state $s_T \in \mathcal{S}_T$ is achieved. Several natural language structured prediction problems can be modeled under the search-based framework including dependency parsing (Nivre, 2008) and neural machine translation (Liang et al., 2006; Sutskever et al., 2014). Table 1 shows the search-based structured prediction view of these two problems.

In the data-driven settings, $\pi(s)$ controls the whole search process and is usually parameterized by a classifier $p(a \mid s)$ which outputs the proba-

Algorithm 1: Generic learning algorithm for search-based structured prediction.

```
Input: training data: ${\mathbf{x}^{(n)}, \mathbf{y}^{(n)}}_{n=1}^{N}$; the
       reference policy: $\pi_{\mathcal{R}}(s, \mathbf{y})$.
Output: classifier $p(a|s)$.
1  $D \leftarrow \emptyset$;
2  for $n \leftarrow 1...N$ do
3      $t \leftarrow 0$;
4      $s_t \leftarrow s_0(\mathbf{x}^{(n)})$;
5      while $s_t \notin \mathcal{S}_T$ do
6          $a_t \leftarrow \pi_{\mathcal{R}}(s_t, \mathbf{y}^{(n)})$;
7          $D \leftarrow D \cup \{s_t\}$;
8          $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$;
9          $t \leftarrow t + 1$;
10     end
11 end
12 optimize $\mathcal{L}_{NLL}$;
```

bility of choosing an action a on the given state s. The commonly adopted greedy policy can be formalized as choosing the most probable action with $\pi(s) = \text{argmax}_a\, p(a \mid s)$ at test stage. To learn an optimal classifier, search-based structured prediction requires constructing a reference policy $\pi_{\mathcal{R}}(s, \mathbf{y})$, which takes an input state s, gold structure $\mathbf{y}$ and outputs its reference action a, and training $p(a \mid s)$ to imitate the reference policy. Algorithm 1 shows the common practices in training $p(a \mid s)$, which involves: first, using $\pi_{\mathcal{R}}(s, \mathbf{y})$ to generate a sequence of reference states and actions on the training data (line 1 to line 11 in Algorithm 1); second, using the states and actions on the reference sequences as examples to train $p(a \mid s)$ with negative log-likelihood (NLL) loss (line 12 in Algorithm 1),

$$\mathcal{L}_{NLL} = \sum_{s \in D} \sum_{a} -\mathbb{1}\{a = \pi_{\mathcal{R}}\} \cdot \log p(a \mid s)$$

where D is a set of training data.

The reference policy is sometimes sub-optimal and ambiguous which means on one state, there can be more than one action that leads to the optimal prediction. In transition-based dependency parsing, Goldberg and Nivre (2012) showed that one dependency tree can be reached by several search sequences using Nivre (2008)'s *arc-standard* algorithm. In machine translation, the ambiguity problem also exists because one source language sentence usually has multiple semantically correct translations but only one reference translation is presented. Similar problems have also been observed in semantic parsing (Goodman et al., 2016). According to Frénay and Verleysen (2014), the widely used NLL loss is vulnerable to ambiguous data which make it worse for search-based structured prediction.

Besides the ambiguity problem, training and testing discrepancy is another problem that lags the search-based structured prediction performance. Since the training process imitates the reference policy, all the states in the training data are optimal which means they are guaranteed to reach the optimal structure. But during the test phase, the model can predict non-optimal states whose search action is never learned. The greedy search which is prone to error propagation also worsens this problem.

2.2 Knowledge Distillation

A cumbersome model, which could be an ensemble of several models or a single model with larger number of parameters, usually provides better generalization ability. *Knowledge distillation* (Buciluǎ et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015) is a class of methods for transferring the generalization ability of the cumbersome *teacher model* into a small *student model*. Instead of optimizing NLL loss, knowledge distillation uses the distribution $q(y \mid x)$ outputted by the teacher model as "soft target" and optimizes the knowledge distillation loss,

$$\mathcal{L}_{KD} = \sum_{x \in D} \sum_{y} -q(y \mid x) \cdot \log p(y \mid x).$$

In search-based structured prediction scenario, x corresponds to the state s and y corresponds to the action a. Through optimizing the distillation loss, knowledge of the teacher model is learned by the student model $p(y \mid x)$. When correct label is presented, NLL loss can be combined with the distillation loss via simple interpolation as

$$\mathcal{L} = \alpha \mathcal{L}_{KD} + (1 - \alpha)\mathcal{L}_{NLL} \qquad (1)$$

3 Knowledge Distillation for Search-based Structured Prediction

3.1 Ensemble

As Hinton et al. (2015) pointed out, although the real objective of a machine learning algorithm is to generalize well to new data, models are usually trained to optimize the performance on training data, which bias the model to the training data.

In search-based structured prediction, such biases can result from either the ambiguities in the training data or the discrepancy between training and testing. It would be more problematic to train $p(a \mid s)$ using the loss which is in-robust to ambiguities and only considering the optimal states.

The effect of ensemble on ambiguous data has been studied in Dietterich (2000). They empirically showed that ensemble can overcome the ambiguities in the training data. Daumé III et al. (2005) also use weighted ensemble of parameters from different iterations as their final structure prediction model. In this paper, we consider to use ensemble technique to improve the generalization ability of our search-based structured prediction model following these works. In practice, we train M search-based structured prediction models with different initialized weights and ensemble them by the average of their output distribution as $q(a \mid s) = \frac{1}{M}\sum_m q_m(a \mid s)$. In Section 4.3.1, we empirically show that the ensemble has the ability to choose a good search action in the optimal-yet-ambiguous states and the non-optimal states.

3.2 Distillation from Reference

As we can see in Section 4, ensemble indeed improves the performance of baseline models. However, real world deployment is usually constrained by computation and memory resources. Ensemble requires running the structured prediction models for multiple times, and that makes it less applicable in real-world problem. To take the advantage of the ensemble model while avoid running the models multiple times, we use the knowledge distillation technique to distill a single model from the ensemble. We started from changing the NLL learning objective in Algorithm 1 into the distillation loss (Equation 1) as shown in Algorithm 2. Since such method learns the model on the states produced by the reference policy, we name it as *distillation from reference*. Blocks connected by in dashed red lines in Figure 1 show the workflow of our *distillation from reference*.

3.3 Distillation from Exploration

In the scenario of search-based structured prediction, transferring the teacher model's generalization ability into a student model not only includes matching the teacher model's soft targets on the reference search sequence, but also imitating the search decisions made by the teacher model. One way to accomplish the imitation can be sampling search sequence from the ensemble and learn from the soft target on the sampled states. More concretely, we change $\pi_{\mathcal{R}}(s, \mathbf{y})$ into a policy $\pi_{\mathcal{E}}(s)$ which samples an action a from $q(a \mid s)^{\frac{1}{T}}$, where T is the temperature that controls the sharpness of the distribution (Hinton et al., 2015). The algorithm is shown in Algorithm 2. Since such distillation generate training instances from exploration, we name it as *distillation from exploration*. Blocks connected by in solid blue lines in Figure 1 show the workflow of our *distillation from exploration*.

On the sampled states, reference decision from $\pi_{\mathcal{R}}$ is usually non-trivial to achieve, which makes learning from NLL loss infeasible. In Section 4, we empirically show that fully distilling from the soft target, i.e. setting $\alpha = 1$ in Equation 1, achieves comparable performance with that both from distillation and NLL.

Algorithm 2: Knowledge distillation for search-based structured prediction.

```
Input: training data: $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$; the
       reference policy: $\pi_{\mathcal{R}}(s, \mathbf{y})$; the
       exploration policy: $\pi_{\mathcal{E}}(s)$ which
       samples an action from the annealed
       ensemble $q(a \mid s)^{\frac{1}{T}}$
Output: classifier $p(a \mid s)$.
1  $D \leftarrow \emptyset$;
2  for $n \leftarrow 1...N$ do
3      $t \leftarrow 0$;
4      $s_t \leftarrow s_0(\mathbf{x}^{(n)})$;
5      while $s_t \notin \mathcal{S}_T$ do
6          if distilling from reference then
7              $a_t \leftarrow \pi_{\mathcal{R}}(s_t, \mathbf{y}^{(n)})$;
8          else
9              $a_t \leftarrow \pi_{\mathcal{E}}(s_t)$;
10         end
11         $D \leftarrow D \cup \{s_t\}$;
12         $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$;
13         $t \leftarrow t + 1$;
14     end
15 end
16 if distilling from reference then
17     optimize $\alpha\mathcal{L}_{KD} + (1 - \alpha)\mathcal{L}_{NLL}$;
18 else
19     optimize $\mathcal{L}_{KD}$;
20 end
```

3.4 Distillation from Both

Distillation from reference can encourage the model to predict the action made by the reference policy and distillation from exploration learns the model on arbitrary states. They transfer the generalization ability of the ensemble from different aspects. Hopefully combining them can further improve the performance. In this paper, we combine distillation from reference and exploration with the following manner: we use $\pi_{\mathcal{R}}$ and $\pi_{\mathcal{E}}$ to generate a set of training states. Then, we learn $p(a \mid s)$ on the generated states. If one state was generated by the reference policy, we minimize the interpretation of distillation and NLL loss. Otherwise, we minimize the distillation loss only.

4 Experiments

We perform experiments on two tasks: transition-based dependency parsing and neural machine translation. Both these two tasks are converted to search-based structured prediction as Section 2.1.

For the transition-based parsing, we use the stack-lstm parsing model proposed by Dyer et al. (2015) to parameterize the classifier.[1] For the neural machine translation, we parameterize the classifier as an LSTM encoder-decoder model by following Luong et al. (2015).[2] We encourage the reader of this paper to refer corresponding papers for more details.

4.1 Settings

4.1.1 Transition-based Dependency Parsing

We perform experiments on Penn Treebank (PTB) dataset with standard data split (Section 2-21 for training, Section 22 for development, and Section 23 for testing). Stanford dependencies are converted from the original constituent trees using Stanford CoreNLP 3.3.0[3] by following Dyer et al. (2015). Automatic part-of-speech tags are assigned by 10-way jackknifing whose accuracy is 97.5%. Labeled attachment score (LAS) excluding punctuation are used in evaluation. For the other hyper-parameters, we use the same settings as Dyer et al. (2015). The best iteration and α is determined on the development set.

[1]The code for parsing experiments is available at: https://github.com/Oneplus/twpipe.

[2]We based our NMT experiments on OpenNMT (Klein et al., 2017). The code for NMT experiments is available at: https://github.com/Oneplus/OpenNMT-py.

[3]stanfordnlp.github.io/CoreNLP/history.html

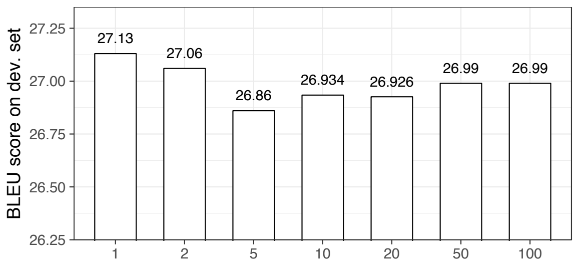


Figure 2: The effect of using different Ks when approximating distillation loss with K-most probable actions in the machine translation experiments.

Reimers and Gurevych (2017) and others have pointed out that neural network training is nondeterministic and depends on the seed for the random number generator. To control for this effect, they suggest to report the average of M differently-seeded runs. In all our dependency parsing, we set $n = 20$.

4.1.2 Neural Machine Translation

We conduct our experiments on a small machine translation dataset, which is the German-to-English portion of the IWSLT 2014 machine translation evaluation campaign. The dataset contains around 153K training sentence pairs, 7K development sentence pairs, and 7K testing sentence pairs. We use the same preprocessing as Ranzato et al. (2015), which leads to a German vocabulary of about 30K entries and an English vocabulary of 25K entries. One-layer LSTM for both encoder and decoder with 256 hidden units are used by following Wiseman and Rush (2016). BLEU (Papineni et al., 2002) was used to evaluate the translator's performance.[4] Like in the dependency parsing experiments, we run $M = 10$ differently-seeded runs and report the averaged score.

Optimizing the distillation loss in Equation 1 requires enumerating over the action space. It is expensive for machine translation since the size of the action space (vocabulary) is considerably large (25K in our experiments). In this paper, we use the K-most probable actions (translations on target side) on one state to approximate the whole probability distribution of $q(a \mid s)$ as $\sum_a q(a \mid s) \cdot \log p(a \mid s) \approx \sum_k^K q(\hat{a}_k \mid s) \cdot \log p(\hat{a}_k \mid s)$, where $\hat{a}_k$ is the k-th probable action. We fix α to

[4]We use multi-bleu.perl to evaluate our model's performance

	LAS
Baseline	90.83
Ensemble (20)	92.73
Distill (reference, α=1.0)	91.99
Distill (exploration, T=1.0)	92.00
Distill (both)	92.14
Ballesteros et al. (2016) (dyn. oracle)	91.42
Andor et al. (2016) (local, B=1)	91.02
Buckman et al. (2016) (local, B=8)	91.19
Andor et al. (2016) (local, B=32)	91.70
Andor et al. (2016) (global, B=32)	92.79
Dozat and Manning (2016)	94.08
Kuncoro et al. (2016)	92.06
Kuncoro et al. (2017)	94.60

Table 2: The dependency parsing results. Significance test (Nilsson and Nivre, 2008) shows the improvement of our *Distill (both)* over *Baseline* is statistically significant with $p < 0.01$.

1 and vary K and evaluate the distillation model's performance. These results are shown in Figure 2 where there is no significant difference between different Ks and in speed consideration, we set K to 1 in the following experiments.

4.2 Results

4.2.1 Transition-based Dependency Parsing

Table 2 shows our PTB experimental results. From this result, we can see that the ensemble model outperforms the baseline model by 1.90 in LAS. For our distillation from reference, when setting $\alpha = 1.0$, best performance on development set is achieved and the test LAS is 91.99.

We tune the temperature T during exploration and the results are shown in Figure 3. Sharpen the distribution during the sampling process generally performs better on development set. Our distillation from exploration model gets almost the same performance as that from reference, but simply combing these two sets of data outperform both models by achieving an LAS of 92.14.

We also compare our parser with the other parsers in Table 2. The second group shows the greedy transition-based parsers in previous literatures. Andor et al. (2016) presented an alternative state representation and explored both greedy and beam search decoding. (Ballesteros et al., 2016) explores training the greedy parser with dynamic oracle. Our distillation parser outperforms all these greedy counterparts. The third group shows parsers trained on different techniques including decoding with beam search (Buckman et al., 2016; Andor et al., 2016), training transition-based parser with beam search (Andor et al., 2016), graph-based parsing (Dozat and Manning, 2016), distilling a graph-based parser from the output of 20 parsers (Kuncoro et al., 2016), and converting constituent parsing results to dependencies (Kuncoro et al., 2017). Our distillation parser still outperforms its transition-based counterparts but lags the others. We attribute the gap between our parser with the other parsers to the difference in parsing algorithms.

	BLEU
Baseline	22.79
Ensemble (10)	26.26
Distill (reference, α=0.8)	24.76
Distill (exploration, T=0.1)	24.64
Distill (both)	25.44
MIXER	20.73
BSO (local, B=1)	22.53
BSO (global, B=1)	23.83

Table 3: The machine translation results. MIXER denotes that of Ranzato et al. (2015), BSO denotes that of Wiseman and Rush (2016). Significance test (Koehn, 2004) shows the improvement of our *Distill (both)* over *Baseline* is statistically significant with $p < 0.01$.

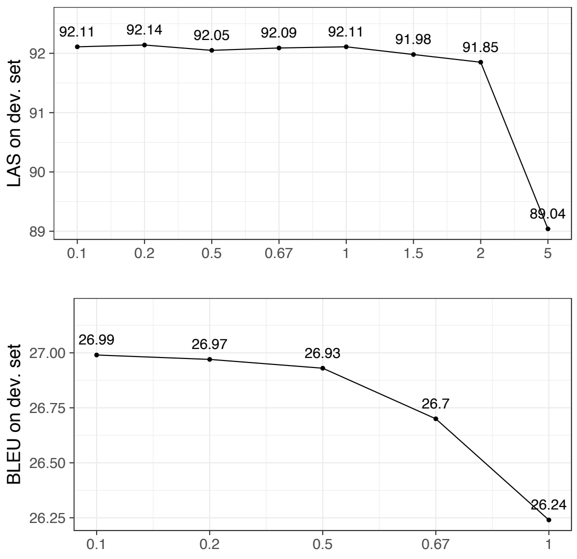


Figure 3: The effect of T on PTB (above) and IWSLT 2014 (below) development set.

4.2.2 Neural Machine Translation

Table 3 shows the experimental results on IWSLT 2014 dataset. Similar to the PTB parsing results, the ensemble 10 translators outperforms the baseline translator by 3.47 in BLEU score. Distilling from the ensemble by following the reference leads to a single translator of 24.76 BLEU score.

Like in the parsing experiments, sharpen the distribution when exploring the search space is more helpful to the model's performance but the differences when $T \leq 0.2$ is not significant as shown in Figure 3. We set $T = 0.1$ in our distillation from exploration experiments since it achieves the best development score. Table 3 shows the exploration result of a BLEU score of 24.64 and it slightly lags the best reference model. Distilling from both the reference and exploration improves the single model's performance by a large margin and achieves a BLEU score of 25.44.

We also compare our model with other translation models including the one trained with reinforcement learning (Ranzato et al., 2015) and that using beam search in training (Wiseman and Rush, 2016). Our distillation translator outperforms these models.

Both the parsing and machine translation experiments confirm that it's feasible to distill a reasonable search-based structured prediction model by just exploring the search space. Combining the reference and exploration further improves the model's performance and outperforms its greedy structured prediction counterparts.

4.3 Analysis

In Section 4.2, improvements from distilling the ensemble have been witnessed in both the transition-based dependency parsing and neural machine translation experiments. However, questions like "Why the ensemble works better? Is it feasible to fully learn from the distillation loss without NLL? Is learning from distillation loss stable?" are yet to be answered. In this section, we first study the ensemble's behavior on "problematic" states to show its generalization ability. Then, we empirically study the feasibility of fully learning from the distillation loss by studying the effect of α in the distillation from reference setting. Finally, we show that learning from distillation loss is less sensitive to initialization and achieves a more stable model.

	optimal-yet-ambiguous	non-optimal
Baseline	68.59	89.59
Ensemble	74.19	90.90
Distill (both)	81.15	91.38

Table 4: The ranking performance of parsers' output distributions evaluated in MAP on "problematic" states.

4.3.1 Ensemble on "Problematic" States

As mentioned in previous sections, "problematic" states which is either ambiguous or non-optimal harm structured prediciton's performance. Ensemble shows to improve the performance in Section 4.2, which indicates it does better on these states. To empirically testify this, we use dependency parsing as a testbed and study the ensemble's output distribution using the dynamic oracle.

The dynamic oracle (Goldberg and Nivre, 2012; Goldberg et al., 2014) can be used to efficiently determine, given any state s, which transition action leads to the best achievable parse from s; if some errors may have already made, what is the best the parser can do, going forward? This allows us to analyze the accuracy of each parser's individual decisions, in the "problematic" states. In this paper, we evaluate the output distributions of the baseline and ensemble parser against the *reference actions* suggested by the dynamic oracle. Since dynamic oracle yields more than one reference actions due to ambiguities and previous mistakes and the output distribution can be treated as their scoring, we evaluate them as a ranking problem. Intuitively, when multiple reference actions exist, a good parser should push probability mass to these actions. We draw problematic states by sampling from our baseline parser. The comparison in Table 4 shows that the ensemble model significantly outperforms the baseline on ambiguous and non-optimal states. This observation indicates the ensemble's output distribution is more "informative", thus generalizes well on problematic states and achieves better performance. We also observe that the distillation model perform better than both the baseline and ensemble. We attribute this to the fact that the distillation model is learned from exploration.

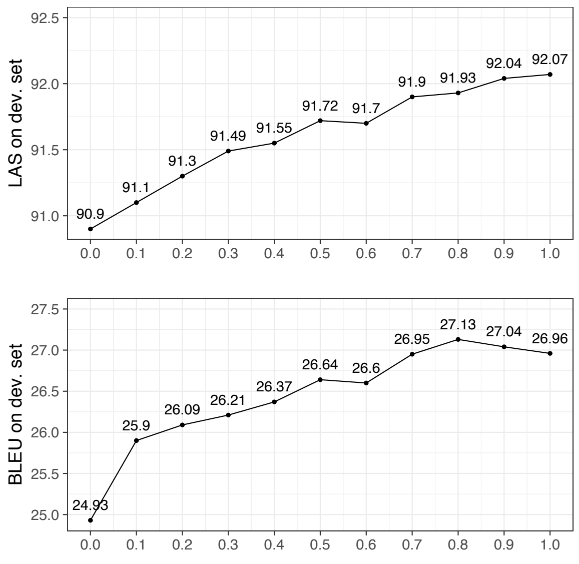


Figure 4: The effect of α on PTB (above) and IWSLT 2014 (below) development set.

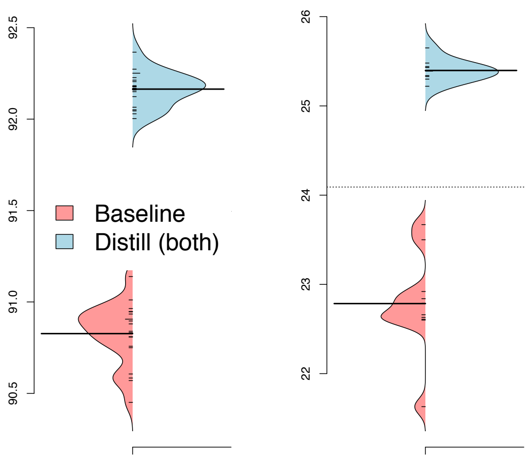


Figure 5: The distributions of scores for the baseline model and our *distillation from both* on PTB test (left) and IWSLT 2014 test (right) on differently-seeded runs.

system	seeds	min	max	σ
PTB test				
Baseline	20	90.45	91.14	0.17
Distill (both)	20	92.00	92.37	0.09
IWSLT 2014 test				
Baseline	10	21.63	23.67	0.55
Distill (both)	10	24.22	25.65	0.12

Table 5: The minimal, maximum, and standard derivation values on differently-seeded runs.

4.3.2 Effect of α

Over our distillation from reference model, we study the effect of α in Equation 1. We vary α from 0 to 1 by a step of 0.1 in both the transition-based dependency parsing and neural machine translation experiments and plot the model's performance on development sets in Figure 4. Similar trends are witnessed in both these two experiments that model that's configured with larger α generally performs better than that with smaller α. For the dependency parsing problem, the best development performance is achieved when we set $\alpha = 1$, and for the machine translation, the best α is 0.8. There is only 0.2 point of difference between the best α model and the one with α equals to 1. Such observation indicates that when distilling from the reference policy paying more attention to the distillation loss rather than the NLL is more beneficial. It also indicates that fully learning from the distillation loss outputted by the ensemble is reasonable because models configured with $\alpha = 1$ generally achieves good performance.

4.3.3 Learning Stability

Besides the improved performance, knowledge distillation also leads to more stable learning. The performance score distributions of differently-seed runs are depicted as violin plot in Figure 5. Table 5 also reveals the smaller standard derivations are achieved by our distillation methods. As Keskar et al. (2016) pointed out that the generalization gap is not due to *overfit*, but due to the network converge to *sharp minimizer* which generalizes worse, we attribute the more stable training from our distillation model as the distillation loss presents less *sharp minimizers*.

5 Related Work

Several works have been proposed to applying knowledge distillation to NLP problems. Kim and Rush (2016) presented a distillation model which focus on distilling the structured loss from a large model into a small one which works on sequence-level. In contrast to their work, we pay more attention to action-level distillation and propose to do better action-level distillation by both from reference and exploration.

Freitag et al. (2017) used an ensemble of 6-translators to generate training reference. Exploration was tried in their work with beam-search. We differ their work by training the single model to match the distribution of the ensemble.

Using ensemble in exploration was also studied in reinforcement learning community (Osband et al., 2016). In addition to distilling the ensemble on the labeled training data, a line of semi-supervised learning works show that it's effective to transfer knowledge of cumbersome model into a simple one on the unlabeled data (Liang et al., 2008; Li et al., 2014). Their extensions to knowledge distillation call for further study.

Kuncoro et al. (2016) proposed to compile the knowledge from an ensemble of 20 transition-based parsers into a voting and distill the knowledge by introducing the voting results as a regularizer in learning a graph-based parser. Different from their work, we directly do the distillation on the classifier of the transition-based parser.

Besides the attempts for directly using the knowledge distillation technique, Stahlberg and Byrne (2017) propose to first build the ensemble of several machine translators into one network by unfolding and then use SVD to shrink its parameters, which can be treated as another kind of knowledge distillation.

6 Conclusion

In this paper, we study knowledge distillation for search-based structured prediction and propose to distill an ensemble into a single model both from reference and exploration states. Experiments on transition-based dependency parsing and machine translation show that our distillation method significantly improves the single model's performance. Comparison analysis gives empirically guarantee for our distillation method.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772153.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *NIPS 27*, pages 2654–2662.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proc. of EMNLP*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS 28*, pages 1171–1179.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proc. of KDD*.

Jacob Buckman, Miguel Ballesteros, and Chris Dyer. 2016. Transition-based dependency parsing with heuristic backtracking. In *Proc. of EMNLP*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proc. of ICML*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*.

Hal Daumé III, John Langford, and Daniel Marcu. 2005. Search-based structured prediction as classification. In *NIPS Workshop on ASLTSP*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3).

Thomas G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Hc-search: A learning framework for search-based structured prediction. *J. Artif. Intell. Res. (JAIR)*, 50.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *CoRR*, abs/1702.01802.

Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25:845–869.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *TACL*, 2.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proc. of ACL*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proc. of NAACL*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. of ACL 2017, System Demonstrations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP 2004*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proc. of EMNLP*.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proc. of ACL*.

P. Liang, H. Daumé, and D. Klein. 2008. Structure compilation: trading structure for features. pages 592–599.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Jens Nilsson and Joakim Nivre. 2008. Malteval: an evaluation and visualization tool for dependency parsing. In *Proc. of LREC*. Http://www.lrec-conf.org/proceedings/lrec2008/.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped dqn. In *NIPS 29*, pages 4026–4034.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proc. of EMNLP*.

Stephane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proc. of AISTATS*, volume 9.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. of AISTATS*, volume 15.

Felix Stahlberg and Bill Byrne. 2017. Unfolding and shrinking neural machine translation ensembles. In *Proc. of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 27*, pages 3104–3112.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of EMNLP*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proc. of EMNLP*.

Stack-Pointer Networks for Dependency Parsing

Xuezhe Ma
Carnegie Mellon University
xuezhem@cs.cmu.edu

Zecong Hu[*]
Tsinghua University
huzecong@gmail.com

Jingzhou Liu
Carnegie Mellon University
liujingzhou@cs.cmu.edu

Nanyun Peng
University of Southern California
npeng@isi.edu

Graham Neubig and **Eduard Hovy**
Carnegie Mellon University
{gneubig, ehovy}@cs.cmu.edu

Abstract

We introduce a novel architecture for dependency parsing: *stack-pointer networks* (**STACKPTR**). Combining pointer networks (Vinyals et al., 2015) with an internal stack, the proposed model first reads and encodes the whole sentence, then builds the dependency tree top-down (from root-to-leaf) in a depth-first fashion. The stack tracks the status of the depth-first search and the pointer networks select one child for the word at the top of the stack at each step. The STACKPTR parser benefits from the information of the whole sentence and all previously derived subtree structures, and removes the left-to-right restriction in classical transition-based parsers. Yet, the number of steps for building any (including non-projective) parse tree is linear in the length of the sentence just as other transition-based parsers, yielding an efficient decoding algorithm with $O(n^2)$ time complexity. We evaluate our model on 29 treebanks spanning 20 languages and different dependency annotation schemas, and achieve state-of-the-art performance on 21 of them.

1 Introduction

Dependency parsing, which predicts the existence and type of linguistic dependency relations between words, is a first step towards deep language understanding. Its importance is widely recognized in the natural language processing (NLP) community, with it benefiting a wide range of NLP applications, such as coreference resolution (Ng, 2010; Durrett and Klein, 2013; Ma et al., 2016), sentiment analysis (Tai et al., 2015), machine translation (Bastings et al., 2017), information extraction (Nguyen et al., 2009; Angeli et al., 2015; Peng et al., 2017), word sense disambiguation (Fauceglia et al., 2015), and low-resource languages processing (McDonald et al., 2013; Ma and Xia, 2014). There are two dominant approaches to dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007): local and greedy *transition-based* algorithms (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Zhang and Nivre, 2011; Chen and Manning, 2014), and the globally optimized *graph-based* algorithms (Eisner, 1996; McDonald et al., 2005a,b; Koo and Collins, 2010).

Transition-based dependency parsers read words sequentially (commonly from left-to-right) and build dependency trees incrementally by making series of multiple choice decisions. The advantage of this formalism is that the number of operations required to build any projective parse tree is linear with respect to the length of the sentence. The challenge, however, is that the decision made at each step is based on local information, leading to error propagation and worse performance compared to graph-based parsers on root and long dependencies (McDonald and Nivre, 2011). Previous studies have explored solutions to address this challenge. Stack LSTMs (Dyer et al., 2015; Ballesteros et al., 2015, 2016) are capable of learning representations of the parser state that are sensitive to the complete contents of the parser's state. Andor et al. (2016) proposed a globally normalized transition model to replace the locally normalized classifier. However, the parsing accuracy is still behind state-of-the-art graph-based parsers (Dozat and Manning, 2017).

Graph-based dependency parsers, on the other hand, learn scoring functions for parse trees and perform exhaustive search over all possible trees for a sentence to find the globally highest scoring

[*]Work done while at Carnegie Mellon University.

tree. Incorporating this global search algorithm with distributed representations learned from neural networks, neural graph-based parsers (Kiperwasser and Goldberg, 2016; Wang and Chang, 2016; Kuncoro et al., 2016; Dozat and Manning, 2017) have achieved the state-of-the-art accuracies on a number of treebanks in different languages. Nevertheless, these models, while accurate, are usually slow (e.g. decoding is $O(n^3)$ time complexity for first-order models (McDonald et al., 2005a,b) and higher polynomials for higher-order models (McDonald and Pereira, 2006; Koo and Collins, 2010; Ma and Zhao, 2012b,a)).

In this paper, we propose a novel neural network architecture for dependency parsing, *stack-pointer networks* (**STACKPTR**). STACKPTR is a transition-based architecture, with the corresponding asymptotic efficiency, but still maintains a global view of the sentence that proves essential for achieving competitive accuracy. Our STACKPTR parser has a pointer network (Vinyals et al., 2015) as its backbone, and is equipped with an internal stack to maintain the order of head words in tree structures. The STACKPTR parser performs parsing in an incremental, top-down, depth-first fashion; at each step, it generates an arc by assigning a child for the head word at the top of the internal stack. This architecture makes it possible to capture information from the whole sentence and all the previously derived subtrees, while maintaining a number of parsing steps linear in the sentence length.

We evaluate our parser on 29 treebanks across 20 languages and different dependency annotation schemas, and achieve state-of-the-art performance on 21 of them. The contributions of this work are summarized as follows:

(i) We propose a neural network architecture for dependency parsing that is simple, effective, and efficient.

(ii) Empirical evaluations on benchmark datasets over 20 languages show that our method achieves state-of-the-art performance on 21 different treebanks[1].

(iii) Comprehensive error analysis is conducted to compare the proposed method to a strong graph-based baseline using biaffine attention (Dozat and Manning, 2017).

[1]Source code is publicly available at `https://github.com/XuezheMax/NeuroNLP2`

2 Background

We first briefly describe the task of dependency parsing, setup the notation, and review Pointer Networks (Vinyals et al., 2015).

2.1 Dependency Parsing and Notations

Dependency trees represent syntactic relationships between words in the sentences through labeled directed edges between head words and their dependents. Figure 1 (a) shows a dependency tree for the sentence, "But there were no buyers".

In this paper, we will use the following notation:

Input: $\mathbf{x} = \{w_1, \ldots, w_n\}$ represents a generic sentence, where w_i is the ith word.

Output: $\mathbf{y} = \{p_1, p_2, \cdots, p_k\}$ represents a generic (possibly non-projective) dependency tree, where each path $p_i = \$, w_{i,1}, w_{i,2}, \cdots, w_{i,l_i}$ is a sequence of words from the root to a leaf. "\$" is an universal virtual root that is added to each tree.

Stack: σ denotes a stack configuration, which is a sequence of words. We use $\sigma|w$ to represent a stack configuration that pushes word w into the stack σ.

Children: $\mathrm{ch}(w_i)$ denotes the list of all the children (modifiers) of word w_i.

2.2 Pointer Networks

Pointer Networks (PTR-NET) (Vinyals et al., 2015) are a variety of neural network capable of learning the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence. This model cannot be trivially expressed by standard sequence-to-sequence networks (Sutskever et al., 2014) due to the variable number of input positions in each sentence. PTR-NET solves the problem by using attention (Bahdanau et al., 2015; Luong et al., 2015) as a pointer to select a member of the input sequence as the output.

Formally, the words of the sentence $\mathbf{x}$ are fed one-by-one into the encoder (a multiple-layer bi-directional RNN), producing a sequence of *encoder hidden states* s_i. At each time step t, the decoder (a uni-directional RNN) receives the input from last step and outputs *decoder hidden state* h_t. The *attention vector* a^t is calculated as follows:

$$\begin{array}{rcl} e_i^t & = & score(h_t, s_i) \\ a^t & = & softmax(e^t) \end{array} \quad (1)$$

where $score(\cdot, \cdot)$ is the *attention scoring function*, which has several variations such as dot-product,

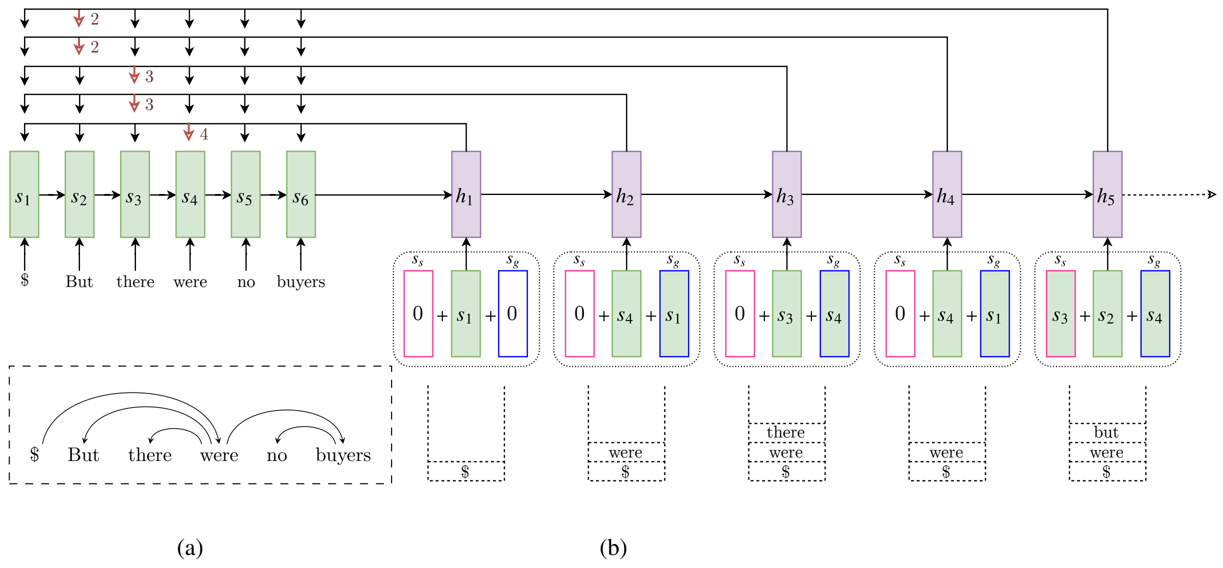


Figure 1: Neural architecture for the StackPtr network, together with the decoding procedure of an example sentence. The BiRNN of the encoder is elided for brevity. For the inputs of decoder at each time step, vectors in red and blue boxes indicate the sibling and grandparent.

concatenation, and biaffine (Luong et al., 2015). Ptr-Net regards the attention vector a^t as a probability distribution over the source words, i.e. it uses a^t_i as pointers to select the input elements.

3 Stack-Pointer Networks

3.1 Overview

Similarly to Ptr-Net, StackPtr first reads the whole sentence and encodes each word into the encoder hidden state s_i. The internal stack σ is always initialized with the root symbol \$. At each time step t, the decoder receives the input vector corresponding to the top element of the stack σ (the head word w_p where p is the word index), generates the hidden state h_t, and computes the attention vector a^t using Eq. (1). The parser chooses a specific position c according to the attention scores in a^t to generate a new dependency arc (w_h, w_c) by selecting w_c as a child of w_h. Then the parser pushes w_c onto the stack, i.e. $\sigma \rightarrow \sigma|w_c$, and goes to the next step. At one step if the parser points w_h to itself, i.e. $c = h$, it indicates that all children of the head word w_h have already been selected. Then the parser goes to the next step by popping w_h out of σ.

At test time, in order to guarantee a valid dependency tree containing all the words in the input sentences exactly once, the decoder maintains a list of "available" words. At each decoding step, the parser selects a child for the current head word, and removes the child from the list of available words to make sure that it cannot be selected as a child of other head words.

For head words with multiple children, it is possible that there is more than one valid selection for each time step. In order to define a deterministic decoding process to make sure that there is only one ground-truth choice at each step (which is necessary for simple maximum likelihood estimation), a predefined order for each $\text{ch}(w_i)$ needs to be introduced. The predefined order of children can have different alternatives, such as left-to-right or inside-out[2]. In this paper, we adopt the inside-out order[3] since it enables us to utilize second-order *sibling* information, which has been proven beneficial for parsing performance (McDonald and Pereira, 2006; Koo and Collins, 2010) (see § 3.4 for details). Figure 1 (b) depicts the architecture of StackPtr and the decoding procedure for the example sentence in Figure 1 (a).

3.2 Encoder

The encoder of our parsing model is based on the bi-directional LSTM-CNN architecture (BLSTM-CNNs) (Chiu and Nichols, 2016; Ma and Hovy, 2016) where CNNs encode character-level information of a word into its character-level repre-

[2]Order the children by the distances to the head word on the left side, then the right side.

[3]We also tried left-to-right order which obtained worse parsing accuracy than inside-out.

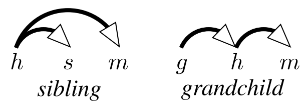
sentation and BLSTM models context information of each word. Formally, for each word, the CNN, with character embeddings as inputs, encodes the character-level representation. Then the character-level representation vector is concatenated with the word embedding vector to feed into the BLSTM network. To enrich word-level information, we also use POS embeddings. Finally, the encoder outputs a sequence of hidden states s_i.

3.3 Decoder

The decoder for our parser is a uni-directional LSTM. Different from previous work (Bahdanau et al., 2015; Vinyals et al., 2015) which uses word embeddings of the previous word as the input to the decoder, our decoder receives the encoder hidden state vector (s_i) of the top element in the stack σ (see Figure 1 (b)). Compared to word embeddings, the encoder hidden states contain more contextual information, benefiting both the training and decoding procedures. The decoder produces a sequence of decoder hidden states h_i, one for each decoding step.

3.4 Higher-order Information

As mentioned before, our parser is capable of utilizing higher-order information. In this paper, we incorporate two kinds of higher-order structures — *grandparent* and *sibling*. A sibling structure is a head word with two successive modifiers, and a grandparent structure is a pair of dependencies connected head-to-tail:



To utilize higher-order information, the decoder's input at each step is the sum of the encoder hidden states of three words:

$$\beta_t = s_h + s_g + s_s$$

where β_t is the input vector of decoder at time t and h, g, s are the indices of the head word and its grandparent and sibling, respectively. Figure 1 (b) illustrates the details. Here we use the element-wise sum operation instead of concatenation because it does not increase the dimension of the input vector β_t, thus introducing no additional model parameters.

3.5 Biaffine Attention Mechanism

For attention score function (Eq. (1)), we adopt the biaffine attention mechanism (Luong et al., 2015; Dozat and Manning, 2017):

$$e_i^t = h_t^T \mathbf{W} s_i + \mathbf{U}^T h_t + \mathbf{V}^T s_i + \mathbf{b}$$

where $\mathbf{W}, \mathbf{U}, \mathbf{V}$, $\mathbf{b}$ are parameters, denoting the weight matrix of the bi-linear term, the two weight vectors of the linear terms, and the bias vector.

As discussed in Dozat and Manning (2017), applying a multilayer perceptron (MLP) to the output vectors of the BLSTM before the score function can both reduce the dimensionality and overfitting of the model. We follow this work by using a one-layer perceptron to s_i and h_i with elu (Clevert et al., 2015) as its activation function.

Similarly, the dependency label classifier also uses a biaffine function to score each label, given the head word vector h_t and child vector s_i as inputs. Again, we use MLPs to transform h_t and s_i before feeding them into the classifier.

3.6 Training Objectives

The STACKPTR parser is trained to optimize the probability of the dependency trees given sentences: $P_\theta(\mathbf{y}|\mathbf{x})$, which can be factorized as:

$$\begin{aligned} P_\theta(\mathbf{y}|\mathbf{x}) &= \prod_{i=1}^{k} P_\theta(p_i|p_{<i}, \mathbf{x}) \\ &= \prod_{i=1}^{k} \prod_{j=1}^{l_i} P_\theta(c_{i,j}|c_{i,<j}, p_{<i}, \mathbf{x}), \end{aligned} \tag{2}$$

where θ represents model parameters. $p_{<i}$ denotes the preceding paths that have already been generated. $c_{i,j}$ represents the jth word in p_i and $c_{i,<j}$ denotes all the proceeding words on the path p_i. Thus, the STACKPTR parser is an autoregressive model, like sequence-to-sequence models, but it factors the distribution according to a top-down tree structure as opposed to a left-to-right chain. We define $P_\theta(c_{i,j}|c_{i,<j}, p_{<i}, \mathbf{x}) = a^t$, where attention vector a^t (of dimension n) is used as the distribution over the indices of words in a sentence.

Arc Prediction Our parser is trained by optimizing the conditional likelihood in Eq (2), which is implemented as the cross-entropy loss.

Label Prediction We train a separated multi-class classifier in parallel to predict the dependency labels. Following Dozat and Manning (2017), the classifier takes the information of the head word and its child as features. The label classifier is trained simultaneously with the parser by optimizing the sum of their objectives.

3.7 Discussion

Time Complexity. The number of decoding steps to build a parse tree for a sentence of length n is $2n-1$, linear in n. Together with the attention mechanism (at each step, we need to compute the attention vector a^t, whose runtime is $O(n)$), the time complexity of decoding algorithm is $O(n^2)$, which is more efficient than graph-based parsers that have $O(n^3)$ or worse complexity when using dynamic programming or maximum spanning tree (MST) decoding algorithms.

Top-down Parsing. When humans comprehend a natural language sentence, they arguably do it in an incremental, left-to-right manner. However, when humans consciously annotate a sentence with syntactic structure, they rarely ever process in fixed left-to-right order. Rather, they start by reading the whole sentence, then seeking the main predicates, jumping back-and-forth over the sentence and recursively proceeding to the sub-tree structures governed by certain head words. Our parser follows a similar kind of annotation process: starting from reading the whole sentence, and processing in a top-down manner by finding the main predicates first and only then search for sub-trees governed by them. When making latter decisions, the parser has access to the entire structure built in earlier steps.

3.8 Implementation Details

Pre-trained Word Embeddings. For all the parsing models in different languages, we initialize word vectors with pretrained word embeddings. For Chinese, Dutch, English, German and Spanish, we use the structured-skipgram (Ling et al., 2015) embeddings. For other languages we use Polyglot embeddings (Al-Rfou et al., 2013).

Optimization. Parameter optimization is performed with the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = \beta_2 = 0.9$. We choose an initial learning rate of $\eta_0 = 0.001$. The learning rate η is annealed by multiplying a fixed decay rate $\rho = 0.75$ when parsing performance stops increasing on validation sets. To reduce the effects of "gradient exploding", we use gradient clipping of 5.0 (Pascanu et al., 2013).

Dropout Training. To mitigate overfitting, we apply dropout (Srivastava et al., 2014; Ma et al., 2017). For BLSTM, we use recurrent dropout (Gal and Ghahramani, 2016) with a drop rate of 0.33 between hidden states and 0.33 between layers. Following Dozat and Manning (2017), we also use embedding dropout with a rate of 0.33 on all word, character, and POS embeddings.

Hyper-Parameters. Some parameters are chosen from those reported in Dozat and Manning (2017). We use the same hyper-parameters across the models on different treebanks and languages, due to time constraints. The details of the chosen hyper-parameters for all experiments are summarized in Appendix A.

4 Experiments

4.1 Setup

We evaluate our StackPtr parser mainly on three treebanks: the English Penn Treebank (PTB version 3.0) (Marcus et al., 1993), the Penn Chinese Treebank (CTB version 5.1) (Xue et al., 2002), and the German CoNLL 2009 corpus (Hajič et al., 2009). We use the same experimental settings as Kuncoro et al. (2016).

To make a thorough empirical comparison with previous studies, we also evaluate our system on treebanks from CoNLL shared task and the Universal Dependency (UD) Treebanks[4]. For the CoNLL Treebanks, we use the English treebank from CoNLL-2008 shared task (Surdeanu et al., 2008) and all 13 treebanks from CoNLL-2006 shared task (Buchholz and Marsi, 2006). The experimental settings are the same as Ma and Hovy (2015). For UD Treebanks, we select 12 languages. The details of the treebanks and experimental settings are in § 4.5 and Appendix B.

Evaluation Metrics Parsing performance is measured with five metrics: unlabeled attachment score (UAS), labeled attachment score (LAS), unlabeled complete match (UCM), labeled complete match (LCM), and root accuracy (RA). Following previous work (Kuncoro et al., 2016; Dozat and Manning, 2017), we report results excluding punctuations for Chinese and English. For each experiment, we report the mean values with corresponding standard deviations over 5 repetitions.

[4]http://universaldependencies.org/

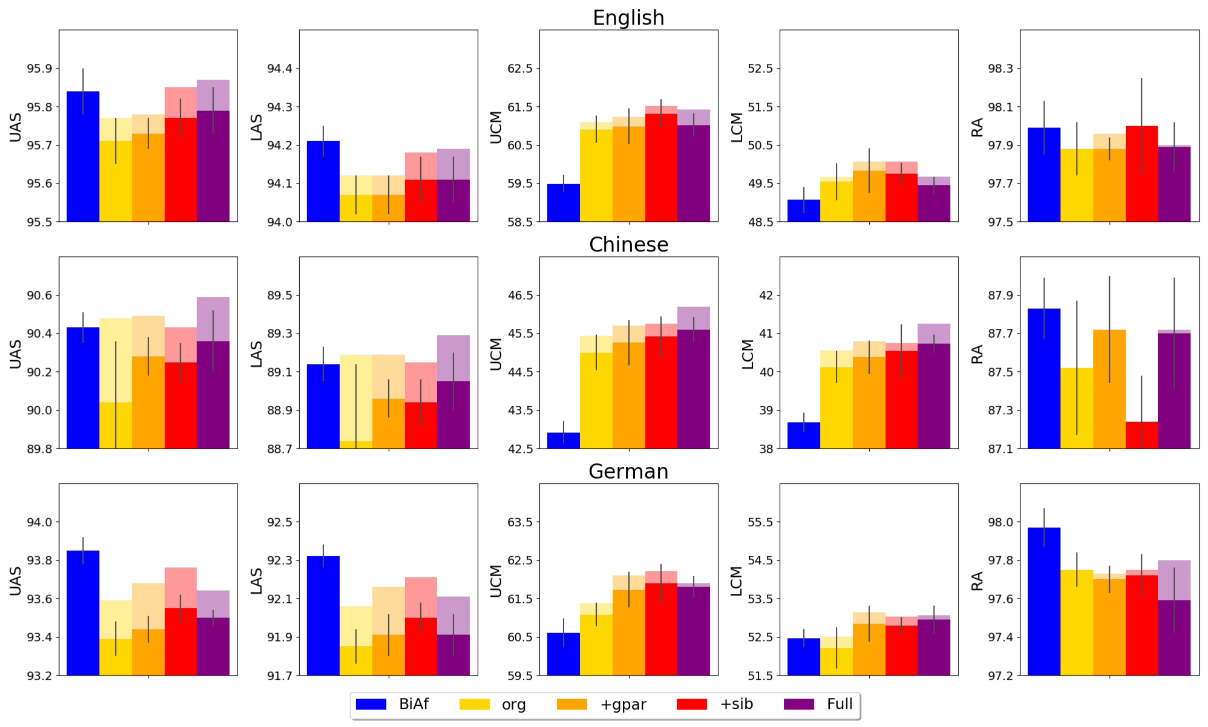


Figure 2: Parsing performance of different variations of our model on the test sets for three languages, together with baseline BIAF. For each of our STACKPTR models, we perform decoding with beam size equal to 1 and 10. The improvements of decoding with beam size 10 over 1 are presented by stacked bars with light colors.

Baseline For fair comparison of the parsing performance, we re-implemented the graph-based Deep Biaffine (BIAF) parser (Dozat and Manning, 2017), which achieved state-of-the-art results on a wide range of languages. Our re-implementation adds character-level information using the same LSTM-CNN encoder as our model (§ 3.2) to the original BIAF model, which boosts its performance on all languages.

4.2 Main Results

We first conduct experiments to demonstrate the effectiveness of our neural architecture by comparing with the strong baseline BIAF. We compare the performance of four variations of our model with different decoder inputs — Org, +gpar, +sib and Full — where the Org model utilizes only the encoder hidden states of head words, while the +gpar and +sib models augments the original one with grandparent and sibling information, respectively. The Full model includes all the three information as inputs.

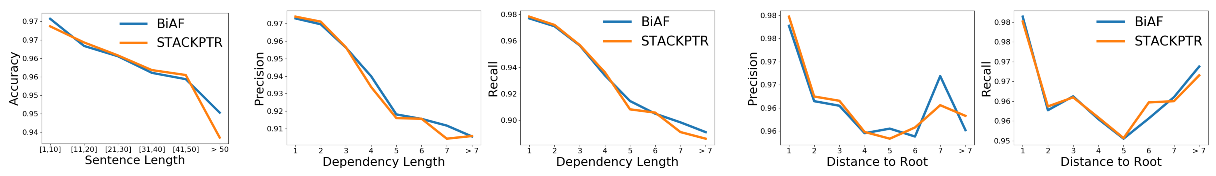
Figure 2 illustrates the performance (five metrics) of different variations of our STACKPTR parser together with the results of baseline BIAF re-implemented by us, on the test sets of the three languages. On UAS and LAS, the Full variation of STACKPTR with decoding beam size 10 outperforms BIAF on Chinese, and obtains competitive performance on English and German. An interesting observation is that the Full model achieves the best accuracy on English and Chinese, while performs slightly worse than +sib on German. This shows that the importance of higher-order information varies in languages. On LCM and UCM, STACKPTR significantly outperforms BIAF on all languages, showing the superiority of our parser on complete sentence parsing. The results of our parser on RA are slightly worse than BIAF. More details of results are provided in Appendix C.

4.3 Comparison with Previous Work

Table 1 illustrates the UAS and LAS of the four versions of our model (with decoding beam size 10) on the three treebanks, together with previous top-performing systems for comparison. Note that the results of STACKPTR and our re-implementation of BIAF are the average of 5 repetitions instead of a single run. Our Full model significantly outperforms all the transition-based parsers on all three languages, and achieves better results than most graph-based parsers. Our

System		English		Chinese		German	
		UAS	LAS	UAS	LAS	UAS	LAS
Chen and Manning (2014)	T	91.8	89.6	83.9	82.4	–	–
Ballesteros et al. (2015)	T	91.63	89.44	85.30	83.72	88.83	86.10
Dyer et al. (2015)	T	93.1	90.9	87.2	85.7	–	–
Bohnet and Nivre (2012)	T	93.33	91.22	87.3	85.9	91.4	89.4
Ballesteros et al. (2016)	T	93.56	91.42	87.65	86.21	–	–
Kiperwasser and Goldberg (2016)	T	93.9	91.9	87.6	86.1	–	–
Weiss et al. (2015)	T	94.26	92.41	–	–	–	–
Andor et al. (2016)	T	94.61	92.79	–	–	90.91	89.15
Kiperwasser and Goldberg (2016)	G	93.1	91.0	86.6	85.1	–	–
Wang and Chang (2016)	G	94.08	91.82	87.55	86.23	–	–
Cheng et al. (2016)	G	94.10	91.49	88.1	85.7	–	–
Kuncoro et al. (2016)	G	94.26	92.06	88.87	87.30	91.60	89.24
Ma and Hovy (2017)	G	94.88	92.98	89.05	87.74	92.58	90.54
BIAF: Dozat and Manning (2017)	G	95.74	94.08	89.30	88.23	93.46	91.44
BIAF: re-impl	G	95.84	**94.21**	90.43	89.14	**93.85**	**92.32**
STACKPTR: Org	T	95.77	94.12	90.48	89.19	93.59	92.06
STACKPTR: +gpar	T	95.78	94.12	90.49	89.19	93.65	92.12
STACKPTR: +sib	T	95.85	94.18	90.43	89.15	93.76	92.21
STACKPTR: Full	T	**95.87**	94.19	**90.59**	**89.29**	93.65	92.11

Table 1: UAS and LAS of four versions of our model on test sets for three languages, together with top-performing parsing systems. "T" and "G" indicate transition- and graph-based models, respectively. For BIAF, we provide the original results reported in Dozat and Manning (2017) and our re-implementation. For STACKPTR and our re-implementation of BiAF, we report the average over 5 runs.



(a) (b) (c)

Figure 3: Parsing performance of BIAF and STACKPTR parsers relative to length and graph factors.

POS	UAS	LAS	UCM	LCM
Gold	96.12±0.03	95.06±0.05	62.22±0.33	55.74±0.44
Pred	95.87±0.04	94.19±0.04	61.43±0.49	49.68±0.47
None	95.90±0.05	94.21±0.04	61.58±0.39	49.87±0.46

Table 2: Parsing performance on the test data of PTB with different versions of POS tags.

re-implementation of BIAF obtains better performance than the original one in Dozat and Manning (2017), demonstrating the effectiveness of the character-level information. Our model achieves state-of-the-art performance on both UAS and LAS on Chinese, and best UAS on English. On German, the performance is competitive with BIAF, and significantly better than other models.

4.4 Error Analysis

In this section, we characterize the errors made by BIAF and STACKPTR by presenting a number of experiments that relate parsing errors to a set of linguistic and structural properties. For simplicity, we follow McDonald and Nivre (2011) and report labeled parsing metrics (either accuracy, precision, or recall) for all experiments.

4.4.1 Length and Graph Factors

Following McDonald and Nivre (2011), we analyze parsing errors related to structural factors.

Sentence Length. Figure 3 (a) shows the accuracy of both parsing models relative to sentence lengths. Consistent with the analysis in McDonald and Nivre (2011), STACKPTR tends to perform better on shorter sentences, which make fewer parsing decisions, significantly reducing the chance of error propagation.

Dependency Length. Figure 3 (b) measures the precision and recall relative to dependency lengths. While the graph-based BIAF parser still performs better for longer dependency arcs and transition-based STACKPTR parser does better for shorter ones, the gap between the two systems is marginal, much smaller than that shown

	Bi-Att	NeuroMST	BIAF	STACKPTR	Best Published	
	UAS [LAS]	UAS [LAS]	UAS [LAS]	UAS [LAS]	UAS	LAS
ar	80.34 [68.58]	80.80 [69.40]	82.15±0.34 [71.32±0.36]	**83.04±0.29 [72.94±0.31]**	81.12	–
bg	93.96 [89.55]	94.28 [90.60]	94.62±0.14 **[91.56±0.24]**	**94.66±0.10** [91.40±0.08]	94.02	–
zh	–	93.40 [90.10]	**94.05±0.27 [90.89±0.22]**	93.88±0.24 [90.81±0.55]	93.04	–
cs	91.16 [85.14]	91.18 [85.92]	92.24±0.22 [87.85±0.21]	**92.83±0.13 [88.75±0.16]**	91.16	85.14
da	91.56 [85.53]	91.86 [87.07]	**92.80±0.26 [88.36±0.18]**	92.08±0.15 [87.29±0.21]	92.00	–
nl	87.15 [82.41]	87.85 [84.82]	90.07±0.18 **[87.24±0.17]**	**90.10±0.27** [87.05±0.26]	87.39	–
en	–	94.66 [92.52]	95.19±0.05 [93.14±0.05]	**93.25±0.05 [93.17±0.05]**	93.25	–
de	92.71 [89.80]	93.62 [91.90]	94.52±0.11 [93.06±0.11]	**94.77±0.05 [93.21±0.10]**	92.71	89.80
ja	93.44 [90.67]	**94.02 [92.60]**	93.95±0.06 [92.46±0.07]	93.38±0.08 [91.92±0.16]	93.80	–
pt	92.77 [88.44]	92.71 [88.92]	93.41±0.08 [89.96±0.24]	**93.57±0.12 [90.07±0.20]**	93.03	–
sl	86.01 [75.90]	86.73 [77.56]	87.55±0.17 [78.52±0.35]	**87.59±0.36 [78.85±0.53]**	87.06	–
es	88.74 [84.03]	89.20 [85.77]	90.43±0.13 [87.08±0.14]	**90.87±0.26 [87.80±0.31]**	88.75	84.03
sv	90.50 [84.05]	91.22 [86.92]	92.22±0.15 [88.44±0.17]	**92.49±0.21 [89.01±0.22]**	91.85	85.26
tr	78.43 [66.16]	77.71 [65.81]	**79.84±0.23 [68.63±0.29]**	79.56±0.22 [68.03±0.15]	78.43	66.16

Table 3: UAS and LAS on 14 treebanks from CoNLL shared tasks, together with several state-of-the-art parsers. Bi-Att is the bi-directional attention based parser (Cheng et al., 2016), and NeuroMST is the neural MST parser (Ma and Hovy, 2017). "Best Published" includes the most accurate parsers in term of UAS among Koo et al. (2010), Martins et al. (2011), Martins et al. (2013), Lei et al. (2014), Zhang et al. (2014), Zhang and McDonald (2014), Pitler and McDonald (2015), and Cheng et al. (2016).

in McDonald and Nivre (2011). One possible reason is that, unlike traditional transition-based parsers that scan the sentence from left to right, STACKPTR processes in a top-down manner, thus sometimes unnecessarily creating shorter dependency arcs first.

Root Distance. Figure 3 (c) plots the precision and recall of each system for arcs of varying distance to the root. Different from the observation in McDonald and Nivre (2011), STACKPTR does not show an obvious advantage on the precision for arcs further away from the root. Furthermore, the STACKPTR parser does not have the tendency to over-predict root modifiers reported in McDonald and Nivre (2011). This behavior can be explained using the same reasoning as above: the fact that arcs further away from the root are usually constructed early in the parsing algorithm of traditional transition-based parsers is not true for the STACKPTR parser.

4.4.2 Effect of POS Embedding

The only prerequisite information that our parsing model relies on is POS tags. With the goal of achieving an end-to-end parser, we explore the effect of POS tags on parsing performance. We run experiments on PTB using our STACKPTR parser with gold-standard and predicted POS tags, and without tags, respectively. STACKPTR in these experiments is the Full model with beam=10.

Table 2 gives results of the parsers with different versions of POS tags on the test data of PTB. The parser with gold-standard POS tags significantly outperforms the other two parsers, showing that dependency parsers can still benefit from accurate POS information. The parser with predicted (imperfect) POS tags, however, performs even slightly worse than the parser without using POS tags. It illustrates that an end-to-end parser that doesn't rely on POS information can obtain competitive (or even better) performance than parsers using imperfect predicted POS tags, even if the POS tagger is relative high accuracy (accuracy $> 97\%$ in this experiment on PTB).

4.5 Experiments on Other Treebanks

4.5.1 CoNLL Treebanks

Table 3 summarizes the parsing results of our model on the test sets of 14 treebanks from the CoNLL shared task, along with the state-of-the-art baselines. Along with BIAF, we also list the performance of the bi-directional attention based Parser (Bi-Att) (Cheng et al., 2016) and the neural MST parser (NeuroMST) (Ma and Hovy, 2017) for comparison. Our parser achieves state-of-the-art performance on both UAS and LAS on eight languages — Arabic, Czech, English, German, Portuguese, Slovene, Spanish, and Swedish. On Bulgarian and Dutch, our parser obtains the best UAS. On other languages, the performance of our parser is competitive with BIAF, and significantly better than others. The only exception is Japanese, on which NeuroMST obtains the best scores.

	Dev				Test			
	BIAF		STACKPTR		BIAF		STACKPTR	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
bg	93.92±0.13	89.05±0.11	**94.09±0.16**	**89.17±0.14**	94.30±0.16	**90.04±0.16**	**94.31±0.06**	89.96±0.07
ca	94.21±0.05	91.97±0.06	**94.47±0.02**	**92.51±0.05**	94.36±0.06	92.05±0.07	**94.47±0.02**	**92.39±0.02**
cs	94.14±0.03	90.89±0.04	**94.33±0.04**	**91.24±0.05**	94.06±0.04	90.60±0.05	**94.21±0.06**	**90.94±0.07**
de	91.89±0.11	88.39±0.17	**92.26±0.11**	**88.79±0.15**	90.26±0.19	86.11±0.25	**90.26±0.07**	**86.16±0.01**
en	**92.51±0.08**	**90.50±0.07**	92.47±0.03	90.46±0.02	91.91±0.17	89.82±0.16	**91.93±0.07**	**89.83±0.06**
es	93.46±0.05	91.13±0.07	**93.54±0.06**	**91.34±0.05**	93.72±0.07	91.33±0.08	**93.77±0.07**	**91.52±0.07**
fr	**95.05±0.04**	**92.76±0.07**	94.97±0.04	92.57±0.06	92.62±0.15	89.51±0.14	**92.90±0.20**	**89.88±0.23**
it	94.89±0.12	92.58±0.12	**94.93±0.09**	**92.90±0.10**	**94.75±0.12**	**92.72±0.12**	94.70±0.07	92.55±0.09
nl	93.39±0.08	90.90±0.07	**93.94±0.11**	**91.67±0.08**	93.44±0.09	91.04±0.06	**93.98±0.05**	**91.73±0.07**
no	95.44±0.05	93.73±0.05	**95.52±0.08**	**93.80±0.08**	95.28±0.05	93.58±0.05	**95.33±0.03**	**93.62±0.03**
ro	91.97±0.13	85.38±0.03	**92.06±0.08**	**85.58±0.12**	**91.94±0.07**	**85.61±0.13**	91.80±0.11	85.34±0.21
ru	93.81±0.05	91.85±0.06	**94.11±0.07**	**92.29±0.10**	94.40±0.03	92.68±0.04	**94.69±0.04**	**93.07±0.03**

Table 4: UAS and LAS on both the development and test datasets of 12 treebanks from UD Treebanks, together with BIAF for comparison.

4.5.2 UD Treebanks

For UD Treebanks, we select 12 languages — Bulgarian, Catalan, Czech, Dutch, English, French, German, Italian, Norwegian, Romanian, Russian and Spanish. For all the languages, we adopt the standard training/dev/test splits, and use the universal POS tags (Petrov et al., 2012) provided in each treebank. The statistics of these corpora are provided in Appendix B.

Table 4 summarizes the results of the STACKPTR parser, along with BIAF for comparison, on both the development and test datasets for each language. First, both BIAF and STACKPTR parsers achieve relatively high parsing accuracies on all the 12 languages — all with UAS are higher than 90%. On nine languages — Catalan, Czech, Dutch, English, French, German, Norwegian, Russian and Spanish — STACKPTR outperforms BIAF for both UAS and LAS. On Bulgarian, STACKPTR achieves slightly better UAS while LAS is slightly worse than BIAF. On Italian and Romanian, BIAF obtains marginally better parsing performance than STACKPTR.

5 Conclusion

In this paper, we proposed STACKPTR, a transition-based neural network architecture, for dependency parsing. Combining pointer networks with an internal stack to track the status of the top-down, depth-first search in the decoding procedure, the STACKPTR parser is able to capture information from the whole sentence and all the previously derived subtrees, removing the left-to-right restriction in classical transition-based parsers, while maintaining linear parsing steps, w.r.t the length of the sentences. Experimental results on 29 treebanks show the effectiveness of our parser across 20 languages, by achieving state-of-the-art performance on 21 corpora.

There are several potential directions for future work. First, we intend to consider how to conduct experiments to improve the analysis of parsing errors qualitatively and quantitatively. Another interesting direction is to further improve our model by exploring reinforcement learning approaches to learn an optimal order for the children of head words, instead of using a predefined fixed order.

Acknowledgements

The authors thank Chunting Zhou, Di Wang and Zhengzhong Liu for their helpful discussions. This research was supported in part by DARPA grant FA8750-18-2-0018 funded under the AIDA program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of CoNLL-2013*. Sofia, Bulgaria, pages 183–192.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL-2016 (Volume 1: Long Papers)*. Berlin, Germany, pages 2442–2452.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction.

In *Proceedings of ACL-2015 (Volume 1: Long Papers)*. Beijing, China, pages 344–354.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR-2015*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of EMNLP-2015*. Lisbon, Portugal, pages 349–359.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of EMNLP-2016*. Austin, Texas, pages 2005–2010.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP-2017*. Copenhagen, Denmark, pages 1957–1967.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP-2012*. Jeju Island, Korea, pages 1455–1465.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceeding of CoNLL-2006*. New York, NY, pages 149–164.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP-2014*. Doha, Qatar, pages 740–750.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of EMNLP-2016*. Austin, Texas, pages 2204–2214.

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

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* .

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR-2017 (Volume 1: Long Papers)*. Toulon, France.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of EMNLP-2013*. Seattle, Washington, USA, pages 1971–1982.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL-2015 (Volume 1: Long Papers)*. Beijing, China, pages 334–343.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-1996 (Volume 1)*. Association for Computational Linguistics, pages 340–345.

Nicolas R Fauceglia, Yiu-Chang Lin, Xuezhe Ma, and Eduard Hovy. 2015. Word sense disambiguation via propstore and ontonotes for event mention detection. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. Denver, Colorado, pages 11–15.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL-2009: Shared Task*. pages 1–18.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL-2010*. Uppsala, Sweden, pages 1–11.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP-2010*. Cambridge, MA, pages 1288–1298.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *Proceedings of EMNLP-2016*. Austin, Texas, pages 1744–1753.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of ACL-2014 (Volume 1: Long Papers)*. Baltimore, Maryland, pages 1381–1391.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of NAACL-2015*. Denver, Colorado, pages 1299–1304.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP-2015*. Lisbon, Portugal, pages 1412–1421.

Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. 2017. Dropout with expectation-linear regularization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-2017)*. Toulon, France.

Xuezhe Ma and Eduard Hovy. 2015. Efficient inner-to-outer greedy algorithm for higher-order labeled dependency parsing. In *Proceedings of EMNLP-2015*. Lisbon, Portugal, pages 1322–1328.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL-2016 (Volume 1: Long Papers)*. Berlin, Germany, pages 1064–1074.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. In *Proceedings of IJCNLP-2017 (Volume 1: Long Papers)*. Taipei, Taiwan, pages 59–69.

Xuezhe Ma, Zhengzhong Liu, and Eduard Hovy. 2016. Unsupervised ranking model for entity coreference resolution. In *Proceedings of NAACL-2016*. San Diego, California, USA.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL-2014*. Baltimore, Maryland, pages 1337–1348.

Xuezhe Ma and Hai Zhao. 2012a. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*. Mumbai, India, pages 785–796.

Xuezhe Ma and Hai Zhao. 2012b. Probabilistic models for high-order projective dependency parsing. *Technical Report, arXiv:1502.04174* .

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2):313–330.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL-2013 (Volume 2: Short Papers)*. Sofia, Bulgaria, pages 617–622.

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP-2011*. Edinburgh, Scotland, UK., pages 238–249.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*. Ann Arbor, Michigan, USA, pages 91–98.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics* 37(1):197–230.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL-2013*. Sofia, Bulgaria, pages 92–97.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceeding of EACL-2006*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP-2005*. Vancouver, Canada, pages 523–530.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of ACL-2010*. Association for Computational Linguistics, Uppsala, Sweden, pages 1396–1411.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of EMNLP-2009*. Singapore, pages 1378–1387.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Prague, Czech Republic, pages 915–932.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*. Geneva, Switzerland, pages 64–70.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML-2013*. pages 1310–1318.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics* 5:101–115.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC-2012*. Istanbul, Turkey, pages 2089–2096.

Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of NAACL-2015*. Denver, Colorado, pages 662–671.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*. pages 159–177.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings ACL-2015 (Volume 1: Long Papers)*. Beijing, China, pages 1556–1566.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of ACL-2016 (Volume 1: Long Papers)*. Berlin, Germany, pages 2306–2315.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL-2015 (Volume 1: Long Papers)*. Beijing, China, pages 323–333.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of COLING-2002*. pages 1–8.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*. Nancy, France, volume 3, pages 195–206.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of ACL-2014 (Volume 2: Short Papers)*. Baltimore, Maryland, pages 656–661.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of EMNLP-2014*. Doha, Qatar, pages 1013–1024.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, pages 188–193.

Twitter Universal Dependency Parsing for African-American and Mainstream American English

Su Lin Blodgett **Johnny Tian-Zheng Wei** **Brendan O'Connor**
College of Information and Computer Sciences
University of Massachusetts Amherst
blodgett@cs.umass.edu jwei@umass.edu brenocon@cs.umass.edu

Abstract

Due to the presence of both Twitter-specific conventions and non-standard and dialectal language, Twitter presents a significant parsing challenge to current dependency parsing tools. We broaden English dependency parsing to handle social media English, particularly social media African-American English (AAE), by developing and annotating a new dataset of 500 tweets, 250 of which are in AAE, within the Universal Dependencies 2.0 framework. We describe our standards for handling Twitter- and AAE-specific features and evaluate a variety of cross-domain strategies for improving parsing with no, or very little, in-domain labeled data, including a new data synthesis approach. We analyze these methods' impact on *performance disparities* between AAE and Mainstream American English tweets, and assess parsing accuracy for specific AAE lexical and syntactic features. Our annotated data and a parsing model are available at: http://slanglab.cs.umass.edu/TwitterAAE/.

1 Introduction

Language on Twitter diverges from well-edited Mainstream American English (MAE, also called Standard American English) in a number of ways, presenting significant challenges to current NLP tools. It contains, among other phenomena, non-standard spelling, punctuation, capitalization, and syntax, as well as Twitter-specific conventions such as hashtags, usernames, and retweet tokens (Eisenstein, 2013). Additionally, it contains an abundance of dialectal language, including African-American English (AAE), a dialect of American English spoken by millions of individuals, which contains lexical, phonological, and syntactic features not present in MAE (Green, 2002; Stewart, 2014; Jones, 2015).

Since standard English NLP tools are typically trained on well-edited MAE text, their performance is degraded on Twitter, and even more so for AAE tweets compared to MAE tweets—gaps exist for part-of-speech tagging (Jørgensen et al., 2016), language identification, and dependency parsing (Blodgett et al., 2016; Blodgett and O'Connor, 2017). Expanding the linguistic coverage of NLP tools to include minority and colloquial dialects would help support equitable language analysis across sociolinguistic communities, which could help information retrieval, translation, or opinion analysis applications (Jurgens et al., 2017). For example, sentiment analysis systems ought to count the opinions of all types of people, whether they use standard dialects or not.

In this work, we broaden Universal Dependencies (Nivre et al., 2016) parsing[1] to better handle social media English, in particular social media AAE. First, we develop standards to handle Twitter-specific and AAE-specific features within Universal Dependencies 2.0 (§3), by selecting and annotating a new dataset of 500 tweets, 250 of which are in AAE.

Second, we evaluate several state-of-the-art dependency parsers, finding that, as expected, they perform poorly on our dataset relative to the UD English Treebank (§4). Third, since the UD English Treebank contains substantial amounts of traditional MAE data for training, we investigate cross-domain training methods to improve Twitter AAE dependency parsing with no, or very little,

[1]http://universaldependencies.org/

in-domain labeled data, by using Twitter-specific taggers, embeddings, and a novel heuristic training data synthesis procedure. This helps close some of the gap between MAE and AAE performance. Finally, we provide an error analysis of the parsers' performance on AAE lexical and syntactic constructions in our dataset (§5.4).[2]

2 Related Work

2.1 Parsing for Twitter

Parsing for noisy social media data presents interesting and significant challenges. Foster et al. (2011) develop a dataset of 519 constituency-annotated English tweets, which were converted to Stanford dependencies. Their analysis found a substantial drop in performance of an off-the-shelf dependency parser on the new dataset compared to a WSJ test set. Sanguinetti et al. (2017) annotated a dataset of 6,738 Italian tweets according to UD 2.0 and examined the performance of two parsers on the dataset, finding that they lagged considerably relative to performance on the Italian UD Treebank.

Kong et al. (2014) develop an English dependency parser designed for Twitter, annotating a dataset of 929 tweets (TWEEBANK V1) according to the unlabeled FUDG dependency formalism (Schneider et al., 2013). It has substantially different structure than UD (for example, prepositions head PPs, and auxiliaries govern main verbs).

More recently, Liu et al. (2018) developed TWEEBANK V2, fully annotating TWEEBANK V1 according to UD 2.0 and annotating additionally sampled tweets, for a total of 3,550 tweets. They found that creating consistent annotations was challenging, due to frequent ambiguities in interpreting tweets; nevertheless, they were able to train a pipeline for tokenizing, tagging, and parsing the tweets, and develop ensemble and distillation models to improve parsing accuracy. Our work encounters similar challenges; in our approach, we intentionally oversample AAE-heavy messages for annotation, detail specific annotation decisions for AAE-specific phenomena (§3.2), and analyze parser performance between dialects and for particular constructions (§5.3–5.4). Future work may be able to combine these annotations for effective multi-dialect Twitter UD parsers, which may allow for the use of pre-existing downstream tools like semantic relation extractors (e.g. White et al. (2016)).

One line of work for parsing noisy social media data, including Khan et al. (2013) and Nasr et al. (2016), examines the effects of the domain mismatches between traditional sources of training data and social media data, finding that matching the data as closely as possible aids performance. Other work focuses on normalization, including Daiber and van der Goot (2016) and van der Goot and van Noord (2017), which develop a dataset of 500 manually normalized and annotated tweets, and uses normalization within a parser. Separately, Zhang et al. (2013) created a domain-adaptable, parser-focused system by directly linking parser performance to normalization performance.

2.2 Parsing for Dialects

For Arabic dialects, Chiang et al. (2006) parse Levantine Arabic by projecting parses from Modern Standard Arabic translations, while Green and Manning (2010) conduct extensive error analysis of Arabic constituency parsers and the Penn Arabic Treebank. Scherrer (2011) parse Swiss German dialects by transforming Standard German phrase structures. We continue in this line of work in our examination of AAE-specific syntactic structures and generation of synthetic data with such structures (§4.2.1).

Less work has examined parsing dialectal language on social media. Recently, Wang et al. (2017) annotate 1,200 Singlish (Singaporean English) sentences from a Singaporean talk forum, selecting sentences containing uniquely Singaporean vocabulary items. Like other work, they observe a drop in performance on dialectal Singlish text, but increase performance through a stacking-based domain adaptation method.

3 Dataset and Annotation

3.1 Dataset

Our dataset contains 500 tweets, with a total of 5,951 non-punctuation edges, sampled from the publicly available TwitterAAE corpus.[3] Each tweet in that corpus is accompanied by a model's demographically-aligned topic model probabilities jointly inferred from Census demographics and word likelihood by Blodgett et al. (2016), including the African-American and White topics.

[2]Our annotated dataset and trained dependency parser are available at http://slanglab.cs.umass.edu/TwitterAAE/ and annotations are available in the public Universal Dependencies repository.

[3]http://slanglab.cs.umass.edu/TwitterAAE/

We create a balanced sample to get a range of dialectal language, sampling 250 tweets from those where the African-American topic has at least 80% probability, and 250 from those where the White topic has at least 80% probability. We refer to these two subcorpora as AA and WH; Blodgett et al. (2016) showed the former exhibits linguistic features typical of AAE.

The 250 AA tweets include many alternate spellings of common words that correspond to well-known phonological phenomena—including *da, tha* (the), *dat, dhat* (that), *dis, dhis* (this), *ion, iont* (I don't), *ova* (over), *yo* (your), *dere, der* (there), *den, dhen* (then), *ova* (over), and *nall, null* (no, nah)—where each of the mentioned italicized AAE terms appears in the AAE data, but never in the MAE data. We examine these lexical variants more closely in §5.4. Across the AA tweets, 18.0% of tokens were not in a standard English dictionary, while the WH tweets' OOV rate was 10.7%.[4] We further observe a variety of AAE syntactic phenomena in our AA tweets, several of which are described in §3.2 and §5.4.

3.2 Annotation

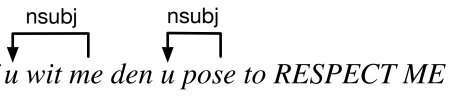
To effectively measure parsing quality and develop better future models, we first focus on developing high-quality annotations for our dataset, for which we faced a variety of challenges. We detail our annotation principles using Universal Dependency 2.0 relations (Nivre et al., 2016).

All tweets were initially annotated by two annotators, and disagreements resolved by one of the annotators. Annotation decisions for several dozen tweets were discussed in a group of three annotators early in the annotation process.

Our annotation principles are in alignment with those proposed by Liu et al. (2018), with the exception of contraction handling, which we discuss briefly in §3.2.2.

3.2.1 Null Copulas

The AAE dialect is prominently characterized by the drop of copulas, which can occur when the copula is present tense, not first person, not accented, not negative, and expressing neither the habitual nor the remote present perfect tenses (Green, 2002). We frequently observed null copulas, as in:



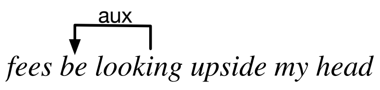
If u wit me den u pose to RESPECT ME

"If you (are) with me, then you (are) supposed to respect me"

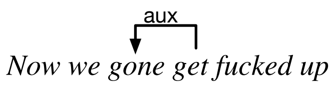
The first dropped *are* is a null copula; UD2.0 would analyze the MAE version as *you* $\xleftarrow{nsubj}$ *me* $\xrightarrow{cop}$ *are*, which we naturally extend to analyze the null copula by simply omitting *cop* (which is now over a null element, so cannot exist in a dependency graph). The second *are* is a null auxiliary (in MAE, *you* $\xleftarrow{nsubj}$ *supposed* $\xrightarrow{aux}$ *are*), a tightly related phenomenon (for example, Green et al. (2007) studies both null copulas and null auxiliary *be* in infant AAE), which we analyze similarly by simply omitting the *aux* edge.

3.2.2 AAE Verbal Auxiliaries

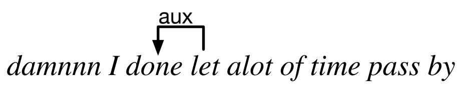
We observed AAE verbal auxiliaries, e.g.,



fees be looking upside my head



Now we gone get fucked up



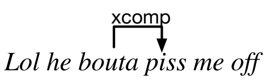
damnnn I done let alot of time pass by

including habitual *be* ("Continually, over and over, fees are looking at me..."), future *gone* ("we are going to get..."), and completive *done* ("I did let time pass by," emphasizing the speaker completed a time-wasting action).

We attach the auxiliary to the main verb with the *aux* relation, as UD2.0 analyzes other English auxiliaries (e.g. *would* or *will*).

3.2.3 Verbs: Auxiliaries vs. Main Verbs

We observed many instances of quasi-auxiliary, "-to" shortened verbs such as *wanna, gotta, finna, bouta, tryna, gonna*, which can be glossed as *want to*, *got to*, *fixing to*, *about to*, etc. They control modality, mood and tense—for example, *finna* and *bouta* denote an immediate future tense; Green (2002, ch. 2) describes *finna* as a preverbal marker. From UD's perspective, it is difficult to decide if they should be subordinate auxiliaries or main verbs. In accordance with the UD Treebank's handling of MAE *want to X* and *going to X* as main verbs (*want* $\xrightarrow{xcomp}$ *X*), we analyzed them similarly, e.g.



Lol he bouta piss me off "He is about to piss me off"

[4]The dictionary of 123,377 words with American spellings was generated using http://wordlist.aspell.net/.

This is an instance of a general principle that, if there is a shortening of an MAE multiword phrase into a single word, the annotations on that word should mirror the edges in and out of the original phrase's subgraph (as in Schneider et al. (2013)'s fudge expressions).

However, in contrast to the UD Treebank, we did not attempt to split up these words into their component words (e.g. *wanna* $\rightarrow$ *want to*), since to do this well, it would require a more involved segmentation model over the dozens or even hundreds of alternate spellings each of the above can take;[5] we instead rely on Owoputi et al. (2013); O'Connor et al. (2010)'s rule-based tokenizer that never attempts to segment within such shortenings. This annotation principle is in contrast to that of Liu et al. (2018), which follows UD tokenization for contractions.

3.2.4 Non-AAE Twitter issues

We also encountered many issues general to Twitter but not AAE; these are still important to deal with since AAE tweets include more non-standard linguistic phenomena overall. When possible, we adapted Kong et al. (2014)'s annotation conventions into the Universal Dependencies context, which are the only published conventions we know of for Twitter dependencies (for the FUDG dependency formalism). Issues include:

- @-mentions, which require different treatment when they are terms of address, versus nominal elements within a sentence.

- Hashtags, which in their tag-like usage are utterances by themselves (*#tweetliketheoppositegender Oh damn .*). or sometimes can be words with standard syntactic relations within the sentence (*#She's A Savage*, having *#She's* $\xleftarrow{\text{nsubj}}$ Savage). Both hashtag and @-mention ambiguities are handled by Owoputi et al. (2013)'s POS tagger.

- Multiple utterances, since we do not attempt sentence segmentation, and in many cases sentential utterances are not separated by explicit punctuation. FUDG allows for multiple roots for a text, but UD does not; instead we follow UD's convention of the *parataxis* relation for what they describe as "side-by-side run-on sentences."

- Emoticons and emoji, which we attach as *discourse* relations to the utterance root, following UD's treatment of interjections.

- Collapsed phrases, like *omw* for "on my way." When possible, we used the principle of annotating according to the root of the subtree of the original phrase. For example, UD 2.0 prescribes *way* $\xrightarrow{xcomp}$ *get* for the sentence *On my way to get...*; therefore we use *omw* $\xrightarrow{xcomp}$ *get* for *omw to get*.

- Separated words, like *uh round* for "around," which we analyze as multiword phrases (*flat* or *compound*).

We discuss details for these and other cases in the online appendix.

4 Experiments

4.1 Models

Our experiments use the following two parsers.

UDPipe (Straka et al., 2016) is a neural pipeline containing a tokenizer, morphological analyzer, tagger, and transition-based parser intended to be easily retrainable. The parser attains 80.2% LAS (labeled attachment score) on the UD English treebank with automatically generated POS tags, and was a baseline system used in the CoNLL 2017 Shared Task (Zeman et al., 2017).[6]

Deep Biaffine (Dozat et al., 2017; Dozat and Manning, 2016) is a graph-based parser incorporating neural attention and biaffine classifiers for arcs and labels. We used the version of the parser in the Stanford CoNLL 2017 Shared Task submission, which attained 82.2% LAS on the UD English treebank with automatically generated tags, and achieved the best performance in the task. The model requires pre-trained word embeddings. [7]

4.2 Experimental Setup

We considered a series of experiments within both a cross-domain scenario (§4.2.1), where we trained only on UD Treebank data, and an in-domain scenario (§4.2.2) using small amounts of our labeled data. We use the parsing systems' default hyperparameters (e.g. minibatch size and learning rate) and the default training/development split of the treebank (both systems perform early stopping based on development set performance).

[5]For example, Owoputi et al. (2013)'s Twitter word cluster 0011000 has 36 forms of *gonna* alone: http://www.cs.cmu.edu/~ark/TweetNLP/cluster_viewer.html

[6]https://github.com/ufal/udpipe

[7]https://github.com/tdozat/UnstableParser/

4.2.1 Cross-Domain Settings

Morpho-Tagger vs. ARK POS tags: The UD Treebank contains extensive fine-grained POS and morphological information, on which UDPipe's morphological analyzer and tagging system is originally trained. This rich information should be useful for parsing, but the analyzers may be highly error-prone on out-of-domain, dialectal Twitter data, and contribute to poor parsing performance. We hypothesize that higher quality, even if coarser, POS information should improve parsing.

To test this, we retrain UDPipe in two different settings. We first retrain the parser component with fine-grained PTB-style POS tags and morphological information provided by the tagger component;[8] we call this the *Morpho-Tagger* setting. Second, we retrain the parser with morphological information stripped and its tags predicted from the ARK Twitter POS tagger (Owoputi et al., 2013), which is both tailored for Twitter and displays a smaller AAE vs MAE performance gap than traditional taggers (Jørgensen et al., 2016); we call this the *ARK Tagger* setting.[9] The ARK Tagger's linguistic representation is impoverished compared to Morpho-Tagger: its coarse-grained POS tag system does not include tense or number information, for example.[10]

Synthetic Data: Given our knowledge of Twitter- and AAE-specific phenomena that do not occur in the UD Treebank, we implemented a rule-based method to help teach the machine-learned parser these phenomena; we generated synthetic data for three Internet-specific conventions and one set of AAE syntactic features. (This is inspired by Scherrer (2011)'s rule transforms between Standard and Swiss German.) We performed each of the following transformations separately on a copy of the UD Treebank data and concatenated the transformed files together for the final training and development files, so that each final file contained several transformed copies of the original UD Treebank data.

1. *@-mentions, emojis, emoticons, expressions, and hashtags:* For each sentence in the UD Treebank we inserted at least one @-mention, emoji, emoticon, expression (Internet-specific words and abbreviations such as *lol*, *kmsl*, and *xoxo*), or hashtag, annotated with the correct relation, at the beginning of the sentence. An item of the same type was repeated with 50% probability, and a second item was inserted with 50% probability. @-mentions were inserted using the *ATMENTION* token and emojis using the *EMOJI* token. Emoticons were inserted from a list of 20 common emoticons, expressions were inserted from a list of 16 common expressions, and hashtags were sampled for insertion according to their frequency in a list of all hashtags observed in the TwitterAAE corpus.

2. *Syntactically participating @-mentions:* To replicate occurrences of syntactically participating @-mentions, for each sentence in the UD Treebank with at least one token annotated with an *nsubj* or *obj* relation and an *NNP* POS tag, we replaced one at random with the *ATMENTION* token.

3. *Multiple utterances:* To replicate occurrences of multiple utterances, we randomly collapsed pairs of two short sentences ($<$ 15 tokens) together, attaching the root of the second to the root of the first with the *parataxis* relation.

4. *AAE preverbal markers and auxiliaries:* We introduced instances of verbal constructions present in AAE that are infrequent or non-existent in the UD Treebank data. First, constructions such as *going to*, *about to*, and *want to* are frequently collapsed to *gonna*, *bouta*, and *wanna*, respectively (see §3.2.2); for each sentence with at least one of these constructions, we randomly chose one to collapse. Second, we randomly replaced instances of *going to* with *finna*, a preverbal marker occurring in AAE and in the American South (Green, 2002). Third, we introduced the auxiliaries *gone* and *done*, which denote future tense and past tense, respectively; for the former, for each sentence containing at least one auxiliary *will*, we replace it with *gone*, and for the latter, for each sentence containing at least one non-auxiliary, non-passive, past-tense verb, we choose one and insert *done* before it. Finally, for each sentence containing at least one copula, we delete one at random.

Word Embeddings: Finally, since a tremendous variety of Twitter lexical items are not present in the UD Treebank, we use 200-dimensional word embeddings that we trained with *word2vec*[11] (Mikolov et al., 2013) on the

[8] We also retrained this component, to maintain consistency of training and development split. We also remove the universal (coarse) POS tags it produces, replacing them with the same PTB tags.

[9] We strip lemmas from training and development files for both settings.

[10] Derczynski et al. (2013)'s English Twitter tagger, which outputs PTB-style tags, may be of interest for future work.

[11] https://github.com/dav/word2vec

TwitterAAE corpus, which contains 60.8 million tweets. Before training, we processed the corpus by replacing @-mentions with ATMENTION, replacing emojis with EMOJI, and replacing sequences of more than two repeated letters with two repeated letters (e.g. *partyyyyy* $\rightarrow$ *partyy*). This resulted in embeddings for 487,450 words.

We retrain and compare UDPipe on each of the *Morpho-Tagger* and *ARK Tagger* settings with synthetic data and pre-trained embeddings, and without. We additionally retrain Deep Biaffine with and without synthetic data and embeddings.[12]

4.2.2 In-domain Training

We additionally investigate the effects of small amounts of in-domain training data from our dataset. We perform 2-fold cross-validation, randomly partitioning our dataset into two sets of 250 tweets. We compare two different settings (all using the UDPipe *ARK Tagger* setting):

Twitter-only: To explore the effect of training with Twitter data alone, for each set of 250 we trained on that set alone, along with our Twitter embeddings, and tested on the remaining 250.

UDT+Twitter: To explore the additional signal provided by the UD Treebank, for each set of 250 we trained on the UD Treebank concatenated with that set (with the tweets upweighted to approximately match the size of the UD Treebank, in order to use similar hyperparameters) and tested on the remaining 250.

5 Results and Analysis

In our evaluation, we ignored punctuation tokens (labeled with *punct*) in our LAS calculation.

5.1 Effects of Cross-Domain Settings

***Morpho-Tagger* vs. ARK Tagger:** As hypothesized, UDPipe's *ARK Tagger* setting outperformed the *Morpho-Tagger* across all settings, ranging from a 2.8% LAS improvement when trained only on the UD Treebank with no pre-trained word embeddings, to 4.7% and 5.4% improvements when trained with Twitter embeddings and both Twitter embeddings and synthetic data, respectively. The latter improvements suggest that the *ARK Tagger* setup is able to take better advantage of Twitter-specific lexical information from the embeddings and syntactic patterns from the synthetic data. Table 1 shows the LAS for our various settings.

[12]As the existing implementation of Deep Biaffine requires pre-trained word embeddings, for the Deep Biaffine baseline experiments we use the CoNLL 2017 Shared Task 100-dimensional embeddings that were pretrained on the English UD Treebank.

Model	LAS
(1) UDPipe, Morpho-Tagger, UDT	50.5
(2) + Twitter embeddings	53.9
(3) + synthetic, Twitter embeddings	58.9
(4) UDPipe, ARK Tagger, UDT	53.3
(5) + Twitter embeddings	58.6
(6) + synthetic, Twitter embeddings	64.3
Deep Biaffine, UDT	
(7) + CoNLL MAE embeddings	62.3
(8) + Twitter embeddings	63.7
(9) + synthetic, Twitter embeddings	65.0

Table 1: Results from cross-domain training settings (see §4.2.1).

Model	LAS
(10) UDPipe, Twitter embeddings	62.2
(11) + UDT	70.3

Table 2: Results from in-domain training settings (with the *ARK Tagger* setting, see §4.2.2).

After observing the better performance of the *ARK Tagger* setting, we opted not to retrain the Deep Biaffine parser in any *Morpho-Tagger* settings due to the model's significantly longer training time; all our Deep Biaffine results are reported for models trained with an *ARK Tagger* setting.

Synthetic data and embeddings: We observed that synthetic data and Twitter-trained embeddings were independently helpful; embeddings provided a 1.4–5.3% boost across the UDPipe and Deep Biaffine models, while synthetic data provided a 1.3–5.7% additional boost (Table 1).

UDPipe vs. Deep Biaffine: While the baseline models for UDPipe and Deep Biaffine are not directly comparable (since the latter required pre-trained embeddings), in the Twitter embeddings setting Deep Biaffine outperformed UDPipe by 5.1%. However, given access to both synthetic data and Twitter embeddings, UDPipe's performance approached that of Deep Biaffine.

5.2 Effects of In-Domain Training

Perhaps surprisingly, training with even limited amounts of in-domain training data aided in parsing performance; training with just in-domain data produced an LAS comparable to that of the baseline Deep Biaffine model, and adding UD Treebank data further increased LAS by 8.1%, indicat-

Model	AA LAS	WH LAS	Gap
(1) UDPipe, Morpho-Tagger	43.0	57.0	14.0
(2) + Twitter embeddings	45.5	61.2	15.7
(3) + synthetic, Twitter embeddings	50.7	66.2	15.5
(4) UDPipe, ARK Tagger	50.2	56.1	5.9
(5) + Twitter embeddings	54.1	62.5	8.4
(6) + synthetic, Twitter embeddings	59.9	68.1	8.2
Deep Biaffine, ARK Tagger			
(7) + CoNLL MAE embeddings	56.1	67.7	11.6
(8) + Twitter embeddings	58.7	66.7	8.0
(9) + synthetic, Twitter embeddings	59.9	70.8	10.9

Table 3: AA and WH tweets' labeled attachment scores for UD Treebank-trained models (see §5.3 for discussion); *Gap* is the WH − AA difference in LAS.

ing that they independently provide critical signal.

5.3 AAE/MAE Performance Disparity

For each model in each of the cross-domain settings, we calculated the LAS on the 250 tweets drawn from highly African-American tweets and the 250 from highly White tweets (see §3 for details); we will refer to these as the AA and WH tweets, respectively. We observed clear disparities in performance between the two sets of tweets, ranging from 5.9% to 15.7% (Table 3). Additionally, across settings, we observed several patterns.

First, the UDPipe *ARK Tagger* settings produced significantly smaller gaps (5.9–8.4%) than the corresponding *Morpho-Tagger* settings (14.0–15.7%). Indeed, most of the performance improvement of the *ARK Tagger* setting comes from the AA tweets; the LAS on the AA tweets jumps 7.2–9.2% from each *Morpho-Tagger* setting to the corresponding *ARK Tagger* setting, compared to differences of −0.9–1.9% for the WH tweets.

Second, the Deep Biaffine *ARK Tagger* settings produced larger gaps (8.0–11.6%) than the UDPipe *ARK Tagger* settings, with the exception of the embeddings-only setting.

Finally, we observed the surprising result that adding Twitter-trained embeddings and synthetic data, which contains both Twitter-specific and AAE-specific features, increases the performance gap across both UDPipe settings. We hypothesize that while UDPipe is able to effectively make use of both Twitter-specific lexical items and annotation conventions within MAE-like syntactic structures, it continues to be stymied by AAE-like syntactic structures, and is therefore unable to make use of the additional information.

We further calculated recall for each relation type across the AA tweets and WH tweets, and the resulting performance gap, under the UDPipe *Morpho-Tagger* and *ARK Tagger* models trained with synthetic data and embeddings. Table 4 shows these calculations for the 15 relation types for which the performance gap was highest and which had at least 15 instances in each of the AA and WH tweet sets, along with the corresponding calculation under the *ARK Tagger* model. The amount by which the performance gap is reduced from the first setting to the second setting is also reported. Of the 15 relations shown, the gap was reduced for 14, and 7 saw a reduction of at least 10%.

5.4 Lexical and Syntactic Analysis of AAE

In this section, we discuss AAE lexical and syntactic variations observed in our dataset, with the aim of providing insight into decreased AA parsing accuracy, and the impact of various parser settings on their parsing accuracy.

AAE contains a variety of phonological features which present themselves on Twitter through a number of lexical variations (Green, 2002; Jones, 2015), many of which are listed in §3.1, instances of which occur a total of 80 times in the AA tweets; notably, none occur in the WH tweets.

We investigated the accuracy of various cross-domain parser settings on these lexical variants; for each of the baseline *Morpho-Tagger*, baseline *ARK Tagger*, *ARK Tagger* with embeddings, and *ARK Tagger* with synthetic data and embeddings models, we counted the number of instances of lexical variants from §3.1 for which the model gave the correct head with the correct label.

While the lexical variants challenged all four models, switching from the *Morpho-Tagger* set-

	Morpho-Tagger			**ARK Tagger**			
Relation	**AA Recall**	**WH Recall**	**Gap (WH - AA)**	**AA Recall**	**WH Recall**	**Gap (WH - AA)**	**Reduction**
compound	36.4	71.2	34.8	42.4	72.9	30.5	4.4
obl:tmod	25.0	51.7	26.7	43.8	55.2	11.4	**15.3**
nmod	28.6	54.4	25.8	45.7	51.5	5.8	**20.1**
cop	56.5	82.1	25.6	65.2	79.1	13.9	**11.7**
obl	41.4	65.4	24.0	56.8	62.5	5.7	**18.3**
cc	56.9	79.0	22.1	78.5	82.7	4.3	**17.8**
ccomp	33.3	54.2	20.8	40.5	54.2	13.7	7.1
obj	61.3	81.5	20.2	72.8	83.5	10.7	9.5
case	60.5	79.8	19.3	75.2	83.4	8.2	**11.1**
det	73.1	90.7	17.5	83.4	92.2	8.8	8.7
advmod	53.8	71.2	17.3	62.9	72.1	9.1	8.2
advcl	31.5	46.8	15.3	25.9	46.8	20.9	-5.6
root	56.4	71.6	15.2	62.8	74.0	11.2	4.0
xcomp	40.0	54.9	14.9	51.2	50.0	1.2	**13.7**
discourse	30.7	44.9	14.2	46.0	51.4	5.4	8.8

Table 4: Recall by relation type under UDPipe's *Morpho-Tagger* and *ARK Tagger* settings (+synthetic+embeddings; (3) and (6) from Table 3; §5.3). *Reduction* is the reduction in performance gap from the *Morpho-Tagger* setting to the *ARK Tagger* setting; bolded numbers indicate a gap reduction of ≥ 10.0.

Feature	**AA Count**	**WH Count**	**Example**
Dropped copula	44	0	*MY **bestfrienddd mad** at me tho*
Habitual *be*, describing repeated actions	10	0	*fees **be** looking upside my head likee ion kno wat **be** goingg on .* *I kno that clown, u don't **be** around tho*
Dropped possessive marker	5	0	*ATMENTION on Tv...tawkn bout dat **man** gf* *Twink rude lol can't be calling ppl ugly that's **somebody** child lol...*
Dropped 3rd person singular	5	0	*When a female **owe** you sex you don't even wanna have a conversation with her*
Future *gone*	4	0	*she **gone** dance without da bands lol*
it is instead of *there is*	2	1	***It was** too much goin on in dat mofo .*
Completive *done*	1	0	*damnnn I **done** let alot of time pass by . .*

Table 5: Examples of AAE syntactic phenomena and occurrence counts in the 250 AA and 250 WH tweet sets.

ting to the *ARK Tagger* settings produced significant accuracy increases (Table 6). We observed that the greatest improvement came from using the *ARK Tagger* setting with Twitter-trained embeddings; the Twitter-specific lexical information provided by the embeddings was critical to recognizing the variants. Surprisingly, adding synthetic data decreased the model's ability to parse the variants.

We next investigated the presence of AAE syntactic phenomena in our dataset. Table 5 shows examples of seven well-documented AAE morphological and syntactic features and counts of their occurrences in our AA and WH tweet sets; again, while several of the phenomena, such as dropped copulas and habitual *be*, occur frequently in our AA tweets, there is only one instance of any of these features occurring in the WH tweet set.

We measured the parsing accuracy for the two most frequent syntactic features, dropped copulas and habitual *be*, across the four models; accuracies are given in Table 6. For dropped copulas, we measured parsing correctness by checking if the parser correctly attached the subject to the correct predicate word via the *nsubj* relation; for the first example in Table 5, for example, we considered the parser correct if it attached *bestfrienddd* to *mad* via the *nsubj* relation. For habitual *be*, we checked for correct attachment via the *aux* or *cop* relations as in the first and second examples in Ta-

AAE Feature	**Morpho-Tagger Baseline**	**ARK Tagger Baseline**	**ARK Tagger with Embeddings**	**ARK Tagger with Synthetic, Embeddings**
Lexical Variants (§3.1)	16.3 (13/80)	61.3 (49/80)	63.8 (51/80)	57.5 (46/80)
Dropped copula	54.5 (24/44)	70.5 (31/44)	61.4 (27/44)	68.2 (30/44)
Habitual *be*	50.0 (5/10)	80.0 (8/10)	90.0 (9/10)	90.0 (9/10)

Table 6: Parsing accuracies of syntactic and lexical variations across four UDPipe models (see §5.4).

ble 5, respectively.

As before, we observed significant increases in accuracy moving from the *Morpho-Tagger* to the *ARK Tagger* settings. However, neither adding embeddings nor synthetic data appeared to significantly increase accuracy for these features. From manual inspection, most of the dropped copulas errors appear to arise either from challenging questions (e.g. *ATMENTION what yo number ?*) or from mis-identification of the word to which to attach the subject (e.g. *He claim he in love llh*, where *he* was attached to *llh* rather than to *love*).

6 Conclusion

While current neural dependency parsers are highly accurate on MAE, our analyses suggest that AAE text presents considerable challenges due to lexical and syntactic features which diverge systematically from MAE. While the cross-domain strategies we presented can greatly increase accurate parsing of these features, narrowing the performance gap between AAE- and MAE-like tweets, much work remains to be done for accurate parsing of even linguistically well-documented features.

It remains an open question whether it is better to use a model with a smaller accuracy disparity (e.g. UDPipe), or a model with higher average accuracy, but a worse disparity (e.g. Deep Biaffine). The emerging literature on fairness in algorithms suggests interesting further challenges; for example, Kleinberg et al. (2017) and Corbett-Davies et al. (2017) argue that as various commonly applied notions of fairness are mutually incompatible, algorithm designers must grapple with such trade-offs. Regardless, the modeling decision should be made in light of the application of interest; for example, applications like opinion analysis and information retrieval may benefit from equal (and possibly weaker) performance between groups, so that concepts or opinions inferred from groups of authors (e.g. AAE speakers) are not under-counted or under-represented in results returned to a user or analyst.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by a Google Faculty Research Award, and a National Science Foundation Graduate Research Fellowship (No. 1451512).

References

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. *Proceedings of EMNLP*.

Su Lin Blodgett and Brendan O'Connor. 2017. Racial disparity in natural language processing: A case study of social media African-American English. *arXiv preprint arXiv:1707.00061; presented at Fairness, Accountability, and Transparency in Machine Learning (FAT/ML) workshop at KDD 2017*.

David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. 2006. Parsing Arabic dialects. In *Proceedings of EACL*.

Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the KDD*. ACM.

Joachim Daiber and Rob van der Goot. 2016. The denoised web treebank: Evaluating dependency parsing under noisy input conditions. In *Proceedings of LREC*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Recent Advances in Natural Language Processing*, pages 198–206.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *Proceedings of ICLR*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *HLT-NAACL*, pages 359–369.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. # hardtoparse: Pos tagging and parsing the twitterverse. In *AAAI 2011 Workshop on Analyzing Microtext*.

Rob van der Goot and Gertjan van Noord. 2017. Parser adaptation for social media by integrating normalization. In *Proceedings of ACL*.

Lisa Green, Toya A Wyatt, and Qiuana Lopez. 2007. Event arguments and 'be' in child African American English. *University of Pennsylvania Working Papers in Linguistics*, 13(2):8.

Lisa J Green. 2002. *African American English: A linguistic introduction*. Cambridge University Press.

Spence Green and Christopher D Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of COLING*. ACL.

Taylor Jones. 2015. Toward a description of African American Vernacular English dialect regions using "Black Twitter". *American Speech*, 90(4).

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2016. Learning a pos tagger for aave-like language. In *Proceedings of NAACL*. Association for Computational Linguistics.

David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. 2017. Incorporating dialectal variability for socially equitable language identification. In *Proceedings of ACL*.

Mohammad Khan, Markus Dickinson, and Sandra Kübler. 2013. Towards domain adaptation for parsing web data. In *RANLP*.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2017. Inherent trade-offs in the fair determination of risk scores. *Proceedings of ITCS*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar. Association for Computational Linguistics.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A Smith. 2018. Parsing tweets into universal dependencies. *Proceedings of NAACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Alexis Nasr, Geraldine Damnati, Aleksandra Guerraz, and Frederic Bechet. 2016. Syntactic parsing of chat language in contact center conversation corpus. In *Annual SIGdial Meeting on Discourse and Dialogue*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory search and topic summarization for Twitter. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

Manuela Sanguinetti, Cristina Bosco, Alessandro Mazzei, Alberto Lavelli, and Fabio Tamburini. 2017. Annotating Italian social media texts in universal dependencies. In *Proceedings of Depling 2017*.

Yves Scherrer. 2011. Syntactic transformations for Swiss German dialects. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*. ACL.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 51–60, Sofia, Bulgaria. Association for Computational Linguistics.

Ian Stewart. 2014. Now we stronger than ever: African-american english syntax in twitter. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 31–37, Gothenburg, Sweden. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Straková. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of LREC*.

Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal dependencies parsing for colloquial Singaporean english. *Proceedings of ACL*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of ACL*.

LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better

Adhiguna Kuncoro♠♣ **Chris Dyer**♠ **John Hale**♠♡
Dani Yogatama♠ **Stephen Clark**♠ **Phil Blunsom**♠♣
♠DeepMind, London, UK
♣Department of Computer Science, University of Oxford, UK
♡Department of Linguistics, Cornell University, NY, USA
`{akuncoro,cdyer,jthale,dyogatama,clarkstephen,pblunsom}@google.com`

Abstract

Language exhibits hierarchical structure, but recent work using a subject-verb agreement diagnostic argued that state-of-the-art language models, LSTMs, fail to learn long-range syntax-sensitive dependencies. Using the same diagnostic, we show that, in fact, LSTMs do succeed in learning such dependencies—provided they have enough capacity. We then explore whether models that have access to explicit syntactic information learn agreement more effectively, and how the way in which this structural information is incorporated into the model impacts performance. We find that the mere presence of syntactic information does not improve accuracy, but when model architecture is determined by syntax, number agreement is improved. Further, we find that the choice of *how* syntactic structure is built affects how well number agreement is learned: top-down construction outperforms left-corner and bottom-up variants in capturing long-distance structural dependencies.

1 Introduction

Recurrent neural networks (RNNs) are remarkably effective models of sequential data. Recent years have witnessed the widespread adoption of recurrent architectures such as LSTMs (Hochreiter and Schmidhuber, 1997) in various NLP tasks, with state of the art results in language modeling (Melis et al., 2018) and conditional generation tasks like machine translation (Bahdanau et al., 2015) and text summarization (See et al., 2017).

Here we revisit the question asked by Linzen et al. (2016): as RNNs model word sequences without explicit notions of hierarchical structure,

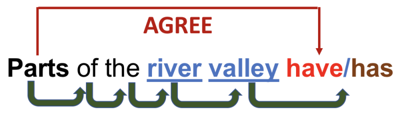


Figure 1: An example of the number agreement task with two attractors and a subject-verb distance of five.

to what extent are these models able to learn non-local syntactic dependencies in natural language? Identifying number agreement between subjects and verbs—especially in the presence of *attractors*—can be understood as a cognitively-motivated probe that seeks to distinguish hierarchical theories from sequential ones, as models that rely on sequential cues like the most recent noun would favor the incorrect verb form. We provide an example of this task in Fig. 1, where the plural form of the verb ***have*** agrees with the distant subject ***parts***, rather than the adjacent attractors (underlined) of the singular form.

Contrary to the findings of Linzen et al. (2016), our experiments suggest that sequential LSTMs are able to capture structural dependencies to a large extent, even for cases with multiple attractors (§2). Our finding suggests that network capacity plays a crucial role in capturing structural dependencies with multiple attractors. Nevertheless, we find that a strong character LSTM language model—which lacks explicit word representation and has to capture much longer sequential dependencies in order to learn non-local structural dependencies effectively—performs much worse in the number agreement task.

Given the strong performance of word-based LSTM language models, are there are any substantial benefits, in terms of number agreement accuracy, to explicitly modeling hierarchical structures as an inductive bias? We discover that a certain class of LSTM language models that explicitly models syntactic structures, the recurrent neural network grammars (Dyer et al., 2016, RNNGs), considerably outperforms sequential LSTM language models for cases with multiple attractors (§3). We present experiments affirming that this gain is due to an explicit *composition* operator rather than the presence of predicted syntactic annotations. Rather surprisingly, syntactic LSTM language models without explicit composition have no advantage over sequential LSTMs that operate on word sequences, although these models can nevertheless be excellent predictors of phrase structures (Choe and Charniak, 2016).

Having established the importance of modeling structures, we explore the hypothesis that *how* we build the structure affects the model's ability to identify structural dependencies in English. As RNNGs build phrase-structure trees through top-down operations, we propose extensions to the structure-building sequences and model architecture that enable left-corner (Henderson, 2003, 2004) and bottom-up (Chelba and Jelinek, 2000; Emami and Jelinek, 2005) generation orders (§4).

Extensive prior work has characterized top-down, left-corner, and bottom-up *parsing* strategies in terms of cognitive plausibility (Pulman, 1986; Abney and Johnson, 1991; Resnik, 1992) and neurophysiological evidence in human sentence processing (Nelson et al., 2017). Here we move away from the realm of parsing and evaluate the three strategies as models of *generation* instead, and address the following empirical question: which generation order is most appropriately biased to model structural dependencies in English, as indicated by number agreement accuracy? Our key finding is that the top-down generation outperforms left-corner and bottom-up variants for difficult cases with multiple attractors.

In theory, the three traversal strategies approximate the same chain rule that decompose the joint probability of words and phrase-structure trees, denoted as $p(\boldsymbol{x}, \boldsymbol{y})$, differently and as such will impose different biases on the learner. In §4.3, we show that the three variants achieve similar perplexities on a held-out validation set. As we observe different patterns in number agreement, this demonstrates that while perplexity can be a useful diagnostic tool, it may not be sensitive enough for comparing models in terms of how well they capture grammatical intuitions.

2 Number Agreement with LSTM Language Models

We revisit the number agreement task with LSTMs trained on language modeling objectives, as proposed by Linzen et al. (2016).

Experimental Settings. We use the same parsed Wikipedia corpus, verb inflectors, preprocessing steps, and dataset split as Linzen et al. (2016).[1] Word types beyond the most frequent 10,000 are converted to their respective POS tags. We summarize the corpus statistics of the dataset, along with the test set distribution of the number of attractors, in Table 1. Similar to Linzen et al. (2016), we only include test cases where *all* intervening nouns are of the opposite number forms than the subject noun. All models are implemented using the DyNet library (Neubig et al., 2017).

	Train	**Test**
Sentences	141,948	1,211,080
Types	10,025	10,025
Tokens	3,159,622	26,512,851

# Attractors	**# Instances**	**% Instances**
$n = 0$	1,146,330	94.7%
$n = 1$	52,599	4.3%
$n = 2$	9,380	0.77%
$n = 3$	2,051	0.17%
$n = 4$	561	0.05%
$n = 5$	159	0.01%

Table 1: Corpus statistics of the Linzen et al. (2016) number agreement dataset.

Training was done using a language modeling objective that predicts the next word given the prefix; at test time we compute agreement error rates by comparing the probability of the correct verb form with the incorrect one. We report performance of a few different LSTM hidden layer configurations, while other hyper-parameters are selected based on a grid search.[2] Following Linzen

[1]The dataset and scripts are obtained from `https://github.com/TalLinzen/rnn_agreement`.

[2]Based on the grid search results, we used the following hyper-parameters that work well across different hidden layer sizes: 1-layer LSTM, SGD optimizers with an initial learning rate of 0.2, a learning rate decay of 0.10 after 10 epochs, LSTM dropout rates of 0.2, an input embedding dimension of 50, and a batch size of 10 sentences. Our use of single-layer LSTMs and 50-dimensional word embedding (learned from scratch) as one of the baselines is consistent with the experimental settings of Linzen et al. (2016).

	n=0	n=1	n=2	n=3	n=4
Random	50.0	50.0	50.0	50.0	50.0
Majority	32.0	32.0	32.0	32.0	32.0
LSTM, H=50†	6.8	32.6	$\approx$50	$\approx$65	$\approx$70
Our LSTM, H=50	2.4	8.0	15.7	26.1	34.65
Our LSTM, H=150	1.5	4.5	9.0	14.3	17.6
Our LSTM, H=250	1.4	3.3	5.9	**9.7**	13.9
Our LSTM, H=350	**1.3**	**3.0**	**5.7**	**9.7**	**13.8**
1B Word LSTM (repl)	2.8	8.0	14.0	21.8	20.0
Char LSTM	**1.2**	5.5	11.8	20.4	27.8

Table 2: Number agreement error rates for various LSTM language models, broken down by the number of attractors. The top two rows represent the random and majority class baselines, while the next row (†) is the reported result from Linzen et al. (2016) for an LSTM language model with 50 hidden units (some entries, denoted by $\approx$, are approximately derived from a chart, since Linzen et al. (2016) did not provide a full table of results). We report results of our LSTM implementations of various hidden layer sizes, along with our re-run of the Jozefowicz et al. (2016) language model, in the next five rows. We lastly report the performance of a state of the art character LSTM baseline with a large model capacity (Melis et al., 2018).

et al. (2016), we include the results of our replication[3] of the large-scale language model of Jozefowicz et al. (2016) that was trained on the One Billion Word Benchmark.[4] Hyper-parameter tuning is based on validation set perplexity.

Discussion. Table 2 indicates that, given enough capacity, LSTM language models without explicit syntactic supervision are able to perform well in number agreement. For cases with multiple attractors, we observe that the LSTM language model with 50 hidden units trails behind its larger counterparts by a substantial margin despite comparable performance for zero attractor cases, suggesting that network capacity plays an especially important role in propagating relevant structural information across a large number of steps.[5] Our experiment independently derives the same finding as the recent work of Gulordava et al. (2018), who also find that LSTMs trained with language modeling objectives are able to learn number agreement well; here we additionally identify model capacity as one of the reasons for the discrepancy with the Linzen et al. (2016) results.

While the pretrained large-scale language model of Jozefowicz et al. (2016) has certain advantages in terms of model capacity, more training data, and richer vocabulary, we suspect that the poorer performance is due to differences between their training domain and the number agreement testing domain, although the model still performs reasonably well in the number agreement test set.

Prior work has confirmed the notion that, in many cases, statistical models are able to achieve good performance under some aggregate metric by overfitting to patterns that are predictive in *most* cases, often at the expense of more difficult, infrequent instances that require deeper language understanding abilities (Rimell et al., 2009; Jia and Liang, 2017). In the vast majority of cases, structural dependencies between subjects and verbs highly overlap with sequential dependencies (Table 1). Nevertheless, the fact that number agreement accuracy gets worse as the number of attractors increases is consistent with a *sequential recency* bias in LSTMs: under this conjecture, identifying the correct structural dependency becomes harder when there are more adjacent nouns of different number forms than the true subject.

If the sequential recency conjecture is correct, then LSTMs would perform worse when the structural dependency is more distant in the sequences, compared to cases where the structural dependency is more adjacent. We empirically test this conjecture by running a strong character-based LSTM language model of Melis et al. (2018) that achieved state of the art results on EnWiki8 from the Hutter Prize dataset (Hutter, 2012), with 1,800 hidden units and 10 million parameters. The character LSTM is trained, validated, and tested[6] on the same split of the Linzen et al. (2016) number agreement dataset.

A priori, we expect that number agreement is harder for character LSTMs for two reasons. First, character LSTMs lack explicit word representa-

[3]When evaluating the large-scale language model, the primary difference is that we do not map infrequent word types to their POS tags and that we subsample to obtain 500 test instances of each number of attractor due to computation cost; both preprocessing were also done by Linzen et al. (2016).

[4]The pretrained large-scale language model is obtained from `https://github.com/tensorflow/models/tree/master/research/lm_1b`.

[5]This trend is also observed by comparing results with H=150 and H=250. While both models achieve near-identical performance for zero attractor, the model with H=250 performs much better for cases with multiple attractors.

[6]For testing, we similarly evaluate number agreement accuracy by comparing the probability of the correct and incorrect verb form given the prefix, as represented by the respective character sequences.

tions, thus succeeding in this task requires identifying structural dependencies between two *sequences* of character tokens, while word-based LSTMs only need to resolve dependencies between *word tokens*. Second, by nature of modeling characters, non-local structural dependencies are sequentially further apart than in the word-based language model. On the other hand, character LSTMs have the ability to exploit and share informative morphological cues, such as the fact that plural nouns in English tend to end with '*s*'.

As demonstrated on the last row of Table 2, we find that the character LSTM language model performs much worse at number agreement with multiple attractors compared to its word-based counterparts. This finding is consistent with that of Sennrich (2017), who find that character-level decoders in neural machine translation perform worse than subword models in capturing morphosyntactic agreement. To some extent, our finding demonstrates the limitations that character LSTMs face in learning structure from language modeling objectives, despite earlier evidence that character LSTM language models are able to implicitly acquire a lexicon (Le Godais et al., 2017).

3 Number Agreement with RNNGs

Given the strong performance of sequential LSTMs in number agreement, is there any further benefit to explicitly modeling hierarchical structures? We focus on recurrent neural network grammars (Dyer et al., 2016, RNNGs), which jointly model the probability of phrase-structure trees and strings, $p(\boldsymbol{x}, \boldsymbol{y})$, through structure-building actions and explicit compositions for representing completed constituents.

Our choice of RNNGs is motivated by the findings of Kuncoro et al. (2017), who find evidence for syntactic headedness in RNNG phrasal representations. Intuitively, the ability to learn heads is beneficial for this task, as the representation for the noun phrase "*The* ***flowers*** *in the* *vase*" would be similar to the syntactic head ***flowers*** rather than *vase*. In some sense, the composition operator can be understood as injecting a *structural recency* bias into the model design, as subjects and verbs that are sequentially apart are encouraged to be close together in the RNNGs' representation.

3.1 Recurrent Neural Network Grammars

RNNGs (Dyer et al., 2016) are language models that estimate the *joint* probability of string terminals and phrase-structure tree nonterminals. Here we use stack-only RNNGs that achieve better perplexity and parsing performance (Kuncoro et al., 2017). Given the current stack configuration, the objective function of RNNGs is to predict the correct structure-building operation according to a top-down, left-to-right traversal of the phrase-structure tree; a partial traversal for the input sentence "*The flowers in the vase are blooming*" is illustrated in Fig. 3(a).[7]

The structural inductive bias of RNNGs derives from an explicit *composition* operator that represents completed constituents; for instance, the constituent (NP *The flowers*) is represented by a single composite element on the stack, rather than as four separate symbols. During each REDUCE action, the topmost stack elements that belong to the new constituent are popped from the stack and then composed by the composition function; the composed symbol is then pushed back into the stack. The model is trained in an end-to-end manner by minimizing the cross-entropy loss relative to a sample of gold trees.

3.2 Experiments

Here we summarize the experimental settings of running RNNGs on the number agreement dataset and discuss the empirical findings.

Experimental settings. We obtain phrase-structure trees for the Linzen et al. (2016) dataset using a publicly available discriminative model[8] trained on the Penn Treebank (Marcus et al., 1993). At training time, we use these predicted trees to derive action sequences on the training set, and train the RNNG model on these sequences.[9] At test time, we compare the probabilities of the correct and incorrect verb forms given the prefix, which now includes both nonterminal and terminal symbols. An example of the stack contents (i.e. the prefix) when predicting the verb is provided in Fig. 3(a). We similarly run a grid search over the same hyper-parameter range as the sequential

[7]For a complete example of action sequences, we refer the reader to the example provided by Dyer et al. (2016).

[8]`https://github.com/clab/rnng`

[9]Earlier work on RNNGs (Dyer et al., 2016; Kuncoro et al., 2017) train the model on gold phrase-structure trees on the Penn Treebank, while here we train the RNNG on the number agreement dataset based on predicted trees from another parser.

LSTM and compare the results with the strongest sequential LSTM baseline from §2.

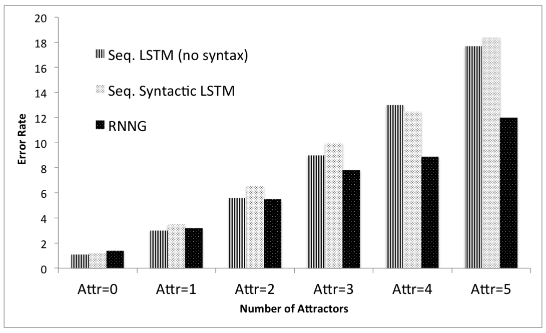


Figure 2: Number agreement error rates for sequential LSTM language models (left), sequential syntactic LSTM language models (Choe and Charniak, 2016, center), and RNNGs (right).

Discussion. Fig. 2 shows that RNNGs (rightmost) achieve much better number agreement accuracy compared to LSTM language models (leftmost) for difficult cases with four and five attractors, with around 30% error rate reductions, along with a 13% error rate reduction (from 9% to 7.8%) for three attractors. We attribute the slightly worse performance of RNNGs on cases with zero and one attractor to the presence of intervening structure-building actions that separate the subject and the verb, such as a REDUCE (step 6 in Fig. 3(a)) action to complete the noun phrase and at least one action to predict a verb phrase (step 15 in Fig. 3(a)) before the verb itself is introduced, while LSTM language models benefit from shorter dependencies for zero and one attractor cases.

The performance gain of RNNGs might arise from two potential causes. First, RNNGs have access to predicted syntactic annotations, while LSTM language models operate solely on word sequences. Second, RNNGs incorporate explicit compositions, which encourage hierarhical representations and potentially the discovery of syntactic (rather than sequential) dependencies.

Would LSTMs that have access to syntactic annotations, but without the explicit composition function, benefit from the same performance gain as RNNGs? To answer this question, we run sequential LSTMs over the same phrase-structure trees (Choe and Charniak, 2016), similarly estimating the joint probability of phrase-structure nonterminals and string terminals but without an explicit composition operator. Taking the example in Fig. 3(a), the sequential syntactic LSTM would have fifteen[10] symbols on the LSTM when predicting the verb, as opposed to three symbols in the case of RNNGs' stack LSTM. In theory, the sequential LSTM over the phrase-structure trees (Choe and Charniak, 2016) may be able to incorporate a similar, albeit implicit, composition process as RNNGs and consequently derive similarly syntactic heads, although there is no inductive bias that explicitly encourages such process.

Fig. 2 suggests that the sequential syntactic LSTMs (center) perform comparably with sequential LSTMs without syntax for multiple attractor cases, and worse than RNNGs for nearly all attractors; the gap is highest for multiple attractors. This result showcases the importance of an explicit composition operator and hierarchical representations in identifying structural dependencies, as indicated by number agreement accuracy. Our finding is consistent with the recent work of Yogatama et al. (2018), who find that introducing elements of hierarchical modeling through a stack-structured memory is beneficial for number agreement, outperforming LSTM language models and attention-augmented variants by increasing margins as the number of attractor grows.

3.3 Further Analysis

In order to better interpret the results, we conduct further analysis into the perplexities of each model, followed by a discussion on the effect of incrementality constraints on the RNNG when predicting number agreement.

Perplexity. To what extent does the success of RNNGs in the number agreement task with multiple attractors correlate with better performance under the perplexity metric? We answer this question by using an importance sampling marginalization procedure (Dyer et al., 2016) to obtain an estimate of $p(\boldsymbol{x})$ under both RNNGs and the sequential syntactic LSTM model. Following Dyer et al. (2016), for each sentence on the validation set we sample 100 candidate trees from a discriminative model[11] as our proposal distribution. As demonstrated in Table 3, the LSTM language model has the lowest validation set perplexity despite substantially worse performance than RNNGs in number agreement with multiple attractors, suggesting that lower perplexity is not neces-

[10]In the model of Choe and Charniak (2016), each nonterminal, terminal, and closed parenthesis symbol is represented as an element on the LSTM sequence.

[11]`https://github.com/clab/rnng`

sarily correlated with number agreement success.

	Validation ppl.
LSTM LM	**72.6**
Seq. Syntactic LSTM	79.2
RNNGs	77.9

Table 3: Validation set perplexity of LSTM language model, sequential syntactic LSTM, and RNNGs.

Incrementality constraints. As the syntactic prefix was derived from a discriminative model that has access to unprocessed words, one potential concern is that this prefix might violate the incrementality constraints and benefit the RNNG over the LSTM language model. To address this concern, we remark that the empirical evidence from Fig. 2 and Table 3 indicates that the LSTM language model *without* syntactic annotation outperforms the sequential LSTM *with* syntactic annotation in terms of both perplexity and number agreement throughout nearly all attractor settings, suggesting that the predicted syntactic prefix does not give any unfair advantage to the syntactic models.

Furthermore, we run an experiment where the syntactic prefix is instead derived from an incremental beam search procedure of Fried et al. (2017).[12] To this end, we take the highest scoring beam entry at the time that the verb is generated to be the syntactic prefix; this procedure is applied to both the correct and incorrect verb forms.[13] We then similarly compare the probabilities of the correct and incorrect verb form given each respective syntactic prefix to obtain number agreement accuracy. Our finding suggests that using the fully incremental tree prefix leads to even *better* RNNG number agreement performance for four and five attractors, achieving 7.1% and 8.2% error rates, respectively, compared to 9.4% and 12% for the RNNG error rates in Fig. 2.

4 Top-Down, Left-Corner, and Bottom-Up Traversals

In this section, we propose two new variants of RNNGs that construct trees using a different construction order than the top-down, left-to-right order used above. These are a bottom-up construction order (§4.1) and a left-corner construction order (§4.2), analogous to the well-known parsing strategies (e.g. Hale, 2014, chapter 3). They differ from these classic strategies insofar as they do not announce the phrase-structural content of an entire branch at the same time, adopting instead a node-by-node enumeration reminescent of Markov Grammars (Charniak, 1997). This step-by-step arrangement extends to the derived string as well; since all variants generate words from left to right, the models can be compared using number agreement as a diagnostic.[14]

Here we state our hypothesis on why the build order matters. The three generation strategies represent different chain rule decompositions of the joint probability of strings and phrase-structure trees, thereby imposing different biases on the learner. Earlier work in *parsing* has characterized the plausibility of top-down, left-corner, and bottom-up strategies as viable candidates of human sentence processing, especially in terms of memory constraints and human difficulties with center embedding constructions (Johnson-Laird, 1983; Pulman, 1986; Abney and Johnson, 1991; Resnik, 1992, *inter alia*), along with neurophysiological evidence in human sentence processing (Nelson et al., 2017). Here we cast the three strategies as models of language generation (Manning and Carpenter, 1997), and focus on the empirical question: which generation order has the most appropriate bias in modeling non-local structural dependencies in English?

These alternative orders organize the learning problem so as to yield intermediate states in generation that condition on different aspects of the grammatical structure. In number agreement, this amounts to making an agreement controller, such as the word *flowers* in Fig. 3, more or less salient. If it is more salient, the model should be better-able to inflect the main verb in agreement with this controller, without getting distracted by the attractors. The three proposed build orders are compared in Fig. 3, showing the respective configurations (i.e. the prefix) when generating the main verb in a sentence with a single attractor.[15] In ad-

[12]As the beam search procedure is time-consuming, we randomly sample 500 cases for each attractor and compute the number agreement accuracy on these samples.

[13]Consequently, the correct and incorrect forms of the sentence might have different partial trees, as the highest scoring beam entries may be different for each alternative.

[14]Only the order in which these models build the nonterminal symbols is different, while the terminal symbols are still generated in a left-to-right manner in all variants.

[15]Although the stack configuration at the time of verb generation varies only slightly, the configurations encountered

dition, we show concrete action sequences for a simpler sentence in each section.

4.1 Bottom-Up Traversal

In bottom-up traversals, phrases are recursively constructed and labeled with the nonterminal type once all their daughters have been built, as illustrated in Fig. 4. Bottom-up traversals benefit from shorter stack depths compared to top-down due to the lack of incomplete nonterminals. As the commitment to label the nonterminal type of a phrase is delayed until its constituents are complete, this means that the generation of a child node cannot condition on the label of its parent node.

In n-ary branching trees, bottom-up completion of constituents requires a procedure for determining how many of the most recent elements on the stack should be daughters of the node that is being constructed.[16] Conceptually, rather than having a single REDUCE operation as we have before, we have a complex REDUCE(X, n) operation that must determine the type of the constituent (i.e., X) as well as the number of daughters (i.e., n).

In step 5 of Fig. 4, the newly formed NP constituent only covers the terminal *worms*, and neither the unattached terminal *eats* nor the constituent (NP *The fox*) is part of the new noun phrase. We implement this extent decision using a stick-breaking construction—using the stack LSTM encoding, a single-layer feedforward network, and a logistic output layer—which decides whether the top element on the stack should be the leftmost child of the new constituent (i.e. whether or not the new constituent is complete after popping the current topmost stack element), as illustrated in Fig. 5. If not, the process is then repeated after the topmost stack element is popped. Once the extent of the new nonterminal has been decided, we parameterize the decision of the nonterminal label type; in Fig. 5 this is an NP. A second difference to top-down generation is that when a single constituent remains on the stack, the sentence is not necessarily complete (see step 3 of Fig. 4 for examples where this happens). We thus introduce an explicit STOP action (step 8, Fig. 4), indicating the tree is complete, which is only assigned non-zero probability when the stack has a single complete constituent.

during the history of the full generation process vary considerably in the invariances and the kinds of actions they predict.

[16]This mechanism is not necessary with strictly binary branching trees, since each new nonterminal always consists of the two children at the top of the stack.

	Avg. stack depth	**Ppl.**
TD	12.29	94.90
LC	11.45	95.86
BU	7.41	96.53

Table 4: Average stack depth and validation set perplexity for top-down (TD), left-corner (LC), and bottom-up (BU) RNNGs.

4.2 Left-Corner Traversal

Left-corner traversals combine some aspects of top-down and bottom-up processing. As illustrated in Fig. 6, this works by first generating the leftmost terminal of the tree, *The* (step 0), before proceeding bottom-up to predict its parent NP (step 1) and then top-down to predict the rest of its children (step 2). A REDUCE action similarly calls the composition operator once the phrase is complete (e.g. step 3). The complete constituent (NP *The fox*) is the leftmost child of its parent node, thus an NT_SW(S) action is done next (step 4).

The NT_SW(X) action is similar to the NT(X) from the top-down generator, in that it introduces an open nonterminal node and must be matched later by a corresponding REDUCE operation, but, in addition, swaps the two topmost elements at the top of the stack. This is necessary because the parent nonterminal node is not built until after its left-most child has been constructed. In step 1 of Fig. 6, with a single element *The* on the stack, the action NT_SW(NP) adds the open nonterminal symbol NP to become the topmost stack element, but after applying the swap operator the stack now contains (NP | *The* (step 2).

4.3 Experiments

We optimize the hyper-parameters of each RNNG variant using grid searches based on validation set perplexity. Table 4 summarizes average stack depths and perplexities[17] on the Linzen et al. (2016) validation set. We evaluate each of the variants in terms of number agreement accuracy as an evidence of its suitability to model structural dependencies in English, presented in Table 5. To account for randomness in training, we report the error rate summary statistics of ten different runs.

[17]Here we measure perplexity over $p(\boldsymbol{x}, \boldsymbol{y})$, where y is the presumptive gold tree on the Linzen et al. (2016) dataset. Dyer et al. (2016) instead used an importance sampling procedure to marginalize and obtain an estimate of $p(\boldsymbol{x})$.

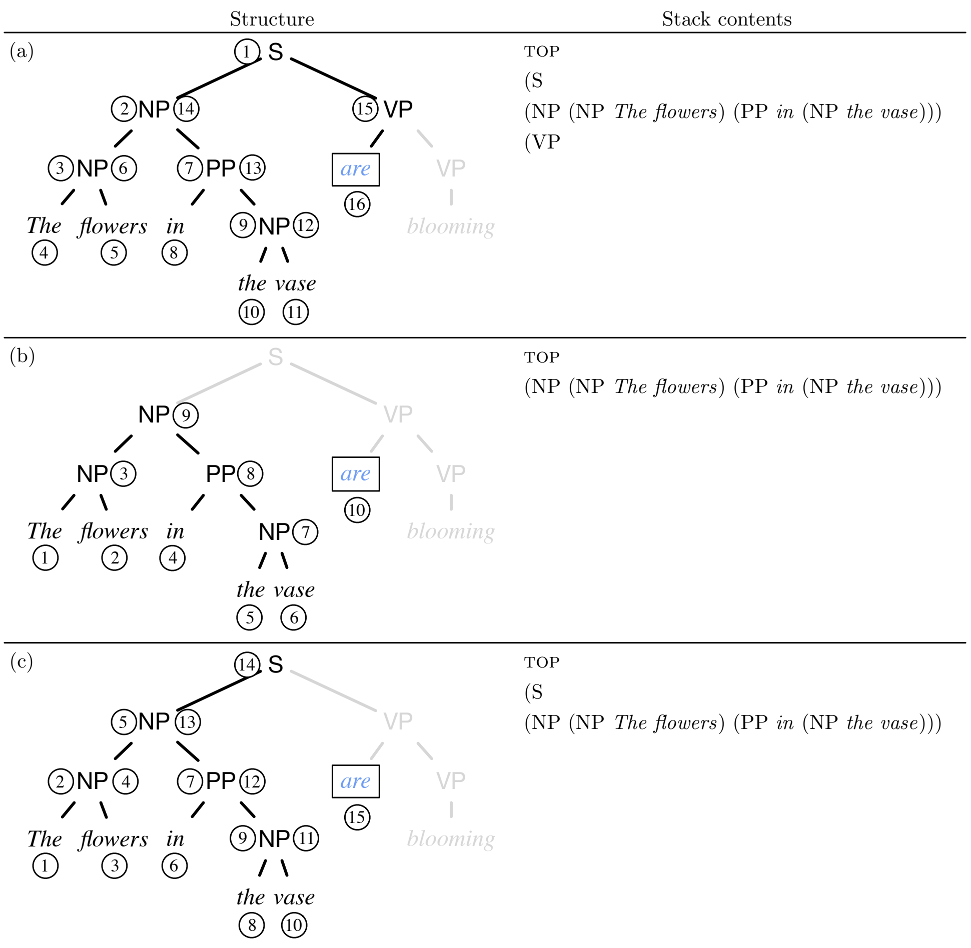

Structure
Stack contents
(a)
TOP
(S
(NP (NP The flowers) (PP in (NP the vase)))
(VP
(b)
TOP
(NP (NP The flowers) (PP in (NP the vase)))
(c)
TOP
(S
(NP (NP The flowers) (PP in (NP the vase)))
S
NP
VP
PP
The flowers in the vase are blooming

Figure 3: The (a) top-down, (b) bottom-up, and (c) left-corner build order variants showing in black the structure that exists as well as the generator's stack contents when the word *are* is generated during the derivation of the sentence *The flowers in the vase are blooming*. Structure in grey indicates material that will be generated subsequent to this. Circled numbers indicate the time when the corresponding structure/word is constructed. In (a) and (c), nonterminals are generated by a matched pair of NT and REDUCE operations, while in (b) they are introduced by a single complex REDUCE operation.

Input: *The fox eats worms*

	Stack	**Action**
0		GEN(*The*)
1	*The*	GEN(*fox*)
2	*The* | *fox*	REDUCE(NP,2)
3	(NP *The fox*)	GEN(*eats*)
4	(NP *The fox*) | *eats*	GEN(*worms*)
5	(NP *The fox*) | *eats* | *worms*	REDUCE(NP,1)
6	(NP *The fox*) | *eats* | (NP *worms*)	REDUCE(VP,2)
7	(NP *The fox*) | (VP *eats* (NP *worms*))	REDUCE(S,2)
8	(S (NP *The fox*) (VP *eats* (NP *worms*)))	STOP

Figure 4: Example Derivation for Bottom-Up Traversal. '|' indicates separate elements on the stack. The REDUCE(X, n) action takes the top n elements on the stack and creates a new constituent of type X with the composition function.

	Avg.(±sdev)/min/max		
	n=2	**n=3**	**n=4**
LM	5.8(±0.2)/5.5/6.0	9.6(±0.7)/8.8/10.1	14.1(±1.2)/13.0/15.3
TD	5.5(±0.4)/4.9/5.8	**7.8**(±0.6)/7.4/8.0	**8.9**(±1.1)/7.9/9.8
LC	**5.4**(±0.3)/5.2/5.5	8.2(±0.4)/7.9/8.7	9.9(±1.3)/8.8/11.5
BU	5.7(±0.3) 5.5/5.8	8.5(±0.7)/8.0/9.3	9.7(±1.1)/9.0/11.3

Table 5: Number agreement error rates for top-down (TD), left-corner (LC), and bottom-up (BU) RNNGs, broken down by the number of attractors. LM indicates the best sequential language model baseline (§2). We report the mean, standard deviation, and minimum/maximum of 10 different random seeds of each model.

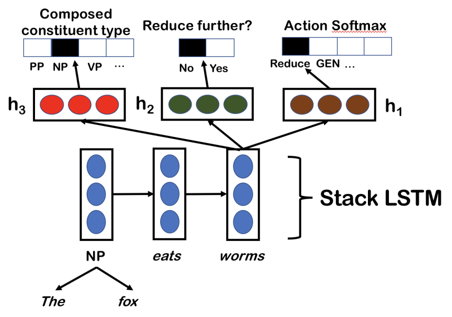


Figure 5: Architecture to determine type and span of new constituents during bottom-up generation.

Input: *The fox eats worms*

	Stack	Action
0		GEN(*The*)
1	*The*	NT_SW(NP)
2	(NP | *The*	GEN(*fox*)
3	(NP | *The* | *fox*	REDUCE
4	(NP *The fox*)	NT_SW(S)
5	(S | (NP *The fox*)	GEN(*eats*)
6	(S | (NP *The fox*) | *eats*	NT_SW(VP)
7	(S | (NP *The fox*) | (VP | *eats*	GEN(*worms*)
8	(S | (NP *The fox*) | (VP | *eats* | *worms*	NT_SW(NP)
9	(S | (NP *The fox*) | (VP | *eats* | (NP | *worms*	REDUCE
10	(S | (NP *The fox*) | (VP | *eats* | (NP *worms*)	REDUCE
11	(S | (NP *The fox*) | (VP *eats* (NP *worms*))	REDUCE
12	(S (NP *The fox*) (VP *eats* (NP *worms*)))	N/A

Figure 6: Example Derivation for left-corner traversal. Each NT_SW(X) action adds the open nonterminal symbol (X to the stack, followed by a deterministic **swap** operator that swaps the top two elements on the stack.

Discussion. In Table 5, we focus on empirical results for cases where the structural dependencies matter the most, corresponding to cases with two, three, and four attractors. All three RNNG variants outperform the sequential LSTM language model baseline for these cases. Nevertheless, the top-down variant outperforms both left-corner and bottom-up strategies for difficult cases with three or more attractors, suggesting that the top-down strategy is most appropriately biased to model difficult number agreement dependencies in English. We run an approximate randomization test by stratifying the output and permuting within each stratum (Yeh, 2000) and find that, for four attractors, the performance difference between the top-down RNNG and the other variants is statistically significant at $p < 0.05$.

The success of the top-down traversal in the domain of number-agreement prediction is consistent with a classical view in parsing that argues top-down parsing is the most human-like parsing strategy since it is the most *anticipatory*. Only anticipatory representations, it is said, could explain the rapid, incremental processing that humans seem to exhibit (Marslen-Wilson, 1973; Tanenhaus et al., 1995); this line of thinking similarly motivates Charniak (2010), among others. While most work in this domain has been concerned with the parsing problem, our findings suggest that anticipatory mechanisms are also beneficial in capturing structural dependencies in language modeling. We note that our results are achieved using models that, in theory, are able to condition on the entire derivation history, while earlier work in sentence processing has focused on cognitive memory considerations, such as the memory-bounded model of Schuler et al. (2010).

5 Conclusion

Given enough capacity, LSTMs trained on language modeling objectives are able to learn syntax-sensitive dependencies, as evidenced by accurate number agreement accuracy with multiple attractors. Despite this strong performance, we discover explicit modeling of structure does improve the model's ability to discover non-local structural dependencies when determining the distribution over subsequent word generation. Recurrent neural network grammars (RNNGs), which jointly model phrase-structure trees and strings and employ an explicit composition operator, substantially outperform LSTM language models and syntactic language models without explicit compositions; this highlights the importance of a hierarchical inductive bias in capturing structural dependencies. We explore the possibility that how the structure is built affects number agreement performance. Through novel extensions to RNNGs that enable the use of left-corner and bottom-up generation strategies, we discover that this is indeed the case: the three RNNG variants have different generalization properties for number agreement, with the top-down traversal strategy performing best for cases with multiple attractors.

Acknowledgments

We would like to thank Tal Linzen for his help in data preparation and answering various questions. We also thank Laura Rimell, Nando de Freitas, and the three anonymous reviewers for their helpful comments and suggestions.

References

Steven Abney and Mark Johnson. 1991. Memory requirements and local ambiguities for parsing strategies. *Journal of Psycholinguistic Research* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Eugene Charniak. 1997. Statistical techniques for natural language parsing. *AI Magazine* .

Eugene Charniak. 2010. Top-down nearly-context-sensitive parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 674–683. http://www.aclweb.org/anthology/D10-1066.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language* 14(4).

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proc. of EMNLP*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Ahmad Emami and Frederick Jelinek. 2005. A neural syntactic language model. *Machine Learning* 60:195–227.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada, pages 161–166.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proc. of NAACL*.

John T Hale. 2014. *Automaton theories of human sentence comprehension*. CSLI Publications.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. of NAACL*.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. of ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* .

Marcus Hutter. 2012. The human knowledge compression contest .

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proc. of EMNLP*.

Philip N. Johnson-Laird. 1983. *Mental Models*. Harvard University Press.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*.

Gaël Le Godais, Tal Linzen, and Emmanuel Dupoux. 2017. Comparing character-level neural language models using a lexical decision task. In *Proc. of EACL*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* .

Christopher D. Manning and Bob Carpenter. 1997. Probabilistic parsing using left corner language models. In *Proc. of IWPT*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* .

William Marslen-Wilson. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature* 244:522–523.

Gabor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *Proc. of ICLR*.

Matthew J. Nelson, Imen El Karoui, Kristof Giber, Xiaofang Yang, Laurent Cohen, Hilda Koopman, Sydney S. Cash, Lionel Naccache, John T. Hale, Christophe Pallier, and Stanislas Dehaene. 2017. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences of the United States of America* .

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Stephen Pulman. 1986. Grammars, parsers, and memory limitations. *Language and Cognitive Processes* .

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *Proc. of COLING*.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proc. of EMNLP*.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-like memory constraints 36(1):1–30.

Abigail See, Peter Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proc. of EACL*.

Michael Tanenhaus, Michael Spivey-Knowlton, Kathleen Eberhard, and Julie Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268:1632–1634.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proc. of COLING*.

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *Proc. of ICLR*.

Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures

Wenqiang Lei[‡][*], Xisen Jin[§][*], Zhaochun Ren[†], Xiangnan He[‡], Min-Yen Kan[‡], Dawei Yin[†]
‡National University of Singapore, Singapore
§Fudan University, Shanghai, China
†Data Science Lab, JD.com, Beijing, China
{wenqianglei,xisenjin}@gmail.com renzhaochun@jd.com
kanmy@comp.nus.edu.sg xiangnanhe@gmail.com yindawei@acm.org

Abstract

Existing solutions to task-oriented dialogue systems follow pipeline designs which introduce architectural complexity and fragility. We propose a novel, holistic, extendable framework based on a single sequence-to-sequence (seq2seq) model which can be optimized with supervised or reinforcement learning. A key contribution is that we design text spans named *belief spans* to track dialogue believes, allowing task-oriented dialogue systems to be modeled in a seq2seq way. Based on this, we propose a simplistic *Two Stage CopyNet* instantiation which demonstrates good scalability: significantly reducing model complexity in terms of number of parameters and training time by an order of magnitude. It significantly outperforms state-of-the-art pipeline-based methods on two datasets and retains a satisfactory entity match rate on out-of-vocabulary (OOV) cases where pipeline-designed competitors totally fail.

1 Introduction

The challenge of achieving both task completion and human-like response generation for task-oriented dialogue systems is gaining research interest. Wen et al. (2017b, 2016a, 2017a) pioneered a set of models to address this challenge. Their proposed architectures follow traditional pipeline designs, where the belief tracking component is the key component (Chen et al., 2017).

In the current paradigm, such a belief tracker builds a complex multi-class classifier for each slot (See §3.2) which can suffer from high complexity, especially when the number of slots and their values grow. Since all the possible slot values have to be pre-defined as classification labels, such trackers also cannot handle the requests that have out-of-vocabulary (OOV) slot values. Moreover, the belief tracker requires delexicalization, i.e., replacing slot values with their slot names in utterances (Mrkšić et al., 2017). It does not scale well, due to the lexical diversity. The belief tracker also needs to be pre-trained, making the models unrealistic for end-to-end training (Eric and Manning, 2017a). While Eric and Manning (2017a,b) investigated building task-oriented dialogue systems by using a seq2seq model, unfortunately, their methods are rather preliminary and do not perform well in either task completion or response generation, due to their omission of a belief tracker.

Questioning the basic pipeline architecture, in this paper, we re-examine the tenets of belief tracking in light of advances in deep learning. We introduce the concept of a *belief span* (bspan), a text span that tracks the belief states at each turn. This leads to a new framework, named *Sequicity*, with a single seq2seq model. Sequicity decomposes the task-oriented dialogue problem into the generation of bspans and machine responses, converting this problem into a sequence optimization problem. In practice, Sequicity decodes in two stages: in the first stage, it decodes a bspan to facilitate knowledge base (KB) search; in the second, it decodes a machine response on the condition of knowledge base search result and the bspan.

Our method represents a shift in perspective compared to existing work. Sequicity employs a single seq2seq model, resulting in a vastly simplified architecture. Unlike previous approaches with an overly parameterized delexicalization-based belief tracker, Sequicity achieves much less train-

[*] Work performed during an internship at Data Science Lab, JD.com.

ing time, better performance on larger a dataset and an exceptional ability to handle OOV cases. Furthermore, Sequicity is a theoretically and aesthetically appealing framework, as it achieves true end-to-end trainability using only one seq2seq model. As such, Sequicity leverages the rapid development of seq2seq models (Gehring et al., 2017; Vaswani et al., 2017; Yu et al., 2017) in developing solutions to task-oriented dialogue scenarios. In our implementation, we improve on CopyNet (Gu et al., 2016) to instantiate Sequicity framework in this paper, as key words present in bspans and machine responses recur from previous utterances. Extensive experiments conducted on two benchmark datasets verify the effectiveness of our proposed method.

Our contributions are fourfold: (1) We propose the *Sequicity* framework, which handles both task completion and response generation in a single seq2seq model; (2) We present an implementation of the Sequicity framework, called *Two Stage CopyNet* (*TSCP*), which has fewer number of parameters and trains faster than state-of-the-art baselines (Wen et al., 2017b, 2016a, 2017a); (3) We demonstrate that TSCP significantly outperforms state-of-the-art baselines on two large-scale datasets, inclusive of scenarios involving OOV; (4) We release source code of TSCP to assist the community to explore Sequicity[1].

2 Related Work

Historically, task-oriented dialog systems have been built as pipelines of separately trained modules. A typical pipeline design contains four components: 1) a user intent classifier, 2) a belief tracker, 3) a dialogue policy maker and a 4) response generator. User intent detectors classify user utterances to into one of the pre-defined intents. SVM, CNN and RNN models (Silva et al., 2011; Hashemi et al., 2016; Shi et al., 2016) perform well for intent classification. Belief trackers, which keep track of user goals and constraints every turn (Henderson et al., 2014a,b; Kim et al., 2017) are the most important component for task accomplishment. They model the probability distribution of values over each slot (Lee, 2013). Dialogue policy makers then generate the next available system action. Recent experiments suggest that reinforcement learning is a promising paradigm to accomplish this task (Young et al., 2013a; Cuayáhuitl et al., 2015; Liu and Lane, 2017), when state and action spaces are carefully designed (Young et al., 2010). Finally, in the response generation stage, pipeline designs usually pre-define fixed templates where placeholders are filled with slot values at runtime (Dhingra et al., 2017; Williams et al., 2017; Henderson et al., 2014b,a). However, this causes rather static responses that could lower user satisfaction. Generating a fluent, human-like response is considered a separate topic, typified by the topic of conversation systems (Li et al., 2015).

[1]http://github.com/WING-NUS/sequicity

3 Preliminaries

3.1 Encoder-Decoder Seq2seq Models

Current seq2seq models adopt encoder–decoder structures. Given a source sequence of tokens $X = x_1 x_2 ... x_n$, an encoder network represent X as hidden states: $\mathbf{H}^{(x)} = \mathbf{h}_1^{(x)}\mathbf{h}_2^{(x)}...\mathbf{h}_n^{(x)}$. Based on $\mathbf{H}^{(x)}$, a decoder network generates a target sequence of tokens $Y = y_1 y_2 ... y_m$ whose likelihood should be maximized given the training corpus.

As of late, the recurrent neural network with attention (Att-RNN) is now considered a baseline encoder–decoder architecture. Such networks employ two (sometimes identical) RNNs, one for encoding (i.e., generating $\mathbf{H}^{(x)}$) and another for decoding. Particularly, for decoding y_j, the decoder RNN takes the embedding $\mathbf{y}_{j-1}$ to generate a hidden vector $\mathbf{h}_j^{(y)}$. Afterwards, the decoder attends to X: calculating attention scores between all $\mathbf{h}_i^{(x)} \in \mathbf{H}^{(x)}$ and $\mathbf{h}_j^{(y)}$ (Eq. (1)), and then sums all $\mathbf{h}_i^{(x)}$, weighted by their corresponding attention scores (Eqs. (2)). The summed result $\tilde{\mathbf{h}}_j^{(x)}$ concatenates $\mathbf{h}_j^{(y)}$ as a single vector which is mapped into an output space for a $softmax$ operation (Eq. (3)) to decode the current token:

$$u_{ij} = \mathbf{v}^T tanh(\mathbf{W}_1 \mathbf{h}_i^{(x)} + \mathbf{W}_2 \mathbf{h}_j^{(y)}) \quad (1)$$

$$\tilde{\mathbf{h}}_j^{(x)} = \sum_{i=1}^{n} \frac{e^{u_{ij}}}{\sum_i e^{u_{ij}}} \mathbf{h}_i^{(x)} \quad (2)$$

$$y_j = softmax(\mathbf{O} \begin{bmatrix} \tilde{\mathbf{h}}_j^{(x)} \\ \mathbf{h}_j^{(y)} \end{bmatrix}) \quad (3)$$

where $\mathbf{v} \in \mathbb{R}^{1\times l}$; $\mathbf{W}_1$, $\mathbf{W}_2 \in \mathbb{R}^{l\times d}$ and $\mathbf{O} \in \mathbb{R}^{|V|\times d}$. d is embedding size and V is vocabulary set and $|V|$ is its size.

3.2 Belief Trackers

In the multi-turn scenarios, a belief tracker is the key component for task completion as it records key information from past turns (Wen et al., 2017b; Henderson et al., 2013, 2014a,b). Early belief trackers are designed as Bayesian networks where each node is a dialogue belief state (Paek and Horvitz, 2000; Young et al., 2013b). Recent work successfully represents belief trackers as discriminative classifiers (Henderson et al., 2013; Williams, 2012; Wen et al., 2017b).

Wen et al. (2017b) apply discrimination approaches (Henderson et al., 2013) to build one classifier for each slot in their belief tracker. Following the terminology of (Wen et al., 2017b), a slot can be either **informable** or **requestable**, which have been annotated in CamRes676 and KVRET. Individually, an informable slot, specified by user utterances in previous turns, is set to a constraint for knowledge base search; whereas a requestable slot records the user's need in the current dialogue. As an example of belief trackers in CamRes676, `food_type` is an informable slot, and a set of food types is also predefined (e.g., `Italian`) as corresponding slot values. In (Wen et al., 2017b), the informable slot `food_type` is recognized by a classifier, which takes user utterances as input to predict if and which type of food should be activated, while the requestable slot of `address` is a binary variable. `address` will be set to true if the slot is requested by the user.

4 Method

We now describe the Sequicity framework, by first explaining the core concept of bspans. We then instantiate the Sequicity framework with our introduction of an improved CopyNet (Gu et al., 2016).

4.1 Belief Spans for Belief Tracking

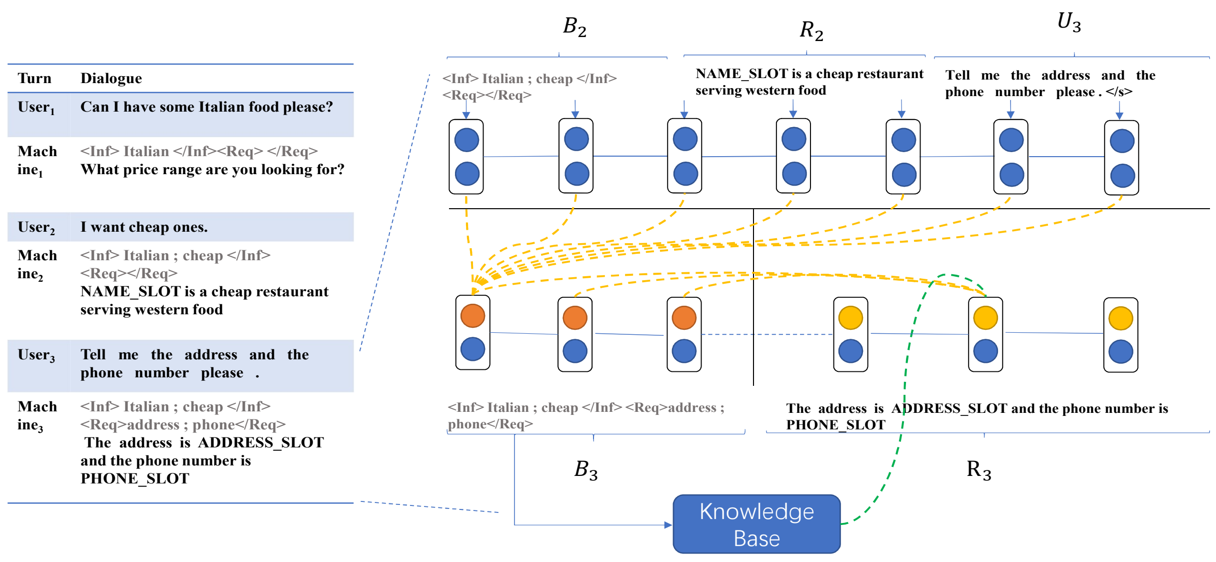
The core of belief tracking is keeping track of informable and requestable slot values when a dialogue progresses. In the era of pipeline-based methods, supervised classification is a straightforward solution. However, we observe that this traditional architecture can be updated by applying seq2seq models directly to the problem. In contrast to (Wen et al., 2017b) which treats slot values as classification labels, we record them in a text span, to be decoded by the model. This leverages the state-of-the-art neural seq2seq models to learn and dynamically generate them. Specifically, our bspan has an information field (marked with `<Inf></Inf>`) to store values of informable slots since only values are important for knowledge base search. Bspans can also feature a requested field (marked with `<Req></Req>`), storing requestable slot names if the corresponding value is `True`.

At turn t, given the user utterance U_t, we show an example of both bspan B_t and machine response R_t generation in Figure 1, where annotated slot values at each turn are decoded into bspans. B_1 contains an information slot `Italian` because the user stated "Italian food" in U_1. During the second turn, the user adds an additional constraint `cheap` resulting in two slot values in B_2's information field. In the third turn, the user further asks for the restaurant's phone and address, which are stored in requested slots of B_3.

Our bspan solution is concise: it simplifies multiple sophisticated classifiers with a single sequence model. Furthermore, it can be viewed as an explicit data structure that expedite knowledge base search as its format is fixed: following (Wen et al., 2017b), we use the informable slots values directly for matching fields of entries in databases.

4.2 The Sequicity Framework

We make a key observation that at turn t, a system only needs to refer to B_{t-1}, R_{t-1} and U_t to generate a new belief span B_t and machine response R_t, without appealing to knowing all past utterances. Such Markov assumption allows Sequicity to concatenate B_{t-1}, R_{t-1} and U_t (denoted as $B_{t-1}R_{t-1}U_t$) as a source sequence for seq2seq modeling, to generate B_t and R_t as target output sequences at each turn. More formally, we represent the dialogue utterances as $\{(B_0R_0U_1; B_1R_1); (B_1R_1U_2; B_2R_2); ...; (B_{t-1}R_{t-1}U_t; B_tR_t)\}$ where B_0 and R_0 are initialized as empty sequences. In this way, Sequicity fulfills both task accomplishment and response generation in an unified seq2seq model. Note that we process B_t and R_t separately, as the belief state B_t depends only on $B_{t-1}R_{t-1}U_t$, while the response R_t is additionally conditioned on B_t and the knowledge base search results (denoted as $\mathbf{k}_t$); that is, B_t informs the R_t's contents. For example, R_t must include all the request slots from B_t when communicating the entities fulfilling the requests found in the knowledge base. Here, $\mathbf{k}_t$ helps generate R_t pragmatically.



Turn	Dialogue
$User_1$	Can I have some Italian food please?
$Machine_1$	<Inf> Italian </Inf><Req> </Req> What price range are you looking for?
$User_2$	I want cheap ones.
$Machine_2$	<Inf> Italian ; cheap </Inf> <Req></Req> NAME_SLOT is a cheap restaurant serving western food
$User_3$	Tell me the address and the phone number please .
$Machine_3$	<Inf> Italian ; cheap </Inf> <Req>address ; phone</Req> The address is ADDRESS_SLOT and the phone number is PHONE_SLOT

B_2
R_2
U_3
<Inf> Italian ; cheap </Inf> <Req></Req>
NAME_SLOT is a cheap restaurant serving western food
Tell me the address and the phone number please . </s>
<Inf> Italian ; cheap </Inf> <Req>address ; phone</Req>
The address is ADDRESS_SLOT and the phone number is PHONE_SLOT
B_3
R_3
Knowledge Base

Figure 1: Sequicity overview. The left shows a sample dialogue; the right illustrates the Sequicity. B_t is employed only by the model, and not visible to users. During training, we substitute slot values with placeholders bearing the slot names for machine response. During testing, this is inverted: the placeholders are replaced by actual slot values, according to the item selected from the knowledge base.

Generally, $\mathbf{k}_t$ has three possibilities: 1) multiple matches, 2) exact match and 3) no match, while the machine responses differ accordingly. As an example, let's say a user requests an Italian restaurant. In the scenario of multiple matches, the system should prompt for additional constraints for disambiguation (such as restaurant price range). In the second exact match scenario where a single target (i.e., restaurant) has been found, the system should inform the user their requested information (e.g., restaurant address). If no entity is obtained, the system should inform the user and perhaps generate a cooperative response to retry a different constraint.

We thus formalize Sequicity as a seq2seq model which encodes $B_{t-1}R_{t-1}U_t$ jointly, but decodes B_t and R_t separately, in two serial stages. In the first stage, the seq2seq model decodes B_t unconditionally (Eq. 4a). Once B_t obtained, the decoding pauses to perform the requisite knowledge base search based on B_t, resulting in $\mathbf{k}_t$. Afterwards, the seq2seq model continues to the second decoding stage, where R_t is generated on the additional conditions of B_t and $\mathbf{k}_t$ (Eq. 4b).

$$B_t = \mathbf{seq2seq}(B_{t-1}R_{t-1}U_t|0,0) \tag{4a}$$
$$R_t = \mathbf{seq2seq}(B_{t-1}R_{t-1}U_t|B_t,\mathbf{k}_t) \tag{4b}$$

Sequicity is a general framework suitably implemented by any of the various seq2seq models. The additional modeling effort beyond a general seq2seq model is to add the conditioning on B_t and $\mathbf{k}_t$ to decode the machine response R_t. Fortunately, natural language generation with specific conditions has been extensively studied (Wen et al., 2016b; Karpathy and Fei-Fei, 2015; Mei et al., 2016) which can be employed within this framework.

4.3 Sequicity Instantiation: A Two Stage CopyNet

Although there are many possible instantiations, in this work we purposefully choose a simplistic architecture, leaving more sophisticated modeling for future work. We term our instantiated model a *Two Stage CopyNet* (*TSCP*). We denote the first m' tokens of target sequence Y are B_t and the rests are R_t, i.e. $B_t = y_1...y_{m'}$ and $R_t = y_{m'+1}...y_m$.

Two-Stage CopyNet. We choose to improve upon CopyNet (Gu et al., 2016) as our seq2seq model. This is a natural choice as we observe that target sequence generation often requires the copying of tokens from the input sequence. Let's discuss this in more detail. From a probabilistic point of view, the traditional encoder–decoder structure learns a language model. To decode y_j, we can employ a $softmax$ (e.g., Eq. 3) to calculate the probability distribution over V i.e., $P^g_j(v)$ where $v \in V$, and then choose the token with the highest generation probability. However, in our case, tokens in the target sequence Y might be exactly copied from the input X (e.g., "Italian"). These copied words need to be explicitly modeled. CopyNet (Gu et al., 2016) is a natural fit here, as it enlarges the decoding output space from V to $V \cup X$. For y_j, it considers an additional copy probability $P_j^c(v)$, indicating the likelihood of y_j copied from $v \in X$. Following (Gu et al., 2016), the simple summation of both probabilities $P_j(v) = P_j^g(v) + P_j^c(v), v \in V \cup X$ is treated as the final probability in the original paper.

In Sequicity, simply applying original CopyNet architecture is insufficient, since B_t and R_t have different distributions. We here employ two separate RNN (GRU in our implementation) in decoder: one for B_t and the other for R_t. In the first decoding stage, we have a copy-attention mechanism on X to decode B_t; then calculate the generation probability through attending to X as introduced in Sec 3.1, as well as the copy probability for each word $v \in X$ following (Gu et al., 2016) by Eq. 5:

$$P_j^c(v) = \frac{1}{Z} \sum_{i:x_i=v}^{|X|} e^{\psi(x_i)}, j \leqslant m' \tag{5}$$

where Z is a normalization term and $\psi(x_i)$ is the score of "copying" word x_i and is calculated by:

$$\psi(x_i) = \sigma({\mathbf{h}_i^{(x)}}^T \mathbf{W}_c)\mathbf{h}_j^{(y)}, j \leqslant m' \tag{6}$$

where $\mathbf{W}_c \in \mathbb{R}^{d \times d}$.

In the second decoding stage (i.e., decoding R_t), we apply the last hidden state of B_t as the initial hidden state of the R_t GRU. However, as we need to explicitly model the dependency on B_t, we have copy-attention mechanism on B_t instead of on X: treating all tokens of B_t as the candidate for copying and attention. Specifically, we use hidden state generated by B_t GRU, i.e., $\mathbf{h}_1^{(y)}, ..., \mathbf{h}_{m'}^{(y)}$, to calculate copying using Eqs. 7 and 8 and attention score as introduced in Sec 3.1. It helps to reduce search space because all key information of X for task completion has been included in B_t.

$$P_j^c(v) = \frac{1}{Z} \sum_{i:y_i=v} e^{\psi(y_i)}, i \leqslant m' < j \leqslant m \tag{7}$$

$$\psi(y_i) = \sigma({\mathbf{h}_i^{(y)}}^T \mathbf{W}_c)\mathbf{h}_j^{(y)}, i \leqslant m' < j \leqslant m \tag{8}$$

In contrast to recent work (Eric and Manning, 2017a) that also employs a copy-attention mechanism to generate a knowledge-base search API and machine responses, our proposed method advances in two aspects: on one hand, bspans reduce the search space from $U_1R_1...U_tR_t$ to $B_{t-1}R_{t-1}U_t$ by compressing key points for the task completion given past dialogues; on the other hand, because bspans revisit context by only handling the B_t with a fixed length, the time complexity of TSCP is only $O(T)$, comparing $O(T^2)$ in (Eric and Manning, 2017a).

Involving $\mathbf{k}_t$ when decoding R_t. As $\mathbf{k}_t$ has three possible values: obtaining only one, multiple or no entities. We let $\mathbf{k}_t$ be a vector of three dimensions, one of which signals a value. We append $\mathbf{k}_t$ to the embeddings $\mathbf{y}_j$, as shown in Eq. (9) that is fed into an GRU for generating $\mathbf{h}_{j+1}^{(y)}$. This approach is also referred to as Language Model Type condition (Wen et al., 2016b)

$$\mathbf{y}'_j = \begin{bmatrix} \mathbf{y}_j \\ \mathbf{k}_t \end{bmatrix}, j \in [m'+1, m] \tag{9}$$

4.4 Training

The standard cross entropy is adopted as our objective function to train a language model:

$$\sum_{j=1}^{m} y_j log P_j(y_j) \tag{10}$$

In response generation, every token is treated equally. However, in our case, tokens for task completion are more important. For example, when a user asks for the address of a restaurant, it matters more to decode the placeholder `<address>` than decode words for language fluency. We can employ reinforcement learning to fine tune the trained response decoder with an emphasis to decode those important tokens.

Inspired by (Wen et al., 2017a), in the context of reinforcement learning, the decoding network can be viewed as a policy network, denoted as $\pi_\Theta(y_j)$ for decoding y_j ($m'+1 \leqslant j \leqslant$ m). Accordingly, the choice of word y_j is an action and its hidden vector generated by decoding GRU is the corresponding state. In reinforcement tuning stage, the trained response decoder is the initial policy network. By defining a proper reward function $r^{(j)}$ for decoding y_j, we can update the trained

Dataset	Cam676		
Size	Train:408 / Test: 136 / Dev: 136		
Domains	restaurant reservation		
Slot types	price, food style etc.		
Distinct slot values	99		

Dataset	KVRET		
Size	Train:2425 / Test: 302 / Dev: 302		
Domains	calendar	weather info.	POI
Slot types	date, etc.	location, etc.	poi, etc.
Distinct slot values	79	65	140

Table 1: Dataset demographics. Following the respective literature, Cam676 is split 3:1:1 and KVRET is split 8:1:1, into training, developing and testing sets, respectively.

response model with policy gradient:

$$\frac{1}{m-m'} \sum_{j=m'+1}^{m} r^{(j)} \frac{\partial log\pi_{\Theta}(y_j)}{\partial\Theta} \quad (11)$$

where $r^{(j)} = r^{(j)} + \lambda r^{(j+1)} + \lambda^2 r^{(j+2)} + ... + \lambda^{m-j+1} r^{(m)}$. To encourage our generated response to answer the user requested information but avoid long-winded response, we set the reward at each step $r^{(j)}$ as follows: once the placeholder of requested slot has been decoded, the reward for current step is 1; otherwise, current step's reward is -0.1. λ is a decay parameter. Sec 5.2 for λ settings.

5 Experiments

We assess the effectiveness of Sequicity in three aspects: the task completion, the language quality, and the efficiency. The evaluation metrics are listed as follows:

· **BLEU** to evaluate the language quality (Papineni et al., 2002) of generated responses (hence top-1 candidate in (Wen et al., 2017b)).
· **Entity match rate** evaluates task completion. According to (Wen et al., 2017b), it determines if a system can generate all correct constraints to search the indicated entities of the user. This metric is either 0 or 1 for each dialogue.
· **Success F_1** evaluates task completion and is modified from the success rate in (Wen et al., 2017b, 2016a, 2017a). The original success rate measures if the system answered all the requested information (e.g. address, phone number). However, this metric only evaluates recall. A system can easily achieve a perfect task success by always responding all possible request slots. Instead, we here use success F_1 to balance both recall and precision. It is defined as the F_1 score of requested slots answered in the current dialogue.
· **Training time.** The training time is important for iteration cycle of a model in industry settings.

5.1 Datasets

We adopt the CamRest676 (Wen et al., 2017a) and KVRET (Eric and Manning, 2017b) datasets. Both datasets are created by a Wizard-of-Oz (Kelley, 1984) method on Amazon Mechanical Turk platform, where a pair of workers are recruited to carry out a fluent conversation to complete an assigned task (e.g. restaurant reservation). During conversation, both informable and requestable slots are recorded by workers.

CamRest676's dialogs are in the single domain of restaurant searching, while KVRET is broader, containing three domains: calendar scheduling, weather information retrieval and point of interest (POI) Navigation. Detailed slot information in each domain are shown in Table 1. We follow the data splits of the original papers as shown in 1.

5.2 Parameter Settings

For all models, the hidden size and the embedding size d is set to 50. $|V|$ is 800 for CamRes676 and 1400 for KVRET. We train our model with an Adam optimizer (Kingma and Ba, 2015), with a learning rate of 0.003 for supervised training and 0.0001 for reinforcement learning. Early stopping is performed on developing set. In reinforcement learning, the decay parameter λ is set to 0.8. We also use beam search strategy for decoding, with a beam size of 10.

5.3 Baselines and Comparisons

We first compare our model with the state-of-the-art baselines as follow:

- NDM (Wen et al., 2017b). As described in Sec 1, it adopts pipeline designs with a belief tracker component depending on delexicalization.

- NDM+Att+SS. Based on the NDM model, an additional attention mechanism is performed on the belief trackers and a snapshot learning mechanism (Wen et al., 2016a) is adopted.

- LIDM (Wen et al., 2017a). Also based on NDM, this model adopts neural variational inference with reinforcement learning.

	CamRes676					KVRET				
	Mat.	BLEU	Succ. F_1	$Time_{full}$	$Time_{N.B.}$	Mat.	BLEU	Succ. F_1	$Time_{full}$	$Time_{N.B.}$
(1) NDM	0.904	0.212	0.832	91.9 min	8.6 min	0.724	0.186	0.741	285.5 min	29.3 min
(2) NDM + Att + SS	0.904	0.240	0.836	93.7 min	10.4 min	0.724	0.188	0.745	289.7 min	33.5 min
(3) LIDM	0.912	0.246	0.840	97.7 min	14.4 min	0.721	0.173	0.762	312.8 min	56.6 min
(4) KVRN	N/A	0.134	N/A	21.4 min	–	0.459	0.184	0.540	46.9 min	–
(5) TSCP	**0.927**	**0.253**	**0.854**	7.3 min	–	**0.845**	**0.219**	**0.811**	25.5 min	–
(6) Att-RNN	0.851	0.248	0.774	7.2 min	–	0.805	0.208	0.801	23.0 min	–
(7) TSCP\\k_t	**0.927**	0.232	0.835	7.2 min	–	**0.845**	0.168	0.759	25.3 min	–
(8) TSCP\\RL	**0.927**	0.234	0.834	**4.1** min	–	**0.845**	0.191	0.774	**17.5** min	–
(9) TSCP\\B_t	0.888	0.197	0.809	22.9 min	–	0.628	0.182	0.755	42.7 min	–

Table 2: Model performance on CamRes676 and KVRET. This table is split into two parts: competitors on the upper side and our ablation study on the bottom side. **Mat.** and **Succ. F1** are for match rate and success F1 respectively. $\mathbf{Time_{full}}$ column reports training time till converge. For NDM, NDM+Att+SS and LIDM, we also calculate the training time for the rest parts except for the belief tracker ($\mathbf{Time_{N.B.}}$).

- KVRN (Eric and Manning, 2017b) uses one seq2seq model to generate response as well as interacting with knowledge base. However, it does not incorporate a belief tracking mechanism.

For NDM, NDM+Att+SS, LIDM, we run the source code released by the original authors[2]. For KVRN, we replicate it since there is no source code available. We also performed an ablation study to examine the effectiveness of each component.

- TSCP\\k_t. We removed the conditioning on k_t when decoding R_t.
- TSCP\\RL. We removed reinforcement learning which fine tunes the models for response generation.
- Att-RNN. The standard seq2seq baseline as described in the preliminary section (See §3.1).
- TSCP\\B_t. We removed bspans for dialogue state tracking. Instead, we adopt the method in (Eric and Manning, 2017a): concatenating all past utterance in a dialogue into a CopyNet to generate user information slots for knowledge base search as well as machine response.

5.4 Experimental Results

As shown in Table 2, TSCP outperforms all baselines (Row 5 vs. Rows 1–4) in task completion (entity match rate, success F1) and language quality (BLEU). The more significant performance of TSCP in KVRET dataset indicates the scalability of TSCP. It is because KVRET dataset has significant lexical variety, making it hard to perform delexicalization for Wen et al.'s model (Rows 1–3)[3]. However, CamRes676 is relatively small with simple patterns where all systems work well. As predicted, KVRN (Row 4) performs worse than TSCP (Row 5) due to lack of belief tracking.

Compared with Wen et al.'s models (Rows 1–3), TSCP takes a magnitude less time to train. Although TSCP is implemented in PyTorch while Wen et al.'s models in Theano, such speed comparison is still valid, as the rest of the NDM model — apart from its belief tracker — has a comparable training speed to TSCP (7.3 mins vs. 8.6 mins on CamRes676 and 25.5 mins vs. 29.3 mins on KVRET), where model complexities are similar. The bottleneck in the time expense is due to belief tracker training. In addition, Wen et al.'s models perform better at the cost of more training time (Rows 1, 2 and 3), suggesting the intrinsic complexity of pipeline designs.

Importantly, ablation studies validate the necessity of bspans. With bspans, even a standard seq2seq model (Att-RNN, Row 6) beats sophisticated models such as attention copyNets (TSCP\\$B_t$, Row 9) in KVRET. Furthermore, TSCP (Row 5) outperforms TSCP\\$B_t$ (Row 9) in all aspects: task completion, language quality and training speed. This validate our theoretical analysis in Sec 4.3. Other components of TSCP are also important. If we only use vanilla Attention-based RNN instead of copyNet, all metrics for model effectiveness decrease, validating our hypothesize that the copied words need to be specifically modeled. Secondly, BLEU score is sensitive to knowl-

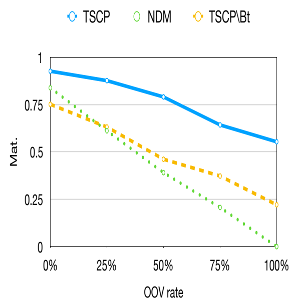
[2]https://github.com/shawnwun/NNDIAL

[3]We use the delexicalization lexicon provided by the original author of KVRET(Eric and Manning, 2017b)

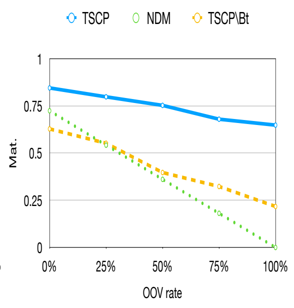
edge base search result $\mathbf{k}_t$ (Row 7 vs. Row 5). By examining error cases, we find that the system is likely to generate common sentences like "you are welcome" regardless of context, due to corpus frequency. Finally, reinforcement learning effectively helps both BLEU and success F_1 although it takes acceptable additional time for training.

5.5 OOV Tests

Previous work predefines all slot values in a belief tracker. However, a user may request new attributes that has not been predefined as a classification label, which results in an *entity mismatch*. TSCP employs copy mechanisms, gaining an intrinsic potential to handle OOV cases. To conduct the OOV test, we synthesize OOV test instances by adding a suffix `unk` to existing slot fillers. For example, we change "I would like Chinese food" into "I would like Chinese_unk food." We then randomly make a proportion of testing data OOV and measure its entity match rate. For simplicity, we only show the three most representative models pre-trained in the in-vocabulary data: TSCP, TSCP$\backslash B_t$ and NDM.



(a) CamRes676



(b) KVRET

Figure 2: OOV tests. 0% OOV rate means no OOV instance while 100% OOV rate means all instances are changed to be OOV.

Compared with NDM, TSCP still performs well when all slot fillers are unknown. This is because TSCP actually learns sentence patterns. For example, CamRes676 dataset contains a frequent pattern "I would like `[food_type]` food" where the `[food_type]` should be copied in B_t regardless what exact word it is. In addition, the performance of TSCP$\backslash B_t$ decreases more sharply than TSCP as more instances set to be OOV. This might be because handling OOV cases is much harder when search space is large.

5.6 Empirical Model Complexity

Traditional belief trackers like (Wen et al., 2017b) are built as a multi-class classifier, which models each individual slot and its corresponding values, introducing considerable model complexities. This is especially severe in large datasets with a number of slots and values. In contrast, Sequicity reduces such a complex classifier to a language model. To compare the model complexities of two approaches, we empirically measure model size. We split KVRET dataset by their domains, resulting in three sub-datasets. We then accumulatively add the sub-datasets into training set to examine how the model size grows. We here selectively present TSCP, NDM and its separately trained belief tracker, since Wen et al.'s set of models share similar model sizes.

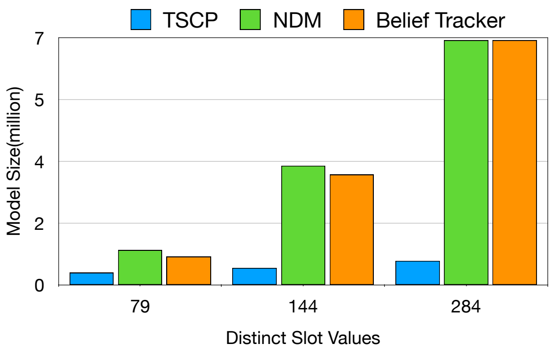


Figure 3: Model size sensitivity with respect to KVRET. Distinct slot values of 79, 144, 284 correspond to the number of slots in KVRET's *calendar*, *calendar + weather info.*, and all 3 domains.

As shown in Figure 3, TSCP has a magnitude less number of parameters than NDM and its model size is much less sensitive to distinct slot values increasing. It is because TSCP is a seq2seq language model which has a approximate linear complexity to vocabulary size. However, NDM employs a belief tracker which dominates its model size. The belief tracker is sensitive to the increase of distinct slot values because it employs complex structures to model each slot and corresponding values. Here, we only perform empirical evaluation, leaving theoretically complexity analysis for future works.

5.7 Discussions

In this section we discuss if Sequicity can tackle *inconsistent user requests* , which happens when users change their minds during a dialogue. Inconsistent user requests happen frequently and are difficult to tackle in belief tracking (Williams, 2012; Williams et al., 2013). Unlike most of previous pipeline-based work that explicitly defines model actions for each situation, Sequicity is proposed to directly handle various situations from the training data with less manual intervention. Here, given examples about restaurant reservation, we provide three different scenarios to discuss:

- **A user totally changes his mind**. For example, the user request a Japanese restaurant first and says "I dont want Japanese food anymore, I'd like French now." Then, all the slot activated before should be invalid now. The slot annotated for this turn is only *French*. Sequicity can learn this pattern, as long as it is annotated in the training set.

- **User requests cannot be found in the KB** (e.g., *Japanese food*). Then the system should respond like "Sorry, there is no Japanese food...". Consequently, the user can choose a different option: "OK, then French food." The activated slot *Japanese* will be replaced as *French*, which our system can learn. Therefore, an important pattern is the machine-response (e.g., "there is no [XXX constraint]") in the immediate previous utterance.

- **Other cases.** Sequicity is expected to generate both slot values in a belief span if it doesn't know which slot to replace. To maintain the belief span, we run a simple post-processing script at each turn, which detects whether two slot values have the same slot name (e.g., `food_type`) in a pre-defined slot name-value table. Then, such script only keeps the slot value in the current turn of user utterance. Given this script, Sequicity can accurately discover the slot requested by a user in each utterance. However, this script only works when slot values are pre-defined. For inconsistent OOV requests, we need to build another classifier to recognize slot names for slot values.

To sum up, Sequicity, as a framework, is able to handle various inconsistent user input despite its simple design. However, detailed implementations should be customized depends on different applications.

6 Conclusion

We propose Sequicity, an extendable framework, which tracks dialogue believes through the decoding of novel text spans: belief spans. Such belief spans enable a task-oriented dialogue system to be holistically optimized in a single seq2seq model. One simplistic instantiation of Sequicity, called Two Stage CopyNet (TSCP), demonstrates better effectiveness and scalability of Sequicity. Experiments show that TSCP outperforms the state-of-the-art baselines in both task accomplishment and language quality. Moreover, our TSCP implementation also betters traditional pipeline architectures by a magnitude in training time and adds the capability of handling OOV. Such properties are important for real-world customer service dialog systems where users' inputs vary frequently and models need to be updated frequently. For our future work, we will consider advanced instantiations for Sequicity, and extend Sequicity to handle unsupervised cases where information and requested slots values are not annotated.

Acknowledgments

We would like to thank the anonymous reviewers for their detailed comments and suggestions for this paper. This work is also supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@SG Funding Initiative.

References

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *arXiv preprint arXiv:1711.01731* .

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099* .

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 484–495.

Mihail Eric and Christopher D Manning. 2017a. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue.

Mihail Eric and Christopher D Manning. 2017b. Key-value retrieval networks for task-oriented dialogue. *SIGDIAL* .

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2017. A convolutional encoder model for neural machine translation. *ACL* .

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *ACL* .

Homa B Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. In *International Conference on Web Search and Data Mining, Workshop on Query Understanding*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *SIGDIAL Conference*. pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 324–329.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 467–471.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3128–3137.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)* 2(1):26–41.

Seokhwan Kim, Luis Fernando DHaro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2017. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, Springer, pages 435–449.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *SIGDIAL Conference*. pages 442–451.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *arXiv preprint arXiv:1709.06136* .

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. *ACL* .

Tim Paek and Eric Horvitz. 2000. Conversation as action under uncertainty. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pages 455–464.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1501–1511.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review* 35(2):137–154.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016a. Conditional generation and snapshot learning in neural dialogue systems. *EMNLP* .

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016b. Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*. ACL, Austin, Texas, pages 2153–2162. https://aclweb.org/anthology/D16-1233.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017a. Latent intention dialogue models. *ICML* .

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017b. A network-based end-to-end trainable task-oriented dialogue system. *EACL* .

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 404–413.

Jason D Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on future directions and needs in the spoken dialog community: tools and data*. Association for Computational Linguistics, pages 23–24.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* .

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language* 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013a. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013b. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*. pages 2852–2858.

An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking

Puyang Xu **Qi Hu**[†]
Mobvoi AI Lab, Redmond, WA
[†]University of Washington, Seattle, WA
{puyangxu, qihuchn}@gmail.com

Abstract

We highlight a practical yet rarely discussed problem in dialogue state tracking (DST), namely handling unknown slot values. Previous approaches generally assume predefined candidate lists and thus are not designed to output unknown values, especially when the spoken language understanding (SLU) module is absent as in many end-to-end (E2E) systems. We describe in this paper an E2E architecture based on the pointer network (PtrNet) that can effectively extract unknown slot values while still obtains state-of-the-art accuracy on the standard DSTC2 benchmark. We also provide extensive empirical evidence to show that tracking unknown values can be challenging and our approach can bring significant improvement with the help of an effective feature dropout technique.

1 Introduction

A dialogue state tracker is a core component in most of today's spoken dialogue systems (SDS). The goal of dialogue state tracking (DST) is to monitor the user's intentional states during the course of the conversation, and provide a compact representation, often called the dialogue states, for the dialogue manager (DM) to decide the next action to take.

In task-oriented dialogues, or *slot-filling* dialogues in the simplistic form, the dialogue agent is tasked with helping the user achieve simple goals such as finding a restaurant or booking a train ticket. As the name itself suggests, a slot-filling dialogue is composed of a predefined set of slots that need to be filled through the conversation. The dialogue states in this case are therefore the values of these slot variables, which are essentially the search constraints the DM has to maintain in order to perform the database lookup.

Traditionally in the research community, as exemplified in the dialogue state tracking challenge (DSTC) (Williams et al., 2013), which has become a standard evaluation framework for DST research, the dialogues are usually constrained by a fixed *domain ontology*, which essentially describes in detail all the possible values that each predefined slot can take. Having access to such an ontology can simplify the tracking problem in many ways, however, in many of the SDS applications we have built in the industry, such an ontology was not obtainable. Oftentimes, the backend databases are only exposed through an external API, which is owned and maintained by our partners. It is usually not possible to gain access to their data or enumerate all possible slot values in their knowledge base. Even if such lists or dictionaries exist, they can be very large in size and highly dynamic (e.g. new songs added, new restaurants opened etc.). It is therefore not amiable to many of the previously introduced DST approaches, which generally rely on classification over a fixed ontology or scoring each slot value pairs separately by enumerating the candidate list.

In this paper, we will therefore focus on this particular aspect of the DST problem which has rarely been discussed in the community – namely how to perform state tracking in the absence of a comprehensive domain ontology and how to handle unknown slot values effectively.

It is worth noting that end-to-end (E2E) modeling for task-oriented dialogue systems has become a popular trend (Williams and Zweig, 2016; Zhao and Eskenazi, 2016; Li et al., 2017; Liu et al.,

[0]The first author is now with Facebook. Qi contributed to the work during an internship at Mobvoi.

2017; Wen et al., 2017), although most of them focus on E2E policy learning and language generation, and still rely on explicit dialogue states in their models. While fully E2E approaches which completely obviate explicit DST have been attempted (Bordes and Weston, 2016; Eric and Manning, 2017a,b; Dhingra et al., 2017), their generality and scalability in real world applications remains to be seen. In reality, a dedicated DST component remains a central piece to most dialogue systems, even in most of the proclaimed E2E models.

E2E approaches for DST, i.e. joint modeling of SLU and DST has also been presented in the literature (Henderson et al., 2014b,c; Mrksic et al., 2015; Zilka and Jurcicek, 2015; Perez and Liu, 2017; Mrksic et al., 2017). In these methods, the conventional practice of having a separate spoken language understanding (SLU) module is replaced by various E2E architectures that couple SLU and DST altogether. They are sometimes called word based state tracking as the dialogue states are derived directly from word sequences as opposed to SLU outputs. In the absence of SLU to generate value candidates, most E2E trackers today can only operate with fixed value sets. To address this limitation, we introduce an E2E tracker that allows us to effectively handle unknown value sets. The proposed solution is based on the recently introduced pointer network (PtrNet) (Vinyals et al., 2015), which essentially performs state tracking in an extractive fashion similar to the sequence labeling techniques commonly utilized for slot tagging in SLU (Tur and Mori, 2011).

Our proposed technique is similar in spirit as the recent work in (Rastogi et al., 2018), which also targets the problem of unbounded and dynamic value sets. They introduce a sophisticated candidate generation strategy followed by a neural network based scoring mechanism for each candidate. Despite the similarity in the motivation, their system relies on SLU to generate value candidates, resulting in an extra module to maintain and potential error propagation as commonly faced by pipelined systems.

The contributions of this paper are three-folds: Firstly, we target a very practical yet rarely investigated problem in DST, namely handling unknown slot values in the absence of a predefined ontology. Secondly, we describe a novel E2E architecture without SLU based on the PtrNet to perform state tracking. Thirdly, we also introduce an effective *dropout* technique for training the proposed model which drastically improves the recall rate of unknown slot values.

The rest of the paper is structured as follows: We give a brief review of related work in the field in Section 2 and point out its limitations. The PtrNet and its proposed application in DST are described in Section 3. In Section 4, we demonstrate some caveats regarding the use of PtrNet and propose an additional classification module as a complementary component. The targeted dropout technique, which can be essential for generalization on some datasets, is described in Section 5. Experimental setup and results are presented in Section 6, followed by conclusions in Section 7.

2 Dialogue State Tracking

In DSTC tasks, the dialogue states are defined as a set of search constraints (i.e. informable slots or goals) the user specified through the dialogue and a set of attribute questions regarding the search results (i.e. requestable slots or requests). The DST component is expected to track the values of the aforementioned slots taking into account the current user utterance as well as the entire dialogue context. As mentioned in the previous section, the values each slot variable can take are specified beforehand through an ontology. This is a hidden assumption that previous techniques usually rely upon implicitly and also what motivates our work in this paper.

Discriminative DST While generative models aiming at modeling the joint distribution of dialogue states and miscellaneous evidences have been a popular modeling choice for DST for many years, the scalability issue resulting from large state spaces has limited the broader application of this family of models, despite the success of various approximation techniques.

The discriminative methods, on the other hand, directly model the posterior distribution of dialogue states given the evidences accumulated through the conversation history. Models such as maximum entropy (Metallinou et al., 2013) and particularly the more recent deep learning based models (Henderson et al., 2014b,c; Zilka and Jurcicek, 2015; Mrksic et al., 2015, 2017; Perez and Liu, 2017) have demonstrated state-of-the-art results on public benchmarks. Such techniques often involve a multi-class classification step at the end (e.g. in the form of a softmax layer) which for each slot predicts the corresponding value based on the dialogue history. Sometimes the multi-class classification is replaced by a binary prediction that decides whether a particular slot value pair was expressed by the user, and the list of candidates comes from either a fixed ontology or the SLU output.

E2E DST Previous work has also investigated joint modeling strategies merging SLU and DST altogether. In this line of work, the SLU module is removed from the standard SDS architecture, resulting in reduced development cost and alleviating the error propagation problem commonly affecting cascaded systems.

In the absence of SLU providing fine-grained semantic features, the E2E approaches these days typically rely on variants of neural networks such as recurrent neural networks (RNN) or memory networks (Weston et al., 2014) to automatically learn features from the raw dialogue history. The deep learning based techniques cited in the previous subsection generally fall into this category.

Current Limitations In short, most of the previous DST approaches, particularly E2E ones, are not designed to handle slot values that are not known to the tracker.

As we have described in the introduction, the assumption that a predefined ontology exists for the dialogue and one can enumerate all possible values for each slot is often not valid in real world scenarios. Such an assumption has implicitly influenced many design choices of previous systems. The methods based on classification or scoring each slot value pair separately can be very difficult to apply when the set of slot values is not enumerable, either due to its size or its constantly changing nature, especially in E2E models where there is no SLU module to generate an enumerable candidate list for the tracker.

It is important to point out the difference between *unseen* states and *unknown* states, as previous work has tried to address the problem of unseen slot values, i.e. values that were not observed during training. E2E approaches in particular, frequently employ a featurization strategy called *delexicalization*, which replaces slots and values mentioned in the dialogue text with generic labels. Such a conversion allows the models to generalize much better to new values that are infrequent or unseen in the training data. However, such slot values are still expected to be known to the tracker, either through a predefined value set or provided by SLU, otherwise the delexicalization cannot be performed, nor can the classifier properly output such values.

3 Pointer Network

In this section, we briefly introduce the PtrNet (Vinyals et al., 2015), which is the main basis of the proposed technique, and how the DST problem can be reformulated to take advantage of the flexibility enabled by such a model.

In the PtrNet architecture, similar as other sequence-to-sequence (seq2seq) models, there is an encoder which takes the input and iteratively produces a sequence of hidden states corresponding to the feature vector at each input position. There is also a decoder which generates outputs with the help of the weighted encoded states where the weights are computed through attention. Here, instead of using softmax to predict the distribution over a set of predefined candidates, the decoder directly normalizes the attention score at each position and obtains an output distribution over the input sequence. The index of the maximum probability is the pointed position, and the corresponding element is selected as decoder output, which is then fed into next decoding step. Both the encoder and decoder are based on various RNN models, capable of dealing with sequences of variable length.

The PtrNet specifically targets the problems where the output corresponds to positions in the input sequence, and it is widely used for seq2seq tasks where some kind of *copying* from the input is needed. Among its various applications, machine comprehension (a form of question answering), such as in (Wang and Jiang, 2016), is the closest to how we apply the model to DST.

The output of DST, same as in machine comprehension, is a word segment in the input sequence most of the time, thus can be naturally formulated as a *pointing* problem. Instead of generating longer output sequences, the decoder only has to predict the starting index and the ending index in order to identify the word segment.

More specifically, words are mapped to embeddings and the dialogue history $w_0, w_1, ..., w_t$ up to the current turn t is bidirectionally encoded using LSTM models. To differentiate words spoken by the user versus by the system, the word embeddings are further augmented with speaker role information. Other features, such as the entity type of each word, can also be fed into the encoder simultaneously in order to extract richer information from the dialogue context.

The encoded state at each position can then be denoted as h_i, which is the concatenation of forward state and backward state ($[h_i^f, h_i^b]$). The final forward state h_t^f is used as the initial hidden state of the decoder. We use a special symbol denoting the type of slot (e.g. <food>) as the first decoder input, which is also mapped to a trainable embedding E_{type}. Therefore, the starting index s^0 of the slot value is computed as the following, where u_i^0 is the attention score of the i_{th} word in the input against the decoder state d^0.

$$d^0 = LSTM(h_t^f, E_{type})$$

$$u_i^0 = v^T \tanh(W_h h_i + W_d d_0)$$

$$a_i^0 = \exp(u_i^0) / \sum_{j=0}^{t} \exp(u_j^0)$$

$$s^0 = \arg\max_i a_i^0$$

The attention scores at the second decoding step are computed similarly as below, where $E_{w_{s^0}}$ is the embedding of the word at the selected starting position, and the ending position s^1 can be obtained in the same way as s^0.

$$d^1 = LSTM(d^0, E_{w_{s^0}})$$

$$u_i^1 = v^T \tanh(W_h h_i + W_d d^1)$$

Note that there is no guarantee that $s^1 > s^0$, although most of the time the model is able to identify consistent patterns in the data and therefore output reasonable word segments. When $s^1 < s^0$, it is often a good indication that the answer does not exist in the input (such as the *none* slot in DSTC2).[1] Depending on the nature of the task, it is certainly possible to set a constraint at the second decoding step, forcing s^1 to be larger than s^0.

One can clearly see how the described model can handle unknown slot values – as long as they are mentioned explicitly during the dialogue, we

[1]It is the backoff strategy we take in our experiments on DSTC2.

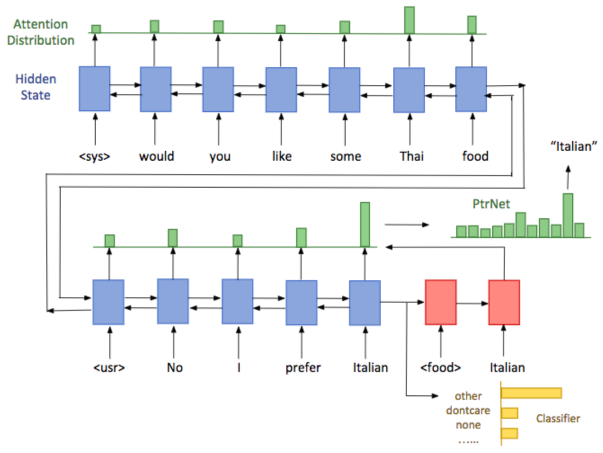


Figure 1: An illustration of the proposed PtrNet based architecture for DST. The classifier outputs "other" indicating the decision should be made by PtrNet; The decoder (red) in PtrNet is predicting the ending word of the slot value given the predicted starting word via attention against the encoded states (blue).

have a chance of finding them. Compared with previous approaches, which all require some kind of candidate lists, the proposed technique takes a different perspective on DST: For most slots in dialogue systems, tracking up-to-date values in a dialogue is not very different from tagging slots in a user query. While sequence labeling models such as conditional random field (CRF) has proven to be a great fit for slot tagging, the same formulation may as well be used for DST.

4 Rephrasing and Non-pointable Values

Our PtrNet based architecture works by directly pinpointing in the conversation history the slot value that the user expressed in its surface form. The model is totally unaware of the different ways of referring to the same entity. Therefore, the derived dialogue states may not have canonical forms that are consistent with the values in the backend database, making it more difficult to retrieve the correct results. A good example from the DSTC2 dataset is the price slot which can take the reference value "moderate", in the actual dialogues however, they are frequently expressed as "moderately priced", causing problems for searching the database and also computing accuracy.

While such a problem can be easily remedied by an extra *canonicalization* step (setting dialogue states to standard forms) before performing the

	Classifer	**PtrNet**
Rephrasing	Yes	*Yes
none, dontcare, etc	Yes	No
ASR errors	Hard	Hard
Unknown values	No	Yes

Table 1: Classifier vs. Pointer network in handling various difficult conditions. *PtrNet requires post-normalization to handle rephrasing.

database lookup, it is a much bigger problem if the slot value is not indicated explicitly by any particular word or phrase in the dialogue history, we describe these slot values as *non-pointable*. To give an example, in DSTC tasks, the special *none* value is given when the user has not specified any constraint for the slot. While this information can be easily inferred from the dialogue, it is not possible to point to any specific word segment in the sentence as the corresponding slot value. The same problem also exists for the *dontcare* value in DSTC, which implies that the user can accept any values for a slot constraint.

To address this issue, we add a classification component into our neural network architecture to handle non-pointable values. For each turn of the dialogue, the classifier makes a multi-class decision on whether the target slot should take any of the non-pointable values (e.g. dontcare or none) or it should be processed by the PtrNet.

As illustrated in Figure 1, the final forward state out of the dialogue encoder is used as the feature vector for the classification layer, which is trained with cross entropy loss and jointly with the PtrNet.

The best choice of the set of values to be handled by the classifier may not be obvious. In most cases both the classifier and the PtrNet are capable of extracting the correct slot value, although they both offer unique advantages over the other. Table 1 briefly summarizes the pros and cons of each model.

The proposed combined architecture, taking the best of both worlds, is similar to the pointer-generator model introduced in (See et al., 2017) for abstractive text summarization. In their approach the PtrNet is also augmented with a classification based word generator, and the model can choose to generate words from a predefined vocabulary or copy words from the input. Other *classify-and-copy* mechanisms have also been explored in (Gu et al., 2016; Gulcehre et al., 2016; Eric and Manning, 2017a), and demonstrated improved performance on various seq2seq tasks such as summarization and E2E dialogue generation. [2] As we have shown in this paper, DST can also be formulated to incorporate such copying mechanisms, allowing itself to handle unknown slot values as well.

5 Targeted Feature Dropout

Feature dropout is an effective technique to prevent feature co-adaption and improve model generalization (Hinton et al., 2012). It is most widely used for neural network based models but may as well be utilized for other feature based models. Targeted feature dropout however, was introduced in (Xu and Sarikaya, 2014) to address a very specific co-adaptation problem in slot filling, namely insufficient training of word context features.

For slot filling, this problem often occurs when 1) the dictionary (a precompiled list of possible slot values) covers the majority of the slot values in the training data, or 2) most slot values repeat frequently resulting in insufficient tail representations. In both cases, the contextual features tend to get severely under-trained and as a result the model is not able to generalize to unknown slot values that are neither in the dictionary nor observed in training.

The way our architecture works essentially extracts slot values in the same way as in slot filling, although the goal is to identify slots considering the entire dialogue context rather than a (usually) single user query. The same problem can also happen for DST if training data are not examined carefully. As an example, the DSTC2 task comes with a fixed ontology, it is not originally designed to track unknown slots (see the OOV rate in Table 2). Taking a closer look at the data, as shown in the histogram in Figure 2, the majority of the food type slot appears more than 10 times in the training data. As a result, the model oftentimes only learns to memorize these frequent slot values, and not the contextual patterns which can be more crucial for extracting slot values not known in advance.

To alleviate the generalization issue, we adapt the targeted dropout trick to work with our neural

[2] The copy-augmented model in (Eric and Manning, 2017a) also outputs API call parameters (which are essentially dialogue states) in a seq2seq fashion, including unknown parameters by copying from dialogue history, although the work focuses entirely on dialogue generation.

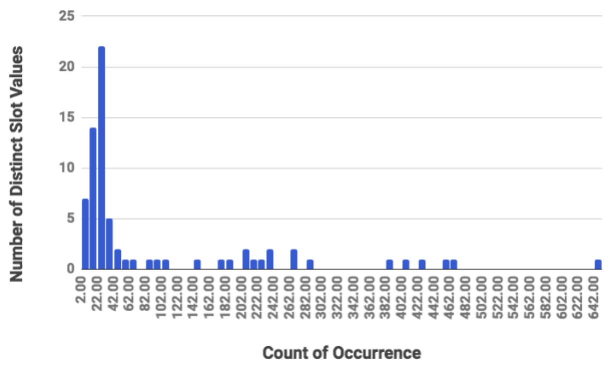


Figure 2: Histogram of *food type* slot on DSTC2 training data.

network based architecture. Instead of randomly disabling unigram and dictionary features for CRF models as done in the original work, we randomly set to zero the input word embeddings that correspond to the slot values in the dialogue utterances. For example, the *italian* food type in DSTC2 appears almost 500 times in the training data. During training, every time “italian” gets mentioned in the dialogue as the labeled user goal, we turn off the word embedding of “italian” in the model input with some probability, forcing the model to learn from the context to identify the slot value. Dictionary features are not used in our experiments, otherwise they can be turned off similarly.

As we will show later in the results, this proves to be a particularly effective yet simple trick for improving generalization to unknown slot values, without sacrificing accuracy for the known and observed ones.

6 Experiments and Results

6.1 Datasets

We conduct our experiments on the DSTC2 dataset (Henderson et al., 2014a), and on the bAbI dialogue dataset as used in (Bordes and Weston, 2016).

The DSTC2 dataset is the standard DST benchmark comprised of real dialogues between human and dialogue systems. We are mainly interested in tracking user goals, whereas the other two components of the dialogue state, namely search methods and requested slots, are not concerned with unknown slot values, and thus are not the focus in this paper. Meanwhile, the non-pointable values, none and dontcare, constitute a significant portion in DSTC2. Overall almost 60% of the user goals

	Original	New
#food types in train	74	48
#train instances	11677	8546
#test instances	9890	9890
OOV food types in test (%)	0	30.4

Table 2: Statistics of the new modified DSTC2 dataset with unknown food types. About 27% of the training instances are discarded. The test set remains the same.

are labeled as either none or dontcare, the two predominant non-pointable values, it is therefore particularly suitable for evaluating our proposed hybrid architecture.

An important part of our experimental evaluation is to demonstrate our ability to identify unknown slots. Although it happens frequently in real world situations, the original DSTC2 dataset does not suffer from this particular problem – on the test data, there are no unknown values that we have not observed in training for all of the three slot types. To conduct our investigation, we pick the food type slot to simulate unknown values. Specifically, we randomly select about 35% of the food types in the training set (26 out of 74) as unknown and discard all the training instances where the correct food type is one of the 26 unknown types that we selected. The statistics of the resulting dataset is shown in Table 2.

On the other hand, the bAbI dialogue dataset is initially designed for evaluating E2E goal oriented dialogue systems and has not been used specifically for DST. The model is expected to predict both the system utterances and the API calls to access the database. We notice that the parameters of the API calls are essentially the dialogue states at the point of the dialogue, it may as well be used as a dataset for measuring the accuracy of the state tracker. We therefore convert Task 5 of the bAbI dataset, which is the full dialogue combination of Tasks 1-4, into a DST dataset for our experiments.

Although simulated and with highly regular behaviors, the nice thing about the bAbI dialogue dataset is that it comes with an out-of-vocabulary (OOV) test set in which the entities (locations and food types) are not seen in any training dialogues. This poses exactly the same problem we are trying to address in this paper, namely predicting the API call parameters when they are not only unseen but also unknown to the system. Many of the previous E2E approaches simplifies the prediction problem as a selection among all API calls appeared in the entire dataset, thus bypassing the problem of tracking unknown dialogue states explicitly, although we believe it is not a realistic simplification.

6.2 Model and Training Details

The proposed model is implemented in TensorFlow. We use the provided development set to tune the hyper-parameters, track the training progress and select the best performing model for reporting the accuracy on test sets. The joint architecture is trained separately for each slot type by minimizing the sum of the cross entropy loss from the PtrNet and the classifier. Mini-batch SGD with a batch size of 50 and Adam optimizer (Kingma and Ba, 2014) is used for training.

Each word is mapped to a *randomly initialized* 100 dimensional embedding and each dialogue instance is represented as a 540 * 100 dimensional vector with zero paddings on the left when necessary. Instead of the using the raw word sequences, the system utterances are replaced by the more succinct and consistent dialogue act representations such as "request slot food". One layer of LSTM is used with a state size of 200 (additional layers did not help noticeably). Standard dropout with a keep probability of 0.5 is performed for training at the input and output of the LSTM cells. To keep it simple, targeted dropout is done only once for the entire training set before training begins, the dataset is therefore static across epochs.

To train the PtrNet, the location of the reference slot value in the dialogue needs to be provided. It does not require manual labeling though, and we simply use the *last* occurrence of the reference slot value in the dialogue history as the reference location. The occurrence is found via exact string match and the two most frequent spelling variations, "moderate" and "moderately", "center" and "centre" are considered equivalent. If no occurrence exists in a training instance (due to ASR errors or rephrasing), it will not be used for training the PtrNet.

On the other hand, the classifier serves as a gatekeeper that decides which slot values should be handed over to the PtrNet. On the bAbI dataset, there are zero non-pointable slots, and therefore everything is handled by the PtrNet. On DSTC2, we train the classifier to perform a *three-way* classification that determines if the slot values is none, dontcare or other. As we have described, other slot values can also become non-pointable in the actual dialogue: Those resulting from different surface forms are usually easier to handle, all we need is an extra post-processing step to normalize the value; The ones caused by ASR errors though, are much more challenging. One can argue that a classifier may be better equipped for these cases since it does not require locating the actual values in the word sequence, but unless there are consistent misrecognition patterns, they are difficult to handle for either the classifier or the PtrNet.

The non-pointable values in DSTC2, besides none and dontcare, are predominantly due to recognition errors, and we decide not to do anything specific about them – the PtrNet is tasked with processing these misrecognized utterances, and no normalization (except for "moderately" and "center") is performed on the network output for computing the accuracy. [3]

6.3 Evaluation Setup

The DSTC2 dataset is a standard benchmark for the task, we therefore compare the joint goal accuracy (a turn is considered correct if values are predicted correctly for all slots) of the proposed model with previous reported numbers to show the efficacy of our approach under regular circumstances, i.e. all slot values are known and observed in training. However, it is not our goal to outperform all previous DST systems – the main theme is that our technique allows identifying unknown slot values effectively and even if used in the standard setting, our model yields state-of-the-art results.

Measuring the accuracy on unknown slot values, however, does not have well-established baselines in the literature. Most previous systems are not concerned with this problem, and many of them are inherently not capable of outputting unknown values. So instead of comparisons with previous techniques, we will focus on demonstrating how this could be a serious problem tracking unknown slot values and how the targeted dropout can improve things drastically.

6.4 Results

The joint goal accuracy on the standard DSTC2 test set is shown in Table 3 comparing our Ptr-

[3]Non-pointable values besides none and dontcare constitute 9.7% of *food*, 7.6% of *location* and 4.7% of *price* on the test data, effectively setting an upper bound on the accuracy.

Models	Joint Acc.
Delexicalizaed RNN	69.1
Delexicalizaed RNN + semdict	72.9
NBT-DNN	72.6
NBT-CNN	73.4
MemN2N	74.0
Scalable Multi-domain DST	70.3
PtrNet	**72.1**

Table 3: Joint goal accuracy on DSTC2 test set vs. various approaches as reported in the literature.

Net based model against various previous reported baselines.

It is important to emphasize that the PtrNet model is an E2E model without using any SLU output and makes use of only the 1-best ASR hypothesis without any confidence measure for testing. Although more sophisticated DST models sometimes demonstrate better accuracy, our PtrNet model holds various advantages against all baseline models: In comparison with our approach, the delexicalized RNN models (Henderson et al., 2014b,c) utilize the n-best list and/or the SLU output; The NBT (Mrksic et al., 2017) and MemN2N (Perez and Liu, 2017) models are E2E but both depend on candidate lists as given and hence are not designed to handle unknown (different from unseen) slot values; The scalable DST model (Rastogi et al., 2018), although addressing the same problem of unbounded value set, relies on SLU to generate value candidates, and also does not perform equally well on the standard test set.

On the modified DSTC2 dataset with the reduced training set, the accuracy of the known/seen and unknown food types is shown in Figure 3. The standard training process with no targeted dropout performs poorly when the food types are not known beforehand, epitomizing the often overlooked challenge of handling unknown slot values. With a small dropout probability of 5%, the accuracy on unknown values essentially increases by three times (from 11.6% to 34.4%), while the accuracy on other values remains roughly the same.

Similar observations can also be made on the bAbI dataset predicting OOV API parameters (Table 4). While the dataset is quite artificial and in most cases we can achieve perfect accuracy on the regular test set, the OOV parameter values are not nearly as easy to predict. The targeted dropout

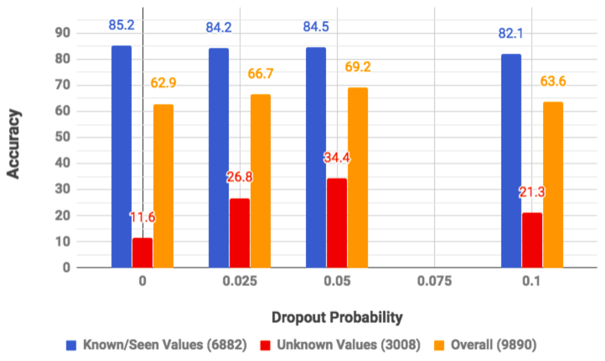


Figure 3: Accuracy of known/seen and unknown food types on the modified DSTC2 dataset with different dropout probabilities.

	Regular Test		OOV Test	
	p=0	p=0.1	p=0	p=0.1
food	100	100	86.2	100
location	100	100	74.7	99.6

Table 4: Accuracy of predicting regular and OOV *food* and *location* parameters in bAbI (Task 5) API calls w/ (p=0.1) and w/o (p=0) targeted dropout.

however, allows us to bridge the accuracy gap entirely.

7 Conclusion

An E2E dialogue state tracker is introduced based on the pointer network. The model outputs slot values in an extractive fashion similar to the slot filling task in SLU. We also add a jointly trained classification component to combine with the pointer network, forming a hybrid architecture that not only achieves state-of-the-art accuracy on the DSTC2 dataset, but also more importantly is able to handle unknown slot values, which is a problem often neglected although particularly valuable in real world situations. A feature dropout trick is also described and proves to be particularly effective.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments. We also would like to thank Mei-Yuh Hwang for helpful discussions.

References

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. In *CoRR*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Chen Yun-Nung, Faisal Ahmed, and Deng Li. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mihail Eric and Christopher Manning. 2017a. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *arXiv preprint arXiv:1701.04024v3 [cs.CL]*.

Mihail Eric and Christopher Manning. 2017b. Key-value retrieval networks for task-oriented dialogue. In *arXiv preprint arXiv:1705.05414v2 [cs.CL]*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014a. The second dialog state tracking challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Robust dialosg state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE Spoken Language Technology.*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv preprint arXiv:1207.0580v1 [cs.NE]*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *arxiv preprint arXiv:1703.01008v3 [cs.CL]*.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2017. End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. In *arxiv preprint arXiv:1711.10712v2 [cs.CL]*.

Metallinou Metallinou, Dan Bohus, and Jason Williams. 2013. Discriminative state tracking for spoken dialog systems. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nikola Mrksic, Diarmuid Seaghdha, Blaise Thomson, Milica Gasic, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nikola Mrksic, Diarmuid Seaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of EACL*.

Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. 2018. Scalable multi-domain dialogue state tracking. In *arXiv preprint arXiv:1712.10224v2 [cs.CL]*.

Abigail See, Peter Liu, and Christopher Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gokhan Tur and Renato De Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Wiley.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. In *arXiv preprint arXiv:1608.07905v2 [cs.CL]*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. In *CoRR*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*.

Jason Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. In *arxiv preprint arXiv:1606.01269v1 [cs.CL]*.

Puyang Xu and Ruhi Sarikaya. 2014. Targeted feature dropout for robust slot filling in natural language understanding. In *ISCA - International Speech Communication Association*.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of SIGDIAL 2016 Conference*.

Lukas Zilka and Filip Jurcicek. 2015. Incremental lstm-based dialog state tracker. In *ASRU*.

Global-Locally Self-Attentive Dialogue State Tracker

Victor Zhong, Caiming Xiong, Richard Socher
Salesforce Research
Palo Alto, CA
{vzhong, cxiong, rsocher}@salesforce.com

Abstract

Dialogue state tracking, which estimates user goals and requests given the dialogue context, is an essential part of task-oriented dialogue systems. In this paper, we propose the Global-Locally Self-Attentive Dialogue State Tracker (GLAD), which learns representations of the user utterance and previous system actions with global-local modules. Our model uses global modules to share parameters between estimators for different types (called slots) of dialogue states, and uses local modules to learn slot-specific features. We show that this significantly improves tracking of rare states and achieves state-of-the-art performance on the WoZ and DSTC2 state tracking tasks. GLAD obtains 88.1% joint goal accuracy and 97.1% request accuracy on WoZ, outperforming prior work by 3.7% and 5.5%. On DSTC2, our model obtains 74.5% joint goal accuracy and 97.5% request accuracy, outperforming prior work by 1.1% and 1.0%.

1 Introduction

Task oriented dialogue systems can significantly reduce operating costs by automating processes such as call center dispatch and online customer support. Moreover, when combined with automatic speech recognition systems, task-oriented dialogue systems provide the foundation of intelligent assistants such as Amazon Alexa, Apple Siri, and Google Assistant. In turn, these assistants allow for natural, personalized interactions with users by tailoring natural language system responses to the dialogue context. Dialogue state tracking (DST) is a crucial part of dialogue systems. In DST, a dialogue state tracker estimates the state of the conversation using the current user utterance and the conversation history. This estimated state is then used by the system to plan the next action and respond to the user. A state in DST typically consists of a set of *requests* and *joint goals*. Consider the task of restaurant reservation as an example. During each turn, the user may inform the system of particular *goals* the user would like to achieve (e.g. `inform(food=french)`), or *request* for more information from the system (e.g. `request(address)`). The set of goal and request slot-value pairs (e.g. `(food, french)`, `(request, address)`) given during a turn are referred to as the *turn goal* and *turn request*. The *joint goal* is the set of accumulated turn goals up to the current turn. Figure 1 shows an example dialogue with annotated turn states, in which the user reserves a restaurant.

Traditional dialogue state trackers rely on Spoken Language Understanding (SLU) systems (Henderson et al., 2012) in order to understand user utterances. These trackers accumulate errors from the SLU, which sometimes do not have the necessary dialogue context to interpret the user utterances. Subsequent DST research forgo the SLU and directly infer the state using the conversation history and the user utterance (Henderson et al., 2014b; Zilka and Jurcicek, 2015; Mrkšić et al., 2015). These trackers rely on hand-crafted semantic dictionaries and delexicalization — the anonymization of slots and values using generic tags — to achieve generalization. Recent work by Mrkšić et al. (2017) apply representation learning using convolutional neural networks to learn features relevant for each state as opposed to hand-crafting features.

A key problem in DST that is not addressed by existing methods is the extraction of rare slot-value pairs that compose the state during each turn. Because task oriented dialogues cover large

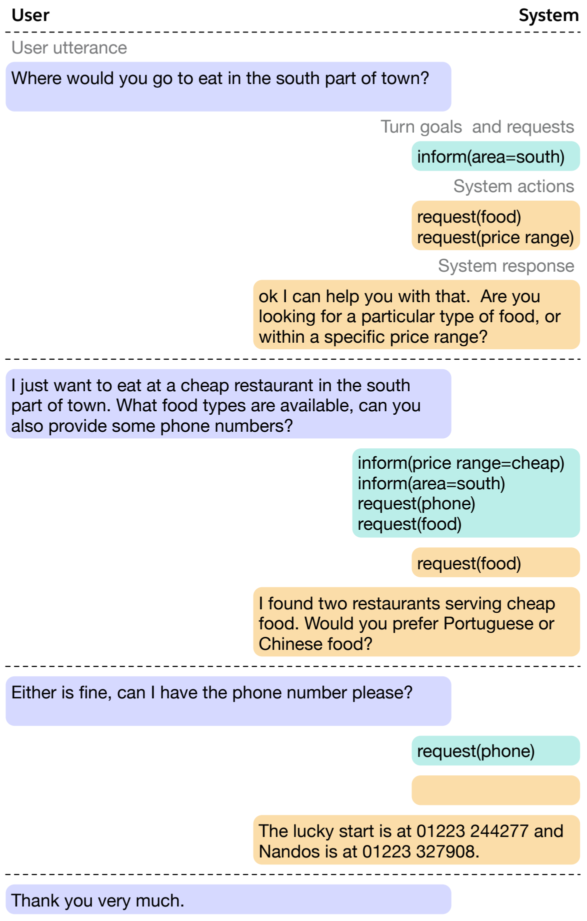

User
System
User utterance
Where would you go to eat in the south part of town?
Turn goals and requests
inform(area=south)
System actions
request(food)
request(price range)
System response
ok I can help you with that. Are you looking for a particular type of food, or within a specific price range?
I just want to eat at a cheap restaurant in the south part of town. What food types are available, can you also provide some phone numbers?
inform(price range=cheap)
inform(area=south)
request(phone)
request(food)
request(food)
I found two restaurants serving cheap food. Would you prefer Portuguese or Chinese food?
Either is fine, can I have the phone number please?
request(phone)
The lucky start is at 01223 244277 and Nandos is at 01223 327908.
Thank you very much.

Figure 1: An example dialogue from the WoZ restaurant reservation corpus. Dashed lines divide turns in the dialogue. A turn contains an user utterance (purple), followed by corresponding turn-level goals and requests (blue). The system then executes actions (yellow), and formulates the result into a natural language response (yellow).

state spaces, many slot-value pairs that compose the state rarely occur in the training data. Although the chance of a particular rare slot-value pair being specified by the user in a turn is small, the chance that at least one rare slot-value pair is specified is large. Failure to predict these rare slot-value pairs results in incorrect turn-level goal and request tracking. Accumulated errors in turn-level goal tracking significantly degrade joint goal-tracking. For example, in the WoZ state tracking dataset, slot-value pairs have 214.9 training examples on average, while 38.6% of turns have a joint goal that contains a rare slot-value pair with less than 20 training examples.

In this work, we propose the **G**lobal-**L**ocally Self-**A**ttentive **D**ialogue State Tracker (GLAD), a new state-of-the-art model for dialogue state tracking. In contrast to previous work that estimate each slot-value pair independently, GLAD uses global modules to share parameters between estimators for each slot and local modules to learn slot-specific feature representations. We show that by doing so, GLAD generalizes on rare slot-value pairs with few training examples. GLAD achieves state-of-the-art results of 88.1% goal accuracy and 97.1% request accuracy on the WoZ dialogue state tracking task (Wen et al., 2017), outperforming prior best by 3.7% and 5.5%. On DSTC2 (Henderson et al., 2014a), we achieve 74.5% goal accuracy and 97.5% request accuracy, outperforming prior best by 1.1% and 1.0%.

2 Global-Locally Self-Attentive Dialogue State Tracker

One formulation of state tracking is to predict the turn state given an user utterance and previous system actions (Williams and Young, 2007). Like previous methods (Henderson et al., 2014b; Wen et al., 2017; Mrkšić et al., 2017), GLAD decomposes the multi-label state prediction problem into a collection of binary prediction problems by using a distinct estimator for each slot-value pair that make up the state. Hence, we describe GLAD with respect to a slot-value pair that is being predicted by the model.

Shown in Figure 2, GLAD is comprised of an encoder module and a scoring module. The encoder module consists of separate global-locally self-attentive encoders for the user utterance, the previous system actions, and the slot-value pair under consideration. The scoring module consists of two scorers. One scorer considers the contribution from the utterance while the other considers the contribution from previous system actions.

2.1 Global-Locally Self-Attentive Encoder

We begin by describing the global-locally self-attentive encoder, which makes up the encoder module. DST datasets tend to be small relative to their state space in that many slot-value pairs rarely occur in the dataset. Because each state is comprised of a set of slot-value pairs, many of them rare, poor inference of rare slot-value pairs subsequently results in poor turn-level tracking. This problem is amplified in joint tracking, due to the accumulation of turn-level errors. In developing this encoder, we seek to better model

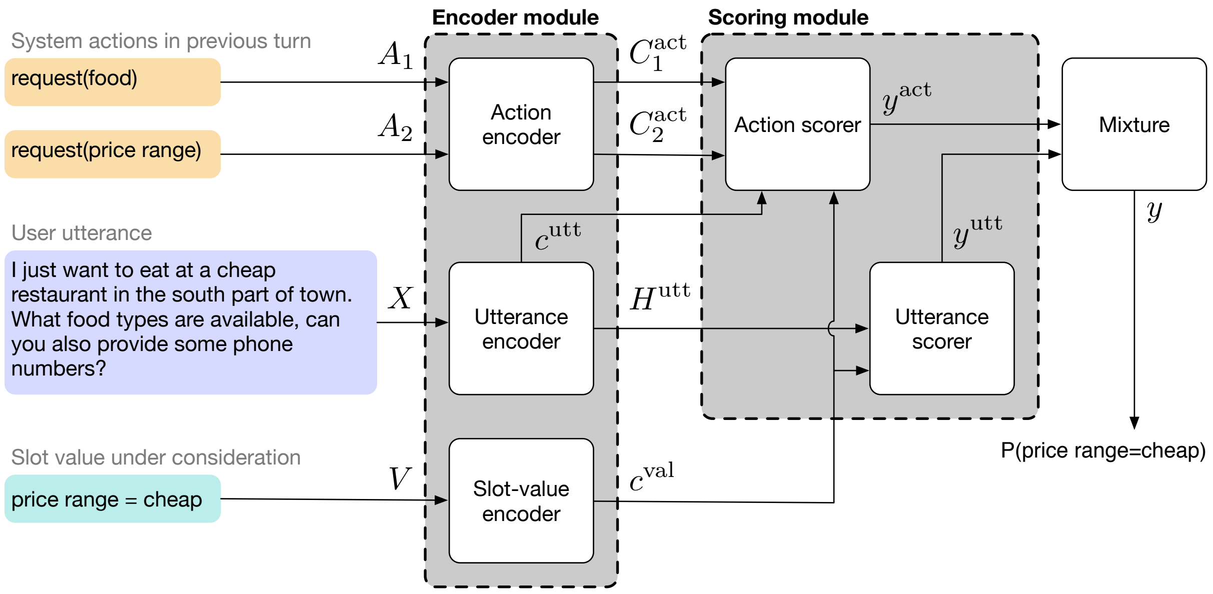


Figure 2: The Global-Locally Self-Attentive Dialogue State Tracker.

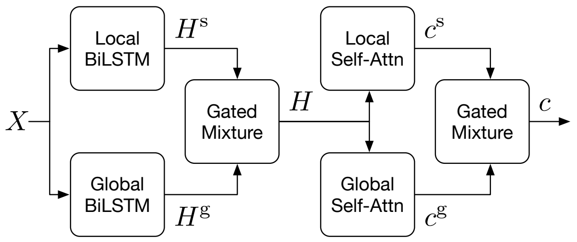


Figure 3: Global-locally self-attentive encoder.

rare slot-value pairs by sharing parameters between each slot through global modules and learning slot-specific features through local modules.

The global-locally self-attentive encoder consists of a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), which captures temporal relationships within the sequence, followed by a self-attention layer to compute the summary of the sequence. Figure 3 illustrates the global-locally self-attentive encoder.

Consider the process of encoding a sequence with respect to a particular slot s. Let n denote the number of words in the sequence, d_{emb} the dimension of the embeddings, and $X \in \mathbb{R}^{n \times d_{\mathrm{emb}}}$ the word embeddings corresponding to words in the sequence. We produce a global encoding H^{g} of X using a global bidirectional LSTM.

$$H^{\mathrm{g}} = \mathrm{biLSTM}^{\mathrm{g}}(X) \in \mathbb{R}^{n \times d_{\mathrm{rnn}}} \qquad (1)$$

where d_{rnn} is the dimension of the LSTM state. We similarly produce a local encoding H^{s} of X, taking into account the slot s, using a local bidirectional LSTM.

$$H^{\mathrm{s}} = \mathrm{biLSTM}^{\mathrm{s}}(X) \in \mathbb{R}^{n \times d_{\mathrm{rnn}}} \qquad (2)$$

The outputs of the two LSTMs are combined through a mixture layer to yield a global-local encoding H of X.

$$H = \beta^{\mathrm{s}} H^{\mathrm{s}} + (1 - \beta^{\mathrm{s}}) H^{\mathrm{g}} \in \mathbb{R}^{n \times d_{\mathrm{rnn}}} \qquad (3)$$

Here, the scalar β^{s} is a learned parameter between 0 and 1 that is specific to the slot s. Next, we compute a global-local self-attention context c over H. Self-attention, or intra-attention, is a very effective method of computing summary context over variable-length sequences for natural language processing tasks (Cheng et al., 2016; Vaswani et al., 2017; He et al., 2017; Lee et al., 2017). In our case, we use a global self-attention module to compute attention context useful for general-purpose state tracking, as well as a local self-attention module to compute slot-specific attention context.

For each ith element H_i, we compute a scalar global self-attention score a_i^{g} which is subsequently normalized across all elements using a softmax function.

$$a_i^{\mathrm{g}} = W^{\mathrm{g}} H_i + b^{\mathrm{g}} \in \mathbb{R} \qquad (4)$$

$$p^{\mathrm{g}} = \mathrm{softmax}(a^{\mathrm{g}}) \in \mathbb{R}^n \qquad (5)$$

The global self-attention context c^{g} is then the sum of each element H_i, weighted by the corresponding normalized global self-attention score p_i^{g}.

$$c^{\mathrm{g}} = \sum_i p_i^{\mathrm{g}} H_i \in \mathbb{R}^{d_{\mathrm{rnn}}} \tag{6}$$

We similarly compute the local self-attention context c^{s}.

$$\begin{aligned} a_i^{\mathrm{s}} &= W^{\mathrm{s}} H_i + b^{\mathrm{s}} \in \mathbb{R} && (7) \\ p^{\mathrm{s}} &= \mathrm{softmax}\left(a^{\mathrm{s}}\right) \in \mathbb{R}^n && (8) \\ c^{\mathrm{s}} &= \sum_i p_i^{\mathrm{s}} H_i \in \mathbb{R}^{d_{\mathrm{rnn}}} && (9) \end{aligned}$$

The global-local self-attention context c is the mixture

$$c = \beta^{\mathrm{s}} c^{\mathrm{s}} + \left(1 - \beta^{\mathrm{s}}\right) c^{\mathrm{g}} \in \mathbb{R}^{n \times d_{\mathrm{rnn}}} \tag{10}$$

For ease of exposition, we define the multi-value encode function $\mathrm{encode}\left(X\right)$.

$$\mathrm{encode} : X \to H, c \tag{11}$$

This function maps the sequence X to the encoding H and the self-attention context c.

2.2 Encoding module

Having defined the global-locally self-attentive encoder, we now build representations for the user utterance, the previous system actions, and the slot-value pair under consideration. Let U denote word embeddings of the user utterance, A_j denote those of the jth previous system action (e.g. `request ( price range )`, and V denote those of the slot-value pair under consideration (e.g. `food = french`). We have

$$\begin{aligned} H^{\mathrm{utt}}, c^{\mathrm{utt}} &= \mathrm{encode}\left(U\right) && (12) \\ H_j^{\mathrm{act}}, C_j^{\mathrm{act}} &= \mathrm{encode}\left(A_j\right) && (13) \\ H^{\mathrm{val}}, c^{\mathrm{val}} &= \mathrm{encode}\left(V\right) && (14) \end{aligned}$$

2.3 Scoring module

Intuitively, there are two sources of contribution to whether the user has expressed the slot-value pair under consideration. The first source of contribution is the user utterance, in which the user directly states the goals and requests. An example of this is the user saying "how about a French restaurant in the centre of town?", after the system asked "how may I help you?" To handle these cases, we determine whether the utterance specifies the slot-value pair. Namely, we attend over the user utterance H^{utt}, taking into account the slot-value pair being considered c^{val}, and use the resulting attention context q^{utt} to score the slot-value pair.

$$\begin{aligned} a_i^{\mathrm{utt}} &= \left(H_i^{\mathrm{utt}}\right)^{\intercal} c^{\mathrm{val}} \in \mathbb{R} && (15) \\ p^{\mathrm{utt}} &= \mathrm{softmax}\left(a^{\mathrm{utt}}\right) \in \mathbb{R}^m && (16) \\ q^{\mathrm{utt}} &= \sum_i p_i^{\mathrm{utt}} H_i^{\mathrm{utt}} \in \mathbb{R}^{d_{\mathrm{rnn}}} && (17) \\ y^{\mathrm{utt}} &= W q^{\mathrm{utt}} + b \in \mathbb{R} && (18) \end{aligned}$$

where m is the number of words in the user utterance. The score y^{utt} indicates the degree to which the value was expressed by the user utterance.

The second source of contribution is the previous system actions. This source is informative when the user utterance does not present enough information and instead refers to previous system actions. An example of this is the user saying "yes", after the system asked "would you like a restaurant in the centre of town?" To handle these cases, we examine previous actions after considering the user utterance. First, we attend over the previous action representations $C^{\mathrm{act}} = [C_1^{\mathrm{act}} \cdots C_l^{\mathrm{act}}]$, taking into account the current user utterance c^{utt}. Here, l is the number of previous system actions. Then, we use the similarity between the attention context q^{act} and the slot-value pair c^{val} to score the slot-value pair.

$$\begin{aligned} a_j^{\mathrm{act}} &= \left(C_j^{\mathrm{act}}\right)^{\intercal} c^{\mathrm{utt}} \in \mathbb{R} && (19) \\ p^{\mathrm{act}} &= \mathrm{softmax}\left(a^{\mathrm{act}}\right) \in \mathbb{R}^{l+1} && (20) \\ q^{\mathrm{act}} &= \sum_j p_j^{\mathrm{act}} C_j^{\mathrm{act}} \in \mathbb{R}^{d_{\mathrm{rnn}}} && (21) \\ y^{\mathrm{act}} &= \left(q^{\mathrm{act}}\right)^{\intercal} c^{\mathrm{val}} \in \mathbb{R} && (22) \end{aligned}$$

In addition to real previous system actions, we introduce a sentinel action to each turn which allows the attention mechanism to ignore previous system actions. The score y^{act} indicates the degree to which the value was expressed by the previous actions.

The final score y is then a weighted sum between the two scores y^{utt} and y^{act}, normalized by the sigmoid function σ.

$$y = \sigma\left(y^{\text{utt}} + w y^{\text{act}}\right) \in \mathbb{R} \qquad (23)$$

Here, the weight w is a learned parameter.

3 Experiments

3.1 Dataset

The Dialogue Systems Technology Challenges (DSTC) provides a common framework for developing and evaluating dialogue systems and dialogue state trackers (Williams et al., 2013; Henderson et al., 2014a). Under this framework, dialogue semantics such as states and actions are based on a task ontology such as restaurant reservation. During each turn, the user may inform the system of particular *goals* (e.g. `inform(food=french)`), or *request* for more information from the system (e.g. `request(address)`). For instance, `food` and `area` are examples of slots in the DSTC2 task, and `french` and `chinese` are example values within the `food` slot. We train and evaluate our model using DSTC2 as well as the Wizard of Oz (WoZ) restaurant reservation task (Wen et al., 2017), which also adheres to the DSTC framework and has the same ontology as DSTC2.

For DSTC2, it is standard to evaluate using the N-best list of the automatic speech recognition system (ASR) that is included with the dataset. Because of this, each turn in the DSTC2 dataset contains several noisy ASR outputs instead of a noise-free user utterance. The WoZ task does not provide ASR outputs, and we instead train and evaluate using the user utterance.

3.2 Metrics

We evaluate our model using turn-level request tracking accuracy as well as joint goal tracking accuracy. Our definition of GLAD in the previous sections describes how to obtain turn goals and requests. To compute the joint goal, we simply accumulate turn goals. In the event that the current turn goal specifies a slot that has been specified before, the new specification takes precedence. For example, suppose the user specifies a `food=french` restaurant during the current turn. If the joint goal has no existing `food` specifications, then we simply add `food=french` to the joint goal. Alternatively, if `food=thai` had been specified in a previous turn, we simply replace it with `food=french`.

3.3 Implementation Details

We use fixed, pretrained GloVe embeddings (Pennington et al., 2014) as well as character n-gram embeddings (Hashimoto et al., 2017). Each model is trained using ADAM (Kingma and Ba, 2015). For regularization, we apply dropout with 0.2 drop rate (Srivastava et al., 2014) to the output of each local module and each global module. We use the development split for hyperparameter tuning and apply early stopping using the joint goal accuracy.

For the DSTC2 task, we train using transcripts of user utterances and evaluate using the noisy ASR transcriptions. During evaluation, we take the sum of the scores resulting from each ASR output as the output score of a particular slot-value. We then normalize this sum using a sigmoid function as shown in Equation (23). We also apply word dropout, in which the embeddings of a word is randomly set to zero with a probability of 0.3. This accounts for the poor quality of ASR outputs in DSTC2, which frequently miss several words in the user utterance. We did not find word dropout to be helpful for the WoZ task, which does not contain noisy ASR outputs.

3.4 Comparison to Existing Methods

Table 1 shows the performance of GLAD compared to previous state-of-the-art models. The delexicalisation models, which replace slots and values in the utterance with generic tags, are from Henderson et al. (2014b) for DSTC2 and Wen et al. (2017) for WoZ. Semantic dictionaries map slot-value pairs to hand-engineered synonyms and phrases. The NBT (Mrkšić et al., 2017) applies CNN over word embeddings learned over a paraphrase database (Wieting et al., 2015) instead of delexicalised n-gram features.

On the WoZ dataset, we find that GLAD significantly improves upon previous state-of-the-art performance by 3.7% on joint goal tracking accuracy and 5.5% on turn-level request tracking accuracy. On the DSTC dataset, which evaluates using noisy ASR outputs instead of user utterances, GLAD improves upon previous state of the art performance by 1.1% on joint goal tracking accuracy and 1.0% on turn-level request tracking accuracy.

Model	DSTC2		WoZ	
	Joint goal	Turn request	Joint goal	Turn request
Delexicalisation-Based Model	69.1%	95.7%	70.8%	87.1%
Delex. Model + Semantic Dictionary	72.9%	95.7%	83.7%	87.6%
Neural Belief Tracker (NBT) - DNN	72.6%	96.4%	84.4%	91.2%
Neural Belief Tracker (NBT) - CNN	73.4%	96.5%	84.2%	91.6%
GLAD	**74.5± 0.2%**	**97.5± 0.1%**	**88.1± 0.4%**	**97.1± 0.2%**

Table 1: Test accuracies on the DSTC2 and WoZ restaurant reservation datasets. The other models are: delexicalisation DSTC2 (Henderson et al., 2014b), delexicalisation WoZ (Wen et al., 2017), and NBT (Mrkšić et al., 2017). We run 10 models using random seeds with early stopping on the development set, and report the mean and standard deviation test accuracies for each dataset.

Model	Tn goal	Jnt goal	Tn request
GLAD	93.7%	88.8%	97.3%
- global	88.8%	73.4%	97.3%
- local	93.1%	86.6%	95.1%
- self-attn	91.6%	84.4%	97.1%
- LSTM	88.7%	71.5%	93.2%

Table 2: Ablation study showing turn goal, joint goal, and turn request accuracies on the dev. split of the WoZ dataset. For "- self-attn", we use mean-pooling instead of self-attention. For "- LSTM", we compute self-attention over word embeddings.

3.5 Ablation study

We perform ablation experiments on the development set to analyze the effectiveness of different components of GLAD. The results of these experiments are shown in Table 2. In addition to the joint goal accuracy and the turn request accuracy, we also show the turn goal accuracy for reference.

Temporal order is important for state tracking. We experiment with an embedding-matching variant of GLAD with self-attention but without LSTMs. The weaker performance by this model suggests that representations that capture temporal dependencies is helpful for understanding phrases for state tracking.

Self-attention allows slot-specific, robust feature learning. We observe a consistent drop in performance for models that use mean-pooling as opposed to self-attention (Equations (4) to (6)). This stems from the flexibility in the attention context computation afforded by the self-attention mechanism, which allows the model to focus on select words relevant to the slot-value pair under consideration. Figure 4 illustrates an example in which local self-attention modules focus on relevant parts of the utterance. We note that the model attends to relevant phrases that n-gram and embedding matching techniques do not capture (e.g. "within 5 miles" for the "area" slot).

Global-local sharing improves goal tracking. We study the two extremes of sharing between the global module and the local module. The first uses only the local module and results in degradation in goal tracking but does not affect request tracking (e.g. $\beta^s = 1$). This is because the former is a joint prediction over several slot-values with few training examples, whereas the latter predicts a single slot that has the most training examples.

The second uses only the global module and underperforms in both goal tracking and request tracking (e.g. $\beta^s = 0$). This model is less expressive, as it lacks slot-specific specializations except for the final scoring modules.

Figure 5 shows the performance of the original model and the two sharing variants across different numbers of occurrences in the training data. GLAD consistently outperforms both variants for rare slot-value pairs. For slot-value pairs with an abundance of training data, there is no significant performance difference between models as there is sufficient data to generalize.

3.6 Qualitative analysis

Table 3 shows example predictions by GLAD. In the first example, the user explicitly outlines requests and goals in a single utterance. In the second example, the model previously prompted the user for confirmation of two requests (e.g. for the restaurant's address and phone number), and the user simply answers in the affirmative. In this case, the model still obtains the correct result by leveraging the system actions in the previous turn. The last example demonstrates an error made by

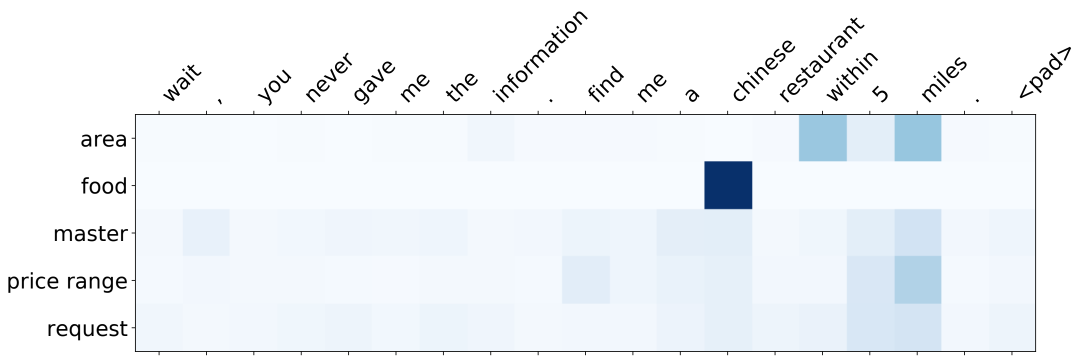


Figure 4: Global and local self-attention scores on user utterances. Each row corresponds to the self-attention score for a particular slot. Slot-specific local self-attention modules emphasize relevant key words and phrases to that slot, whereas the global module attends to all relevant words.

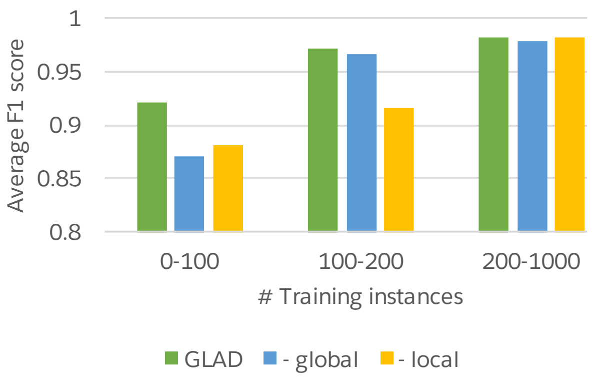


Figure 5: F1 performance for each slot-value pair in the development split of the WoZ task, grouped by the number of training instances.

the model. Here, the user does not answer the system's previous request for the choice of food and instead asks for what food is available. The model misinterprets the lack of response as the user not caring about the choice of food.

4 Related Work

Dialogue State Tracking. Traditional dialogue state trackers rely on a separate SLU component that serves as the initial stage in the dialogue agent pipeline. The downstream tracker then combines the semantics extracted by the SLU with previous dialogue context in order to estimate the current dialogue state (Thomson and Young, 2010; Wang and Lemon, 2013; Williams, 2014; Perez and Liu, 2017). Recent results in dialogue state tracking show that it is beneficial to jointly learn speech understanding and dialogue tracking (Henderson et al., 2014b; Zilka and Jurcicek, 2015; Wen et al., 2017). These approaches directly take as input the N-best list produced by the ASR system. By avoiding the accumulation of errors from the initial SLU component, these joint approaches typically achieved stronger performance on tasks such as DSTC2. One drawback to these approaches is that they rely on hand-crafted features and complex domain-specific lexicon (in addition to the ontology), and consequently are difficult to extend and scale to new domains. The recent Neural Belief Tracker (NBT) by Mrkšić et al. (2017) avoids reliance on hand-crafted features and lexicon by using representation learning. The NBT employs convolutional filters over word embeddings in lieu of previously-used hand-engineered features. Moreover, to outperform previous methods, the NBT uses pretrained embeddings tailored to retain semantic relationships by injecting semantic similarity constraints from the Paraphrase Database (Wieting et al., 2015; Ganitkevitch et al., 2013). On the one hand, these specialized embeddings are more difficult to obtain than word embeddings from language modeling. On the other hand, these embeddings are not specific to any dialogue domain and are directly usable for new domains.

Neural attention models in NLP. Attention mechanisms have led to improvements on a variety of natural language processing tasks. Bahdanau et al. (2015) propose attentional sequence to sequence models for neural machine translation. Luong et al. (2015) analyze various attention techniques and highlight the effectiveness of the simple, parameterless dot product attention. Similar models have also proven successful in tasks such as summarization (See et al., 2017; Paulus et al., 2018). Self-attention, or intra-attention, has led improvements in language modeling, sentiment

System actions in previous turn	User utterance	Predicted turn belief state

N/A	I would like Polynesian food in the South part of town. Please send me phone number and address.	request(phone)
		request(address)
		inform(food=polynesian)
		inform(area=south)

request(address)
request(phone)
 	
There is a moderately priced italian place called Pizza hut at cherry hilton. would you like the address and phone number?
	Yes please.	request(phone)
		request(address)

request(food)
request(price range)
 	
ok I can help you with that. Are you looking for a particular type of food, or within a specific price range?
	I just want to eat at a cheap restaurant in the south part of town. What food types are available, can you also provide some phone numbers?	request(phone)
		inform(price range=cheap)
		inform(area=south)
		-inform(food=dontcare)
		+request(food)

Table 3: Example predictions by Global-Locally Self-Attentive Dialogue State Tracker on the development split of the WoZ restaurant reservation dataset. Model predicted slot-value pairs that are not in the ground truth (e.g. false positives) are prefaced with a "+" symbol. Ground truth slot-value pairs that are not predicted by the model (e.g. false negatives) are prefaced with a "-" symbol.

analysis, natural language inference (Cheng et al., 2016), semantic role labeling (He et al., 2017), and coreference resolution (Lee et al., 2017). Deep self-attention has also achieved state-of-the-art results in machine translation (Vaswani et al., 2017). Coattention, or bidirectional attention that codependently encode two sequences, have led to significant gains in question answering (Xiong et al., 2017; Seo et al., 2017) as well as visual question answering (Lu et al., 2016).

Parameter sharing between related tasks. Sharing parameters between related tasks to improve joint performance is prominent in multi-task learning (Caruana, 1998; Thrun, 1996). Early works in multi-tasking use Gaussian processes whose covariance matrix is induced from shared kernels (Lawrence and Platt, 2004; Yu et al., 2005; Seeger et al., 2005; Bonilla et al., 2008). Hashimoto et al. (2017) propose a progressively trained joint model for NLP tasks. When a new task is introduced, a new section is added to the network whose inputs are intermediate representations from sections for previous tasks. In this sense, tasks share parameters in a hierarchical manner. Johnson et al. (2016) propose a single model that jointly learns to translate between multiple language pairs, including one-to-many, many-to-one, and many-to-many translation. Kaiser et al. (2017) propose a model that jointly learns multiple tasks across modalities. Each modality has a corresponding modality net, which extracts a representation that is fed into a shared encoder.

5 Conclusion

We introduced the Global-Locally Self-Attentive Dialogue State Tracker (GLAD), a new state-of-the-art model for dialogue state tracking. At the core of GLAD is the global-locally self-attention encoder, whose global modules allow parameter sharing between slots and local modules allow slot-specific feature learning. This allows GLAD to generalize on rare slot-value pairs with few training data. GLAD achieves state-of-the-art results of 88.1% goal accuracy and 97.1% request accuracy on the WoZ dialogue state tracking task, as well as 74.5% goal accuracy and 97.5% request accuracy on DSTC2.

Acknowledgement

We thank Nikola Mrkšić for providing us with a preprocessed version of the DSTC2 dataset.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Edwin V Bonilla, Kian M Chai, and Christopher Williams. 2008. Multi-task gaussian process prediction. In *NIPS*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *ACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *ACL*.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative spoken language understanding using word confusion networks. In *Spoken Language Technology Workshop (SLT)*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *SIGDIAL*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Compututation* 9(8).

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. Technical report, Google.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137* .

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Neil D Lawrence and John C Platt. 2004. Learning to learn with the informative vector machine. In *ICML*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *ACL*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *ACL*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *EACL*.

Abigail See, Peter Liu, and Christopher Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Matthias Seeger, Yee-Whye Teh, and Michael Jordan. 2005. Semiparametric latent factor models. In *AISTATS*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1).

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language* 24(4).

Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. In *ACL*.

Jason D Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *SIGDIAL*.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *SIGDIAL*.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language* 21.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Kai Yu, Volker Tresp, and Anton Schwaighofer. 2005. Learning gaussian processes from multiple tasks. In *ICML*.

Lukas Zilka and Filip Jurcicek. 2015. Incremental lstm-based dialog state tracker. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems

Andrea Madotto[*], **Chien-Sheng Wu**[*], **Pascale Fung**
Human Language Technology Center
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
`[eeandreamad,cwuak,pascale]@ust.hk`

Abstract

End-to-end task-oriented dialog systems usually suffer from the challenge of incorporating knowledge bases. In this paper, we propose a novel yet simple end-to-end differentiable model called memory-to-sequence (Mem2Seq) to address this issue. Mem2Seq is the first neural generative model that combines the multi-hop attention over memories with the idea of pointer network. We empirically show how Mem2Seq controls each generation step, and how its multi-hop attention mechanism helps in learning correlations between memories. In addition, our model is quite general without complicated task-specific designs. As a result, we show that Mem2Seq can be trained faster and attain the state-of-the-art performance on three different task-oriented dialog datasets.

1 Introduction

Task-oriented dialog systems help users to achieve specific goals with natural language such as restaurant reservation and schedule arrangement. Traditionally, they have been built with several pipelined modules: language understanding, dialog management, knowledge query, and language generation (Williams and Young, 2007; Hori et al., 2009; Lee et al., 2009; Levin et al., 2000; Young et al., 2013). Moreover, the ability to query external Knowledge Bases (KBs) is essential in task-oriented dialog systems, since the responses are guided not only by the dialog history but also by the query results (e.g. Table 1). However, despite the stability of such pipelined systems via combining domain-specific knowledge and slot-filling techniques, modeling the dependencies between modules is complex and the KB interpretation requires human effort.

Point of interest (poi)	Distance	Traffic info	Poi type	Address
The Westin	5 miles	moderate traffic	rest stop	329 El Camino Real
Round Table	4 miles	no traffic	pizza restaurant	113 Anton Ct
Mandarin Roots	5 miles	no traffic	chinese restaurant	271 Springer Street
Palo Alto Cafe	4 miles	moderate traffic	coffee or tea place	436 Alger Dr
Dominos	6 miles	heavy traffic	pizza restaurant	776 Arastradero Rd
Stanford Express Care	6 miles	no traffic	hospital	214 El Camino Real
Hotel Keen	2 miles	heavy traffic	rest stop	578 Arbol Dr

1th Turn	**DRIVER**	Where can I get tea?
	Seq2Seq	I have a away from away would you like the address
	+Attn	I have a listing for a place that serves tea that is 5 miles away
	Ptr-Unk	There is a away you like would you like more info
	Mem2Seq	The nearest tea shop is Palo Alto Cafe located 4 miles away, would you like directions there?
	GOLD	Palo Alto Cafe is 4 miles away and serves coffee and tea. Do you want the address?
2th Turn	**DRIVER**	Yes.
	Seq2Seq	The is at at
	+Attn	The address is 329 El Camino Real and it's 3 miles away and there is no traffic
	Ptr-Unk	Palo Alto Cafe is 4 miles away and PAD is no traffic
	Mem2Seq	Palo Alto Cafe is 4 miles away at 436 Alger Drive
	GOLD	Palo Alto is located at 436 Alger Dr.

Table 1: Example of generated responses for the In-Car Assistant on the navigation domain.

Recently, end-to-end approaches for dialog modeling, which use recurrent neural networks (RNN) encoder-decoder models, have shown promising results (Serban et al., 2016; Wen et al., 2017; Zhao et al., 2017). Since they can directly map plain text dialog history to the output responses, and the dialog states are latent, there is no need for hand-crafted state labels. Moreover, attention-based copy mechanism (Gulcehre et al., 2016; Eric and Manning, 2017) have been recently introduced to copy words directly from the input sources to the output responses. Using such mechanism, even when unknown tokens appear in the dialog history, the models are still able to produce correct and relevant entities.

However, although the above mentioned approaches were successful, they still suffer from two main problems: 1) They struggle to effectively incorporate external KB information into the RNN hidden states (Sukhbaatar et al., 2015),

[*] These two authors contributed equally.

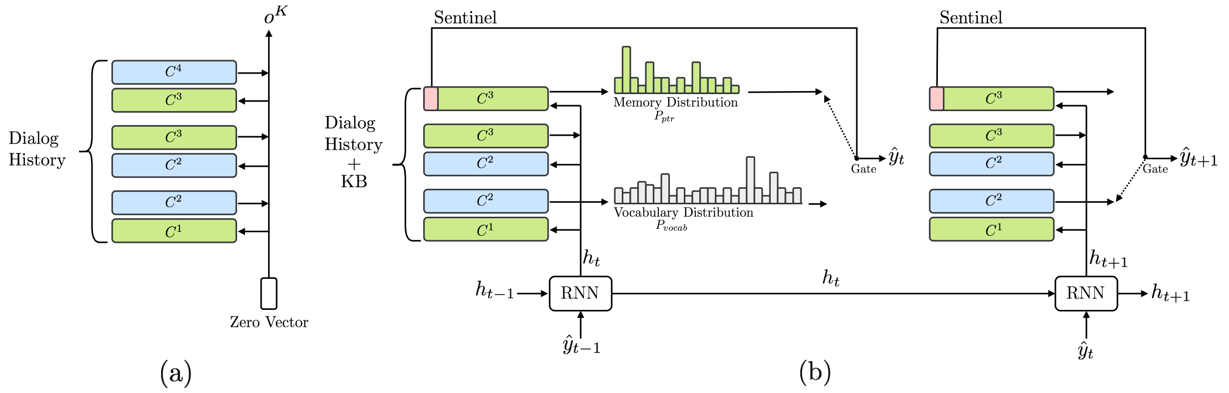


Figure 1: The proposed Mem2Seq architecture for task-oriented dialog systems. (a) Memory encoder with 3 hops; (b) Memory decoder over 2 step generation.

since RNNs are known to be unstable over long sequences. 2) Processing long sequences is very time-consuming, especially when using attention mechanisms.

On the other hand, end-to-end memory networks (MemNNs) are recurrent attention models over a possibly large external memory (Sukhbaatar et al., 2015). They write external memories into several embedding matrices, and use query vectors to read memories repeatedly. This approach can memorize external KB information and rapidly encode long dialog history. Moreover, the multi-hop mechanism of MemNN has empirically shown to be essential in achieving high performance on reasoning tasks (Bordes and Weston, 2017). Nevertheless, MemNN simply chooses its responses from a predefined candidate pool rather than generating word-by-word. In addition, the memory queries need explicit design rather than being learned, and the copy mechanism is absent.

To address these problems, we present a novel architecture that we call Memory-to-Sequence (Mem2Seq) to learn task-oriented dialogs in an end-to-end manner. In short, our model augments the existing MemNN framework with a sequential generative architecture, using global multi-hop attention mechanisms to copy words directly from dialog history or KBs. We summarize our main contributions as such: 1) Mem2Seq is the first model to combine multi-hop attention mechanisms with the idea of pointer networks, which allows us to effectively incorporate KB information. 2) Mem2Seq learns how to generate dynamic queries to control the memory access. In addition, we visualize and interpret the model dynamics among hops for both the memory controller and the attention. 3) Mem2Seq can be trained faster and achieve state-of-the-art results in several task-oriented dialog datasets.

2 Model Description

Mem2Seq [1] is composed of two components: the MemNN encoder, and the memory decoder as shown in Figure 1. The MemNN encoder creates a vector representation of the dialog history. Then the memory decoder reads and copies the memory to generate a response. We define all the words in the dialog history as a sequence of tokens $X = \{x_1, \ldots, x_n, \$\}$, where \$ is a special charter used as a sentinel, and the KB tuples as $B = \{b_1, \ldots, b_l\}$. We further define $U = [B; X]$ as the concatenation of the two sets X and B, $Y = \{y_1, \ldots, y_m\}$ as the set of words in the expected system response, and $PTR = \{ptr_1, \ldots, ptr_m\}$ as the pointer index set:

$$ptr_i = \begin{cases} max(z) & \text{if } \exists z \text{ s.t. } y_i = u_z \\ n+l+1 & \text{otherwise} \end{cases} \quad (1)$$

where $u_z \in U$ is the input sequence and $n + l + 1$ is the sentinel position index.

2.1 Memory Encoder

Mem2Seq uses a standard MemNN with adjacent weighted tying (Sukhbaatar et al., 2015) as an encoder. The input of the encoder is word-level information in U. The memories of MemNN are represented by a set of trainable embedding matrices $C = \{C^1, \ldots, C^{K+1}\}$, where each C^k maps tokens to vectors, and a query vector q^k is used as a reading head. The model loops over K hops and

[1]The code is available at https://github.com/HLTCHKUST/Mem2Seq

it computes the attention weights at hop k for each memory i using:

$$p_i^k = \text{Softmax}((q^k)^T C_i^k), \tag{2}$$

where $C_i^k = C^k(x_i)$ is the memory content in position i, and $\text{Softmax}(z_i) = e^{z_i}/\Sigma_j e^{z_j}$. Here, p^k is a soft memory selector that decides the memory relevance with respect to the query vector q^k. Then, the model reads out the memory o^k by the weighted sum over C^{k+1} [2],

$$o^k = \sum_i p_i^k C_i^{k+1}. \tag{3}$$

Then, the query vector is updated for the next hop by using $q^{k+1} = q^k + o^k$. The result from the encoding step is the memory vector o^K, which will become the input for the decoding step.

2.2 Memory Decoder

The decoder uses RNN and MemNN. The MemNN is loaded with both X and B, since we use both dialog history and KB information to generate a proper system response. A Gated Recurrent Unit (GRU) (Chung et al., 2014), is used as a dynamic query generator for the MemNN. At each decoding step t, the GRU gets the previously generated word and the previous query as input, and it generates the new query vector. Formally:

$$h_t = \text{GRU}(C^1(\hat{y}_{t-1}), h_{t-1}); \tag{4}$$

Then the query h_t is passed to the MemNN which will produce the token, where h_0 is the encoder vector o^K. At each time step, two distribution are generated: one over all the words in the vocabulary (P_{vocab}), and one over the memory contents (P_{ptr}), which are the dialog history and KB inofrmation. The first, P_{vocab}, is generated by concatenating the first hop attention read out and the current query vector.

$$P_{vocab}(\hat{y}_t) = \text{Softmax}(W_1[h_t; o^1]) \tag{5}$$

where W_1 is a trainable parameter. On the other hand, P_{ptr} is generated using the attention weights at the last MemNN hop of the decoder: $P_{ptr} = p_t^K$. Our decoder generates tokens by pointing to the input words in the memory, which is a similar mechanism to the attention used in pointer networks (Vinyals et al., 2015).

[2]Here is C^{k+1} since we use adjacent weighted tying.

We designed our architecture in this way because we expect the attention weights in the first and the last hop to show a "looser" and "sharper" distribution, respectively. To elaborate, the first hop focuses more on retrieving memory information and the last one tends to choose the exact token leveraging the pointer supervision. Hence, during training all the parameters are jointly learned by minimizing the sum of two standard cross-entropy losses: one between $P_{vocab}(\hat{y}_t)$ and $y_t \in Y$ for the vocabulary distribution, and one between $P_{ptr}(\hat{y}_t)$ and $ptr_t \in PTR$ for the memory distribution.

2.2.1 Sentinel

If the expected word is not appearing in the memories, then the P_{ptr} is trained to produce the sentinel token \$, as shown in Equation 1. Once the sentinel is chosen, our model generates the token from P_{vocab}, otherwise, it takes the memory content using the P_{ptr} distribution. Basically, the sentinel token is used as a hard gate to control which distribution to use at each time step. A similar approach has been used in (Merity et al., 2017) to control a soft gate in a language modeling task. With this method, the model does not need to learn a gating function separately as in Gulcehre et al. (2016), and is not constrained by a soft gate function as in See et al. (2017).

2.3 Memory Content

We store word-level content X in the memory module. Similar to Bordes and Weston (2017), we add temporal information and speaker information in each token of X to capture the sequential dependencies. For example, "hello $t1$ $\$u$" means "hello" at time step 1 spoken by a user.

On the other hand, to store B, the KB information, we follow the works of Miller et al. (2016); Eric et al. (2017) that use a (subject, relation, object) representation. For example, we represent the information of The Westin in Table 1: (The Westin, Distance, 5 miles). Thus, we sum word embeddings of the subject, relation, and object to obtain each KB memory representation. During decoding stage, the object part is used as the generated word for P_{ptr}. For instance, when the KB tuple (The Westin, Distance, 5 miles) is pointed, our model copies "5 miles" as an output word. Notice that only a specific section of the KB, relevant to a specific dialog, is loaded into the memory.

Task	1	2	3	4	5	DSTC2	In-Car
Avg. User turns	4	6.5	6.4	3.5	12.9	6.7	2.6
Avg. Sys turns	6	9.5	9.9	3.5	18.4	9.3	2.6
Avg. KB results	0	0	24	7	23.7	39.5	66.1
Avg. Sys words	6.3	6.2	7.2	5.7	6.5	10.2	8.6
Max. Sys words	9	9	9	8	9	29	87
Pointer Ratio	.23	.53	.46	.19	.60	.46	.42
Vocabulary	3747					1229	1601
Train dialogs	1000					1618	2425
Val dialogs	1000					500	302
Test dialogs	1000 + 1000 OOV					1117	304

Table 2: Dataset statistics for 3 different datasets.

3 Experimental Setup

3.1 Dataset

We use three public multi-turn task-oriented dialog datasets to evaluate our model: the bAbI dialog (Bordes and Weston, 2017), DSTC2 (Henderson et al., 2014) and In-Car Assistant (Eric et al., 2017). The train/validation/test sets of these three datasets are split in advance by the providers. The dataset statistics are reported in Table 2.

The bAbI dialog includes five end-to-end dialog learning tasks in the restaurant domain, which are simulated dialog data. Task 1 to 4 are about API calls, refining API calls, recommending options, and providing additional information, respectively. Task 5 is the union of tasks 1-4. There are two test sets for each task: one follows the same distribution as the training set and the other has out-of-vocabulary (OOV) entity values that does not exist in the training set.

We also used dialogs extracted from the Dialog State Tracking Challenge 2 (DSTC2) with the refined version from Bordes and Weston (2017), which ignores the dialog state annotations. The main difference with bAbI dialog is that this dataset is extracted from real human-bot dialogs, which is noisier and harder since the bots made mistakes due to speech recognition errors or misinterpretations.

Recently, In-Car Assistant dataset has been released. which is a human-human, multi-domain dialog dataset collected from Amazon Mechanical Turk. It has three distinct domains: calendar scheduling, weather information retrieval, and point-of-interest navigation. This dataset has shorter conversation turns, but the user and system behaviors are more diverse. In addition, the system responses are variant and the KB information is much more complicated. Hence, this dataset requires stronger ability to interact with KBs, rather than dialog state tracking.

3.2 Training

We trained our model end-to-end using Adam optimizer (Kingma and Ba, 2015), and chose learning rate between $[1e^{-3}, 1e^{-4}]$. The MemNNs, both encoder and decoder, have hops $K = 1, 3, 6$ to show the performance difference. We use simple greedy search and without any re-scoring techniques. The embedding size, which is also equivalent to the memory size and the RNN hidden size (i.e., including the baselines), has been selected between $[64, 512]$. The dropout rate is set between $[0.1, 0.4]$, and we also randomly mask some input words into unknown tokens to simulate OOV situation with the same dropout ratio. In all the datasets, we tuned the hyper-parameters with grid-search over the validation set, using as measure to the Per-response Accuracy for bAbI dialog and DSTC2, and BLEU score for the In-Car Assistant.

3.3 Evaluation Metrics

Per-response/dialog Accuracy: A generative response is correct only if it is exactly the same as the gold response. A dialog is correct only if every generated responses of the dialog are correct, which can be considered as the task-completion rate. Note that Bordes and Weston (2017) tests their model by selecting the system response from predefined response candidates, that is, their system solves a multi-class classification task. Since Mem2Seq generates each token individually, evaluating with this metric is much more challenging for our model.

BLEU: It is a measure commonly used for machine translation systems (Papineni et al., 2002), but it has also been used in evaluating dialog systems (Eric and Manning, 2017; Zhao et al., 2017) and chat-bots (Ritter et al., 2011; Li et al., 2016). Moreover, BLEU score is a relevant measure in task-oriented dialog as there is not a large variance between the generated answers, unlike open domain generation (Liu et al., 2016). Hence, we include BLEU score in our evaluation (i.e. using Moses `multi-bleu.perl` script).

Entity F1: We micro-average over the entire set of system responses and compare the entities in plain text. The entities in each gold system response are selected by a predefined entity list. This metric evaluates the ability to generate relevant entities from the provided KBs and to capture the semantics of the dialog (Eric and Manning, 2017; Eric et al., 2017). Note that the original In-Car Assis-

Task	QRN	MemNN	GMemNN	Seq2Seq	Seq2Seq+Attn	Ptr-Unk	**Mem2Seq H1**	**Mem2Seq H3**	**Mem2Seq H6**
T1	99.4 (-)	99.9 (99.6)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T2	99.5 (-)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T3	74.8 (-)	74.9 (2.0)	74.9 (0)	74.8 (0)	74.8 (0)	85.1 (19.0)	87.0 (25.2)	94.5 (59.6)	**94.7** (**62.1**)
T4	57.2 (-)	59.5 (3.0)	57.2 (0)	57.2 (0)	57.2 (0)	100 (100)	97.6 (91.7)	100 (100)	**100** (**100**)
T5	**99.6** (-)	96.1 (49.4)	96.3 (52.5)	98.8 (81.5)	98.4 (87.3)	99.4 (91.5)	96.1 (45.3)	98.2 (72.9)	97.9 (69.6)
T1-OOV	83.1 (-)	72.3 (0)	82.4 (0)	79.9 (0)	81.7 (0)	92.5 (54.7)	93.4 (60.4)	91.3 (52.0)	**94.0** (**62.2**)
T2-OOV	78.9 (-)	78.9 (0)	78.9 (0)	78.9 (0)	78.9 (0)	83.2 (0)	81.7 (1.2)	84.7 (7.3)	**86.5** (**12.4**)
T3-OOV	75.2 (-)	74.4 (0)	75.3 (0)	74.3 (0)	75.3 (0)	82.9 (13.4)	86.6 (26.2)	**93.2** (**53.3**)	90.3 (38.7)
T4-OOV	56.9 (-)	57.6 (0)	57.0 (0)	57.0 (0)	57.0 (0)	100 (100)	97.3 (90.6)	100 (100)	**100** (**100**)
T5-OOV	67.8 (-)	65.5 (0)	66.7 (0)	67.4 (0)	65.7 (0)	73.6 (0)	67.6 (0)	78.1 (0.4)	**84.5** (**2.3**)

Table 3: Per-response and per-dialog (in the parentheses) accuracy on bAbI dialogs. Mem2Seq achieves the highest average per-response accuracy and has the least out-of-vocabulary performance drop.

	Ent. F1	**BLEU**	**Per-Resp.**	**Per-Dial.**
Rule-Based	-	-	33.3	-
QRN	-	-	43.8	-
MemNN	-	-	41.1	0.0
GMemNN	-	-	**47.4**	1.4
Seq2Seq	69.7	55.0	46.4	**1.5**
+Attn	67.1	**56.6**	46.0	1.4
+Copy	71.6	55.4	47.3	1.3
Mem2Seq H1	72.9	53.7	41.7	0.0
Mem2Seq H3	**75.3**	55.3	45.0	0.5
Mem2Seq H6	72.8	53.6	42.8	0.7

Table 4: Evaluation on DSTC2. Seq2Seq (+attn and +copy) is reported from Eric and Manning (2017).

	BLEU	**Ent. F1**	**Sch. F1**	**Wea. F1**	**Nav. F1**
*Human**	13.5	60.7	64.3	61.6	55.2
*Rule-Based**	6.6	43.8	61.3	39.5	40.4
*KV Retrieval Net**	13.2	48.0	62.9	47.0	41.3
Seq2Seq	8.4	10.3	09.7	14.1	07.0
+Attn	9.3	19.9	23.4	25.6	10.8
Ptr-Unk	8.3	22.7	26.9	26.7	14.9
Mem2Seq H1	11.6	32.4	39.8	**33.6**	**24.6**
Mem2Seq H3	**12.6**	**33.4**	**49.3**	32.8	20.0
Mem2Seq H6	9.9	23.6	34.3	33.0	4.4

Table 5: Evaluation on In-Car Assistant. Human, rule-based and KV Retrieval Net evaluation (with *) are reported from (Eric et al., 2017), which are not directly comparable. Mem2Seq achieves highest BLEU and entity F1 score over baselines.

tant F1 scores reported in Eric et al. (2017) uses the entities in their canonicalized forms, which are not calculated based on real entity value. Since the datasets are not designed for slot-tracking, we report entity F1 rather than the slot-tracking accuracy as in (Wen et al., 2017; Zhao et al., 2017).

4 Experimental Results

We mainly compare Mem2Seq with hop 1,3,6 with several existing models: query-reduction networks (QRN, Seo et al. (2017)), end-to-end memory networks (MemNN, Sukhbaatar et al. (2015)), and gated end-to-end memory networks (GMemNN, Liu and Perez (2017)). We also implemented the following baseline models: standard sequence-to-sequence (Seq2Seq) models with and without attention (Luong et al., 2015), and pointer to unknown (Ptr-Unk, Gulcehre et al. (2016)). Note that the results we listed in Table 3 and Table 4 for QRN are different from the original paper, because based on their released code, [3] we discovered that the per-response accuracy was not correctly computed.

bAbI Dialog: In Table 3, we follow Bordes and Weston (2017) to compare the performance based on per-response and per-dialog accuracy. Mem2Seq with 6 hops can achieve per-response 97.9% and per-dialog 69.6% accuracy in T5, and 84.5% and 2.3% for T5-OOV, which surpass existing methods by far. One can find that in T3 especially, which is the task to recommend restaurant based on their ranks, our model can achieve promising results due to the memory pointer. In terms of per-response accuracy, this indicates that our model can generalize well with few performance loss for test OOV data, while others have around 15-20% drop. The performance gain in OOV data is also mainly attributed to the use of copy mechanism. In addition, the effectiveness of hops is demonstrated in tasks 3-5, since they require reasoning ability over the KB information. Note that QRN, MemNN and GMemNN viewed bAbI dialog tasks as classification problems. Although their tasks are easier compared to our generative methods, Mem2Seq models can still overpass the performance. Finally, one can find that Seq2Seq and Ptr-Unk models are also strong baselines, which further confirms that generative methods can also achieve good performance in task-oriented dialog systems (Eric and Manning, 2017).

[3] We simply modified the evaluation part and reported the results. (https://github.com/uwnlp/qrn)

DSTC2: In Table 4, the Seq2Seq models from Eric and Manning (2017) and the rule-based from Bordes and Weston (2017) are reported. Mem2Seq has the highest 75.3% entity F1 score and an high of 55.3 BLEU score. This further confirms that Mem2Seq can perform well in retrieving the correct entity, using the multiple hop mechanism without losing language modeling. Here, we do not report the results using match type (Bordes and Weston, 2017) or entity type (Eric and Manning, 2017) feature, since this meta-information are not commonly available and we want to have an evaluation on plain input output couples. One can also find out that, Mem2Seq comparable per-response accuracy (i.e. 2% margin) among other existing solution. Note that the per-response accuracy for every model is less than 50% since the dataset is quite noisy and it is hard to generate a response that is exactly the same as the gold one.

In-Car Assistant: In Table 5, our model can achieve highest 12.6 BLEU score. In addition, Mem2Seq has shown promising results in terms of Entity F1 scores (33.4%), which are, in general, much higher than those of other baselines. Note that the numbers reported from Eric et al. (2017) are not directly comparable to ours as we mention below. The other baselines such as Seq2Seq or Ptr-Unk especially have worse performances in this dataset since it is very inefficient for RNN methods to encode longer KB information, which is the advantage of Mem2Seq.

Furthermore, we observe an interesting phenomenon that humans can easily achieve a high entity F1 score with a low BLEU score. This implies that stronger reasoning ability over entities (hops) is crucial, but the results may not be similar to the golden answer. We believe humans can produce good answers even with a low BLEU score, since there could be different ways to express the same concepts. Therefore, Mem2Seq shows the potential to successfully choose the correct entities.

Note that the results of KV Retrieval Net baseline reported in Table 5 come from the original paper (Eric et al., 2017) of In-Car Assistant, where they simplified the task by mapping the expression of entities to a canonical form using named entity recognition (NER) and linking. Hence the evaluation is not directly comparable to our system. For example, their model learned to generate responses such as "You have a football game at football_time with football_party," instead of generating a sentence such as "You have a football game at 7 pm with John." Since there could be more than one football_party or football_time, their model does not learn how to access the KBs, but it rather learns the canonicalized language model.

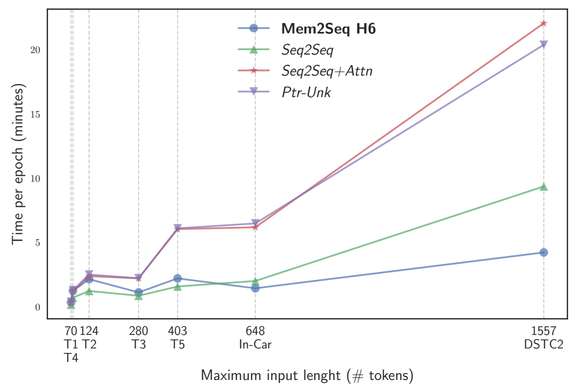


Figure 2: Training time per-epoch for different tasks (lower is better). The speed difference becomes larger as the maximal input length increases.

Time Per-Epoch: We also compare the training time [4] in Figure 2. The experiments are set with batch size 16, and we report each model with the hyper-parameter that can achieved the highest performance. One can observe that the training time is not that different for short input length (bAbI dialog tasks 1-4) and the gap becomes larger as the maximal input length increases. Mem2Seq is around 5 times faster in In-Car Assistant and DSTC2 compared to Seq2Seq with attention. This difference in training efficiency is mainly attributed to the fact that Seq2Seq models have input sequential dependencies which limit any parallelization. Moreover, it is unavoidable for Seq2Seq models to encode KBs, instead Mem2Seq only encodes with dialog history.

5 Analysis and Discussion

Memory Attention: Analyzing the attention weights has been frequently used to show the memory read-out, since it is an intuitive way to understand the model dynamics. Figure 3 shows the attention vector at the last hop for each generated token. Each column represents the P_{ptr} vector at the corresponding generation step. Our model has a sharp distribution over the memory, which im-

[4]Intel(R) Core(TM) i7-3930K CPU@3.20GHz, using a GeForce GTX 1080 Ti

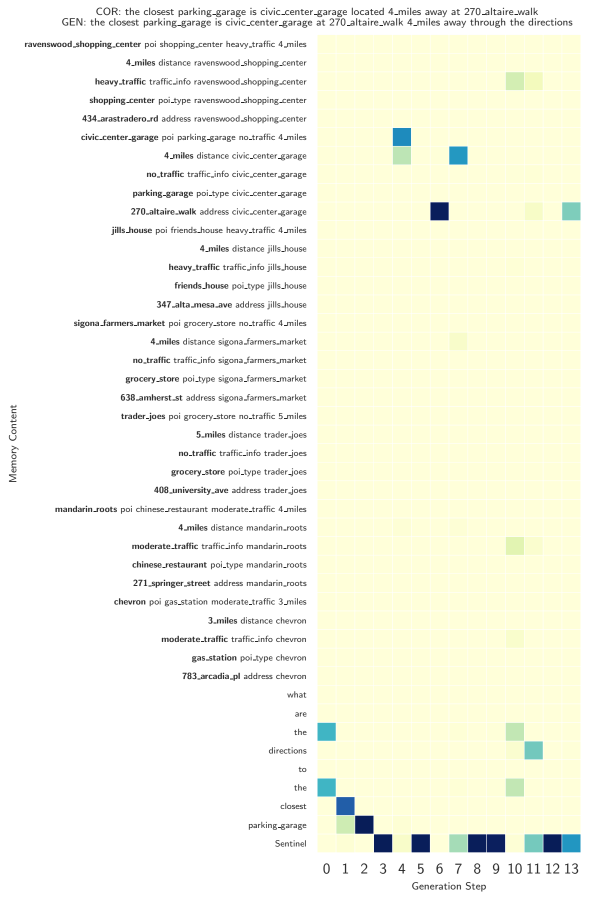


Figure 3: Last hop memory attention visualization from the In-Car dataset. COR and GEN on the top are the correct response and our generated one.

plies that it is able to select the right token from the memory. For example, the KB information "270_altarie_walk" was retrieved at the sixth step, which is an address for "civic_center_garage". On the other hand, if the sentinel is triggered, then the generated word comes from vocabulary distribution P_{vocab}. For instance, the third generation step triggered the sentinel, and "is" is generated from the vocabulary as the word is not present in the dialog history.

Multiple Hops: Mem2Seq shows how multiple hops improve the model performance in several datasets. Task 3 in the bAbI dialog dataset serves as an example, in which the systems need to recommend restaurants to users based on restaurant ranking from highest to lowest. Users can reject the recommendation and the system has to reason over the next highest restaurant. We found out there are two common patterns between hops among different samples: 1) the first hop is usually used to score all the relevant memories and

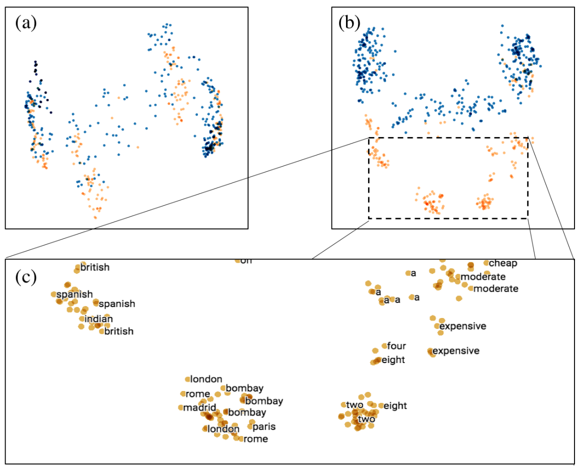


Figure 4: Principal component analysis of query vectors in hop (a) 1 and (b) 6 for bAbI dialog.

room	time	date	party	event
conference room 100	3pm	Thursday	sales team	conference
-	11am	Tuesday	-	tennis activity
-	7pm	Wednesday	-	dentist appointment
conference room 100	1pm	Tuesday	HR	meeting
-	7pm	Friday	-	doctor appointment
-	11am	Tuesday	-	lab appointment

DRIVER	When's my tennis match?
Seq2seq	Your lab appointment is on Monday at
+Att	Your lab appointment is on Tuesday at 11am
Ptr-Unk	Your tennis match is on PAD at 1pm
Mem2Seq H1	Your is very welcome
Mem2Seq H3	Your tennis is on Tuesday at 11am
Mem2Seq H6	Your tennis is Monday at 11am
GOLD	Your tennis match is Tuesday at 11am

Table 6: Example of generated responses for the In-Car Assistant on the scheduling domain.

retrieve information; 2) the last hop tends to focus on a specific token and makes mistakes when the attention is not sharp. Such mistakes can be attributed to lack of hops, for some samples. For more information, we report two figures in the supplementary material.

Query Vectors: In Figure 4, the principal component analysis of Mem2Seq queries vectors is shown for different hops. Each dot is a query vector h_t during each decoding time step, and it has its corresponding generated word y_t. The blue dots are the words generated from P_{vocab}, which triggered the sentinel, and orange ones are from P_{ptr}. One can find that in (a) hop 1, there is no clear separation of two different colors but each of which tends to group together. On the other hand, the separation becomes clearer in (b) hop 6 as each color clusters into several groups such as location, cuisine, and number. Our model tends to retrieve more information in the first hop, and points into the memories in the last hop.

Examples: Table 1 and 6 show the generated responses of different models in the two test set samples from the In-Car Assistant dataset. We report examples from this dataset since their answers are more human-like and not as structured and repetitive as others. Seq2Seq generally cannot produce related information, and sometimes fail in language modeling. Instead, using attention helps with this issue, but it still rarely produces the correct entities. For example, Seq2Seq with attention generated 5 miles in Table 1 but the correct one is 4 miles. In addition, Ptr-Unk often cannot copy the correct token from the input, as shown by "PAD" in Table 1. On the other hand, Mem2Seq is able to produce the correct responses in this two examples. In particular in the navigation domain, shown in Table 1, Mem2Seq produces a different but still correct utterance. We report further examples from all the domains in the supplementary material.

Discussions: Conventional task-oriented dialog systems (Williams and Young, 2007), which are still widely used in commercial systems, require a multitude of human efforts in system designing and data collection. On the other hand, although end-to-end dialog systems are not perfect yet, they require much less human interference, especially in the dataset construction, as raw conversational text and KB information can be used directly without the need of heavy preprocessing (e.g. NER, dependency parsing). To this extent, Mem2Seq is a simple generative model that is able to incorporate KB information with promising generalization ability. We also discovered that the entity F1 score may be a more comprehensive evaluation metric than per-response accuracy or BLEU score, as humans can normally choose the right entities but have very diversified responses. Indeed, we want to highlight that humans may have a low BLEU score despite their correctness because there may not be a large n-gram overlap between the given response and the expected one. However, this does not imply that there is no correlation between BLEU score and human evaluation. In fact, unlike chat-bots and open domain dialogs where BLEU score does not correlate with human evaluation (Liu et al., 2016), in task-oriented dialogs the answers are constrained to particular entities and recurrent patterns. Thus, we believe BLEU score still can be considered as a relevant measure. In future works, several methods could be applied (e.g. Reinforcement Learning (Ranzato et al., 2016), Beam Search (Wiseman and Rush, 2016)) to improve both responses relevance and entity F1 score. However, we preferred to keep our model as simple as possible in order to show that it works well even without advanced training methods.

6 Related Works

End-to-end task-oriented dialog systems train a single model directly on text transcripts of dialogs (Wen et al., 2017; Serban et al., 2016; Williams et al., 2017; Zhao et al., 2017; Seo et al., 2017; Serban et al., 2017). Here, RNNs play an important role due to their ability to create a latent representation, avoiding the need for artificial state labels. End-to-End Memory Networks (Bordes and Weston, 2017; Sukhbaatar et al., 2015), and its variants (Liu and Perez, 2017; Wu et al., 2017, 2018) have also shown good results in such tasks. In each of these architectures, the output is produced by generating a sequence of tokens, or by selecting a set of predefined utterances.

Sequence-to-sequence (Seq2Seq) models have also been used in task-oriented dialog systems (Zhao et al., 2017). These architectures have better language modeling ability, but they do not work well in KB retrieval. Even with sophisticated attention models (Luong et al., 2015; Bahdanau et al., 2015), Seq2Seq fails to map the correct entities to the generated input. To alleviate this problem, copy augmented Seq2Seq models Eric and Manning (2017), were used. These models outperform utterance selection methods by copying relevant information directly from the KBs. Copy mechanisms has also been used in question answering tasks (Dehghani et al., 2017; He et al., 2017), neural machine translation (Gulcehre et al., 2016; Gu et al., 2016), language modeling (Merity et al., 2017), and summarization (See et al., 2017).

Less related to dialog systems, but related to our work, are the memory based decoders and the non-recurrent generative models: 1) Mem2Seq query generation phase used to access our memories can be seen as the memory controller used in Memory Augmented Neural Networks (MANN) (Graves et al., 2014, 2016). Similarly, memory encoders have been used in neural machine translation (Wang et al., 2016), and meta-learning application (Kaiser et al., 2017). However, Mem2Seq differs from these models as such: it uses multi-hop attention in combination with copy mechanism, whereas other models use a single matrix representation. 2) non-recurrent generative models (Vaswani et al., 2017), which only rely on self-attention mechanism, are related to the multi-hop attention mechanism used in MemNN.

7 Conclusion

In this work, we present an end-to-end trainable Memory-to-Sequence model for task-oriented dialog systems. Mem2Seq combines the multi-hop attention mechanism in end-to-end memory networks with the idea of pointer networks to incorporate external information. We empirically show our model's ability to produce relevant answers using both the external KB information and the predefined vocabulary, and visualize how the multi-hop attention mechanisms help in learning correlations between memories. Mem2Seq is fast, general, and able to achieve state-of-the-art results in three different datasets.

Acknowledgments

This work is partially funded by ITS/319/16FP of Innovation Technology Commission, HKUST 16214415 & 16248016 of Hong Kong Research Grants Council, and RDC 1718050-0 of EMOS.AI.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *International Conference on Learning Representations*, abs/1605.07683.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS Deep Learning and Representation Learning Workshop*.

Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1747–1756, New York, NY, USA. ACM.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49. Association for Computational Linguistics.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473, Valencia, Spain. Association for Computational Linguistics.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 199–208, Vancouver, Canada. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Chiori Hori, Kiyonori Ohtake, Teruhisa Misu, Hideki Kashioka, and Satoshi Nakamura. 2009. Statistical dialog management applied to wfst-based dialog systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.*, pages 4793–4796. IEEE.

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events.

International Conference on Learning Representations.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *International Conference on Learning Representations*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *International Conference on Learning Representations*.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. *International Conference on Learning Representations*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 278–286, Austin, Texas. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677, Vancouver, Canada. Association for Computational Linguistics.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Genta Winata, and Pascale Fung. 2017. End-to-end recurrent entity network for entity-value independent goal-oriented dialog learning. In *Dialog System Technology Challenges Workshop, DSTC6*.

Chien-Sheng Wu, Andrea Madotto, Genta Winata, and Pascale Fung. 2018. End-to-end dynamic query memory network for entity-value independent task-oriented dialog. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 27–36. Association for Computational Linguistics.

Tailored Sequence to Sequence Models to Different Conversation Scenarios

Hainan Zhang, Yanyan Lan, Jiafeng Guo, Jun Xu and Xueqi Cheng
University of Chinese Academy of Sciences, Beijing, China
CAS Key Lab of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences
zhanghainan@software.ict.ac.cn, {lanyanyan, guojiafeng, junxu, cxq}@ict.ac.cn

Abstract

Sequence to sequence (Seq2Seq) models have been widely used for response generation in the area of conversation. However, the requirements for different conversation scenarios are distinct. For example, customer service requires the generated responses to be specific and accurate, while chatbot prefers diverse responses so as to attract different users. The current Seq2Seq model fails to meet these diverse requirements, by using a general average likelihood as the optimization criteria. As a result, it usually generates safe and commonplace responses, such as *'I don't know'*. In this paper, we propose two tailored optimization criteria for Seq2Seq to different conversation scenarios, i.e., the maximum generated likelihood for specific-requirement scenario, and the conditional value-at-risk for diverse-requirement scenario. Experimental results on the Ubuntu dialogue corpus (Ubuntu service scenario) and Chinese Weibo dataset (social chatbot scenario) show that our proposed models not only satisfies diverse requirements for different scenarios, but also yields better performances against traditional Seq2Seq models in terms of both metric-based and human evaluations.

1 Introduction

This paper focuses on the problem of the single-turn dialogue generation, which is critical in many natural language processing applications such as customer services, intelligent assistant and chatbot. Recently, sequence to sequence (Seq2Seq) models (Sutskever et al., 2014) have been widely used in this area. In these Seq2Seq models, a recurrent neural network (RNN) based encoder is first utilized to encode the input post to a vector, and another RNN decoder is then used to automatically generate the response word by word. The parameters of the encoder and decoder are learned by maximizing the averaged likelihood of the training data.

It is clear that the requirements for generated responses are distinct in different dialogue scenarios. For instance, in the scenario of customer service or mobile assistant, users mainly expect the system to help them solve a problem. Therefore, the responses should be specific and accurate to provide useful assistance. For example, if the user asks a question *'How can I get the AMD driver running on Ubuntu 12.10?'*, the system is expected to reply *'The fglrx driver is in the repo. But it may depend on your exact chipset.'*, rather than *'I do not know about the package.'*, even though the latter can also be viewed as relevant for the proposed question. We called this kind of scenario as specific-requirement scenario. While in other scenarios such as chatbot, users are interacting with the dialogue system for fun. Therefore, the generated responses should be diverse to attract different users. Take the post *'Can you recommend me a tourist city?'* as an example. If the user prefers the magnificent mountains and rivers, it is better to reply *'You may like the Bernina Express to the Alps'*. While if the user loves literature, it is better to reply *'Paris is a beautiful city with full of the literary atmosphere'*. This kind of scenario is called diverse-requirement scenario.

However, the current generation model Seq2Seq (Sutskever et al., 2014) usually tend to generate common responses, such as *'I don't know'* and *'What does this mean?'* (Li et al., 2016a,b; Zhou et al., 2017), which fails to meet diverse requirements for different conversation scenarios. Intrinsically, conversation is a typical one-to-many application, i.e., multiple responses with different semantic meanings are correspondent to a same post. That means there are various post-response matching patterns in the training data. Seq2Seq optimizes an averaged likelihood, so it can only capture the common matching patterns, leading to common responses.

The purpose of this paper is to propose two tailored optimization criteria for Seq2Seq models to accommodate different conversation scenarios, i.e. specific-requirement scenario and diverse-requirement scenario. The key idea is to how capture the required post-response matching patterns. For the specific-requirement scenario, we define the maximum generated likelihood as the objective function. With this kind of criterion, we just require one ground-truth response to be close to the given post, instead of requiring the average of multiple ground-truth responses to be close to the post. Therefore, the most significant post-response matching pattern will be learned from the data, to facilitate generating a specific response. While for the diverse-requirement scenario, the conditional value-at-risk (CVaR) is used as the objective function. CVaR is a risk-sensitive function widely used in finances (Rockafellar and Uryasev, 2002; Alexander et al., 2006; Chen et al., 2015), defined to assessing the likelihood (at a specific confidence level) that a specific loss will exceed the value at risk. With CVaR as the objective function, the worst 1-α responses are required to be close to the post, therefore various post-response patterns can be captured, and the learned model has the ability to generate diverse responses.

We use public data to evaluate our proposed models. For the specific-requirement scenario, the experiments on public Ubuntu dialogue corpus(Ubuntu service) show that optimizing the maximum generated likelihood produces more specific and accurate responses than traditional Seq2Seq models. While for the diverse-requirement scenario, the experiments on the public Chinese Weibo dataset (social chatbot) show that optimizing CVaR produces diverse responses, as compared with Seq2Seq and the variants.

2 Related Work

The basic neural-based Seq2Seq framework for dialogue generation is inspired by the studies of statistical machine translation. Sutskever et al. (Sutskever et al., 2014) proposed the original Seq2Seq framework(Seq2Seq), which used a multilayered Long Short-Term Memory(LSTM) to map the input sequence to a fixed dimension vector and then used another LSTM to decode the target sequence from the vector. Then Cho et al. (Cho et al., 2014) followed the above architecture, and proposed to feed the last hidden state of encoder to every cell of decoder(RNN-encdec), which enhanced the influence of contexts in generating each word of the targets. To further alleviate the long dependency problem, Bahdanau et al. (Bahdanau et al., 2015) introduced the attention mechanism into the neural network and achieved encouraging performances(Seq2Seq-att). Many studies (Shang et al., 2015; Vinyals and Le, 2015) directly applied the above neural SMT models to the task of dialogue generation, and gained some promising performances.

Although the current Seq2Seq model is capable to generate fluent responses, these responses are usually general. Therefore, many researchers focused on how to improve the generation quality and specification. Li et al. (Li et al., 2016a) proposed a mutual information model(MMI) to tackle this problem. However, it is not a unified training model, instead it still trained original Seq2Seq model, and used the Maximum Mutual Information criterion only for testing to rerank the primary top-n list. Mou et al. (Mou et al., 2017) proposed a forward-backward keyword method which used a pointwise mutual information to predict a noun as a keyword and then used two Seq2Seq models to generate the forward sentence and the backward sentence. Xing et al. (Xing et al., 2017) proposed a joint attention mechanism model, which modified the generation probability by adding the topic keywords likelihood to the generated maximum likelihood with extra corpus. The recent works such as seqGAN (Yu et al., 2017) and Adver-REGS (Li et al., 2017) try to use Generative Adversarial Networks(GAN) for generation, where the discriminator scores are used as rewards for reinforcement learning.

For the study of generating diverse responses, Vijayakumar et al. (Vijayakumar et al., 2016) introduced a diverse beam search which decoded a list of diverse outputs by optimizing for a diversity-augmented objective, which can control for the exploration and exploitation of the search space. Zhou (Zhou et al., 2017) proposed to apply a hidden state as a generating style(Mechanism). They make an assumption that some latent responding mechanisms can generate different responses, and model these mechanisms as latent embedding. With these latent embedding in the mid of Seq2Seq, the mechanism-aware Seq2Seq can generate different mechanism responses.

However, most of these models are using an averaged approach for optimization, similar to that in Seq2Seq. This paper proposes two new criteria for different conversation scenarios. For the specific-requirement scenario, the maximum generated likelihood is used as the objective function. While for the diverse-requirement scenario, CVaR is used for optimization.

3 Sequence to Sequence Models

We first introduce the typical LSTM-based Seq2Seq framework (Bahdanau et al., 2015) used in dialogue generation.

Given a post $X = \{x_1, \dots, x_M\}$ as the input, a standard LSTM first maps the input sequence to a fixed-dimension vector h_M as follows.

$$\begin{aligned} i_k &= \sigma(W_i[h_{k-1}, w_k]),\ f_k = \sigma(W_f[h_{k-1}, w_k]), \\ o_k &= \sigma(W_o[h_{k-1}, w_k]),\ l_k = \tanh(W_l[h_{k-1}, w_k]), \\ c_k &= f_k c_{k-1} + i_k l_k, \qquad h_i = o_k \tanh(c_k), \end{aligned} \tag{1}$$

where i_k, f_k and o_k are the input gate, the memory gate, and the output gate, respectively. w_k is the word embedding for x_k, and h_k stands for the vector computed by LSTM at time k by combining w_k and h_{k-1}. c_k is the cell at time k, and σ denotes the sigmoid function. W_i, W_f, W_o and W_l are parameters.

Then another LSTM is used as the decoder to map the vector h_M to the ground-truth response $Y = \{y_1, \cdots, y_N\}$. Typically, the decoder is trained to predict the next word g_i, given the context vector h_M and the previous generated words $\{g_1, \dots, g_{i-1}\}$. In other words, the decoder defines a probability over the output Y by decomposing the joint probability into the ordered conditionals by chain rule in the probability theory:

$$\begin{aligned} P(Y|X) &= \prod_{i=1}^{N} p(y_i | h_M, y_1, \dots, y_{i-1}) \\ &= \prod_{i=1}^{N} g(h_M, y_{i-1}, h'_i), \end{aligned}$$

where g_θ is a softmax function, h'_i is the hidden state in the decoder LSTM.

Usually the attention mechanism is further introduced to the above Seq2Seq framework in real applications. Instead of using h_M as the context vector in the decoder, we let the context vector, denoted as s_i, to be dependent on the sequence $(h_1, \cdots, h_M)$. Each h_k contains information about the input sequence with a strong focus on the parts surrounding the k-th word of the input sentence. The context vector s_i is then computed as a weighted sum of these h_k:

$$s_i = \sum_{k=1}^{M} \alpha_{ik} h_k.$$

The weight α_{ik} of each representation h_k is computed by:

$$\begin{aligned} \alpha_{ik} &= \frac{\exp(e_{ik})}{\sum_{j=1}^{M} \exp(e_{ij})}, \\ e_{ik} &= v^T \tanh(W_1 h'_{i-1} + W_2 h_k), \end{aligned}$$

where v^T,W_1 and W_2 are learned parameters. e_{ik} is an alignment model which scores how well the inputs around position k and the output at position i match. The score is based on the LSTM hidden state h'_{i-1} (just before emitting y_i), and h_k of the input sentence.

Given a set of training data $\mathcal{D}$, Seq2Seq assumes that data are i.i.d. sampled from a probability P, and uses the following log likelihood as the objective for maximization:

$$\mathcal{L} = \sum_{(X,Y)\in\mathcal{D}} \log P(Y|X). \tag{2}$$

4 Tailored Sequence to Sequence Models

We can see that a general averaged likelihood of the training data is used as the objective function in Seq2Seq. However, this objective function is usually criticized for generating common responses, such as *'I don't know'* and *'What does this mean?'*. Clearly, this kind of responses cannot satisfy either the specific or the diverse requirements. The underlying reason is not difficult to understand. Intrinsically, conversation is a typical one-to-many application, i.e., multiple responses with different semantic meanings are correspondent to a same post. That means there are various post-response matching patterns in the training data. If we optimize an averaged likelihood, we can only capture the common matching patterns, which leads to generating common responses. Therefore, if we want to generate specific responses, we need to capture the most significant matching pattern; while if we want to generate diverse responses, we need to define a criteria which has the ability to capture the various matching patterns. Motivated by this idea, we propose two optimization criteria, i.e. maximum generated likelihood, and CVaR, to adapt two different scenarios.

4.1 Maximum Generated Likelihood Criteria

To meet the specific requirement, we need to capture a specific matching pattern between post and response, rather than the common matching pattern. Therefore, instead of optimizing the averaged likelihood, we turn to use the maximum generated likelihood (MGL) as the objective function. Mathematically, for a given post X and its associated ground-truth responses $(Y_X^{(1)}, Y_X^{(2)}, \cdots, Y_X^{(m_X)})$, the objective function is defined as:

$$\mathcal{L} = \sum_X \max_{k=1}^{m_X} \log P(Y_X^{(k)}|X).$$

From the definition, we can see that we aim to capture the most significant post-response matching pattern in the training data. Therefore, the learned model can output specific responses for a given post. Since there is a max operator in the objective function, which is difficult for accurate optimization, we approximate it by the softmax function. Then the objective function becomes the following form:

$$\mathcal{L} = \sum_X \sum_{k=1}^{m_X} \log \frac{P(Y_X^{(k)}|X)}{\sum_{j=1}^{m_X} P(Y_X^{(j)}|X)}.$$

If the probability for one ground-truth $Y_X^{(k)}$ is small, it contributes little to the objective function. That is to say, we just require the top ground-truth responses with relative large probabilities to be close to the post.

4.2 CVaR Criteria

To meet the diverse requirements, we need to capture various matching patterns between post and its multiple ground-truth responses. Therefore, instead of optimizing the averaged likelihood, we turn to optimize the conditional value-at-risk, named CVaR for short. CVaR is a prominent risk measure used extensively in finance, and it is proved to be coherent (Artzner et al., 1999) and numerically effective (Krokhmal et al., 2002; Uryasev, 2013).

The definitions of VaR and CVaR are as follows. For a confidence level $\alpha \in [0, 1]$, and a continuous random cost Z whose distribution is parameterized by a controllable parameter θ, the α-VaR of the cost Z, denoted by $\nu_\alpha(\theta)$, is defined as:

$$\nu_\alpha(\theta) = \inf\{\nu \in \mathbb{R} | P(Z \leq \nu) \geq \alpha\}.$$

α-VaR denotes the maximum cost that might be incurred with probability at least α, or can be simply regarded as the α-quantile of Z. And the α-CVaR, denoted by $\Phi_\alpha(\theta)$, is defined as:

$$\Phi_\alpha(\theta) = \frac{1}{1-\alpha} \int_\alpha^1 \nu_r(\theta) dr = \mathbb{E}^\theta[Z | Z \geq \nu_\alpha(\theta)].$$

It can be viewed as the expected cost over the $(1-\alpha)$ worst outcomes of Z.

Applying CVaR to generating diverse responses, we can define the random cost Z as $-\log P(Y|X)$, the corresponding CVaR is:

$$\Phi_\alpha(\theta) = \frac{1}{1-\alpha} \int_\alpha^1 \nu_r(\theta) dr,$$

where $\nu_r(\theta) = \inf\{\nu \in \mathbb{R} | P(-\log P(Y|X) \leq \nu) \geq r\}$, and θ are parameters of the Seq2Seq model. Therefore, we have:

$$\nu_r(\theta) = \inf\{\nu \in \mathbb{R} | P(P(Y|X) \geq e^\nu) \geq r\}.$$

Therefore, for a given post X and its ground-truth responses $(Y_X^{(1)}, Y_X^{(2)}, \cdots, Y_X^{(m_X)})$, optimizing CVaR is equivalent to maximizing the following objective function:

$$\mathcal{L} = \sum_X \frac{1}{1-\alpha} \sum_{Y_X^{(k)} \in \mathcal{Y}_{1-\alpha}} P(Y_X^{(k)}|X),$$

where $\mathcal{Y}_{1-\alpha}$ is a collection of ground-truth responses such that:

$$\sup\{P(Y_X^{(i)}|X) : Y_X^i \in \mathcal{Y}_{1-\alpha}\} \leq \alpha.$$

We can see that maximizing the above objective function requires the worst $1-\alpha$ responses to be close to the post. Therefore, we aim to capture each distinct post-response matching pattern by optimizing the CVaR criteria, which can meet the requirement for generating diverse responses.

5 Experiments

In this section, we conduct experiments on both specific-requirement and diverse-requirement scenarios, to evaluate the performances of our proposed methods.

5.1 Experimental Settings

5.1.1 Datasets

We use two public datasets in our experiments. For the specific-requirement scenario, we use the Ubuntu dialogue corpus[1] extracted from Ubuntu question-answering forum, named Ubuntu (Lowe et al., 2015). The original training data consists of 7 million conversational post-responses pairs from 2014 to April 27,2012. The validation data are conversational pairs from April 27,2014 to August 7,2012, and the test data are from August 7,2012 to December 1,2012. We set the number of positive examples as 4,000,000 in the Github to directly sample data from the whole corpus. Then we construct post and response pairs based on the period from both context and utterance. We also conduct some data pro-processing. For example, we use the official script to tokenize, stem and lemmatize, and the duplicates and sentences with length less than 5 or longer than 50 are removed. Finally, we obtain 3,200,000, 100,000 and 100,000 for training, validation and testing, respectively.

For the diverse-requirement scenario, we use the Chinese Weibo dataset, named STC (Shang et al., 2015). It consists of 3,788,571 post-response pairs extracted from the Chinese Weibo website and cleaned by the data publishers. We randomly split the data to training, validation, and testing sets, which contains 3,000,000, 388,571 and 400,000 pairs, respectively. [2]

5.1.2 Baseline Methods

Six baseline methods are used for comparison, including traditional Seq2Seq (Sutskever et al., 2014), RNN-encdec (Cho et al., 2014), Seq2Seq with attention(Seq2Seq-att) (Bahdanau et al., 2015), mutual information(MMI) (Li et al., 2016b), Adver-REGS (Li et al., 2017) and Mechanism model (Zhou et al., 2017). Here are some empirical settings. We first introduce the input embeddings. For STC, we utilize character-level embeddings rather than word-level embeddings, due to the word sparsity, segmentation mistakes and unknown Chinese words which may lead to inferior performance (Hu et al., 2015). For Ubuntu, we use word embeddings trained by word2vec on the training dataset. In the training process, the dimension is set to be 300, the size of negative sample is set to be 3, and the learning rate is 0.05. For fair comparison among all the baseline methods and our methods, the number of hidden nodes is all set to 300, and batch size is set to 200. Stochastic gradient decent (SGD) is utilized in our experiment for optimization, instead of Adam, because SGD yields better performances in our experiments. The learning rate is set to be 0.5, and adaptively decays with rate 0.99 in the optimization process. We run our model on a Tesla K80 GPU card with Tensorflow framework. All the methods are pretrained with the same Seq2Seq model. For maximum generated likelihood(MGL) model, some people may argue that the specific results may be due to the usage of single post-response pair. Thus we also implement the baseline of using a single post-response pair, by random selecting the response from the ground-truth for each post, denoted as Single Model.

5.1.3 Evaluation Measures

We use both quantitative metrics and human judgements to evaluate the proposed MGL model and the CVaR model. Specifically, we use two kinds of metrics for quantitative comparisons. The first one kind is the traditional metric, including PPL and Bleu score (Xing et al., 2017). They are both widely used in natural language processing, and here we use them to evaluate the quality of the generated responses. The other kind is to evaluate the specific degree[3] in (Li et al., 2016a,b). It measures the specific degree of the generated responses, by calculating the number of distinct unigrams and bigrams in the generated responses, denoted as *distinct*. If a model usually generates common responses, the *distinct* will be low.

For the diverse-requirement scenario, we define two measures to evaluate the performance. Specifically, we set the beam as 10. Group-diversity is

[1]https://github.com/rkadlec/ubuntu-ranking-dataset-creator

[2]https://github.com/zhanghainan/TailoredSeq2Seq2DifferentConversationScenarios

[3]Though it is named as diversity in Li's paper, this diversity is not the same as that used in our paper. This diversity measures the specific degree of the generated responses over all generations. While the diversity used in our paper means that the responses are required to be relevant to a post from different aspects.

model	distinct-1	distinct-2	BLEU	PPL
Seq2Seq	0.140	1.11	1.231	51.26
RNN-encdec	0.125	1.24	1.231	46.97
Seq2Seq-att	0.351	4.36	1.294	47.84
MMI	0.283	4.84	1.297	42.52
Adver-REGS	0.268	5.07	1.279	37.71
Single	0.324	5.27	1.342	30.36
MGL	**0.358**	**6.30**	**1.354**	**26.34**
CVaR	0.294	5.52	1.290	30.03

Table 1: The metric-based evaluation results(%) of different models on Ubuntu.

model	human score distribution(%)			Ave.	Kappa
	1	2	3		
Seq2Seq-att	46.5	38.6	14.9	1.684	0.387
MMI	42	38	20	1.78	0.395
Adver-REGS	42	26	32	1.9	0.379
Single	49	14	37	1.88	0.383
MGL	33	16	51	2.18	0.372
CVaR	40	12	48	2.08	0.381

Table 2: The comparisons of different models by human evaluation on Ubuntu.

defined to calculate the difference between each two generations for one post, denoted as divrs. Group-overlap is defined to calculate the overlap between each two generations for one post, denoted as overlap. The detailed definitions are shown as follows.

$$divrs = \frac{1}{N}\sum_{i=1}^{N}\sum_{X_i} cosine(G_{i1}, G_{i2}),$$

$$overlap = \frac{1}{N}\sum_{i=1}^{N}\sum_{X_i} overlap(G_{i1}, G_{i2}),$$

where G_{i1} and G_{i2} are the generated responses from the model for post X, $cosine(G_{i1}, G_{i2})$ is the cosine similarity, and the $overlap(G_{i1}, G_{i2})$ is defined as the intersection divided by union.

For human evaluation, given 200 randomly sampled post and it's generated responses, three annotators, randomly selected from a class of computer science majored students(48 students), are required to give 3-graded judgements. The annotation criteria are defined as follows:

1. the response is nonfluent or has wrong logic; or the response is fluent but not related with the post;
2. the response is fluent and weak related, but it's common which can reply many other posts;
3. the response is fluent and strong related with its post, which is like following a real person's tone.

5.2 Specific-Requirement Scenario

We demonstrate the experimental results on the specific-requirement scenario, based on the Ubuntu dataset.

5.2.1 Metric-based Evaluation

The quantitative evaluation results are shown in Table 1. From the results, we can see that both MMI and Adver-REGS outperform Seq2Seq baselines in terms of BLUE, PPL and *distinct* measures. That's because both MMI and Adver-REGS further consider some reward functions in the optimization process to encourage specific results. Specifically, MMI uses a predefined reward function to penalize generating common responses, and Adver-REGS uses a learned discriminator to define the reward function. Our MGL model obtains higher BLEU and lower PPL than baseline models. Take the BLEU score on Ubuntu dataset for example, the BLEU score of MGL model is 1.354, which is significantly better than that of MMI and Adver-REGS, i.e., 1.297 and 1.279. These results indicate that our MGL generates responses with higher quality. When compared with the Single model, MGL is also better because MGL considers more data in the model computation process. The *distinct* scores of MGL are also higher than baseline models, which indicate that our model can generate more specific responses. That's because it has the ability to learn the significant matching pattern between post and responses, by optimizing the maximum generated likelihood rather than the averaged one. In summary, our maximum generated likelihood model produces more fluent and specific results, as compared with baseline methods.

5.2.2 Human Evaluation

The human evaluation results are shown in Table 2, in which the percentage of sentences belonging to each grade and the averaged grade are demonstrated to evaluate the quality of generated responses. Kappa (Fleiss, 1971) value is presented to demonstrate the consistency of different annotators. From the results, we can see that MGL significantly outperforms baseline methods. The averaged score of MGL Model is 2.18, which is much higher than that of MMI and Adver-REGS, i.e., 1.78 and 1.9, respectively. The percentage of

post	Install grub to the hdd where the bios is set to boot.
response1	So I would not just install it to /boot?
Seq2Seq-att	A log file a bug with the install cd.
MMI	What do you mean?
Adver-REGS	How do I install grub?
Single	Do you delete grub?
MGL	I have boot with the window from the grub and it go from the boot menu.
CVaR	If you have grub on the same pc, it is 0 0 boot partition.
post	I want to have a backup , sad I use up all my space with dual boot and game.
response1	Oh you can get a sata to usb adapt :)
Seq2Seq-att	What about your computer ?
MMI	It is not in the install , you can do that.
Adver-REGS	Game game game game.
Single	And other user will do too ?
MGL	And do you buy direct driver from the kernel and it cause run ?
CVaR	Yes , you can also use a text file to your file .
post	Take a look at install .
response1	I am to cd to the directories contain the folder.
Seq2Seq-att	How do you install it?
MMI	I think it be a good idea to do that.
Adver-REGS	I have no idea what I am looking for.
Single	I think it is a bite , but I do not know a good thing to do that. I am use.
MGL	I think so, I have a lot of nautilus. I am already install.
CVaR	I just install it from synaptics, but I want to install it on the same repository.

Table 3: The generated responses from different models on Ubuntu.

strongly related sentences (i.e., the grade '3') of MGL Model is 51%, which is also higher than that of MMI, Adver-REGS and Single Model, i.e., 20% , 32% and 37%. In summary, our maximum generated likelihood model produces better responses compared with baselines. As compared with MMI and Adver-REGS, both the metric-based improvements and human evaluation improvements of MGL are significant on Ubuntu datasets (p-value < 0.01).

5.2.3 Case Study

Here we show some generated responses for demonstration. Specifically, Table 3 gives one example post and its ground-truth responses from Ubuntu. We also list the generated responses from different models. We can see that Seq2Seq-att, MMI and Adver-REGS all produce common responses, such as *'What do you mean?'*,*'I have no idea what I am looking for.'* and*'What about your computer?'*. Our models give interesting responses with specific meanings. Take the post *'Install grub to the hdd where the bios is set to boot.'* as an example, our model conveys more specific information by replying *'I have boot with the window from the grub and it go from the boot menu.'* . And in another case, for the given post *'I want to have a backup , sad I use up all my space with dual boot and game.'*, our MGL model generates a question for the post *'And do you buy direct driver from the kernel and it cause run?'*, which is more intelligent. Similar observations have been obtained for many other posts, and we omit them for space limitations.

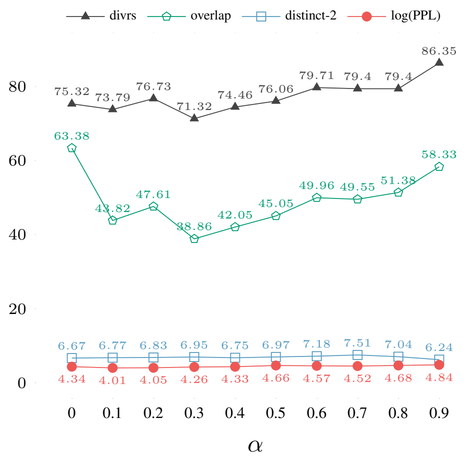


Figure 1: Influences of different α in CVaR.

model	BLEU	PPL	overlap	divrs
Seq2Seq	1.616	132.93	67.26	87.83
RNN-encdec	1.636	130.56	65.72	87.85
Seq2Seq-att	1.620	76.95	63.38	85.32
Adver-REGS	1.635	84.77	57.96	84.94
Mechanism	1.642	90.48	57.67	84.64
MGL	**1.703**	**36.25**	66.92	86.22
CVaR	1.652	70.94	**38.96**	**71.38**

Table 4: The metric-based evaluation results(%) of different models on STC.

5.3 Diverse-Requirement Scenario

Now we introduce the experimental results for the diverse-requirement scenario, based on STC.

5.3.1 Parameters Setting

First, we study the influences of different parameter α in CVaR. Specifically, we show the validation result with α ranging from 0 to 0.9 with step 0.1, to see the change of CVaR performances. Figure 1 show the results of different α in terms of divrs , overlap, distinct-2 and PPL. From the results, we can see that the performances of divrs , overlap and PPL are all changing in a similar trend, i.e. first drop and then increase. The best α for CVaR is 0.3, which is used in the following experiments.

5.3.2 Metric-based Evaluation

The quantitative evaluation results are shown in Table 4. From the results, we can see that both Adver-REGS and Mechanism outperform Seq2Seq models in terms of BLUE and PPL measures. That's because they both use some techniques to enhance the generation ability. Adver-REGS uses a learned discriminator to define the reward function, while Mechanism uses a style

model	human score distribution(%)			Ave.	Kappa
	1	2	3		
Seq2Seq-att	54.5	21	24.5	1.7	0.452
MMI	56	15.5	28.5	1.725	0.447
Adver-REGS	48.5	20	31.5	1.83	0.436
Mechanism	52.5	17.5	30	1.775	0.427
MGL	37	11	52	2.15	0.451
CVaR	44.5	11.5	44	1.995	0.437

Table 5: The comparisons of different models by human evaluation on STC.

hidden state to describe the generation mechanism. Both MGL and CVaR obtain better results in terms of BLUE and PPL, compared with other baselines. These results indicate that our proposed models generate more fluent responses in the diverse-requirement scenario. As for the evaluation for the diversity, we can see that CVaR model obtains the lowest overlap and divrs among all the baseline models. Take the overlap score on STC for example, the overlap score of CVaR model is 38.86, which is significantly lower than that of Adver-REGS, Mechanism and GLM, i.e., 57.96, 57.67 and 66.92. These results indicate that our CVaR model can generate responses with higher diversity. That's because it has the capability to capture various matching patterns in the training data, by optimizing the worst $1 - \alpha$ costs. Therefore, our CVaR model produces both fluent and diverse results, as compared with baseline methods.

5.3.3 Human Evaluation

The human evaluation results are shown in Table 5. From the results, we can see MGL and CVaR models achieve comparable results, which are significantly better than baseline methods. Specifically, the averaged score of MGL and CVaR is 2.15 and 1.995, which is significantly higher than that of Adver-REGS and Mechanism, i.e., 1.83 and 1.775, respectively. The percentage of strongly related sentences (i.e., the grade '3') of MGL Model and CVaR are 52% and 44%, which are also significantly higher than that of Adver-REGS and Mechanism, i.e., 31.5% and 30%. We conducted significant test for the improvement. As compared with Adver-REGS and Mechanism, both the metric-based improvements and human evaluation improvements of CVaR are significant on STC datasets (p-value < 0.01).

5.3.4 Case Study

Here we show some generated responses for demonstration. Specifically, Table 6 gives one post and its three ground-truth responses from STC. We also give three generated responses from Mechanism and CVaR model. We can see that Mechanism produces responses with the same meaning, such as *'Wade is so amazing'* and *'It is really good'*. However, our CVaR models give specific responses with diverse meanings. Take the post *'Waiting for Wade in the final games.'* for example, CVaR's responses are related to different topics. The response *'I must go and see the final games '* focuses on the game, while another response of *'James is so fast '* focuses on the person, James. For the other case, the post is about the docking of two spacecrafts and the CVaR responses are related to different users, such as the supporter of the event, the newspaper reader and the children who have a father concerned with the current news . We have obtained similar observations for many other posts, but we have to omit them for space limitations.

post	总决赛继续等待韦德.(Waiting for Wade in the final games.)
response	每个人都有每个人的喜爱(Everyone has his favorite stars.)
response	分析比新浪分析的好多了(Analysis is much better than Sina)
response	等待闪电侠彻底爆发！(Waiting for the explosion of Mr.Flash)
Mechanism	韦德真伟大啊!支持!(Wade is really great! Support him!)
Mechanism	韦德越来越好。(Wade will be better)
Mechanism	韦德威武!(Wade is mighty)
CVaR	决赛一定要去看看的!(I must go and see the final games)
CVaR	詹皇怎么还能这么快啊(James is so fast)
CVaR	决赛是一种对对手的打击,热火加油 (The final games is a blow for the opposite. Heat come on)
post	神舟八号飞船与天宫一号第二次对接成功. (Shenzhou 8 spacecraft and Tiangong-1 has the second successful docking)
response	希望其它方面也赶快领先世界(Hope other aspects will soon lead the world)
response	过硬的技术，紧密地配合促成了这次成功。 Strong technology and close cooperation contributed to this success
response	下一次就是载人对接啦.(Next will have a human in it)
Mechanism	周末报还会直播?(Will broadcast in the Weekend Newspaper?)
Mechanism	不错，就是不错(It is really good)
Mechanism	不错，就是在一起。(It is really good, they should be together)
CVaR	是啊，还是要坚持在一起。(Yes, they should insisted on being together)
CVaR	您这是在看头版吗？(Are you see it in the front page of the newspaper?)
CVaR	不错，有空推荐给爸爸！ (It is really good, you could recommend it to your father if you have time)

Table 6: The generated responses from different models on STC.

6 Conclusion

In this paper, we propose two new optimization criteria for Seq2Seq model to adapt different conversation scenario. For the specific-requirement scenario, such as customer service, which requires specific and high quality responses, maximum generated likelihood is used as the objective function instead of the averaged one. While for the diverse-requirement, such as chatbot, which requires diverse and high quality responses even if for the same post, CVaR is used as the objective function for worst case optimization. Experimental results on both specific-requirement

(Ubuntu data) and diverse-requirement scenarios (STC data) demonstrate that the proposed optimization criteria can meet the corresponding requirement, yielding better performances against traditional Seq2Seq models in terms of both metric-based and human evaluations.

The contribution of this paper is to use tailored Seq2Seq model for different conversation scenarios. The study shows that if we want to generate specific responses, it is important to design the model to learn the most significant matching pattern between post and response. While if we want to generate diverse responses, a risk-sensitive objective functions is helpful. In future work, we plan to further investigate the impact of risk-sensitive objective functions, including the relations between model robustness and diverse generations.

Acknowledgments

This work was funded by the 973 Program of China under Grant No. 2014CB340401, the National Natural Science Foundation of China (NSFC) under Grants No. 61425016, 61472401, 61722211, 61773362, and 20180290, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, and the National Key R&D Program of China under Grants No. 2016QY02D0405.

References

S. Alexander, T. F. Coleman, and Y. Li. 2006. Minimizing cvar and var for a portfolio of derivatives. *Journal of Banking and Finance* 30(2):583–605.

P. Artzner, F. Delbaen, J. M. Eber, and D. Heath. 1999. Coherent measures of risk mathematical finance 9. *Mathematical Finance Theory Modeling Implementation* volume 9(3):203–228(26).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *The International Conference on Learning Representations* .

Youhua Chen, Minghui Xu, and Zhe George Zhang. 2015. Technical note—a risk-averse newsvendor model under the cvar criterion. *Operations Research* 57(4):1040–1044.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science* .

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *American Psychological Association* .

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. *arXiv preprint arXiv:1506.05865* .

Pavlo Krokhmal, Jonas Palmquist, and Stanislav Uryasev. 2002. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of Risk* 4:11–27.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. *The North American Chapter of the Association for Computational Linguistics* .

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *The Conference on Empirical Methods in Natural Language Processing* .

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *The Conference on Empirical Methods in Natural Language Processing* .

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *Computer Science* .

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2017. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. *The Annual Meeting of the Association for Computational Linguistics* .

R. Tyrrell Rockafellar and Stanislav Uryasev. 2002. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance* 26(7):1443–1471.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *The Annual Meeting of the Association for Computational Linguistics* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *The Annual Conference on Neural Information Processing Systems*. pages 3104–3112.

Stanislav Uryasev. 2013. Probabilistic constrained optimization: methodology and applications. *Springer Science and Business Media* .

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv* .

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *Computer Science* .

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *The Association for the Advancement of Artificial Intelligence*. pages 3351–3357.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *The Association for the Advancement of Artificial Intelligence*. pages 2852–2858.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *The Association for the Advancement of Artificial Intelligence*. pages 3400–3407.

Knowledge Diffusion for Neural Dialogue Generation

Shuman Liu[†,§*], **Hongshen Chen**[‡], **Zhaochun Ren**[‡], **Yang Feng**[†], **Qun Liu**[◇], **Dawei Yin**[‡],
[†] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
[‡] Data Science Lab, JD.com
[◇] ADAPT centre, School of Computing, Dublin City University
[§] University of Chinese Academy of Sciences
liushuman@ict.ac.cn, chenhongshen,renzhaochun@jd.com,
fengyang,liuqun@ict.ac.cn, yindawei@acm.org

Abstract

End-to-end neural dialogue generation has shown promising results recently, but it does not employ knowledge to guide the generation and hence tends to generate short, general, and meaningless responses. In this paper, we propose a neural knowledge diffusion (NKD) model to introduce knowledge into dialogue generation. This method can not only match the relevant facts for the input utterance but diffuse them to similar entities. With the help of facts matching and entity diffusion, the neural dialogue generation is augmented with the ability of convergent and divergent thinking over the knowledge base. Our empirical study on a real-world dataset proves that our model is capable of generating meaningful, diverse and natural responses for both factoid-questions and knowledge grounded chi-chats. The experiment results also show that our model outperforms competitive baseline models significantly.

1 Introduction

Dialogue systems are receiving more and more attention in recent years. Given previous utterances, a dialogue system aims to generate a proper response in a natural way. Compared with the traditional pipeline based dialogue system, the new method based on sequence-to-sequence model (Shang et al., 2015; Vinyals and Le, 2015; Cho et al., 2014) impressed the research communities with its elegant simplicity. Such methods are usually in an end-to-end manner: utterances are encoded by a recurrent neural network while responses are generated sequentially by another (sometimes identical) recurrent neural network. However, due to lack of universal background knowledge and common senses, the end-to-end data-driven structure inherently tends to generate meaningless and short responses, such as "haha" or "I don't know."

To bridge the gap of the common knowledge between human and computers, different kinds of knowledge bases (e.g., the freebase (Google, 2013) and DBpedia (Lehmann et al., 2017)) are leveraged. A related application of knowledge bases is question answering, where the given questions are first analyzed, followed by retrieving related facts from knowledge bases (KBs), and finally the answers are generated.The facts are usually presented in the form of "subject-relation-object" triplets, where the subject and object are entities. With the aid of knowledge triplets, neural generative question answering systems are capable of answering facts related inquiries (Yin et al., 2016; Zhu et al., 2017; He et al., 2017a), *WH* questions in particular, like "*who is Yao Ming's wife ?*".

Although answering enquiries is essential for dialogue systems, especially for task-oriented dialogue systems (Eric et al., 2017), it is still far behind a natural knowledge grounded dialogue system, which should be able to understand the facts involved in current dialogue session (so-called facts matching), as well as diffuse them to other similar entities for knowledge-based chit-chats (i.e. entity diffusion):

1) *facts matching*: in dialogue systems, matching utterances to exact facts is much harder than explicit factoid inquiries answering. Though some utterances are facts related inquiries, whose subjects and relations can be easily recognized, for some utterances, the subjects and relations are elusive, which leads the trouble in exact facts matching.

[*] Work done when the first author was an intern at Data Science Lab, JD.com.

ID	Dialogue
1	A: Who is the director of the Titanic? 泰坦尼克号的导演是谁? B: James Cameron. 詹姆斯卡梅隆。
2	A: Titanic is my favorite film! 泰坦尼克号是我最爱的电影! B: The love inside it is so touching. 里面的爱情太感人了。
3	A: Is there anything like the Titanic? 有什么像泰坦尼克号一样的电影吗? B: I think the love story in film Waterloo Bridge is beautiful, too. 我觉得魂断蓝桥中的爱情故事也很美。
4	A: Is there anything like the Titanic? 有什么像泰坦尼克号一样的电影吗? B: Poseidon is also a classic marine film. 海神号也是一部经典的海难电影。

Table 1: Examples of knowledge grounded conversations. Knowledge entities are underlined.

Table 1 shows an example: Item 1 and 2 are talking about the film "Titanic", Unlike item 1, which is a typical question answering conversation,item 2 is a knowledge related chit-chat without any explicit relation. It is difficult to define the exact fact match for item 2.

2) *entity diffusion*: another noticeable phenomenon is that the conversation usually drifts from one entity to another. In Table 1, utterances in item 3 and 4 are about entity "Titanic", however, the entity of responses are other similar films. Such entity diffusion relations are rarely captured by the current knowledge triplets. The response in item 3 shows that the two entities "Titanic" and "Waterloo Bridge" are relevant through "love stories". Item 4 suggests another similar shipwreck film of "Titanic".

To deal with the aforementioned challenges, in this paper, we propose a **n**eural **k**nowledge **d**iffusion (NKD) dialogue system to benefit the neural dialogue generation with the ability of both convergent and divergent thinking over the knowledge base, and handle factoid QA and knowledge grounded chit-chats simultaneously. NKD learns to match utterances to relevant facts; the matched facts are then diffused to similar entities; and finally, the model generates the responses with respect to all the retrieved knowledge items.

In general, our contributions are as follows:

- We identify the problem of incorporating knowledge bases and dialogue systems as facts matching and entity diffusion.
- We manage both facts matching and entity diffusion by introducing a novel knowledge diffusion mechanism and generate the responses with the retrieved knowledge items, which enable the convergent and divergent thinking over the knowledge base.
- The experimental results show that the proposed model effectively generate more diverse and meaningful responses involving more accurate relevant entities compared with the state-of-the-art baselines.

The corpus will be released upon publication.

2 Model

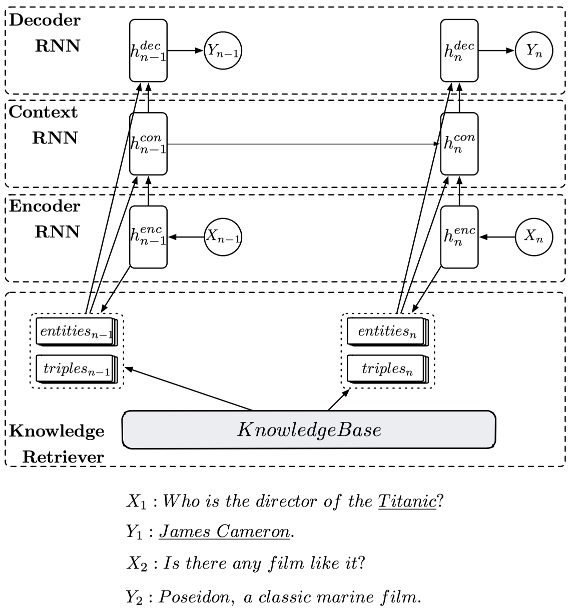


Figure 1: Neural Knowledge Diffusion Dialogue System.

Given the input utterance $X = (x_1, x_2, ..., x_{N_X})$, NKD produces a response $Y = (y_1, y_2, ..., y_{N_Y})$ containing the entities from the knowledge base K. N_X and N_Y are the number of tokens in the utterance and response respectively. The knowledge base K is a collection of knowledge facts in the form of triplets *(subject, relation, object)*. In particular, both subjects and objects are entities in this work. As illustrated in Figure 1, the model mainly consists of four components:

1. An encoder encodes the input utterance X into a vector representation.

2. A context RNN keeps the dialogue state along a conversation session. It takes the utterance representation as input, and outputs a vector guiding the response generation each turn.

3. A decoder generates the final response Y.

4. A knowledge retriever performs the facts matching and diffuses to similar entities at each turn.

Our work is built on hierarchical recurrent encoder-decoder architecture (Sordoni et al., 2015a), and a knowledge retriever network integrates the structured knowledge base into the dialogue system.

2.1 Encoder

The encoder transforms discrete tokens into vector representations. To capture information at different aspects, we learn utterance representations with two independent RNNs resulting with two hidden state sequences $H^C = (h_1^C, h_2^C, ..., h_{N_X}^C)$ and $H^K = (h_1^K, h_2^K, ..., h_{N_X}^K)$ respectively. One final hidden state $h_{N_X}^C$ is used as the input of context RNN to track the dialogue state. The other final hidden state $h_{N_X}^K$ is utilized in knowledge retriever and is designed to encode the knowledge entities and relations within the input utterances. For instance, in Figure 1, *"director"* and *"Titanic"* in X_1 are knowledge elements.

2.2 Knowledge Retriever

Knowledge retriever extracts a certain number of facts from knowledge base and specifies their importance. It enables the knowledge grounded neural dialogue system with convergent and divergent thinking ability through facts matching and entity diffusion. Figure 2 illustrates the process.

2.2.1 Facts Matching

Given the input utterance X and $h_{N_X}^K$, relevant facts are extracted from both the knowledge base and the dialogue history. A predefined number of relevant facts $F = \{f_1, f_2, ..., f_{N_f}\}$ are obtained through string matching, entity linking or named entity recognition. As shown in Figure 2, in the first sentence, *"Titanic"* is recognized as an entity, all the relevant knowledge triplets are extracted. Then, these entities and knowledge triplets are transformed into fact representations $h_f = \{h_{f1}, h_{f2}, ...h_{fN_f}\}$ by averaging the entity embedding and relation embedding. The relevance coefficient r^f between a fact and the input utterances, ranging from 0 to 1, is calculated by a nonlinear function or a sub neural network. Here, we apply a multi-layer perceptron (MLP):

$$r_k^f = MLP([h_{N_X}^K, h_{fk}]).$$

For the multi-turn conversation, entities in previous utterances are also inherited and reserved as depicted in Figure 2 the dotted lines. For instance, in the second sentence of Figure 2 (right one), no new fact is extracted from the input utterance. Thus it is necessary to record the history entities *"Titanic"* and *"James Cameron"*. We summarize the facts as *relevant fact representation* C^f through a weighted average of fact representations h_f:

$$C^f = \frac{\sum_{k=1}^{N_f} r_k^f h_{fk}}{\sum_{k=1}^{N_f} r_k^f}.$$

2.2.2 Entity Diffusion

To retrieve other relevant entities, which are typically not mentioned in the dialogue utterance, we diffuse the matched facts. We calculate the similarity between the entities (except the entities that have occurred in previous utterances) in the knowledge base and the relevant fact representation through a multi-layer perceptron, resulting with a similarity coefficient r^e, ranging from 0 to 1:

$$r_k^e = MLP([h_{N_X}^K, C^f, e_k]),$$

where e_k is the entity embedding. The top N_e number of entities $E = \{e_1, e_2, ..., e_{N_e}\}$ are selected as similar entities. Then, the *similar entity representation* C^s is formalized as:

$$C^s = \frac{\sum_{k=1}^{N_e} r_k^e e_k}{\sum_{k=1}^{N_e} r_k^e}.$$

Back to the example in Figure 2, in the first turn, the matched fact of the input utterance $(Titanic, direct\_by, JamesCameron)$ is of a high relevance coefficient in "facts matching" as expected. When a fact getting matched, intuitively it is not necessary for entity diffusion. In such case, from the Figure 2, we observe that the entities in "entity diffusing" are of low similarities. In the second turn, there is no triplets matched to the utterance, while the entity *"Titanic"* achieves a much higher relevance score. Then in "entity

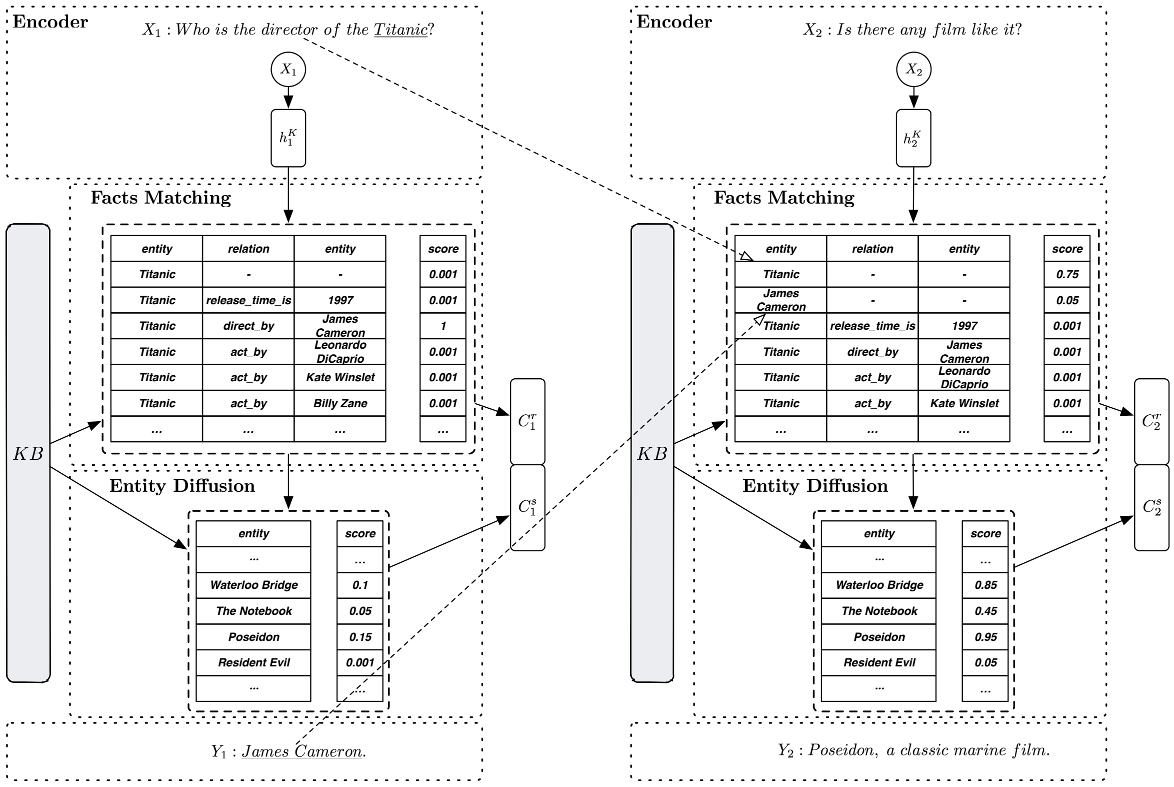

Encoder
X_1 : Who is the director of the Titanic?
X_1
h_1^K
Facts Matching
entity | relation | entity | score
Titanic | - | - | 0.001
Titanic | release_time_is | 1997 | 0.001
Titanic | direct_by | James Cameron | 1
Titanic | act_by | Leonardo DiCaprio | 0.001
Titanic | act_by | Kate Winslet | 0.001
Titanic | act_by | Billy Zane | 0.001
KB
C_1^r
C_1^s
Entity Diffusion
entity | score
Waterloo Bridge | 0.1
The Notebook | 0.05
Poseidon | 0.15
Resident Evil | 0.001
Y_1 : James Cameron.
Encoder
X_2 : Is there any film like it?
X_2
h_2^K
Facts Matching
entity | relation | entity | score
Titanic | - | - | 0.75
James Cameron | - | - | 0.05
Titanic | release_time_is | 1997 | 0.001
Titanic | direct_by | James Cameron | 0.001
Titanic | act_by | Leonardo DiCaprio | 0.001
Titanic | act_by | Kate Winslet | 0.001
KB
C_2^r
C_2^s
Entity Diffusion
entity | score
Waterloo Bridge | 0.85
The Notebook | 0.45
Poseidon | 0.95
Resident Evil | 0.05
Y_2 : Poseidon, a classic marine film.

Figure 2: Knowledge Retriever. Facts related to input utterance are extracted by facts matching. Similar entities are then figured out by entity diffusion. The dotted lines show the inheritance of previous facts.

diffusion", the similar entities *"Waterloo Bridge"* and *"Poseidon"* get relatively higher similarity weights than in the first turn.

2.3 Context RNN

Context RNN records the utterance level dialogue state. It takes in the utterance representation and the knowledge representations. The hidden state of the context RNN is updated as:

$$h_t^T = RNN(h_t^C, [C^f, C^s], h_{t-1}^T).$$

h_t^T is then conveyed to the decoder to guide the response generation.

2.4 Decoder

The decoder generates the response sequentially through a word generator conditioned on h_t^T, C^f and C^s. Let C denotes the concatenation of h_t^T, C^f and C^s. Knowledge items coefficient R is the concatenation of relevance coefficient r^f and similarity coefficient r^e. We introduce two variants of word generator:

Vanilla decoder simply generates the response $Y = (y_1, y_2, ..., y_{N_y})$ according to C, R. The probability of Y is defined as

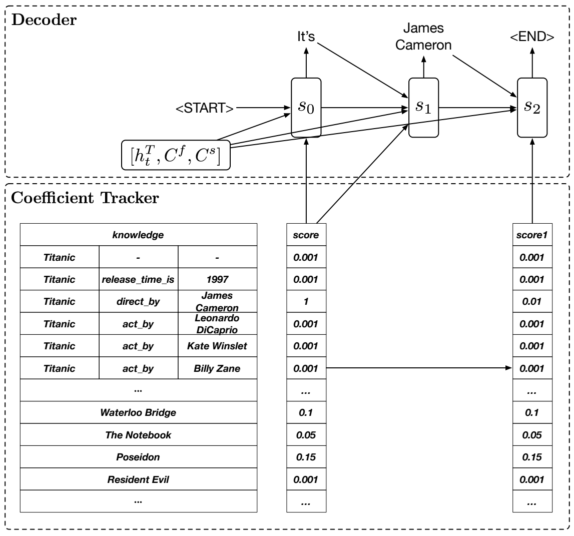

Decoder
It's
James Cameron
<END>
<START>
s_0
s_1
s_2
$[h_t^T, C^f, C^s]$
Coefficient Tracker
knowledge | score | score1
Titanic | - | - | 0.001 | 0.001
Titanic | release_time_is | 1997 | 0.001 | 0.001
Titanic | direct_by | James Cameron | 1 | 0.01
Titanic | act_by | Leonardo DiCaprio | 0.001 | 0.001
Titanic | act_by | Kate Winslet | 0.001 | 0.001
Titanic | act_by | Billy Zane | 0.001 | 0.001
Waterloo Bridge | 0.1 | 0.1
The Notebook | 0.05 | 0.05
Poseidon | 0.15 | 0.15
Resident Evil | 0.001 | 0.001

Figure 3: The decoder generates words from both vocabulary and knowledge base. A score updater keeps tracking of the knowledge item coefficients to ensure its coverage during response generation.

$$p(y_1, .., y_{N_y} | C, R; \theta)$$
$$= p(y_1 | C, R; \theta) \prod_{t=2}^{N_y} p(y_t | y_1, .., y_{t-1}, C, R; \theta),$$

where θ denotes the model parameters. The conditional probability of y_t is specified by

$$\begin{aligned} &p(y_t|y_1, ..., y_{t-1}, C, R; \theta) \\ &\quad = p(y_t|y_{t-1}, s_t, C, R; \theta), \end{aligned}$$

where y_t is the embedding of the vocabulary or object entities of retrieved knowledge items, s_t is the decoder RNN hidden state .

Probabilistic gated decoder utilizes a gating variable z_t (Yin et al., 2016) to indicate whether the t^{th} word is generated from common vocabulary or knowledge entities. The probability of generating the t^{th} word is given by:

$$\begin{aligned} &p(y_t|y_{t-1}, s_t, C, R; \theta) \\ &\quad = p(z_t = 0|s_t; \theta)p(y_t|y_{t-1}, s_t, C, R, z_t = 0; \theta) \\ &\quad + p(z_t = 1|s_t; \theta)p(y_t|R, z_t = 1; \theta), \end{aligned}$$

where $p(z_t|s_t; \theta)$ is computed by a logistic regression, $p(y_t|R, z_t = 1; \theta)$ is approximated with the knowledge items coefficient R, and θ is the model parameter.

During response generation, if an entity is overused, the response diversity will be reduced. Therefore, once a knowledge item occurred in the response, the corresponding coefficient should be reduced in case that an item occurs multiple times. To keep tracking the coverage of knowledge items, we update the knowledge items coefficient R at each time step. We also explore two coverage tracking mechanisms: 1) *Mask coefficient tracker* directly reduces the coefficient of the chosen knowledge item to 0 to ensure it can never be selected as the response word again. 2) *Coefficient attenuation tracker* calculates an attenuation score i_t based on s_t, R_0, R_{t-1} and y_{t-1}:

$$i_t = DNN(s_t, y_{t-1}, R_0, R_{t-1}),$$

and then update the coefficient as:

$$R_t = i_t \cdot R_{t-1},$$

where i_t ranges from 0 to 1 to gradually decrease the coefficient.

2.5 Training

The model parameters include the embedding of vocabulary, entities, relations, and all the model components. The model is differential and can be optimized in an end-to-end manner using back-propagation. Given the training data

$$D = \{(X_1^{N_d}, Y_1^{N_d}, F_1^{N_d}, E_1^{N_d})\}$$

where N_d is the max turns of a dialogue, F denotes the set of relevant knowledge and E denotes the set of similar knowledge in response, the objective function is to minimize the negative log-likelihood:

$$\ell(D, \theta) = -\sum \sum_{i=1}^{N_D} \log p(Y_i|X_i, F_i, E_i)$$

3 Experiment

3.1 Dataset

Most existing knowledge related datasets are mainly focused on single-turn factoid question answering (Yin et al., 2016; He et al., 2017b). We here collect a multi-turn conversation corpus grounded on the knowledge base, which includes not only facts related inquiries but also knowledge-based chit-chats. The data is publicly available online[1].

We first obtain the element information of each movie, including the movie's title, publication time, directors, actors and other attributes from `https://movie.douban.com/`, a popular Chinese social network for movies. Then, entities and relations are extracted as triplets to build the knowledge base K.

To collect the question-answering dialogues, we crawled the corpus from a question-answering forum `https://zhidao.baidu.com/`. To gather the knowledge related chit-chat corpus, we mined the dataset from the social forum `https://www.douban.com/group/`. Users post their comments, feedbacks, and impressions of films and televisions on it.

The conversations are grounded on the knowledge using NER, string match, and artificial scoring and filtering rules. The statistical information of the dataset is shown in Table 2. We observed that the conversations follow the long tail distribution, where famous films and televisions are discussed repeatedly and the low rating ones are rarely mentioned.

3.2 Experiment Detail

The total 32977 conversations consisting of 104567 utterances are divided into training (32177) and testing set (800). Bi-directional LSTM (Schuster and Paliwal, 1997) is used for encoder, and the dimension of the LSTM hidden

[1]https://github.com/liushuman/neural-knowledge-diffusion

Knowledge base			**Community QA**	**Multi-round dialogue**	
#entities	#relations	#triplets	#QA pairs	#dialogues	#sentences
152568	4	766854	8121	24856	88325

Table 2: Statistics of knowledge base and conversations.

layer is set to 512. For the context RNN, the dimension of the LSTM unit is set to 1024. The dimension of word embedding shared by the vocabulary, entities and relations is also set to 512 empirically. We use Adam learning (Kingma and Ba, 2014) to update the gradient and clip the gradient in 5.0. It takes 140 to 150 epochs to train the model with a batch size of 80.

3.3 Baselines

We compare our neural knowledge diffusion model with three state-of-the-art baselines:

- **Seq2Seq**: a sequence to sequence model with vanilla RNN encoder-decoder (Shang et al., 2015; Vinyals and Le, 2015).
- **HRED**: a hierarchical recurrent encoder-decoder model.
- **GenDS**: a neural generative dialogue system that is capable of generating responses based on input message and related knowledge base (KB) (Zhu et al., 2017) .

Three variants of the neural diffusion dialogue generation model are implemented to verify different configurations of decoders.

- **NKD-ori** is the original model with a vanilla decoder and a mask coefficient tracker.
- **NKD-gated** is augmented with a probabilistic gated decoder and a mask coefficient tracker.
- **NKD-atte** utilizes a vanilla decoder and the coefficient attenuation tracker.

3.4 Evaluation Metric

Both automatic and human evaluation metrics are used to analyze the model's performance. To validate the effectiveness of facts matching and diffusion, we calculate **entity accuracy and recall** on factoid QA data set as well as the whole data set. Human evaluation rates the model in three aspects: **fluency**, **knowledge relevance** and **correctness** of the response. All these metrics range from 0 to 3, where 0 represents complete error, 1 for partially correct, 2 for almost correct, 3 for absolutely correct.

model	**accuracy(%)**	**recall(%)**
LSTM	7.8	7.5
HRED	3.7	3.9
GenDS	70.3	63.1
NKD-ori	67.0	56.2
NKD-gated	**77.6**	**77.3**
NKD-atte	55.1	46.6

Table 3: Evaluation results on factoid question answering dialogues.

model	**accuracy(%)**	**recall(%)**	**entity number**
LSTM	2.6	2.5	1.65
HRED	1.4	1.5	1.79
GenDS	20.9	17.4	1.34
NKD-ori	22.9	19.7	2.55
NKD-gated	**24.8**	**25.6**	1.59
NKD-atte	18.4	16.0	**3.41**

Table 4: Evaluation results on entire dataset.

3.5 Experiment Result

Table 3 displays the accuracy and recall of entities on factoid question answering dialogues. The performance of NKD is slightly better than the specific QA solution GenDS, while LSTM and HRED which are designed for chi-chat almost fail in this task. All the variants of NKD models are capable of generating entities with an accuracy of 60% to 70%, and NKD-gated achieves the best performance with an accuracy of 77.6% and a recall of 77.3%.

Table 4 lists the accuracy and recall of entities on the entire dataset including both the factoid QA and knowledge grounded chit-chats. Not surprisingly, both NKD-ori and NKD-gated outperform GenDS on the entire dataset, and the relative improvement over GenDS is even higher than the improvement in QA dialogues. It confirms that although NKD and GenDS are comparable in answering factoid questions, NKD is better at introducing the knowledge entities for knowledge grounded chit-chats.

All the NKD variants in Table 4 generate more entities than GenDS. LSTM and HRED also produce a certain amount of entities, but are of low

model	Fluency	Appropriateness of knowledge	Entire Correctness
LSTM	2.52	0.88	0.8
HRED	2.48	0.36	0.32
GenDS	**2.76**	1.36	1.34
NKD-ori	2.42	**1.92**	**1.58**
NKD-gated	2.08	1.72	1.44
NKD-atte	2.7	1.54	1.38

Table 5: Human evaluation result.

accuracies and recalls. We also noticed that NKD-gated achieves the highest accuracy and recall, but generates fewer entities compared with NKD-ori and NKD-gated, whereas NKD-atte generates more entities but also with relatively low accuracies and recalls.This demonstrates that NKD-gated not only learns to generate more entities but also maintains the quality (with a relatively high accuracy and recall).

The results of human evaluation in Table 5 also validate the superiority of the proposed model, especially on appropriateness. Responses generated by LSTM and HRED are of high fluency, but are simply repetitions, or even dull responses as "I don't know.", "Good.". NKD-gated is more adept at incorporating the knowledge base with respect to appropriateness and correctness, while NKD-atte generates more fluent responses. NKD-ori is a compromise, and obtains the best correctness in completing an entire dialogue. Four evaluators rated the scores independently. The pairwise Cohen's Kappa agreement scores are 0.67 on fluency, 0.54 on appropriateness, and 0.60 on entire correctness, which indicate a strong annotator agreement.

To our surprise, one of the variant model of NKD, which utilized both probabilistic gated decoder and coefficient attenuation tracker does not perform well on entire dataset. The accuracy of the model is quite high, but the recall is very low compared to others. We speculate that this is due to the method of minimizing negative log-likelihood during the training process, which makes the model tend to generate completely correct answers, and therefore reduces the number of generated entities.

3.6 Case Study

Table 6 shows typical examples of the generated responses. Both Item 1 and 2 are based on facts relevant utterances. NKD handles these questions by facts matching. Item 3 asks for a recommendation. NKD obtains similar entities by diffusing the entities. For item 4, 5 and 6, no explicit entity appears in the utterances. NKD is able to output appropriate recommendations through entity diffusion. The entities are recorded during the whole dialogue session, so NKD keeps recommending for several turns. Item 7 fails to generate an appropriate response because the entity in the golden response does not appear in the training set, which suggests the future work for out-of-vocabulary cases.

4 Related Work

The successes of sequence-to-sequence architecture (Cho et al., 2014; Sutskever et al., 2014) motivated investigation in dialogue systems that can effectively learn to generate a response sequence given the previous utterance sequence (Shang et al., 2015; Sordoni et al., 2015b; Vinyals and Le, 2015). The model is trained to minimize the negative log-likelihood of the training data. Despite the current progress, the lack of response diversity is a notorious problem, where the model inherently tends to generate short, general responses in spite of different inputs. Li et al. (2016a); Serban et al. (2017); Cao and Clark (2017) suggested that theses boring responses are common in training data and shorter responses are more likely to be given a higher likelihood. To tackle the problem, Li et al. (2016a) introduced a maximum mutual information training objective. Serban et al. (2017), Cao and Clark (2017) and Chen et al. (2018) used latent variables to introduce stochasticity to enhance the response diversity. Vijayakumar et al. (2016),Shao et al. (2017) and Li et al. (2016b) recognized that the greedy search decoding process, especially beam-search with a wide beam size, leads the short responses possess higher likelihoods. They reserved more diverse candidates during beam-search decoding. In this paper, we present that the absence of background knowledge and common sense is another source of lacking diversity. We augment the knowledge base to end-to-end dialogue generation.

Another research line comes from the utilizing of knowledge bases. A typical application is question-answering (QA) systems. The end-to-end QA also resort to the encoder-decoder framework (Yin et al., 2016; He et al., 2017a). Yin et al. (2016) enquired the knowledge-base to achieve one fact and answer the simple factoid questions

ID	Utterance	Response
1	X_1:Duckweed, casts 乘风破浪，演员表	Y_1:Hongda Guan Jiufeng Shi Qi Wang Zhi Cheng 关宏达 史久峰 汪漪 程之
		NKD:Hongda Guan Jiufeng Shi Qi Wang Zhi Cheng 关宏达 史久峰 汪漪 程之
2	X_1: Just spend the whole night watching the movie Silenced, a little depressed now 刚看完熔炉... 连夜看完的，略低落	Y_1:I saw it last night and had a hard time 昨晚看了，心里好难过
		NKD:The film has an atmosphere make us rethink sth 气氛有，会反思一些事
3	X_1:Which movie most representative of youth? I'd like In the Heat of the Sun, Flipped, First Love(Sing Street) 你觉得最能代表青春的电影有哪些呢 我觉得有阳光灿烂的日子，怦然心动，初恋这件小事（唱街）	Y_1:Kids Return 坏孩子的天空
		NKD:《Love Letter》《Once》 《情书》《曾经》
4	X_1:Can anyone recommend me some touching movie, tears streaming after watching 谁能给我推荐两部感人的电影，直接点的，看了可以流两缸眼泪那种	Y_1:Gift From Room 7 Hearty Paws Memories of Matsuko 7号房的礼物 人狗奇缘 被嫌弃的松子的一生
		NKD:Wedding Dress, these ones,,, 婚纱，这些，，，
5	X_1:burn brain or suspense movies please, recommend me some burn brain or suspense or horror films please! suspense films which are less popular are ok!!! 求推荐烧脑的推理悬疑电影 求各位大神推荐烧脑的悬疑恐怖电影 冷门的推理剧也爱！！！	Y_1:《The Skeleton Key》《Identity》 《万能钥匙》《致命id》
		NKD:Identity 致命id
	X_2:I've seen it, it's nice 看过哦不错呢	Y_1:《Orphan》？ 《孤儿怨》？
		NKD: Also Inception Whisper of the Heart Whisper of the Heart 还有 盗梦空间 侧耳倾听 侧耳倾听
6	X_1:It's a bit boring recently. Any good horror movies, guys? I've watched a lot. 最近有点无聊，各位豆友有什么好看的恐怖片推荐啊？看了好多了。	Y_1:Rec, A Wicked Ghost, Dead Silence, 1408 phontom horror, 死亡录像，山村老尸，死寂，1408，
		NKD:Recommend a few I've seen, 《A Wicked Ghost》,《hide and seek》 推荐几部我看过《山村老尸》、《捉迷藏》
	X_2:I've seen it all, anything else? 卤煮都看过了。还有推荐？	Y_1:The Others, hardly call to mind 小岛惊魂，冷不丁还真想不起来
		NKD: Identity, Fight Club, Interstellar, Snowpiercer 致命id、搏击俱乐部、星际穿越、雪国列车
	X_3:Have seen it before, really great 看过，真的很不错的说	Y_3:《The Revenant》, nice movie 《亡灵》，非常好看
		NKD:So what type do you like? 那你喜欢什么类型的？
7	X_1:any awesome animation like Song of the Sea, please 求一些画风超赞的动画 类似于海洋之歌这种的，谢谢	Y_1::The Girl Without Hands 无手的少女
		NKD:strongly recommended 强烈推荐

Table 6: Examples of the generated response. Entities are underlined and Y_i denotes the gold response.

by referring to the fact. He et al. (2017a) extended this approach by augmenting the copying mechanism and enabled the output words to copy from the original input sequence. Eric et al. (2017) noticed that neural task-oriented dialogue systems often struggle to smoothly interface with a knowledge base and they addressed the problem by augmenting the end-to-end structure with a key-value retrieval mechanism where a separate attention is performed over the key of each entry in the KB. Ghazvininejad et al. (2017) represented the unstructured text as bag of words representation and also performed soft attention over the facts to retrieve a facts vector. Zhu et al. (2017) generated responses with any number of answer entities in the structured KB, even when these entities never appear in the training set. Dhingra et al. (2017) proposed a multi-turn dialogue agent which helps users search knowledge base by soft KB lookup. In our model, we perform not only facts matching to answer factoid inquiries, but also entity diffusion to infer similar entities. Given previous utterances, we retrieve the relevant facts, diffuse them, and generate responses based on diversified relevant knowledge items.

5 Conclusion

In this paper, we identify the knowledge diffusion in conversations and propose an end-to-end neural knowledge diffusion model to deal with the problem. The model integrates the dialogue system with the knowledge base through both facts matching and entity diffusion, which enable the convergent and divergent thinking over the knowledge base. Under such mechanism, the factoid question answering and knowledge grounded chitchats can be tackled together. Empirical results show the proposed model is able to generate more meaningful and diverse responses, compared with the state-of-the-art baselines. In future work, we plan to introduce reinforcement learning and knowledge base reasoning mechanisms to improve the performance.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.61662077, No.61472428). We also would like to thank all the reviewers for their insightful and valuable comments and suggestions.

References

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 182–187, Valencia, Spain. Association for Computational Linguistics.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1653–1662. International World Wide Web Conferences Steering Committee.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49. Association for Computational Linguistics.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *arXiv preprint arXiv:1702.01932*.

Google. 2013. Freebase data dumps.

Shizhu He, Cao Liu, Kang Liu, Jun Zhao, Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017a. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Meeting of the Association for Computational Linguistics*, pages 199–208.

Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2017b. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv eprint arXiv:1711.05073*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2017. Dbpedia – a large-scale, multilingual knowledge base extracted from wikipedia.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Jurafsky Dan. 2016b. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI Conference on Artificial Intelligence*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China. Association for Computational Linguistics.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, and Brian Strope. 2017. Generating long and diverse responses with neural conversation models. *arXiv preprint arXiv:1701.03185*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *International Joint Conference on Artificial Intelligence*, pages 2972–2978.

Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. 2017. Flexible end-to-end dialogue system for knowledge grounded conversation. *arXiv eprint arXiv:1709.04264*.

Generating Informative Responses with Controlled Sentence Function

Pei Ke[1], **Jian Guan**[2], **Minlie Huang**[1]*, **Xiaoyan Zhu**[1]
[1]Conversational AI group, AI Lab., Dept. of Computer Science, Tsinghua University
[1]Beijing National Research Center for Information Science and Technology, China
[2]Dept. of Physics, Tsinghua University, Beijing 100084, China
kepei1106@outlook.com, guanj15@mails.tsinghua.edu.cn
aihuang@tsinghua.edu.cn, zxy-dcs@tsinghua.edu.cn

Abstract

Sentence function is a significant factor to achieve the purpose of the speaker, which, however, has not been touched in large-scale conversation generation so far. In this paper, we present a model to generate informative responses with controlled sentence function. Our model utilizes a continuous latent variable to capture various word patterns that realize the expected sentence function, and introduces a type controller to deal with the compatibility of controlling sentence function and generating informative content. Conditioned on the latent variable, the type controller determines the type (i.e., *function-related*, *topic*, and *ordinary* word) of a word to be generated at each decoding position. Experiments show that our model outperforms state-of-the-art baselines, and it has the ability to generate responses with both controlled sentence function and informative content.

1 Introduction

Sentence function is an important linguistic feature and a typical taxonomy in terms of the purpose of the speaker (Rozakis, 2003). There are four major function types in the language including *interrogative, declarative, imperative*, and *exclamatory*, as described in (Rozakis, 2003). Each sentence function possesses its own structure, and transformation between sentence functions needs a series of changes in word order, syntactic patterns and other aspects (Akmajian, 1984; Yule, 2010).

Since sentence function is regarding the purpose of the speaker, it can be a significant factor indicating the conversational purpose during interactions, but surprisingly, this problem is rather untouched in dialogue systems. As shown in Figure 1, responses with different functions can be used to achieve different conversational purposes: **Interrogative** responses can be used to acquire further information from the user; **imperative** responses are used to make requests, directions, instructions or invitations to elicit further interactions; and **declarative** responses commonly make statements to state or explain something.[1] Interrogative and imperative responses can be used to avoid stalemates (Li et al., 2016b), which can be viewed as important proactive behaviors in conversation (Yu et al., 2016). Thus, conversational systems equipped with the ability to control the sentence function can adjust its strategy for different purposes within different contexts, behave more proactively, and may lead the dialogue to go further.

Post		I'm really hungry now.
Response	**Interrogative**	**What did** you have at breakfast**?**
	Imperative	**Let's** have dinner together**!**
	Declarative	Me, **too. But** you ate too much at lunch**.**

Figure 1: Responses with three sentence functions. Function-related words are in **red**, topic words in blue, and others are ordinary words.

Generating responses with controlled sentence functions differs significantly from other tasks on controllable text generation (Hu et al., 2017; Ficler and Goldberg, 2017; Asghar et al., 2017; Ghosh et al., 2017; Zhou and Wang, 2017; Dong et al., 2017; Murakami et al., 2017). These studies, involving the control of sentiment polarity, emotion, or tense, fall into **local** control, more or less, because the controllable variable can be *locally* re-

*Corresponding author: Minlie Huang.

[1] Note that we did not include the *exclamatory* category in this paper because an exclamatory sentence in conversation is only a strong emotional expression of the original sentence with few changes.

flected by decoding local variable-related words, e.g., *terrible* for negative sentiment (Hu et al., 2017; Ghosh et al., 2017), *glad* for happy emotion (Zhou et al., 2018; Zhou and Wang, 2017), and *was* for past tense (Hu et al., 2017). By contrast, sentence function is a **global** attribute of text, and controlling sentence function is more challenging in that it requires to adjust the **global structure** of the entire text, including changing word order and word patterns.

Controlling sentence function in conversational systems faces another challenge: in order to generate informative and meaningful responses, it has to deal with the compatibility of the sentence function and the content. Similar to most existing neural conversation models (Li et al., 2016a; Mou et al., 2016; Xing et al., 2017), we are also struggling with universal and meaningless responses for different sentence functions, e.g., "*Is that right?*" for interrogative responses, "*Please!*" for imperative responses and "*Me, too.*" for declarative responses. The lack of meaningful topics in responses will definitely degrade the utility of the sentence function so that the desired conversational purpose can not be achieved. Thus, the task needs to generate responses with both informative content and controllable sentence functions.

In this paper, we propose a conversation generation model to deal with the global control of sentence function and the compatibility of controlling sentence function and generating informative content. We devise an encoder-decoder structure equipped with a latent variable in conditional variational autoencoder (CVAE) (Sohn et al., 2015), which can not only project different sentence functions into different regions in a latent space, but also capture various word patterns within each sentence function. The latent variable, supervised by a discriminator with the expected function label, is also used to realize the global control of sentence function. To address the compatibility issue, we use a type controller which lexicalizes the sentence function and the content explicitly. The type controller estimates a distribution over three word types, i.e., **function-related**, **topic**, and **ordinary** words. During decoding, the word type distribution will be used to modulate the generation distribution in the decoder. The type sequence of a response can be viewed as an abstract representation of sentence function. By this means, the model has an explicit and strong control on the function and the content. Our contributions are as follows:

- We investigate how to control sentence functions to achieve different conversational purposes in open-domain dialogue systems. We analyze the difference between this task and other controllable generation tasks.

- We devise a structure equipped with a latent variable and a type controller to achieve the global control of sentence function and deal with the compatibility of controllable sentence function and informative content in generation. Experiments show the effectiveness of the model.

2 Related Work

Recently, language generation in conversational systems has been widely studied with sequence-to-sequence (seq2seq) learning (Sutskever et al., 2014; Bahdanau et al., 2015; Vinyals and Le, 2015; Shang et al., 2015; Serban et al., 2016, 2017). A variety of methods has been proposed to address the important issue of content quality, including enhancing diversity (Li et al., 2016a; Zhou et al., 2017) and informativeness (Mou et al., 2016; Xing et al., 2017) of the generated responses.

In addition to the content quality, controllability is a critical problem in text generation. Various methods have been used to generate texts with controllable variables such as sentiment polarity, emotion, or tense (Hu et al., 2017; Ghosh et al., 2017; Zhou and Wang, 2017; Zhou et al., 2018) . There are mainly two solutions to deal with controllable text generation. First, the variables to be controlled are embedded into vectors which are then fed into the models to reflect the characteristics of the variables (Ghosh et al., 2017; Zhou et al., 2018). Second, latent variables are used to capture the information of controllable attributes as in the variational autoencoders (VAE) (Zhou and Wang, 2017). (Hu et al., 2017) combined the two techniques by disentangling a latent variable into a categorical code and a random part to better control the attributes of the generated text.

The task in this paper differs from the above tasks in two aspects: (1) Unlike other tasks that realize controllable text generation by decoding attribute-related words locally, our task requires to not only decode function-related words, but also

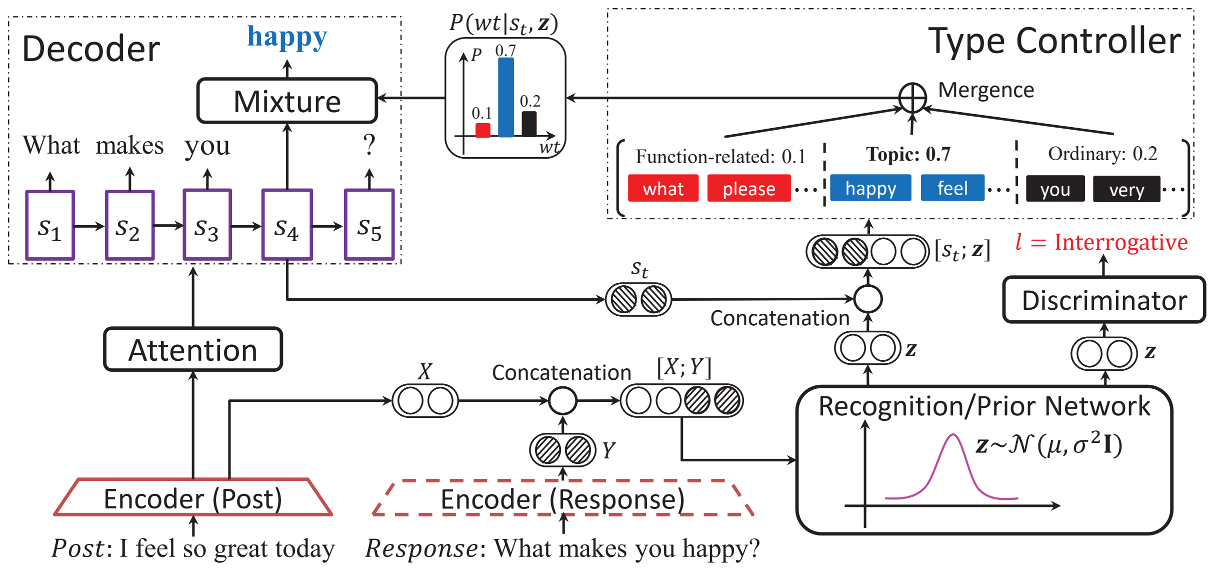


Figure 2: Model overview. During training, the latent variable z is sampled from the recognition network which is supervised by the function label in the discriminator. In the type controller, the latent variable and the decoder's state are used to estimate a type distribution which modulates the final generation distribution. During test, z is sampled from the prior network whose input is only the post. The response encoder in the dotted box appears only in training.

plan the words globally to realize the function type to be controlled. (2) The compatibility of controllable variables and content quality is less studied in the literature. The most similar work in (Zhao et al., 2017) proposed to control the dialogue act of a response, which is also a global attribute. However, the model controls dialog act by directly feeding a latent variable into the decoder, instead, our model has a stronger control on the generation process via a type controller in which words of different types are concretely modeled.

3 Model

3.1 Task Definition and Model Overview

Our problem is formulated as follows: given a post $X = x_1x_2\cdots x_n$ and a sentence function category l, our task is to generate a response $Y = y_1y_2\cdots y_m$ that is not only coherent with the specified function category l but also informative in content. We denote $\boldsymbol{c}$ as the concatenation of all the input information, i.e. $\boldsymbol{c} = [X; l]$. Essentially, the goal is to estimate the conditional probability:

$$P(Y, \boldsymbol{z}|\boldsymbol{c}) = P(\boldsymbol{z}|\boldsymbol{c}) \cdot P(Y|\boldsymbol{z}, \boldsymbol{c}) \quad (1)$$

The latent variable $\boldsymbol{z}$ is used to capture the sentence function of a response. $P(\boldsymbol{z}|\boldsymbol{c})$, parameterized as the prior network in our model, indicates the sampling process of $\boldsymbol{z}$, i.e., drawing $\boldsymbol{z}$ from $P(\boldsymbol{z}|\boldsymbol{c})$. And $P(Y|\boldsymbol{z}, \boldsymbol{c}) = \prod_{t=1}^{m} P(y_t|y_{<t}, \boldsymbol{z}, \boldsymbol{c})$ is applied to model the generation of the response Y conditioned on the latent variable $\boldsymbol{z}$ and the input $\boldsymbol{c}$, which is implemented by a decoder in our model.

Figure 2 shows the overview of our model. As aforementioned, the model is constructed in the encoder-decoder framework. The encoder takes a post and a response as input, and obtains the hidden representations of the input. The recognition network and the prior network, adopted from the CVAE framework (Sohn et al., 2015), sample a latent variable $\boldsymbol{z}$ from two normal distributions, respectively. Supervised by a discriminator with the function label, the latent variable encodes meaningful information to realize a sentence function. The latent variable, along with the decoder's state, is also used to control the type of a word in generation via the type controller. In the decoder, the final generation distribution is mixed by the type distribution which is obtained from the type controller. By this means, the latent variable encodes information not only from sentence function but also from word types, and in return, the decoder and the type controller can deal with the compatibility of realizing sentence function and information content in generation.

3.2 Encoder-Decoder Framework

The encoder-decoder framework has been widely used in language generation (Sutskever et al., 2014; Vinyals and Le, 2015). The encoder transforms the post sequence $X = x_1 x_2 \cdots x_n$ into hidden representations $\boldsymbol{H} = \boldsymbol{h}_1 \boldsymbol{h}_2 \cdots \boldsymbol{h}_n$, as follows:

$$\boldsymbol{h}_t = \mathbf{GRU}(\boldsymbol{e}(x_t), \boldsymbol{h}_{t-1}) \tag{2}$$

where $\mathbf{GRU}$ is gated recurrent unit (Cho et al., 2014), and $\boldsymbol{e}(x_t)$ denotes the embedding of the word x_t.

The decoder first updates the hidden states $\boldsymbol{S} = \boldsymbol{s}_1 \boldsymbol{s}_2 \cdots \boldsymbol{s}_m$, and then generates the target sequence $Y = y_1 y_2 \cdots y_m$ as follows:

$$\boldsymbol{s}_t = \mathbf{GRU}(\boldsymbol{s}_{t-1}, \boldsymbol{e}(y_{t-1}), \boldsymbol{cv}_{t-1}) \tag{3}$$

$$y_t \sim P(y_t | y_{<t}, \boldsymbol{s}_t) = softmax(\boldsymbol{W}\boldsymbol{s}_t) \tag{4}$$

where this $\mathbf{GRU}$ does not share parameters with the encoder's network. The context vector $\boldsymbol{cv}_{t-1}$ is a dynamic weighted sum of the encoder's hidden states, i.e., $\boldsymbol{cv}_{t-1} = \sum_{i=1}^{n} \alpha_i^{t-1} \boldsymbol{h}_i$, and α_i^{t-1} scores the relevance between the decoder's state $\boldsymbol{s}_{t-1}$ and the encoder's state $\boldsymbol{h}_i$ (Bahdanau et al., 2015).

3.3 Recognition/Prior Network

On top of the encoder-decoder structure, our model introduces the recognition network and the prior network of CVAE framework, and utilizes the two networks to draw latent variable samples during training and test respectively. The latent variable can project different sentence functions into different regions in a latent space, and also capture various word patterns within a sentence function.

In the training process, our model needs to sample the latent variable from the posterior distribution $P(\boldsymbol{z}|Y, \boldsymbol{c})$, which is intractable. Thus, the recognition network $q_\phi(\boldsymbol{z}|Y, \boldsymbol{c})$ is introduced to approximate the true posterior distribution so that we can sample $\boldsymbol{z}$ from this deterministic parameterized model. We assume that z follows a multivariate Gaussian distribution whose covariance matrix is diagonal, i.e., $q_\phi(\boldsymbol{z}|Y, \boldsymbol{c}) \sim \mathcal{N}(\mu, \sigma^2 \mathbf{I})$. Under this assumption, the recognition network can be parameterized by a deep neural network such as a multi-layer perceptron (MLP):

$$[\mu, \sigma^2] = \mathbf{MLP}_{posterior}(Y, \boldsymbol{c}) \tag{5}$$

During test, we use the prior network $p_\theta(\boldsymbol{z}|\boldsymbol{c}) \sim \mathcal{N}(\mu', \sigma'^2 \mathbf{I})$ instead to draw latent variable samples, which can be implemented in a similar way:

$$[\mu', \sigma'^2] = \mathbf{MLP}_{prior}(\boldsymbol{c}) \tag{6}$$

To bridge the gap between the recognition and the prior networks, we add the KL divergence term that should be minimized to the loss function:

$$\mathcal{L}_1 = KL(q_\phi(\boldsymbol{z}|Y, \boldsymbol{c}) || p_\theta(\boldsymbol{z}|\boldsymbol{c})) \tag{7}$$

3.4 Discriminator

The discriminator supervises $\boldsymbol{z}$ to encode function-related information in a response with supervision signals. It takes $\boldsymbol{z}$ as input instead of the generated response Y to avoid the vanishing gradient of z, and predicts the function category conditioned on $\boldsymbol{z}$:

$$P(l|\boldsymbol{z}) = softmax(\boldsymbol{W}_D \cdot \mathbf{MLP}_{dis}(\boldsymbol{z})) \tag{8}$$

This formulation can enforce $\boldsymbol{z}$ to capture the features of sentence function and enhance the influence of $\boldsymbol{z}$ in word generation. The loss function of the discriminator is given by:

$$\mathcal{L}_2 = -E_{q_\phi(\boldsymbol{z}|Y, \boldsymbol{c})}[\log P(l|\boldsymbol{z})] \tag{9}$$

3.5 Type Controller

The type controller is designed to deal with the compatibility issue of controlling sentence function and generating informative content. As aforementioned, we classify the words in a response into three types: *function-related, topic, and ordinary* words. The type controller estimates a distribution over the word types at each decoding position, and the type distribution will be used in the mixture model of the decoder for final word generation. During the decoding process, the decoder's state $\boldsymbol{s}_t$ and the latent variable $\boldsymbol{z}$ are taken as input to estimate the type distribution as follows:

$$P(wt|\boldsymbol{s}_t, \boldsymbol{z}) = softmax(\boldsymbol{W}_0 \cdot \mathbf{MLP}_{type}(\boldsymbol{s}_t, \boldsymbol{z})) \tag{10}$$

Noticeably, the latent variable z introduced to the RNN encoder-decoder framework often fails to learn a meaningful representation and has little influence on language generation, because the RNN decoder may ignore z during generation, known as the issue of vanishing latent variable (Bowman et al., 2016). By contrast, our model allows $\boldsymbol{z}$ to directly control the word type at each decoding position, which has more influence on language generation.

3.6 Decoder

Compared with the traditional decoder described in Section 3.2, our decoder updates the hidden state $\boldsymbol{s}_t$ with both the input information $\boldsymbol{c}$ and the latent variable $\boldsymbol{z}$, and generates the response in a *mixture* form which is combined with the type distribution obtained from the type controller:

$$\boldsymbol{s}_t = \mathbf{GRU}(\boldsymbol{s}_{t-1}, \boldsymbol{e}(y_{t-1}), \boldsymbol{cv}_{t-1}, \boldsymbol{c}, \boldsymbol{z}) \quad (11)$$

$$P(y_t|y_{<t}, \boldsymbol{c}, \boldsymbol{z}) = P(y_t|y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}, \boldsymbol{z})$$
$$= \sum_{i=1}^{3} P(wt = i|\boldsymbol{s}_t, \boldsymbol{z}) P(y_t|y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}, \boldsymbol{z}, wt = i) \quad (12)$$

where $wt = 1, 2, 3$ stand for function-related words, topic words, and ordinary words, respectively. The probability for choosing different word types at time t, $P(wt = i|\boldsymbol{s}_t, \boldsymbol{z})$, is obtained from the type controller, as shown in Equation (10). The probabilities of choosing words in different types are introduced as follows:

Function-related Word: Function-related words represent the typical words for each sentence function, e.g., *what* for interrogative responses, and *please* for imperative responses. To select the function-related words at each position, we simultaneously consider the decoder's state $\boldsymbol{s}_t$, the latent variable $\boldsymbol{z}$ and the function category l.

$$P(y_t|y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}, \boldsymbol{z}, wt = 1) = softmax(\boldsymbol{W}_1 \cdot [\boldsymbol{s}_t, \boldsymbol{z}, \boldsymbol{e}(l)]) \quad (13)$$

where $\boldsymbol{e}(l)$ is the embedding vector of the function label. Under the control of $\boldsymbol{z}$, our model can learn to decode function-related words at proper positions automatically.

Topic Word: Topic words are crucial for generating an informative response. The probability for selecting a topic word at each decoding position depends on the current hidden state s_t:

$$P(y_t|y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}, \boldsymbol{z}, wt = 2) = softmax(\boldsymbol{W}_2 \boldsymbol{s}_t) \quad (14)$$

This probability is over the topic words we predict conditioned on a post. Section 3.8 will describe the details.

Ordinary Word: Ordinary words play a functional role in making a natural and grammatical sentence. The probability of generating ordinary words is estimated as below:

$$P(y_t|y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}, \boldsymbol{z}, wt = 3) = softmax(\boldsymbol{W}_3 \boldsymbol{s}_t) \quad (15)$$

The generation loss of the decoder is given as below:

$$\mathcal{L}_3 = -E_{q_\phi(\boldsymbol{z}|Y,\boldsymbol{c})}[\log P(Y|\boldsymbol{z}, \boldsymbol{c})]$$
$$= -E_{q_\phi(\boldsymbol{z}|Y,\boldsymbol{c})}[\sum_t \log P(y_t|y_{<t}, \boldsymbol{z}, \boldsymbol{c})] \quad (16)$$

3.7 Loss Function

The overall loss $\mathcal{L}$ is a linear combination of the KL term $\mathcal{L}_1$, the classification loss of the discriminator $\mathcal{L}_2$, and the generation loss of the decoder $\mathcal{L}_3$:

$$\mathcal{L} = \alpha\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \quad (17)$$

We let α gradually increase from 0 to 1. This technique of *KL cost annealing* can address the optimization challenges of vanishing latent variables in the RNN encoder-decoder (Bowman et al., 2016).

3.8 Topic Word Prediction

Topic words play a key role in generating an informative response. We resort to pointwise mutual information (PMI) (Church and Hanks, 1990) for predicting a list of topic words that are relevant to a post. Let x and y indicate a word in a post X and its response Y respectively, and PMI is computed as follows:

$$PMI(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \quad (18)$$

Then, the relevance score of a topic word to a given post $x_1x_2 \cdots x_n$ can be approximated as follows, similar to (Mou et al., 2016):

$$REL(x_1, ..., x_n, y) \approx \sum_{i=1}^{n} PMI(x_i, y) \quad (19)$$

During training, the words in a response with high REL scores to the post are treated as topic words. During test, we use REL to select the top ranked words as topic words for a post.

4 Experiment

4.1 Data Preparation

We collected a Chinese dialogue dataset from Weibo [2]. We crawled about 10 million post-responses pairs. Since our model needs the sentence function label for each pair, we built a classifier to predict the sentence function automatically to construct large-scale labeled data. Thus,

[2]http://www.weibo.com

we sampled about 2,000 pairs from the original dataset and annotated the data manually with four categories, i.e., *interrogative, imperative, declarative* and *other*. This small dataset was partitioned into the training, validation, and test sets with the ratio of 6:1:1. Three classifiers, including LSTM (Hochreiter and Schmidhuber, 1997), Bi-LSTM (Graves et al., 2005) and a self-attentive model (Lin et al., 2017), were attempted on this dataset. The results in Table 1 show that the self-attentive classifier outperforms other models and achieves the best accuracy of 0.78 on the test set.

Model	Accuracy
LSTM	0.60
Bi-LSTM	0.75
Self-Attentive	0.78

Table 1: Accuracy of sentence function classification on the 2,000 post-response pairs.

We then applied the self-attentive classifier to annotate the large dataset and obtained a dialogue dataset with noisy sentence function labels[3]. To balance the distribution of sentence functions, we randomly sampled about 0.6 million pairs for each sentence function to construct the final dataset. The statistics of this dataset are shown in Table 2. The dataset[4] is available at `http://coai.cs.tsinghua.edu.cn/hml/dataset`.

Training	#Post	1,963,382	
	#Response	Interrogative	618,340
		Declarative	672,346
		Imperative	672,696
Validation	#Post	24,034	
	#Response	Interrogative	7,045
		Declarative	9,685
		Imperative	7,304
Test	#Post	6,000	

Table 2: Corpus statistics.

4.2 Experiment Settings

Our model was implemented with TensorFlow[5]. We applied bidirectional GRU with 256 cells to the encoder and GRU with 512 cells to the decoder. The dimensions of word embedding and function category embedding were both set to 100. We also set the dimension of latent variables to 128. The vocabulary size was set to 40,000. Stochastic gradient descent (Qian, 1999) was used to optimize our model, with a learning rate of 0.1, a decay rate of 0.9995, and a momentum of 0.9. The batch size was set to 128. Our codes are available at `https://github.com/kepei1106/SentenceFunction`.

We chose several state-of-the-art baselines, which were implemented with the settings provided in the original papers:

Conditional Seq2Seq (c-seq2seq): A Seq2Seq variant which takes the category (i.e., function type) embedding as additional input at each decoding position (Ficler and Goldberg, 2017).

Mechanism-aware (MA): This model assumes that there are multiple latent responding mechanisms (Zhou et al., 2017). The number of responding mechanisms is set to 3, equal to the number of function types.

Knowledge-guided CVAE (KgCVAE): A modified CVAE which aims to control the dialog act of a generated response (Zhao et al., 2017).

4.3 Automatic Evaluation

Metrics: We adopted *Perplexity (PPL)* (Vinyals and Le, 2015), *Distinct-1 (Dist-1)*, *Distinct-2 (Dist-2)* (Li et al., 2016a), and *Accuracy (ACC)* to evaluate the models at the content and function level. Perplexity can measure the grammaticality of generated responses. Distinct-1/distinct-2 is the proportion of distinct unigrams/bigrams in all the generated tokens, respectively. Accuracy measures how accurately the sentence function can be controlled. Specifically, we compared the prespecified function (as input to the model) with the function of a generated response, which is predicted by the self-attentive classifier (see Section 4.1).

Model	PPL	Dist-1	Dist-2	ACC
c-seq2seq	57.14	949/.007	5177/.041	0.973
MA	**46.08**	745/.005	2952/.027	0.481
KgCVAE	56.81	1531/**.009**	10683/.070	0.985
Our Model	55.85	**1833**/.008	**15586/.075**	**0.992**

Table 3: Automatic evaluation with perplexity (PPL), distinct-1 (Dist-1), distinct-2 (Dist-2), and accuracy (ACC). The integers in the Dist-* cells denote the total number of distinct n-grams.

Results: Our model has lower perplexity than c-seq2seq and KgCVAE, indicating that the model is comparable with other models in generating grammatical responses. Note that MA has the lowest perplexity because it tends to generate generic responses.

[3]Though the labels are noisy, the data are sufficient to train a generation model in practice.

[4]Note that we strictly obeyed the policies of Weibo and anonymized potential private information in dialogues. This dataset is strictly limited for academic use.

[5]https://github.com/tensorflow/tensorflow

Model	Interrogative			Declarative			Imperative		
	Gram.	Appr.	Info.	Gram.	Appr.	Info.	Gram.	Appr.	Info.
Ours vs. c-seq2seq	0.534	0.536	0.896*	0.630*	0.573*	0.764*	0.685*	0.504	0.893*
Ours vs. MA	0.802*	0.602*	0.675*	0.751*	0.592*	0.617*	0.929*	0.568*	0.577*
Ours vs. KgCVAE	0.510	0.626*	0.770*	0.546*	0.515*	0.744*	0.780*	0.521*	0.837*

Table 4: Manual evaluation results for different functions. The scores indicate the percentages that our model wins the baselines after removing tie pairs. The scores of our model marked with * are significantly better than the competitors (Sign Test, *p-value* < 0.05).

As for distinct-1 and distinct-2, our model generates remarkably more distinct unigrams and bigrams than the baselines, indicating that our model can generate more diverse and informative responses compared to the baselines.

In terms of sentence function accuracy, our model outperforms all the baselines and achieves the best accuracy of 0.992, which indicates that our model can control the sentence function more precisely. MA has a very low score because there is no direct way to control sentence function, instead, it learns automatically from the data.

4.4 Manual Evaluation

To evaluate the generation quality and how well the models can control sentence function, we conducted pair-wise comparison. 200 posts were randomly sampled from the test set and each model was required to generate responses with three function types to each post. For each pair of responses (one by our model and the other by a baseline, along with the post), annotators were hired to give a preference (win, lose, or tie). The total annotation amounts to $200 \times 3 \times 3 \times 3$=5,400 since we have three baselines, three function types, and three metrics. We resorted to a crowdsourcing service for annotation, and each pair-wise comparison was judged by 5 curators.

Metrics: We designed three metrics to evaluate the models from the perspectives of sentence function and content: *grammaticality* (whether a response is grammatical and coherent with the sentence function we prespecified), *appropriateness* (whether a response is a logical and appropriate reply to its post), and *informativeness* (whether a response provides meaningful information via the topic words relevant to the post). Note that the three metrics were separately evaluated.

Results: The scores in Table 4 represent the percentages that our model wins a baseline after removing tie pairs. A value larger than 0.5 indicates that our model outperforms its competitor. Our model outperforms the baselines significantly in most cases (Sign Test, with *p-value* < 0.05). Among the three function types, our model performs significantly better than the baselines when generating declarative and imperative responses. As for interrogative responses, our model is better but the difference is not significant in some settings. This is because interrogative patterns are more apparent and easier to learn, thereby all the models can capture some of the patterns to generate grammatical and appropriate responses, resulting in more ties. By contrast, declarative and imperative responses have less apparent patterns whereas our model is better at capturing the global patterns through modeling the word types explicitly.

We can also see that our model obtains particularly high scores in informativeness. This demonstrates that our model is better to generate more informative responses, and is able to control sentence functions at the same time.

The annotation statistics are shown in Table 5. The percentage of annotations that at least 4 judges assign the same label (at least 4/5 agreement) is larger than 50%, and the percentage for at least 3/5 agreement is about 90%, indicating that annotators reached a moderate agreement.

	At least 3/5	At least 4/5
Grammaticality	91.7%	60.1%
Appropriateness	88.6%	52.5%
Informativeness	95.9%	71.2%

Table 5: Annotation statistics. *At least n/5* means there are no less than n judges assigning the same label to a record during annotation.

4.5 Words and Patterns in Function Control

To further analyze how our model realizes the global control of sentence function, we presented frequent words and frequent word patterns within each function. Specifically, we counted the frequency of a function-related word in the generated responses. The type of a word is predicted by the type controller. Further, we replaced the

Function	Frequent Words		Frequent Patterns		Response Examples	
	Chinese	English	Chinese	English	Chinese	English
Interrogative	？ 是 吗 说 什么	? be particle mean what	x是说y吗？	Does x mean y?	你是说我帅吗？	Do you mean I'm handsome?
			x是在y吗？	Is x y?	你是在夸我吗？	Are you praising me?
			x在哪y啊？	Where does x y?	你在哪上班啊？	Where do you work?
			x想y什么z？	What z does x want to y?	你想要什么类型的？	What type do you want to choose?
Imperative	！ 要 可以 来 请	! will can come please	那就y吧	Do y, then.	那就好好养着吧	Take care of yourself, then.
			x要把y给z	Let x give y to z.	我要把你的房子给你	Let me give your house to you.
Declarative	是 也 觉得 可是 没	be also/too think but no	x也是y，可是z	x also y, but z	我也是这么想的，可是我要找一个人，哈哈	I also think so, but I will find a person. Ha-ha.
			x也是，a都b	x, too, and a has b.	我也是，我的粉丝都被我震精了	Me, too, and my fans have been shocked by me.

Figure 3: Frequent function-related words and frequent patterns containing at least 3 function-related words. The letters denote the variables which replace ordinary and topic words in the generated responses. The underlined words in responses are those occurring in patterns.

ordinary and topic words of a generated response with variables and treated each response as a sequence of function-related words and variables. We then used the Apriori algorithm (Agrawal and Srikant, 1994) to mine frequent patterns in these sequences. We retained frequent patterns that consist of at most 5 words and appear in at least 2% of the generated responses.

Figure 3 presents the most frequent words (the second and third columns) and patterns (the fourth and fifth columns) for each function type. Note that the word patterns can be viewed as an abstract representation of sentence function. We observed that:

First, function-related words are distributed at multiple positions of a sentence, indicating that realizing a sentence function needs a global control by not only predicting the word types but also planning the words of different types properly.

Second, the frequent words clearly reveal the difference between function types. For instance, interrogatives like 什么(what), ？(?) and 吗(particle) are commonly seen in interrogative responses, words like 请(please), 来(come), and 要(will) occur frequently in imperative responses. Further, word patterns in different function types differ significantly (see the fourth/fifth columns), indicating that the model is able to learn function-specific word patterns.

Third, interrogative and imperative responses have explicit patterns, while declarative responses are more implicit and divergent in pattern. Interrogative responses fall into *Yes-No, Wh-, How-*, and other questions. Imperative responses generally start with the base form of verbs or imperative words including 请(please). Our model succeeds in capturing two typical forms in declarative responses: adversative and progressive structures.

4.6 Case Study

Post:	如果有一天我突然跟你绝交，你会怎么样？ What would you do if I suddenly broke up with you someday?
Target Function:	Interrogative
c-seq2seq:	你是谁？ Who are you?
MA:	我会彻查关于法律相关司法措施的结论吗？ Should I investigate the conclusion of judicial measure related to the law?
KgCVAE:	你是不是想我了 Did you miss me?
Our Model:	你**说**的**是**我的错**吗？** **Do** you **mean** that it's my fault**?**
Target Function:	Imperative
c-seq2seq:	我会好好的！ I'll be fine!
MA:	绝交基佬，绝交基佬 Break up with guys. Break up with guys.
KgCVAE:	我会告诉你的！ I will tell you!
Our Model:	我**会把**你的礼物**给**你的，**放心** **Rest assured** that I **would give** your gift **to** you.
Target Function:	Declarative
c-seq2seq:	我也是这么认为的，我觉得我是个好孩子 I think so, too. I feel that I'm a good man.
MA:	我也不知道怎么办，但是我怕怕 I don't know what to do, but I'm in fear.
KgCVAE:	我也觉得我是个好男人。 I think I'm a good man, too.
Our Model:	我**觉得**我**会**很傻，我**会**被我妈骂的 I **would think** that I was stupid and I **would** be blamed by my mother**.**

Figure 4: Generated responses of all the models for different sentence functions. In the responses of our model, function-related words are in **red** and topic words in blue. The word type is predicted by the type controller.

We presented an example in Figure 4 to show that our model can generate responses of different function types better compared to baselines. We can see that each function type can be realized by a natural composition of function-related words (in red) and topic words (in blue). Moreover, function-related words are different and are placed at different positions across function types, indicating that the model learns function-specific word patterns. These examples also show that the compatibility issue of controlling sentence function and generating informative content is well addressed by planning function-related and topic words properly.

Post	如果有一天我突然跟你绝交，你会怎么样？ What would you do if I suddenly broke up with you someday?
Interrogative Response #1	你**说的是**我的错**吗？** **Do** you **mean** that it's my fault**?**
Interrogative Response #2	你**会不会**说话**？** **Can** you speak normally**?**
Interrogative Response #3	你想我**怎样？**我**要不要**绝交**？** **What do** you think I should do**? Shall** I break up with you**?**

Figure 5: Different patterns of interrogative responses generated by our model.

Furthermore, we verified the ability of our model to capture fine-grained patterns within a sentence function. We took interrogative responses as example and obtained responses by drawing latent variable samples repeatedly. Figure 5 shows interrogative responses with different patterns generated by our model given the same post. The model generates several Yes-No questions led by words such as 吗(do), 会(can) and 要(shall), and a Wh-question led by 怎样(what). This example shows that the latent variable can capture the fine-grained patterns and improve the diversity of responses within a function.

5 Conclusion

We present a model to generate responses with both controllable sentence function and informative content. To deal with the global control of sentence function, we utilize a latent variable to capture the various patterns for different sentence functions. To address the compatibility issue, we devise a type controller to handle function-related and topic words explicitly. The model is thus able to control sentence function and generate informative content simultaneously. Extensive experiments show that our model performs better than several state-of-the-art baselines.

As for future work, we will investigate how to apply the technique to multi-turn conversational systems, provided that the most proper sentence function can be predicted under a given conversation context.

Acknowledgments

This work was partly supported by the National Science Foundation of China under grant No.61272227/61332007 and the National Basic Research Program (973 Program) under grant No. 2013CB329403.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499.

Adrian Akmajian. 1984. Sentence types and the form-function fit. *Natural Language Linguistic Theory*, 2(1):1–23.

Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. 2017. Affective neural response generation. *arXiv preprint arXiv:1709.03968*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, pages 22–29.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–632.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 634–642.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Xiang Li, Lili Mou, Rui Yan, and Ming Zhang. 2016b. Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2845–2851.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of International Conference on Learning Representations*.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of 26th International Conference on Computational Linguistics*, pages 3349–3358.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1374–1384.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

Laurie E. Rozakis. 2003. *The complete idiot's guide to grammar and style*. Alpha.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI conference on Artificial Intelligence*.

Iulian Vlad. Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI conference on Artificial Intelligence*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1577–1586.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *International Conference on Machine Learning Deep Learning Workshop*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of AAAI conference on Artificial Intelligence*.

Zhou Yu, Ziyu Xu, Alan W Black, and Alex I. Rudnicky. 2016. Strategy and policy learning for non-task-oriented conversational systems. In *Proceedings of 17th Annual SIGdial Meeting on Discourse and Dialogue*.

George Yule. 2010. *The study of language*. Cambridge university press.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Proceedings of AAAI conference on Artificial Intelligence*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of AAAI conference on Artificial Intelligence*.

Xianda Zhou and William Yang Wang. 2017. Mojitalk: Generating emotional responses at scale. *arXiv preprint arXiv:1711.04090*.

Sentiment Adaptive End-to-End Dialog Systems

Weiyan Shi
[24]7.ai
`weiyan.shi@247.ai`

Zhou Yu
University of California, Davis
`joyu@ucdavis.edu`

Abstract

End-to-end learning framework is useful for building dialog systems for its simplicity in training and efficiency in model updating. However, current end-to-end approaches only consider user semantic inputs in learning and under-utilize other user information. Therefore, we propose to include user sentiment obtained through multimodal information (acoustic, dialogic and textual), in the end-to-end learning framework to make systems more user-adaptive and effective. We incorporated user sentiment information in both supervised and reinforcement learning settings. In both settings, adding sentiment information reduced the dialog length and improved the task success rate on a bus information search task. This work is the first attempt to incorporate multimodal user information in the adaptive end-to-end dialog system training framework and attained state-of-the-art performance.

1 Introduction

Most of us have had frustrating experience and even expressed anger towards automated customer service systems. Unfortunately, none of the current commercial systems can detect user sentiment and let alone act upon it. Researchers have included user sentiment in rule-based systems (Acosta, 2009; Pittermann et al., 2010), where there are strictly-written rules that guide the system to react to user sentiment. Because traditional modular-based systems are harder to train, to update with new data and to debug errors, end-to-end trainable systems are more popular. However, no work has tried to incorporate sentiment information in the end-to-end trainable systems so far to create sentiment-adaptive systems that are easy to train. The ultimate evaluators of dialog systems are users. Therefore, we believe dialog system research should strive for better user satisfaction. In this paper, we not only included user sentiment information as an additional context feature in an end-to-end supervised policy learning model, but also incorporated user sentiment information as an immediate reward in a reinforcement learning model. We believe that providing extra feedback from the user would guide the model to adapt to user behaviour and learn the optimal policy faster and better.

There are three contributions in this work: 1) an audio dataset[1] with sentiment annotation (the annotators were given the complete dialog history); 2) an automatic sentiment detector that considers conversation history by using dialogic features, textual features and traditional acoustic features; and 3) end-to-end trainable dialog policies adaptive to user sentiment in both supervised and reinforcement learning settings. We believe such dialog systems with better user adaptation are beneficial in various domains, such as customer services, education, health care and entertainment.

2 Related Work

Many studies in emotion recognition (Schuller et al., 2003; Nwe et al., 2003; Bertero et al., 2016) have used only acoustic features. But there has been work on emotion detection in spoken dialog systems incorporating extra information as well (Lee and Narayanan, 2005; Devillers et al., 2003; Liscombe et al., 2005; Burkhardt et al., 2009; Yu et al., 2017). For example, Liscombe et al. (2005) explored features like users' dialog act, lexical context and discourse context of the previous turns. Our approach considered accumulated di-

[1]The dataset is available here.

alogic features, such as total number of interruptions, to predict user sentiment along with acoustic and textual features.

The traditional method to build dialog system is to train modules such as language understanding component, dialog manager and language generator separately (Levin et al., 2000; Williams and Young, 2007; Singh et al., 2002). Recently, more and more work combines all the modules in an end-to-end training framework (Wen et al., 2016; Li et al., 2017; Dhingra et al., 2016; Williams et al., 2017; Liu and Lane, 2017a). Specifically related to our work, Williams et al. (2017) built a model, which combined the traditional rule-based system and the modern deep-learning-based system, with experts designing actions masks to regulate the neural model. Action masks are bit vectors indicating allowed system actions at certain dialog state. The end-to-end framework made dialog system training simpler and model updating easier.

Reinforcement learning (RL) is also popular in dialog system building (Zhao and Eskenazi, 2016; Liu and Lane, 2017b; Li et al., 2016). A common practice is to simulate users. However, building a user simulator is not a trivial task. Zhao and Eskenazi (2016) combines the strengths of reinforcement learning and supervised learning to accelerate the learning of a conversational game simulator. Li et al. (2016) provides a standard framework for building user simulators, which can be modified and generalized to different domains. Liu and Lane (2017b) describes a more advanced way to build simulators for both the user and the agent, and train both sides jointly for better performance. We simulated user sentiment by sampling from real data and incorporated it as immediate rewards in RL, which is different from common practice of using task success as delayed rewards in RL training.

Some previous module-based systems integrated user sentiment in dialog planning (Acosta, 2009; Acosta and Ward, 2011; Pittermann et al., 2010). They all integrated user sentiment in the dialog manager with manually defined rules to react to different user sentiment and showed that tracking sentiment is helpful in gaining rapport with users and creating interpersonal interaction in the dialog system. In this work, we include user sentiment into end-to-end dialog system training and make the dialog policy learn to choose dialog actions to react to different user sentiments automatically. We achieve this through integrating user sentiment into reinforcement reward design. Many previous RL studies used delayed rewards, mostly task success. However, delayed rewards make the converging speed slow, so some studies integrated estimated per-turn immediate reward. For example, Ferreira and Lefèvre (2013) explored expert-based reward shaping in dialog management and Ultes et al. (2017) proposed *Interaction Quality (IQ)*, a less subjective variant of user satisfaction, as immediate reward in dialog training. However, both methods are not end-to-end trainable, and require manual input as prior, either in designing proper form of reward, or in annotating the *IQ*. Our approach is different as we detect the multimodal user sentiment on the fly and does not require manual input. Because sentiment information comes directly from real users, our method will adapt to user sentiment as the dialog evolves in real time. Another advantage of our model is that the sentiment scores come from a pre-trained sentiment detector, so no manual annotation of rewards is required. Furthermore, the sentiment information is independent of the user's goal, so no prior domain knowledge is required, which makes our method generalizable and independent of the task.

3 Dataset

We experimented our methods on DSTC1 dataset (Raux et al., 2005), which has a bus information search task. Although DSTC2 dataset is a more commonly-used dataset in evaluating dialog system performance, the audio recordings of DSTC2 are not publicly available and therefore, DSTC1 was chosen. There are a total of 914 dialogs in DSTC1 with both text and audio information. Statistics of this dataset are shown in Table 1. We used the automatic speech recognition (ASR) as the user text inputs instead of the transcripts, because the system's action decisions heavily depend on ASR. There are 212 system action templates in this dataset. Four types of entities are involved, `<place>`, `<time>`, `<route>`, and `<neighborhood>`.

4 Annotation

We manually annotated 50 dialogs consisting of 517 conversation turns for user sentiment. Sentiment is categorized into `negative`, `neutral` and `positive`. The annotator had access to the

Category	Total
total dialogs	914
total dialogs in train	517
total dialogs in test	397

Statistics	Total
avg dialog len	13.8
vocabulary size	685

Table 1: Statistics of the text data.

Category	Total
total dialogs	50
total audios	517
total audios in train	318
total audios in dev	99
total audios in test	100

Category	Total
neutral	254
negative	253
positive	10

Table 2: Statistics of the annotated audio set.

entire dialog history in the annotation process because the dialog context gives the annotators a holistic view of the interactions, and annotating user sentiment in a dialog without the context is really difficult. Some previous studies have also performed similar user information annotation given context, such as Devillers et al. (2002). The annotation scheme is described in Table 10 in Appendix A.2. To address the concern that dialog quality may bias the sentiment annotation, we explicitly asked the annotators to focus on users' behaviour instead of the system, and hid all the details of multimodal features from the annotators. Moreover, two annotators were calibrated on 37 audio files, and reached an inter-annotator agreement (kappa) of 0.74. The statistics of the annotation results are shown in Table 2. The skewness in the dataset is due to the data's nature. In the annotation scheme, `positive` is defined as "excitement or other positive feelings", but people rarely express obvious excitement towards automated task-oriented dialog systems. What we really want to distinguish is neutral and positive cases from negative cases so as to avoid the negative sentiment, and the dataset is balanced for these two cases. To the best of our knowledge, our dataset is the first publicly available dataset that annotated user sentiment with respect to the entire dialog history. There are similar datasets with emotion annotations (Schuller et al., 2013) but are not labeled under dialog contexts.

5 Multimodal Sentiment Classification

To detect user sentiment, we extracted a set of acoustic, dialogic and textual features.

5.1 Acoustic features

We used openSMILE (Eyben et al., 2013) to extract acoustic features. Specifically, we used the paralinguistics configuration from Schuller et al. (2003), which includes 1584 acoustic features, such as pitch, volume and jitter. In order to avoid possible overfitting caused by the large number of acoustic features, we performed tree-based feature selection (Pedregosa et al., 2011) to reduce the size of acoustic features to 20. The selected features are listed in Table 12 in Appendix A.4.

5.2 Dialogic features

Four categories of dialogic features are selected according to previous literature (Liscombe et al., 2005) and the statistics observed in the dataset. We used not only the per-turn statistics of these features, but also the accumulated statistics of them throughout the entire conversation so that the sentiment classifier can also take the entire dialog context into consideration.

Interruption is defined as the user interrupting the system speech. Interruptions occurred fairly frequently in our dataset (4896 times out of 14860 user utterances).

Button usage When the user is not satisfied with the ASR performance of the system, he/she would rather choose to press a button for ”yes/no” questions, so the usage of buttons can be an indication of negative sentiment. During DSTC1 data collection, users were notified about the option to use buttons, so this kind of information is available in the data.

Repetitions There are two kinds of repetitions: the user asks the system to repeat the previous sentence, and the system keeps asking the same question due to failures to catch some important entity. In our model, we combined these two situations as one feature because very few user repetitions occur in our data ($<1\%$). But for other data, it might be helpful to separate them.

Start over is active when the user chooses to restart the task in the middile of the conversation. The system is designed to give the user an option to start over after several turns. If the user takes this offer, he/she might have negative sentiment.

5.3 Textual features

We also noticed that the semantic content of the utterance was relevant to sentiment. So we used the entire dataset as a corpus and created a tf-idf vector for each utterance as textual features.

5.4 Classification results

The sentiment classifier was trained on the 50 dialogs annotated with sentiment labels. The predictions made by this classifier were used for the supervised learning and reinforcement learning in the later sections. We used random forest as our classifier (an implementation from scikit-learn (Pedregosa et al., 2011)), as we had limited annotated data. We separated the data to be 60% for training, 20% for validation and 20% for testing. Due to the randomness in the experiments, we ran all the experiments 20 times and reported the average results of different models in Table 4. We also conducted unpaired one-tailed t-test to assess the statistical significance.

We extracted 20 acoustic features, eight dialogic features and 164 textual features. From Table 4, we see that the model combining all the three categories of features performed the best (0.686 in F-1, $p < 1\mathrm{e}{-6}$ compared to acoustic baseline). One interesting observation is that by only using eight dialogic features, the model already achieved 0.596 in F-1. Another interesting observation is that using 164 textual features alone reached a comparable performance (0.664), but the combination of acoustic and textual features actually brought down the performance to 0.647. One possible reason is that the acoustic information has noise that confused the textual information when combined. But this observation doesn't necessarily apply to other datasets. The significance tests show that adding dialogic features improved the baseline significantly. For example, the model with both acoustic features and dialogic features are significantly better than the one with only acoustic features ($p < 1\mathrm{e}{-6}$). In Table 3, we listed the dialogic features with their relative importance rank, which were obtained from ranking their feature importance scores in the classifier. We observe that "total interruptions so far" is the most useful dialogic features to predict user sentiment. The sentiment detector trained will be integrated in the end-to-end learning described later.

Dialogic Features	**Relative Rank of importance**
total interruptions so far	1
interruptions	2
total button usages so far	3
total repetitions so far	4
repetition	5
button usage	6
total start over so far	7
start over	8

Table 3: Dialogic features' relative importance rank in sentiment detection.

Model	**Avg. of F-1**	**Std. of F-1**	**Max of F-1**
Acoustic features only	0.635	0.027	0.686
Dialogic features only	0.596	0.001	0.596
Textual features only *	0.664	0.010	0.685
Textual + Dialogic *	0.672	0.011	0.700
Acoustic + Dialogic *	0.680	0.019	0.707
Acoustic + Textual	0.647	0.025	0.686
Acoustic + Dialogic + Text *	**0.686**	**0.028**	**0.756**

Table 4: Results of sentiment detectors using different features. The best result is highlighted in **bold** and * indicates statistical significance compared to the baseline, which is using acoustic features only. ($p < 0.0001$)

6 Supervised Learning (SL)

We incorporated the detected user sentiment from the previous section into a supervised learning framework for training end-to-end dialog systems. There are many studies on building a dialog system in a supervised learning setting (Bordes and Weston (2016); Eric and Manning (2017); Seo et al. (2016); Liu and Lane (2017a); Li et al. (2017); Williams et al. (2017)). Following these approaches, we treated the problem of dialog policy learning as a classification problem, which is to select actions among system action templates given conversation history. Specifically, we decided to adopt the framework of *Hybrid Code Network (HCN)* introduced in Williams et al. (2017), because it is the current state-of-the-art model. We reimplemented *HCN* and used it as the baseline system, given the absence of direct comparison on DSTC1 data. One caveat is that *HCN* used action masks (bit vectors indicating allowed actions at certain dialog states) to prevent impossible system actions, but we didn't use hand-crafted action masks in the supervised learning setting because manually designing action masks for 212 action templates is very labor-intensive. This makes our method more general and adaptive to different tasks. All the dialog modules were trained together instead of separately. Therefore, our method is end-to-end trainable and doesn't require human expert involvement.

We listed all the context features used in Williams et al. (2017) in Table 11 in Appendix A.3. In our model, we added one more set of context features, the user-sentiment-related features. For entity extraction, given that the entity values in our dataset form a simple unique fixed set, we used simple string matching. We conducted three experiments: the first one used entity presences as context features, which serves as the baseline; the second one used entity presences in addition to all the raw dialogic features mentioned in Table 3; the third experiment used the baseline features plus the predicted sentiment label by the prebuilt sentiment detector (converted to one-hot vector) instead of the raw dialogic features. We used the entire DSTC1 dataset to train the supervised model. The input is the normalized natural language and the contexutal features, and the output is the action template id. We kept the same experiment setting in Williams et al. (2017), e.g. `last_action_taken` was also used as a feature, along with word embeddings (Mikolov et al., 2013) and bag-of-words; LSTM with 128 hidden-units and AdaDelta optimizer (Zeiler, 2012) were used to train the model.

The results of different models are shown in Table 5. We observe that using the eight raw dialogic features did not improve turn-level F-1 score. One possible reason is that a total of eight dialogic features were added to the model, and some of them might contain noises and therefore caused the model to overfit. However, using predicted sentiment information as an extra feature, which is a more condensed information, outperformed the other models both in terms of turn-level F-1 score and dialog accuracy which indicates if all turns in a dialog are correct. The difference in absolute F-1 score is small because we have a relatively large test set (5876 turns). But the unpaired one-tailed t-test shows that $p < 0.01$ for both the F-1 and the dialog accuracy. This suggests that including user sentiment information in action planning is helpful in a supervised learning setting.

Model	**Weighted F-1**	**Dialog Acc.**
HCN	0.4198	6.05%
HCN + raw dialogic features	0.4190	5.79%
HCN + predicted sentiment label$*$	**0.4261**	**6.55%**

Table 5: Results of different SL models. The best result is highlighted in **bold**. $*$ indicates that the result is significantly better than the baseline ($p < 0.01$). Dialog accuracy indicates if all turns in a dialog are correct, so it's low. For DSTC2 data, the state-of-art dialog accuracy is 1.9%, consistent with our results.

7 Reinforcement Learning (RL)

In the previous section, we discussed including sentiment features directly as a context feature in a supervised learning model for end-to-end dialog system training, which showed promising results. But once a system operates at scale and interacts with a large number of users, it is desirable for the system to continue to learn autonomously using reinforcement learning (RL). With RL, each turn receives a measurement of goodness called *reward* (Williams et al., 2017). Previously, training task-oriented systems mainly relies on the delayed reward about task success. Due to the lack of informative immediate reward, the training takes a long time to converge. In this work, we propose to include user sentiment as immediate rewards to expedite the reinforcement learning training process and create a better user experience.

To use sentiment scores in the reward function, we chose the *policy gradient* approach (Williams, 1992) and implemented the algorithm based on Zhu (2017). The traditional reward function uses a positive constant (e.g. 20) to reward the success of the task, 0 or a negative constant to penalize the failure of the task after certain number of turns, and gives -1 to each extra turn to encourage the system to complete the task sooner. However, such reward function doesn't consider any feedback from the end-user. It is natural for human to consider conversational partner's sentiment in planning dialogs. So, we propose a set of new reward functions that incorporate user sentiment to emulate human behaviors.

The intuition of integrating sentiment in reward functions is as follows. The ultimate evaluator of dialog systems is the end-users. And user sentiment is a direct reflection of user satisfaction. Therefore, we detected the user sentiment scores from multimodal sources on the fly, and used them as immediate rewards in an adaptive end-to-end dialog training setting. This sentiment information came directly from real users, which made the system adapt to individual user's sentiment as the dialog proceeds. Furthermore, the sentiment information is independent of the task, so our method doesn't require any prior domain knowledge and can be easily generalized to other domains. There have been works that incorporated user information into reward design (Su et al., 2015; Ultes et al., 2017). But they used information from one single channel and sometimes required manual labelling of the reward. Our approach utilizes information from multiple channels and doesn't involve manual work once a sentiment detector is ready.

We built a simulated system in the same bus information search domain to test the effectiveness of using sentiment scores in the reward function. In this system, there are 3 entity types - `<departure>`, `<arrival>`, and `<time>` - and 5 actions, asking for different entities, and giving information. A simple action mask was used to prevent impossible actions, such as giving information of an uncovered place. The inputs to the system are the simulated user's dialog acts and the simulated sentiment sampled from a subset of DSTC1, the $CleanData$, which will be described later. The output of the system is the system action template.

7.1 User simulator

Given that reinforcement learning requires feedback from the environment - in our case, the users - and interacting with real users is always expensive, we created a user simulator to interact with the system. At the beginning of each dialog, the simulated user is initiated with a goal consisting of the three entities mentioned above and the goal remains unchanged throughout the conversation. The user responds to system's questions with entities, which are placeholders like `<departure>` instead of real values. To simulate ASR errors, the simulated user's act type occasionally changes from "informing slot values" to "making noises" at certain probabilities set by hand (10% in our case). Some example dialogs along with their associated rewards are shown in Table 8 and 9 in Appendix A.1.

We simulated user sentiment by sampling from real data, the DSTC1 dataset. There are three steps involved. First, we cleaned the DSTC1 dialogs by removing the audio files with no ASR output and high ASR errors. This resulted in a dataset $CleanData$ with 413 dialogs and 1918 user inputs. We observed that users accumulate their sentiment as the conversation unfolds. When the system repeatedly asks for the same entity, they express stronger sentiment. Therefore, summary statistics that record how many times certain entities have been asked during the conversation is representative of users' accumulating sentiment. We designed a set of summary statistics S that record the statistics of system actions, e.g. how many times the arrival place has been asked or the schedule information has been given.

The second step is to create a mapping between the five simulated system actions and the DSTC1 system actions. We do this by calculating a vector s_{real} consisting of the values in S for each user utterance in $CleanData$. s_{real} is used to compare the similarity between the real dialog and the simulated dialog.

The final step is to sample from $CleanData$. For each simulated user utterance, we calculated the same vector s_{sim} and compared it with each s_{real}. There are two possible results. If there are s_{real} equal to s_{sim},we would randomly sample one from all the matched user utterances to represent the sentiment of the simulated user. But if there is no s_{real} matching s_{sim}, different strategies would be applied based on the reward function used, which will be described in details later. Once we have a sample, the eight dialogic features of the sample utterance are used to calculate the sentiment score. We didn't use the acoustic or the textual features because in a simulated setting, only the dialogic features are valid.

7.2 Experiments

We designed four experiments with different reward functions. A discount factor of 0.9 was applied to all the experiments. And the maximum number of turns is 15. Following Williams et al. (2017), we used LSTM with 32 hidden units for the RNN in the *HCN* and AdaDelta for the optimization, and updated the reinforcement learning policy after each dialog. The ϵ-greedy exploration strategy (Tokic, 2010) was applied here. Given that the entire system was simulated, we only used the presence of each entity and the last action taken by the system as the context features, and didn't use bag-of-words or utterance embedding features.

In order to evaluate the method, we froze the policy after every 200 updates, and ran 500 simulated dialogs to calculate the task success rate. We repeated the process 20 times and reported the average performance in Figure 1, 2 and Table 6.

7.2.1 Baseline

We define the baseline reward as follows without any sentiment involvement.

Reward 1 Baseline

```
if success then
    $R_1 = 20$
else if failure then
    $R_1 = -10$
else if each proceeding turn then
    $R_1 = -1$
end if
```

7.2.2 Sentiment reward with random samples (SRRS)

We designed the first simple reward function with user sentiment as the immediate reward: sentiment with random samples (SRRS). We first drew a sample from real data with matched context; if there was no matched data, a random sample was used instead. Because the amount of $CleanData$ is relatively small, so only 36% turns were covered by matched samples. If the sampled dialogic features were not all zeros, the sentiment reward (SR) was calculated as a linear combination with tunable parameters. We chose it to be $-5P_{neg}-P_{neu}+10P_{pos}$ for simplicity. When the dialogic features were all zero, in most cases it meant the user didn't express an obvious sentiment, we set the reward to be -1.

Reward 2 SRRS

```
if success then
    $R_2 = 20$
else if failure then
    $R_2 = -10$
else if sample with all-zero dialogic features then
    $R_2 = -1$
else if sample with non-zero dialogic features then
    $R_2 = -5P_{neg} - P_{neu} + 10P_{pos}$
end if
```

7.2.3 Sentiment reward with repetition penalty (SRRP)

Random samples in SRRS may result in extreme sentiment data. So we used dialogic features to approximate sentiment for the unmatched data. Specifically, if there were repetitions, which correlate with negative sentiment (see Table 3), we assigned a penalty to that utterance. See Reward 3 Formula below for detailed parameters. 36% turns were covered by real data samples, 15% turns had no match in real data and had repetitions, and 33% turns had no match and no repetition.

Moreover, we experimented with different penalty weights. When we increased the repetition penalty to 5, the success rate was similar to penalty of 2.5. However, when we increased the penalty even further to 10, the success rate was brought down by a large margin. Our interpretation is that increasing the repetition penalty to a big value made the focus less on the real sentiment samples but more on the repetitions, which did not help the learning.

Reward 3 SRRP

```
if success then
    $R_3 = 20$
else if failure then
    $R_3 = -10$
else
    if match then
        if all-zero dialogic features then
            $R_3 = -1$
        else if non-zero dialogic features then
            $R_3 = -5P_{neg} - P_{neu} + 10P_{pos}$
        end if
    else if repeated question then
        $R_3 = -2.5$
    else
        $R_3 = -1$
    end if
end if
```

7.2.4 Sentiment reward with repetition and interruption penalties (SRRIP)

We observed in Section 5 that `interruption` is the most important feature in detecting sentiment, so if an interruption existed in the simulated user input, we assumed it had a negative sentiment and added an additional penalty of -1 to the previous sentiment reward SRRP to test the effect of interruption. 7.5% turns have interruptions.

Reward 4 SRRIP

```
if success then
    $R_4 = 20$
else if failure then
    $R_4 = -10$
else
    $R_4 = R_3(SRRP)$
    if interruption then
        $R_4 = R_4 - 1$
    end if
end if
```

7.3 Experiment results

We evaluated every model on two metrics: dialog lengths and task success rates. We observed in Figure 1 that all the sentiment reward functions, even SRRS with random samples, reduced the average length of the dialogs, meaning that the system finished the task faster. The rationale behind is that by adapting to user sentiment, the model can avoid unnecessary system actions to make systems more effective.

In terms of success rate, sentiment reward with both repetition and interruption penalties (SRRIP) performed the best (see Figure 2). In Figure 2, SRRIP is converging faster than the baseline. For example, around 5000 iterations, it outperforms the baseline by 5% in task success rate (60% vs 55%) with statistical significance ($p < 0.01$). It also converges to a better task success rate after 10000 iterations (92.4% vs 94.3%, $p < 0.01$).

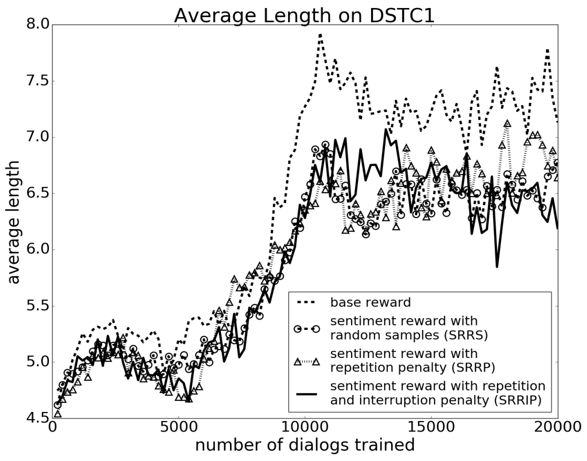


Figure 1: Average dialog length of RL models with different reward functions.

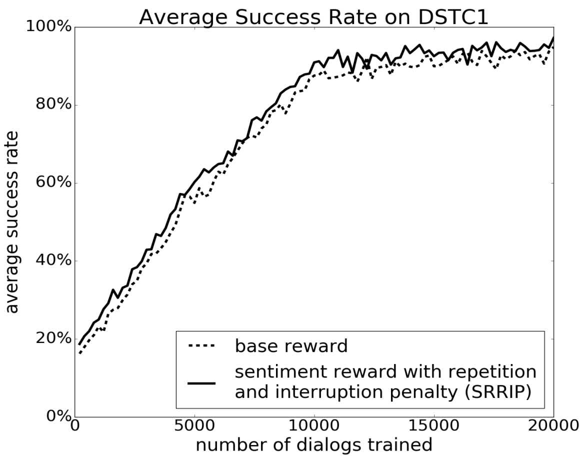


Figure 2: Average success rate of the baseline and the best performing model, SRRIP.

We describe all models' performance in Table 6 in terms of the convergent success rate calculated as the mean success rate after 10000 dialogs. We observed that incorporating various sentiment rewards improved the success rate and expedited the training process overall with statistical significance. We found that even sentiment reward with random samples (SRRS) outperformed the baseline after convergence. By adding penalties for repetition, the algorithm covered more data points, and therefore, the task success rate and the convergence speed improved. We also found that penalizing interruption and repetition together (SRRIP) achieved a slightly better performance compared to penalizing repetition only (SRRP). This suggests that interruptions is another factor to consider when approximating sentiment. But the performances between SRRP and SRRIP is not significant. Our guess is that only 7.5% turns in our data contains interruption and the penalty is just an extra -1, so the model confused this signal with noises. But given more interruptions in the data, interruptions could still be helpful.

Model	**Convergent success rate**
Baseline	0.924
SRRS	0.938$*$
SRRP	0.941$*$
SRRIP	**0.943**$*$

Table 6: Convergent success rate of RL models with different reward functions. It is calculated as the mean success rate after 10000 dialogs. The best result is highlighted in **bold**. $*$ indicates that the result is significantly better than the baseline ($p < 0.01$).

8 Discussion and Future Work

The intuition behind the good performance of models with user sentiment is that the learned policy is in general more sentiment adaptive. For example, there are some system actions that have the same intention but with different surface forms, especially for error-handling strategies. By analyzing the results, we found that when the sentiment adaptive system detected a negative sentiment from the user, it chose to respond the user with a more detailed error-handling strategy than a general one. For example, it chose the template "Where are you leaving from? For example, you can say, <place>", while the baseline model would respond with "Where would you like to leave from?", which doesn't provide details to help the user compared with the previous template. As we all know, dealing with a disappointed user to proceed, providing more details is always better. One example dialog is shown in Table 7. There was no written rules to force the model to choose one specific template under certain situ-

Sentiment Adaptive System	**Baseline System without Sentiment**
SYS: The `<route>`. Where would you like to leave from?	**SYS**: The `<route>`. Where would you like to leave from?
USR: *Yeah **[negative sentiment]***	**USR**: *Yeah*
SYS: Where are you leaving from? For example, you can say, `<place>`.	**SYS**: Right. Where would you like to leave from?

Table 7: An example dialog by different systems in the supervised learning setting. The sentiment-adaptive system gives a more detailed error-handling strategy than the baseline system.

ations, so the model learned these subtle differences on its own. Some may argue that the system could always use a more detailed template to better guide the user instead of distinguishing between two similar system templates. But this is not necessarily true. Ideally, we want the system to be succinct initially to save users' time, because we observe that users, especially repeated users, tend to interrupt long and detailed system utterances. If the user has attempted to answer the system question but failed, then it's beneficial to provide detailed guidance.

The performance of the sentiment detector is a key factor in our work. So in the future, we plan to incorporate features from more channels such as vision to further improve the sentiment predictor's performance, and potentially further improve the performance of the dialog system. We also want to explore more in user sentiment simulation, for example, instead of randomly sampling data for the uncovered cases, we could use linear interpolation to create a similarity score between s_{sim} and s_{real}, and choose the user utterance with the highest score. Furthermore, reward shaping (Ng et al., 1999; Ferreira and Lefèvre, 2013) is an important technique in RL. Specifically, Ferreira and Lefèvre (2013) talked about incorporating expert knowledge in reward design. We also plan to integrate information from different sources into reward function and apply reward shaping. Besides, creating a good user simulator is also very important in the RL training. There are some more advanced methods to create user simulators. For example, Liu and Lane (2017b) described how to optimize the agent and the user simulators jointly using RL. We plan to apply our sentiment reward functions in this framework in the future.

9 Conclusion

We proposed to detect user sentiment from multimodal channels and incorporate the detected sentiment as feedback into adaptive end-to-end dialog system training to make the system more effective and user-adaptive. We included sentiment information directly as a context feature in the supervised learning framework and used sentiment scores as immediate rewards in the reinforcement learning setting. Experiments suggest that incorporating user sentiment is helpful in reducing the dialog length and increasing the task success rate in both SL and RL settings. This work proposed an adaptive methodology to incorporate user sentiment in end-to-end dialog policy learning and showed promising results on a bus information search task. We believe this approach can be easily generalized to other domains given its end-to-end training procedure and task independence.

Acknowledgments

The work is partly supported by Intel Lab Research Gift.

References

Jaime C Acosta. 2009. Using emotion to gain rapport in a spoken dialog system. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 49–54. Association for Computational Linguistics.

Jaime C Acosta and Nigel G Ward. 2011. Achieving rapport with turn-by-turn, user-responsive emotional coloring. *Speech Communication*, 53(9-10):1137–1148.

Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu, Yan Wan, Ricky Ho Yin Chan, and Pascale Fung. 2016. Real-time speech emotion and sentiment recognition for interactive dialogue systems. In *EMNLP*, pages 1042–1047.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Felix Burkhardt, Markus Van Ballegooy, Klaus-Peter Engelbrecht, Tim Polzehl, and Joachim Stegmann. 2009. Emotion detection in dialog systems: applications, strategies and challenges. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1–6. IEEE.

Laurence Devillers, Lori Lamel, and Ioana Vasilescu. 2003. Emotion detection in task-oriented spoken dialogues. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 3, pages III–549. IEEE.

Laurence Devillers, Ioana Vasilescu, and Lori Lamel. 2002. Annotation and detection of emotion in a task-oriented human-human dialog corpus. In *proceedings of ISLE Workshop*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024*.

Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. 2013. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 835–838, New York, NY, USA. ACM.

Emmanuel Ferreira and Fabrice Lefèvre. 2013. Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 108–113. IEEE.

C. M. Lee and Shrikanth Narayanan. 2005. Toward Detecting Emotions in Spoken Dialogs. In *IEEE Transactions on Speech and Audio Processing*, volume 12, pages 293–303.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Jackson Liscombe, Giuseppe Riccardi, and Dilek Hakkani-Tür. 2005. Using context to improve emotion detection in spoken dialog systems. In *Ninth European Conference on Speech Communication and Technology*.

Bing Liu and Ian Lane. 2017a. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *arXiv preprint arXiv:1708.05956*.

Bing Liu and Ian Lane. 2017b. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *arXiv preprint arXiv:1709.06136*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287.

Tin Lay Nwe, Say Wei Foo, and Liyanage C De Silva. 2003. Speech emotion recognition using hidden markov models. *Speech communication*, 41(4):603–623.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Johannes Pittermann, Angela Pittermann, and Wolfgang Minker. 2010. Emotion recognition and adaptation in spoken dialogue systems. *International Journal of Speech Technology*, 13(1):49–60.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*. Citeseer.

Björn Schuller, Gerhard Rigoll, and Manfred Lang. 2003. Hidden markov model-based speech emotion recognition. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 1, pages I–401. IEEE.

Björn Schuller, Stefan Steidl, Anton Batliner, Alessandro Vinciarelli, Klaus Scherer, Fabien Ringeval, Mohamed Chetouani, Felix Weninger, Florian Eyben, Erik Marchi, et al. 2013. The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism. In *Proceedings INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France*.

Minjoon Seo, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Query-regression networks for machine comprehension. *arXiv preprint arXiv:1606.04582*.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Pei-Hao Su, David Vandyke, Milica Gasic, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve

Young. 2015. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. *arXiv preprint arXiv:1508.03386*.

Michel Tokic. 2010. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer.

Stefan Ultes, Paweł Budzianowski, Inigo Casanueva, Nikola Mrkšic, Lina Rojas-Barahona, Pei-Hao Su, Tsung-Hsien Wen, Milica Gašic, and Steve Young. 2017. Domain-independent user satisfaction reward estimation for dialogue policy learning. In *Proc. Interspeech*, pages 1721–1725.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Jason Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*. Association for Computational Linguistics.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Zhou Yu, Alan W Black, and Alexander I Rudnicky. 2017. Learning conversational systems that interleave task and non-task content. *IJCAI*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

Yuke Zhu. 2017. tensorflow-reinforce. `https://github.com/yukezhu/tensorflow-reinforce`.

Embedding Learning Through Multilingual Concept Induction

Philipp Dufter[1], Mengjie Zhao[2], Martin Schmitt[1], Alexander Fraser[1], Hinrich Schütze[1]
[1] Center for Information and Language Processing (CIS) LMU Munich, Germany
[2] École Polytechnique Fédérale de Lausanne, Switzerland
{philipp,martin,fraser}@cis.lmu.de, mengjie.zhao@epfl.ch

Abstract

We present a new method for estimating vector space representations of words: embedding learning by concept induction. We test this method on a highly parallel corpus and learn semantic representations of words in 1259 different languages in a single common space. An extensive experimental evaluation on crosslingual word similarity and sentiment analysis indicates that concept-based multilingual embedding learning performs better than previous approaches.

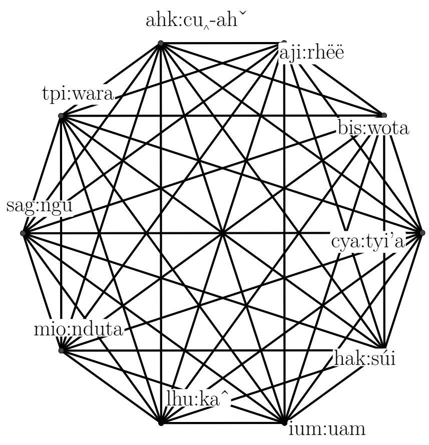


Figure 1: Example of a CLIQUE concept: "water"

1 Introduction

Vector space representations of words are widely used because they improve performance on monolingual tasks. This success has generated interest in multilingual embeddings, shared representation of words across languages (Klementiev et al., 2012). Such embeddings can be beneficial in machine translation in sparse data settings because multilingual embeddings provide meaning representations of source and target in the same space. Similarly, in transfer learning, models trained in one language on multilingual embeddings can be deployed in other languages (Zeman and Resnik, 2008; McDonald et al., 2011; Tsvetkov et al., 2014). Automatically learned embeddings have the added advantage of requiring fewer resources for training (Klementiev et al., 2012; Hermann and Blunsom, 2014b; Guo et al., 2016). Thus, massively multilingual word embeddings (i.e., covering 100s or 1000s of languages) are likely to be important in NLP.

The basic information many embedding learners use is *word-context information*; e.g., the embedding of a word is optimized to predict a representation of its context. We instead learn embeddings from *word-concept information*. As a first approximation, a concept is a set of semantically similar words. Figure 1 shows an example concept and also indicates one way we learn concepts: *we interpret cliques in the dictionary graph as concepts*. The nodes of the dictionary graph are words, its edges connect words that are translations of each other. A dictionary node has the form prefix:word, e.g., "tpi:wara" (upper left node in the figure). The prefix is the ISO 639-3 code of the language; tpi is Tok Pisin.

Our method takes a parallel corpus as input and induces a dictionary graph from the parallel corpus. Concepts and word-concept pairs are then induced from the dictionary graph. Finally, embeddings are learned from word-concept pairs.

A key application of multilingual embeddings is transfer learning. Transfer learning is mainly of interest if the target is resource-poor. We therefore select as our dataset 1664 translations in 1259 languages of the New Testament from PBC, the Parallel Bible Corpus. Since "translation" is an ambiguous word, we will from now on refer to the 1664 translations as "editions". PBC is aligned

English King James Version (KJV)	German Elberfelder 1905	Spanish Americas
And he said , Do it the second time . And they did it the second time ...	Und er sprach : Füllet vier Eimer mit Wasser , und gießet es auf das Brandopfer und auf das Holz . Und er sprach : Tut es zum zweiten Male ! Und sie taten es zum zweiten Male ...	Y dijo : Llenad cuatro cántaros de agua y derramadla sobre el holocausto y sobre la leña . Después dijo : Hacedlo por segunda vez ; y lo hicieron por segunda vez ...

Table 1: Instances of verse 11018034. This multi-sentence verse is an example of verse misalignment.

on the verse level; most verses consist of a single sentence, but some contain several (see Table 1). PBC is a good model for resource-poverty; e.g., the training set (see below) of KJV contains fewer than 150,000 tokens in 6458 verses.

We evaluate multilingual embeddings on two tasks, roundtrip translation (RT) and sentiment analysis. RT on the word level is – to our knowledge – a novel evaluation method: a query word w of language L_1 is translated to its closest (with respect to embedding similarity) neighbor v in L_2 and then backtranslated to its closest neighbor w' in L_1. RT is successful if $w = w'$. There are well-known concerns about RT when it is used in the context of machine translation. A successful roundtrip translation does not necessarily imply that v is of high quality and it is not possible to decide whether an error occurred in the forward or backward translations. Despite these concerns about RT on the sentence level, we show that RT on the word level is a difficult task and an effective measure of embedding quality.

Contributions. (i) We introduce a new embedding learning method, multilingual embedding learning through concept induction. (ii) We show that this new concept-based method outperforms previous approaches to multilingual embeddings. (iii) We propose both word-level and character-level dictionary induction methods and present evidence that concepts induced from word-level dictionaries are better for easily tokenizable languages and concepts induced from character-level dictionaries are better for difficult-to-tokenize languages. (iv) We evaluate our methods on a corpus of 1664 editions in 1259 languages. To the best of our knowledge, this is the first detailed evaluation, involving challenging tasks like word translation and crosslingual sentiment analysis, that has been done on such a large number of languages.

2 Methods

2.1 Pivot languages

Most of our methods are based on bilingual dictionary graphs. With 1664 editions, it is computationally expensive to consider all editions simultaneously (more than 10^6 dictionaries). Thus we split the set of editions in 10 pivot and 1654 remaining editions, and do not compute nor use dictionaries within the 1654 editions. We refer to the ten pivot editions as *pivot languages* and give them a distinct role in concept induction. We refer to all editions (including pivot editions) as *target editions*. Thus, a pivot edition has two roles: as a pivot language and as a target edition.

We select the pivot languages based on their sparseness. Sparseness is a challenge in NLP. In the case of embeddings, it is hard to learn a high-quality embedding for any infrequent word. Many of the world's languages (including many PBC languages) exhibit a high degree of sparseness. But some languages suffer comparatively little from sparseness when simple preprocessing like downcasing and splitting on whitespace is employed.

A simple measure of sparseness that affects embedding learning is the number of types. Fewer types is better since their average frequency will be higher. Table 2 shows the ten languages in PBC that have the smallest number of types in 5000 randomly selected verses. We randomly sample 5000 verses per edition and compare the number of types based on this selection because most editions do not contain a few of the selected 6458 verses.

2.2 Character-level modeling (CHAR)

We will see that tokenization-based models have poor performance on a subset of the 1259 languages. To overcome tokenization problems, we represent a verse of length m bytes, as a sequence of $m - (n - 1) + 2$ overlapping byte n-grams. In this paper, "n-gram" always refers to "byte n-gram". We pad the verse with initial and final space, resulting in two additional n-grams (hence "+2"). This representation is in the spirit of earlier byte-level processing, e.g., (Gillick et al., 2016). There are several motivations for this. (i) We can take advantage of byte-level generalizations. (ii) This is robust if there is noise in the byte encoding. (iii) Characters have different properties in different languages and encodings, e.g., English

iso	name	family; (example) region	types	tokens
lhu	Lahu	Sino-Tibetan; Thailand	1452	268
ahk	Akha	Sino-Tibetan; China	1550	315
hak	Hakka Chinese	Chinese; China	1596	242
ium	Iu Mien	Hmong-Mien; Laos	1779	191
tpi	Tok Pisin	Creole; PNG	1815	177
mio	Pinotepa Mixtec	Oto-Manguean; Oaxaca	1828	208
cya	Highland Chatino	Oto-Manguean; Oaxaca	1868	231
bis	Bislama	Creole; Vanuatu	1872	226
aji	Ajië	Austronesian; Houaïlou	1876	194
sag	Sango	Creole; Central Africa	1895	192

Table 2: Our ten pivot languages, the languages in PBC with the lowest number of types. Tokens in 1000s. Tok Pisin and Bislama are English-based and Sango is a Ngbandi-based creole. PNG = Papua New Guinea

Algorithm 1 χ^2-based dictionary induction

```
 1: procedure DictionaryGraph($C$)
 2:     $A$ = all-edges($C$), $E = []$
 3:     for $d \in [1, 2, \dots, d_{\max}]$ do
 4:         $f_{\max} = 2$
 5:         while $f_{\max} \leq |C|$ do
 6:             $f_{\min} = \max(\min(5, f_{\max}), \frac{1}{10} f_{\max})$
 7:             $(\chi^2, s, t)$ = max-$\chi^2$-edge$(A, f_{\min}, f_{\max}, d)$
 8:             if $\chi^2 < \chi_{\min}$ then
 9:                 $f_{\max} = f_{\max} + 1$; continue
10:             end if
11:             $T$ = extend-ngram$(A, f_{\min}, f_{\max}, d, s, t)$
12:             append$(E, s, T)$
13:             remove-edges$(A, s, T)$
14:         end while
15:     end for
16:     return dictionary-graph = (nodes$(E), E)$
17: end procedure
```

Figure 2: χ^2-based dictionary induction. C is a sentence-aligned corpus. A is initialized to contain all edges, i.e., the fully connected bipartite graph for each parallel verse. E collects the selected dictionary edges. d is the edge degree: in each pass through the loop only edges are considered whose participating units have a degree less than d. $f_{\max}$ is the maximum frequency during this pass. $|C|$ is the number of sentences in the corpus. extend-ngram extends a target ngram to left / right; e.g., if s = "jisas" is aligned with ngram t = "Jesu" in English, then "esus" is added to T. t is always a member of T. remove-edges removes edges in A between s and a member of T.

UTF-8 has properties different from Chinese UTF-8. Thus, universal language processing is easier to design on the byte level.

We refer to this ngram representation as CHAR and to standard tokenization as WORD.

2.3 Dictionary induction

Alignment-based dictionary. We use fastalign (Dyer et al., 2013) to compute word alignments and use GDFA for symmetrization. All alignment edges that occurred at least twice are added to the dictionary graph. Initial experiments indicated that alignment-based dictionaries have poor quality for CHAR, probably due to the fact that overlapping ngram representations of sentences have properties quite different from the tokenized sentences that aligners are optimized for. Thus we use this dictionary induction method only for WORD and developed the following alternative for CHAR.

Correlation-based dictionary (χ^2). χ^2 is a greedy algorithm, shown in Figure 2, that selects, in each iteration, the pair of units that has the highest χ^2 score for cooccurrence in verses. Each selected pair is added to the dictionary and removed from the corpus. Low-frequency units are selected first and high-frequency units last; this prevents errors due to spurious association of high-frequency units with low-frequency units. We perform $d_{\max} = 5$ passes; in each pass, the maximum degree of a dictionary node is $1 \leq d \leq d_{\max}$. So if the node has reached degree d, it is ineligible for additional edges during this pass. Again, this avoids errors due to spurious association of high-frequency units that already participate in many edges with low-frequency units. Recall that this method is only applied for CHAR.

Intra-pivot dictionary. We assume that pivot languages are easily tokenizable. Thus we only consider alignment-based dictionaries (in total 45) within the set of pivot languages.

Pivot-to-target dictionary. We compute an alignment-based and a χ^2-based dictionary between each pivot language and each target edition, yielding a total of 10*1664 dictionaries per dictionary type. (Note that this implies that, for χ^2, the WORD version of the pivot language is aligned with its CHAR version.)

2.4 Concepts

A concept is defined as a set of units that has two subsets: (i) a defining set of words from the ten pivot languages and (ii) a set of target units (words or n-grams) that are linked, via dictionary edges,

Algorithm 2 CLIQUE concept induction

```
 1: procedure CONCEPTS($I \in \mathbb{R}^{n\times n}, \theta, \nu$)
 2:     $G = ([n], \{(i,j) \in [n] \times [n] \mid I_{ij} > \theta\})$
 3:     cliques = get-cliques($G$, 3)
 4:     $G_c := (V_c, E_c) = (\emptyset, \emptyset)$
 5:     for $c_1, c_2 \in$ cliques × cliques do
 6:         if $|c_1 \cap c_2| \geq \nu \min\{|c_1|, |c_2|\}$ then
 7:             $V_c = V_c \cup \{c_1, c_2\}$, $E_c = E_c \cup \{(c_1, c_2)\}$
 8:         end if
 9:     end for
10:     metacliques = get_cliques($G_c$, 1)
11:     concepts = {flatten($c$) | $c \in$ metacliques}
12:     return concepts
13: end procedure
```

Figure 3: CLIQUE concept induction. I is a normalized adjacency matrix of a dictionary graph (i.e., relative frequency of alignment edges with respect to possible alignment edges). get-cliques(G, n) returns all cliques in G of size greater or equal to n. flatten(A) flattens a set of sets. $[n]$ denotes $\{1, 2, \ldots, n\}$. $\theta = 0.4$, $\nu = 0.6$.

to the pivot subset. We selected the ten "easiest" of the 1664 editions as pivot languages. Our premise is that semantic information is encoded in a simply accessible form in the pivot languages and so they should offer a good basis for learning concepts.

We induce concepts from the dictionary graph, a multipartite graph consisting of ten pivot language node/word sets and all target edition node/unit sets (where units are words or n-grams). Edges either connect pivot nodes with other pivot nodes or pivot nodes with target units.

2.4.1 CLIQUE concept induction

If concepts corresponded to each other in the overtly coding pivot languages, if words were not ambiguous and if alignments were perfect, then concepts would be cliques in the pivot part of the dictionary graph. These conditions are too strict for natural languages, so we relax them in our CLIQUE concept induction algorithm (Figure 3). The algorithm identifies maximal multilingual cliques (size ≥ 3) within the dictionary graph of the pivot languages and then merges two cliques if they share enough common words. The merging lets us identify clique-based concepts even if, e.g., a dictionary edge between two words is missing. It also accommodates the situation where more than one word of a pivot language should be part of a concept. The merging step can also be interpreted as metaconcept induction.

Once we have identified the cliques, we project

$N(t)$ ={`bis:Jorim, ium:yo-lim, sag:Yorim, tpi:Jorim`}

$t \in T$={`ac0:Yorim,atg0:iJorimu,bav0:Jorim,bom0:Yorim, dik0:Jorim, dtp0:Yorim, duo0:Jorim, eng1:Jorim, engb:Jorim, fij2:Lorima, fij3:Jorima, gor0:Yorim, hvn0:Yorim, ibo0:Jorim, iri0:Jorri, kmr0:Yorîm, ksd0:Iorim, kwd0:Jorim, lia0:Yorimi, loz0:Jorimi, mbd0:Hurim, mfh0:Yorim, min0:Yorim, mrw0:Yorim,mse0:Jorimma,naq0:Jorimmi, smo1:Iorimo, srn1:Yorim, tsn2:Jorime, yor2:Jórímù`}

Figure 4: Target neighborhood concept example: $N(t) \cup T$. $N(t)$ is the target neighborhood for each of the target words in T.

them to the target editions: a target-unit is added to a clique if it is connected to a proportion $\nu = 0.6$ of its member words (to allow for missing edges). This identifies around 150k clique concepts that cover around 8k of the total vocabulary of 24k English words (WORD).

As an alternative to cliques, Ammar et al. (2016) use connected components (CCs). The reachability relation (induced by CC) is the transitive closure of the edge relation. This results in semantically unrelated words being in the same concept for very low levels of noise. In contrast, cliques are more "strict": only node subsets are considered whose corresponding edge relation is already transitive (or almost so for $\nu = 0.6$). Transitivity across languages often does not hold in alignments or dictionaries; see, e.g., Simard (1999). This is why we only consider cliques (which reflect already existent transitivity) rather than CCs, which impose transitivity where it does not hold naturally.

2.4.2 $N(t)$ (target neighborhood) concept induction

Let $N(t)$ be the neighborhood of target node t in the multipartite dictionary graph, i.e., the set of pivot words that are linked to t. We refer to $N(t)$ as *target neighborhood*. Figure 4 shows an example of such a target neighborhood, the set $N(t)$ consisting of four words.[1] A *target neighborhood concept* consists of a set T of pivot words and all target words t for which $T = N(t)$ holds.

Motivation. Suppose $N(t) = N(u)$ for target nodes t and u from two different languages and $|N(t)|$ covers several pivot languages, e.g., $|N(t)| = |N(u)| = 4$ as in the figure. Again, if units closely corresponded to concepts, if there were no ambiguity, if the dictionary were perfect,

[1]We use numbers and lowercase letters at the fourth position of the prefix to distinguish different editions in the same language, e.g., "0", "3" and "e" in "ace0", "fij3", "enge".

then we could safely conclude that the meanings of t and u are similar; if the meanings of t and u were unrelated, it is unlikely that they would be aligned to the exact same words in four different languages. In reality, there is no exact meaning-form correspondence, there is ambiguity and the dictionary is not perfect. Still, we will see below that defining concepts as target neighborhoods works well.

2.4.3 Filtering target neighborhood concepts

In contrast to CLIQUE, we do not put any constraint on the pivot-to-pivot connections within target neighborhoods; e.g., in Figure 4, we do not require that "bis:Jorim" and "sag:Yorim" are connected by an edge. We evaluate three post-filtering steps of target neighborhoods to increase their quality: restricting target neighborhoods to those that are cliques in $N(t)$**-CLIQUE**; to those that are connected components in $N(t)$**-CC**; and to those of size two that are valid edges in the dictionary in $N(t)$**-EDGE**. For $N(t)$**-EDGE**, we found that taking all edges performs well, so we also consider edges that are proper subsets of target neighborhoods.

2.5 Embedding learning

We adopt the framework of embedding learning algorithms that define contexts and then sample pairs of an input word (more generally, an input unit) and a context word (more generally, a context unit) from each context. The only difference is that our contexts are concepts. For simplicity, we use word2vec (Mikolov et al., 2013a) as the implementation of this model.[2]

2.6 Baselines

Baselines for **multilingual embedding learning.** One baseline is inspired by (Vulić and Moens, 2015). We consider words of one aligned verse in the pivot languages and one target language as a bag of words (BOW) and consider this bag as a context.[3]

Levy et al. (2017) show that sentence ID features (interpretable as an abstract representation of the word's context) are effective. We use a corpus with lines consisting of pairs of an identifier of a verse and a unit extracted from that verse as input to word2vec and call this baseline S-ID.

Lardilleux and Lepage (2009) propose a simple and efficient baseline: **sample-based concept induction**. Words that strictly occur in the same verses are assigned to the same concept. To increase coverage, they propose to sample many different subcorpora.[4] We induce concepts using this method and project them analogous to CLIQUE. We call this baseline SAMPLE.

One novel contribution of this paper is **roundtrip evaluation** of embeddings. We learn embeddings based on a dictionary. The question arises: are the embeddings simply reproducing the information already in the dictionary or are they improving the performance of roundtrip search?

As a baseline, we perform RTSIMPLE, a simple dictionary-based roundtrip translation method. Retrieve the pivot word p in pivot language L_p (i.e., $p \in L_p$) that is closest to the query $q \in L_q$. Retrieve the target unit $t \in L_t$ that is closest to p. Retrieve the pivot word $p' \in L_p$ that is closest to t. Retrieve the unit $q' \in L_q$ that is closest to p'. If $q = q'$, this is an exact hit. We run this experiment for all pivot and target languages.

Note that roundtrip evaluation tests the capability of a system to go from any language to any other language. In an embedding space, this requires two hops. In a highly multilingual dataset of n languages in which not all $O(n^2)$ bilingual dictionaries exist, this requires four hops.

3 Experiments and results

3.1 Data

We use PBC (Mayer and Cysouw, 2014). The version we pulled on 2017-12-11 contains 1664 Bible editions in 1259 languages (based on ISO 639-3 codes) after we discarded editions that have low coverage of the New Testament. We use 7958 verses that have good coverage in these 1664 editions. The data is verse aligned; a verse of the New Testament can consist of multiple sentences. We randomly split verses 6458/1500 into train/test.

3.2 Evaluation

For **sentiment analysis**, we represent a verse as the IDF-weighted sum of its embeddings. Sentiment classifiers (linear SVMs) are trained on the training set of the World English Bible edition

[2]We use `code.google.com/archive/p/word2vec`

[3]The actual implementation slightly differs to avoid very long lines. It does only consider two pivot languages at a time, but writes each verse multiple times.

[4]We use this implementation: `anymalign.limsi.fr`

for the two decision problems positive vs. non-positive and negative vs. non-negative. We create a silver standard by labeling verses in English editions with the NLTK (Bird et al., 2009) sentiment classifier.

A positive vs. negative classification is not reasonable for the New Testament because a large number of verses is mixed, e.g., “Now is come salvation … the power of his Christ: for the accuser … cast down, which accused them before our God …” Note that this verse also cannot be said to be neutral. Splitting the sentiment analysis into two subtasks (“contains positive sentiment: yes/no” and “contains negative sentiment: yes/no”) is an effective solution for this paper.

The two trained models are then applied to the test set of all 1664 editions. All embeddings in this paper are learned on the training set only. So no test information was used for learning the embeddings.

Roundtrip translation. There are no gold standards for the genre of our corpus (the New Testament); for only a few languages out-of-domain gold standards are available. Roundtrip evaluation is an evaluation method for multilingual embeddings that can be applied if no resources are available for a language. Loosely speaking, for a query q in a query language L_q (in our case English) and a target language L_t, roundtrip translation finds the unit w_t in L_t that is closest to q and then the English unit w_e that is closest to w_t. If the semantics of q and w_e are identical (resp. are unrelated), this is deemed evidence for (resp. counter-evidence against) the quality of the embeddings. We work on the level of Bible edition, i.e., two editions in the same language are considered different “languages”.

For a query q, we denote the set of its k_I nearest neighbors in the target edition e by $I_e(q) = \{u_1, u_2, \ldots, u_{k_I}\}$. For each intermediate entry we then consider its k_T nearest neighbors in English. Overall we get a set $T_e(q)$ with $k_I k_T$ predictions for each intermediate Bible edition e. See Figure 5 for an example.

We evaluate the predictions $T_e(q)$ using two sets $G_s(q)$ (strict) and $G_r(q)$ (relaxed) of ground-truth semantic equivalences in English. Precision for a query q is defined as

$$p_i(q) := 1/|E| \textstyle\sum_{e \in E} \min\{1, |T_e(q) \cap G_i(q)|\}$$

where E is the set of all Bible editions and $i \in \{s, r\}$. We report the mean and median across a

query	inter-mediate		predictions
woman ⇒	mujer	⇒	wife woman women widows daughters daughter marry married
⇒	esposa	⇒	marry wife woman married marriage virgin daughters bridegroom

Figure 5: Roundtrip translation example for KJV and Americas Bible (Spanish). In this example $\min\{1, |T_e(q) \cap G_i(q)|\}$ equals 0 for S1 and R1, and 1 for S4 and S16.

```
connu(3), connais(3), connaissent(3), savez(2),
sachant(2), sait(2), sachiez(2), savoir,
sçai, ignorez, connaissiez, sache connaissez,
connaissais, savent, savaient, connoissez,
connue, reconnaîtrez, sais, connaissant,
savons, connaissait, savait
```

Figure 6: Intermediates aggregated over 17 French editions. q=“know”, $N(t)$ embeddings, S16. Intermediates are correct with two possible exceptions: “ignorez” ‘you do not know’; “reconnaîtrez” ‘you recognize’.

set of 70 queries selected from Swadesh (1946)’s list of 100 universal linguistic concepts.

We create G_s and G_r as follows. For WORD, we define $G_s(q) = \{q\}$ and $G_r(q) = L(q)$ where $L(q)$ is the set of words with the same lemma and POS as q. For CHAR, we need to find ngrams that correspond uniquely to the query q. Given a candidate ngram g we consider $c_{qg} := 1/c(g) \sum_{q' \in L(q), \text{substring}(g,q')} c(q')$ where $c(x)$ is the count of character sequence x across all editions in the query language. We add g to $G_i(q)$ if $c_{qg} > \sigma_i$ where $\sigma_s = .75$ and $\sigma_r = .5$. We only consider queries where $G_s(q)$ is non-empty.

We vary the evaluation parameters (i, k_I, k_T) as follows: “S1” represents $(s, 1, 1)$, “S4” $(s, 2, 2)$, “S16” $(s, 2, 8)$, and “R1” $(r, 1, 1)$.

3.3 Corpus generation and hyperparameters

We train with the skipgram model and set vector dimensionality to 200; word2vec default parameters are used otherwise. Each concept – the union of a set of pivot words and a set of target units linked to the pivot words – is written out as a line or (if the set is large) as a sequence of shorter lines. Training corpus size is approximately 50 GB for all experiments. We write several copies of each line (shuffling randomly to ensure lines are different) where the multiplication factor is chosen to result in an overall corpus size of approximately 50 GB.

There are two exceptions. For BOW, we did not find a good way of reducing the corpus size, so this

		roundtrip translation																		sentiment analysis			
		WORD									CHAR									WORD		CHAR	
		S1		R1		S4		S16			S1		R1		S4		S16						
		μ	Md	μ	Md	μ	Md	μ	Md	N	μ	Md	μ	Md	μ	Md	μ	Md	N	pos	neg	pos	neg
1	RTSIMPLE	33	24	37	36					67	24	13	32	21					70				
2	BOW	7	5	8	7	13	12	26	28	69	3	2	3	2	5	4	10	11	70	33	81	13	83
3	S-ID	46	46	52	55	63	76	79	91	65	9	5	9	5	14	9	25	22	70	79	88	65	86
4	SAMPLE	33	23	43	42	54	59	82	96	65	53	**59**	59	**72**	67	85	79	99	58	82	89	77	89
5	CLIQUE	43	36	59	63	67	77	93	99	69	42	46	48	55	60	76	73	98	53	84	89	69	88
6	N(t)	**54**	**59**	**61**	69	**80**	**87**	**94**	**100**	69	50	53	54	59	73	82	90	99	66	82	89	**87**	**90**
7	N(t)-CLIQUE	11	0	11	0	16	0	22	0	18	39	45	41	47	58	74	76	94	56	22	84	61	84
8	N(t)-CC	3	0	3	0	5	0	7	0	5	11	0	11	0	16	0	25	0	21	4	84	40	83
9	N(t)-EDGE	35	30	43	36	56	55	87	94	69	39	29	49	52	64	78	88	**100**	63	84	**90**	84	89

Table 3: Roundtrip translation (mean/median accuracy) and sentiment analysis (F_1) results for word-based (WORD) and character-based (CHAR) multilingual embeddings. N (coverage): # queries contained in the embedding space. The best result *across WORD and CHAR* is set in bold.

corpus is 10 times larger than the others. For S-ID, we use Levy et al. (2017)'s hyperparameters; in particular, we trained for 100 iterations and we wrote each verse-unit pair to the corpus only once, resulting in a corpus of about 4 GB.

We set the n parameter of n-grams to $n = 4$ for Bible editions with $\rho < 2$, $n = 8$ for Bible editions with $2 \leq \rho < 3$ and $n = 12$ for Bible editions with $\rho \geq 3$ where ρ is the ratio between size in bytes of the edition and median size of the 1664 editions. In χ^2 dictionary induction, we set $\chi_{\text{min}} = 100$. In the concept induction algorithm we set $\theta = 0.4$ and $\nu = 0.6$. Except for SAMPLE and CLIQUE, we filter out hapax legomena.

3.4 Results

Table 3 presents evaluation results for roundtrip translation and sentiment analysis.

Validity of roundtrip (RT) evaluation results. RTSIMPLE (line 1) is not competitive; e.g., its accuracy is lower by almost half compared to $N(t)$. We also see that RT is an excellent differentiator of poor multilingual embeddings (e.g., BOW) vs. higher-quality ones like S-ID and $N(t)$. This indicates that RT translation can serve as an effective evaluation measure.

The **concept-based multilingual embedding learning** algorithms CLIQUE and $N(t)$ (lines 5-6) consistently (except S1 WORD) outperform BOW and S-ID (lines 2-3) that are not based on concepts. BOW performs poorly in our low-resource setting; this is not surprising since BOW methods rely on large datasets and are therefore expected to fail in the face of severe sparseness. S-ID performs reasonably well for WORD, but even in that case it is outperformed by $N(t)$, in some cases by a large margin, e.g., μ of 63 for S-ID vs. 80 for $N(t)$ for S4. For CHAR, S-ID results are poor. On sentiment classification, $N(t)$ also consistently outperforms S-ID.

While S-ID provides a clearer signal to the embedding learner than BOW, it is still relatively crude to represent a word as – essentially – its binary vector of verse occurrence. Concept-based methods perform better because they can exploit the more informative dictionary graph.

Comparison of graph-theoretic definitions of concepts: $N(t)$-CLIQUE, $N(t)$-CC. $N(t)$ (line 6) has the most consistent good performance across tasks and evaluation measures. Postfiltering target neighborhoods down to cliques (line 7) and CCs (line 8) does not work. The reason is that the resulting number of concepts is too small; see, e.g., low coverages of $N = 18$ ($N(t)$-CLIQUE) and $N = 5$ ($N(t)$-CC) for WORD and $N = 21$ ($N(t)$-CC) for CHAR. $N(t)$-CLIQUE results are highly increased for CHAR, but still poorer by a large margin than the best methods. We can interpret this result as an instance of a precision-recall tradeoff: presumably the quality of the concepts found by $N(t)$-CLIQUE and $N(t)$-CC is better (higher precision), but there are too few of them (low recall) to get good evaluation numbers.

Comparison of graph-theoretic definitions of concepts: CLIQUE. CLIQUE has strong performance for a subset of measures, e.g., ranks consistently second for RT (except S1 WORD) and sentiment analysis in WORD. Although CLIQUE is perhaps the most intuitive way of inducing a concept from a dictionary graph, it may suffer in relatively high-noise settings like ours.

Comparison of graph-theoretic definitions of concepts: $N(t)$ vs. $N(t)$-EDGE. Recall that $N(t)$-EDGE postfilters target neighborhoods by

[ksw] ဒီးတ◌ၢ်ကမၣ◌်လၢအဟ်လၢယလိၤခဲကနံၣ◌်အံၤ⋆,
⋆ထုးပှ◌ၤအှၣ◌်အသးတန့ၢ်ဘၣ◌်⋆.
[cso] Hi^{3}⋆sa^{3}jun^{3}⋆lɨ́^{13}⋆ma^{3}tson^{2}⋆tsú^{2}⋆
lɨ^{3}ua^{3}⋆cáun^{2}⋆tso^{3}⋆ñí^{1}⋆hná^{1}⋆nɨ́^{2}⋆.
[eng] Neither⋆can⋆they⋆prove⋆the⋆things⋆
whereof⋆they⋆now⋆accuse⋆me⋆.

Figure 7: Verse 44024013. "*" = tokenization boundary. S'gaw Karen (ksw) is difficult to tokenize and CHAR > WORD for $N(t)$. Chinanteco de Sochiapan (cso) has few types, similar to a pivot language, and CHAR < WORD for $N(t)$.

N(t) [CHAR]		S-ID [WORD]		SAMPLE [WORD]		CLIQUE [WORD]	
iso	Δ	iso	Δ	iso	Δ	iso	Δ
arb1	54	pua0	61	jpn1	42	mya2	38
arz0	53	sun2	54	khm2	40	jpn1	36
cop3	49	jpn1	53	cap2	40	khm3	34
srp0	44	khm3	53	khm3	40	bsn0	28
cop2	44	khm2	50	mya2	39	khm2	27
...	...	...	...	...	...	...	...
pis0	-23	vie7	-24	eng8	-7	haw0	-22
pcm0	-23	kri0	-25	enm1	-9	eng4	-23
ksw0	-24	tdt0	-27	lzh2	-9	enm2	-26
lzh2	-41	eng2	-27	eng4	-12	enm1	-26
lzh1	-51	vie6	-29	lzh1	-13	engj	-28

Table 4: Comparison of $N(t)$[WORD] with four other methods. Difference in mean performance (across queries) in R1 per edition. Positive number means better performance of $N(t)$[WORD].

only considering pairs of pivot words that are linked by a dictionary edge. This "quality" filter does seem to work in some cases, e.g., best performance S16 Md for CHAR. But results for WORD are much poorer.

SAMPLE performs best for CHAR: best results in five out of eight cases. However, its coverage is low: $N = 58$. This is also the reason that it does not perform well on sentiment analysis for CHAR ($F_1 = 77$ for pos).

Target neighborhoods $N(t)$. The overall best method is $N(t)$. It is the best method more often than any other method and in the other cases, it ranks second. This result suggests that the assumption that two target units are semantically similar if they have dictionary edges with exactly the same set of pivot words is a reasonable approximation of reality. Postfiltering by putting constraints on eligible sets of pivot words (i.e., the pivot words themselves must have a certain dictionary link structure) does not consistently improve upon target neighborhoods.

WORD vs. CHAR. For roundtrip, WORD is a better representation than CHAR if we just count the bold winners: seven (WORD) vs. three (CHAR), with two ties. For sentiment, the more difficult task is pos and for this task, CHAR is better by 3 points than WORD ($F_1 = 87$, line 6, vs. $F_1 = 84$, lines 9/5). However, Table 4 shows that CHAR < WORD for one subset of editions (exemplified by cso in Figure 7) and CHAR > WORD for a different subset (exemplified by ksw). So there are big differences between CHAR and WORD in both directions, depending on the language. For some languages, WORD performs a lot better, for others, CHAR performs a lot better.

We designed RT evaluation as a word-based evaluation that disfavors CHAR in some cases. The fourgram "ady@" in the World English Bible occurs in "already" (32 times), "ready" (31 times) and "lady" (9 times). Our RT evaluation thus disqualifies "ady@" as a strict match for "ready". But all 17 *aligned* occurrences of "ady@" are part of "ready" – all others were not aligned. So in the χ^2-alignment interpretation, $P(\text{ready}|\text{ady@}) = 1.0$. In contrast to RT, we only used aligned ngrams in the sentiment evaluation. This discrepancy may explain why the best method for sentiment is a CHAR method whereas the best method for RT is a WORD method.

First NLP task evaluation on more than 1000 languages. Table 3 presents results for 1664 editions in 1259 languages. To the best of our knowledge, this is the first detailed evaluation, involving two challenging NLP tasks, that has been done on such a large number of languages. For several methods, the results are above baseline for all 1664 editions; e.g., S1 measures are above 20% for all 1664 editions for $N(t)$ on CHAR.

4 Related Work

Following Upadhyay et al. (2016), we group **multilingual embedding** methods into classes A, B, C, D.

Group A trains monolingual embedding spaces and subsequently uses a transformation to create a unified space. Mikolov et al. (2013b) find the transformation by minimizing the Euclidean distance between word pairs. Similarly, Zou et al. (2013), Xiao and Guo (2014) and Faruqui and Dyer (2014) use different data sources for identifying word pairs and creating the transformation (e.g., by CCA). Duong et al. (2017) is also similar. These approaches need large datasets to obtain high quality monolingual embedding spaces and are thus inappropriate for a low-resource setting of 150,000 tokens per language.

Group B starts from the premise that representation of aligned sentences should be similar. Neural network approaches include (Hermann and Blunsom, 2014a) (BiCVM) and (Sarath Chandar et al., 2014) (autoencoders). Again, we have not enough data for training neural networks of this size. Søgaard et al. (2015) learn an interlingual space by using Wikipedia articles as concepts and applying inverted indexing. Levy et al. (2017) show that what we call S-ID is a strongly performing embedding learning method. We use S-ID as a baseline.

Group C combines mono- and multilingual information in the embedding learning objective. Klementiev et al. (2012) add a word-alignment based term in the objective. Luong et al. (2015) extend Mikolov et al. (2013a)'s skipgram model to a bilingual model. Gouws et al. (2015) introduce a crosslingual term in the objective, which does not rely on any word-pair or alignment information. For n editions, including $O(n^2)$ bilingual terms in the objective function does not scale.

Group D creates pseudocorpora by merging data from multiple languages into a single corpus. One such method, due to Vulić and Moens (2015), is our baseline BOW.

Östling (2014) generates **multilingual concepts** using a Chinese Restaurant process, a computationally expensive method. Wang et al. (2016) base their concepts on cliques. We extend their notion of clique from the bilingual to the multilingual case. Ammar et al. (2016) use connected components. Our baseline SAMPLE, based on (Lardilleux and Lepage, 2007, 2009), samples aligned sentences from a multilingual corpus and extracts perfect alignments.

Malaviya et al. (2017), Asgari and Schütze (2017), Östling and Tiedemann (2017) and Tiedemann (2018) perform **evaluation** on the language level (e.g., typology prediction) for 1000+ languages or perform experiments on 1000+ languages without evaluating each language. We present the first work that evaluates on 1000+ languages on the sentence level on a difficult task.

Somers (2005) criticizes RT evaluation on the sentence level; but see Aiken and Park (2010). We demonstrated that when used on the word/unit level, it distinguishes weak from strong embeddings and correlates well with an independent sentiment evaluation.

Any alignment algorithm can be used for **dictionary induction**. We only used a member of the IBM class of models (Dyer et al., 2013), but presumably we could improve results by using either higher performing albeit slower aligners or non-IBM aligners (e.g., (Och and Ney, 2003; Tiedemann, 2003; Melamed, 1997)). Other alignment algorithms include 2D linking (Kobdani et al., 2009), sampling based methods (e.g., Vulic and Moens (2012)) and EFMARAL (Östling and Tiedemann, 2016). EFMARAL is especially intriguing as it is based on IBM1 and Agić et al. (2016) find IBM2-based models to favor closely related languages more than models based on IBM1. However, the challenge is that we need to compute tens of thousands of alignments, so speed is of the essence. We ran character-based and word-based induction separately; combining them is promising future research; cf. (Heyman et al., 2017).

There is much work on embedding learning that does not require **parallel corpora**, e.g., (Vulić and Moens, 2012; Ammar et al., 2016). This work is more generally applicable, but a parallel corpus provides a clearer signal and is more promising (if available) for low-resource research.

5 Summary

We presented a new method for estimating vector space representations of words: embedding learning by concept induction. We tested this method on a highly parallel corpus and learned semantic representations of words in 1259 different languages in a single common space. Our extensive experimental evaluation on crosslingual word similarity and sentiment analysis indicates that concept-based multilingual embedding learning performs better than previous approaches.

The embedding spaces of the 1259 languages (SAMPLE, CLIQUE and $N(t)$) are available: `http://cistern.cis.lmu.de/comult/`.

We gratefully **acknowledge** funding from the European Research Council (grants 740516 & 640550) and through a Zentrum Digitalisierung.Bayern fellowship awarded to the first author. We are indebted to Michael Cysouw for making PBC available to us.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Alonso Martínez, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4.

Milam Aiken and Mina Park. 2010. The efficacy of round-trip translation for MT evaluation. *Translation Journal*, 14(1).

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Ehsaneddin Asgari and Hinrich Schütze. 2017. Past, present, future: A computational investigation of the typology of tense in 1000 languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual training of crosslingual word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Karl Moritz Hermann and Phil Blunsom. 2014a. Multilingual distributed representations without word alignment. In *Proceedings of the 2014 International Conference on Learning Representations*.

Karl Moritz Hermann and Phil Blunsom. 2014b. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2017. Bilingual lexicon induction by learning to combine word-level and character-level representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *Proceedings of the 24th International Conference on Computational Linguistics*.

Hamidreza Kobdani, Alex Fraser, and Hinrich Schütze. 2009. Word alignment by thresholded two-dimensional normalization. In *Proceeedings of the 12th Machine Translation Summit*.

Adrien Lardilleux and Yves Lepage. 2007. The contribution of the notion of hapax legomena to word alignment. In *Proceedings of the 4th Language and Technology Conference*.

Adrien Lardilleux and Yves Lepage. 2009. Sampling-based multilingual alignment. In *Proceedings of 7th Conference on Recent Advances in Natural Language Processing*.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel bible corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*.

Ryan T. McDonald, Slav Petrov, and Keith B. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

I. Dan Melamed. 1997. A word-to-word model of translational equivalence. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Robert Östling. 2014. Bayesian word alignment for massively parallel texts. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with Markov Chain Monte Carlo. *Prague Bulletin of Mathematical Linguistics*, 106.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

AP Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of the 2014 Annual Conference on Neural Information Processing Systems*.

Michel Simard. 1999. Text-translation alignment: Three languages are better than two. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing*.

Harold Somers. 2005. Round-trip translation: What is it good for? In *Proceedings of the Australasian Language Technology Workshop 2005*.

Morris Swadesh. 1946. South Greenlandic (Eskimo). In Cornelius Osgood, editor, *Linguistic Structures of Native America*. Viking Fund Inc. (Johnson Reprint Corp.), New York.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*.

Jörg Tiedemann. 2018. Emerging language spaces learned from massively multilingual corpora. *arXiv preprint arXiv:1802.00273*.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Ivan Vulić and Marie-Francine Moens. 2012. Detecting highly confident word translations from comparable corpora without any prior knowledge. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Ivan Vulic and Marie-Francine Moens. 2012. Subcorpora sampling with an application to bilingual lexicon extraction. In *Proceedings of the 24th International Conference on Computational Linguistics*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2.

Rui Wang, Hai Zhao, Sabine Ploux, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2016. A novel bilingual word embedding method for lexical translation using bilingual sense clique. *arXiv preprint arXiv:1607.08692*.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the 18th Conference on Computational Natural Language Learning*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Isomorphic Transfer of Syntactic Structures in Cross-Lingual NLP

Edoardo Maria Ponti
LTL, University of Cambridge
ep490@cam.ac.uk

Roi Reichart
Technion, IIT
roiri@ie.technion.ac.il

Anna Korhonen
LTL, University of Cambridge
alk23@cam.ac.uk

Ivan Vulić
LTL, University of Cambridge
iv250@cam.ac.uk

Abstract

The transfer or share of knowledge between languages is a popular solution to resource scarcity in NLP. However, the effectiveness of cross-lingual transfer can be challenged by variation in syntactic structures. Frameworks such as Universal Dependencies (UD) are designed to be cross-lingually consistent, but even in carefully designed resources trees representing equivalent sentences may not always overlap. In this paper, we measure cross-lingual syntactic variation, or *anisomorphism*, in the UD treebank collection, considering both morphological and structural properties. We show that reducing the level of anisomorphism yields consistent gains in cross-lingual transfer tasks. We introduce a source language selection procedure that facilitates effective cross-lingual parser transfer, and propose a typologically driven method for syntactic tree processing which reduces anisomorphism. Our results show the effectiveness of this method for both machine translation and cross-lingual sentence similarity, demonstrating the importance of syntactic structure compatibility for boosting cross-lingual transfer in NLP.

1 Introduction

Linguistic information can be transferred from resource-rich to resource-poor languages using approaches such as annotation projection, model transfer, and/or translation (Agić et al., 2014). Such cross-lingual transfer may rely on syntactic information. Structured and more cross-lingually consistent than linear sequences (Ponti, 2016), syntactic information has proved useful for cross-lingual parsing (Tiedemann, 2015; Rasooli and Collins, 2017), multilingual representation learning (Vulić and Korhonen, 2016; Vulić, 2017), causal relation identification (Ponti and Korhonen, 2017), and neural machine translation (Eriguchi et al., 2016; Aharoni and Goldberg, 2017). It can also guide the generation of synthetic data for multilingual tasks (Wang and Eisner, 2016).

Universal Dependencies (UD) (Nivre et al., 2016) is a collection of treebanks for a variety of languages, annotated with a scheme optimised for knowledge transfer. The tag sets are language-independent and there are direct links between content words. This reduces the variation of dependency trees, because content words are cross-lingually more stable than function words (Croft et al., 2017), and benefits semantically-oriented applications (de Marneffe et al., 2014)[1]. Importantly, although UD is tailored to offer support to cross-lingual transfer, it also supports monolingual applications with a quality comparable to language-specific annotations (Vincze et al., 2017, *inter alia*).

Despite the careful design of this resource, there are still substantial variations in morphological richness and strategies employed to express the same syntactic constructions across languages. These variations posit challenges for syntax-based knowledge transfer. The first challenge is how to match the source and target languages so that differences are minimised. The common criteria are based on the typology of word order (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015) or part-of-speech n-grams (Rosa and Zabokrtsky, 2015; Agić, 2017). The second one is how to make knowledge transfer effective by harmonising syntactic trees (Smith and Eisner, 2009; Vilares et al., 2016) as to enable a better correspondence between source and target nodes.

[1] It is controversial whether it improves parsing: e.g., Groß and Osborne (2015, *inter alia*) argue against whereas Attardi et al. (2015, *inter alia*) argue in favour.

In this paper we address these two challenges. We propose the concept of *isomorphism* (i.e., identity of shapes: syntactic structures) and its opposite, *anisomorphism*, as a probe to measuring quantitatively the extent to which syntactic tree pairs are cross-lingually compatible. We assess the variation of syntactic constructions by **a)** the average Zhang and Shasha (1989)'s tree edit distance between UD treebanks, and **b)** the variation in morphology by the Jaccard index of morphological feature sets. We show that these metrics are strong indicators for source language selection, and even preferable over widespread metrics such as genealogical language relatedness.

Moreover, the concept of isomorphism facilitates the process of reshaping trees to make them compatible across languages via operations of deletion, addition, and relabeling. To this end, we propose a tree processing method which increases the level of isomorphism between trees of cross-lingually compatible sentences. This method leads to consistent improvements on cross-lingual tasks achieved through transfer.

To verify the relevance of isomorphism for cross-lingual transfer in NLP, we perform experiments on three tasks. Firstly, we use the Jaccard index of morphological feature sets to choose source languages for cross-lingual dependency parsing. Secondly, we use syntactic trees harmonised by our method in syntax-based neural machine translation of two typologically distant language pairs (Arabic to Dutch; Indonesian to Portuguese). Finally, we evaluate cross-lingual sentence similarity in a real-life resource-lean scenario where the target language has no annotated data. In all experiments, we enhance performance compared to baselines where the source shows a lower degree of isomorphism.

In §2, we define the concept of (an)isomorphism, propose novel metrics for measuring it quantitatively, and introduce the tree processing algorithm. We then desribe the data (§3), methods (§4), and experimental results (§5). Related work is summarised in §6 and conclusions are drawn in §7.

2 Anisomorphism

The ideal situation for knowledge transfer from one (syntactic) structure into another is when these structures are equivalent. In graph theory, there is isomorphism between the nodes V_S of graph S and the nodes V_T of graph T if there exists a bijection $f(V_S) \rightarrow V_T$ such that $\forall s_i, s_j \in V_S$, it holds that: $s_i \bullet\!\!-\!\!\bullet s_j \Leftrightarrow f(s_i) \bullet\!\!-\!\!\bullet f(s_j)$, where the symbol $\bullet\!\!-\!\!\bullet$ stands for adjacency between nodes. In simple words, the mapping must preserve adjacencies between corresponding nodes.

Syntactic trees are a special case of such graphs. However, vocabularies (the words in their nodes) are peculiar to each language, making their comparison impractical across languages. In this work, we probe isomorphism on delexicalised trees, where each node is the (cross-lingually consistent) dependency relation of the word in that position. Even so, however, isomorphic bijection is often impossible between trees of equivalent sentences in different languages owing to typological variation (see §2.1).

Adopting the term from Ambati (2008), we define this property as *an*isomorphism, which can be quantified as the extent to which two structures differ in their morphological and syntactic properties (§2.2). We present a tree processing method to mitigate anisomorphism in §2.3. Afterwards, in §§3-5 we show how the concepts defined in this section facilitate cross-lingual transfer in three NLP tasks.

2.1 Sources of Anisomorphism

Two main causes underpin anisomorphism. The first cause is the *morphological type* of a language: the same grammatical function may be expressed via morphemes, via separate words (so-called function words), or may not be expressed at all (Bybee, 1985, ch. 2). For instance, consider the following Latin-English example:

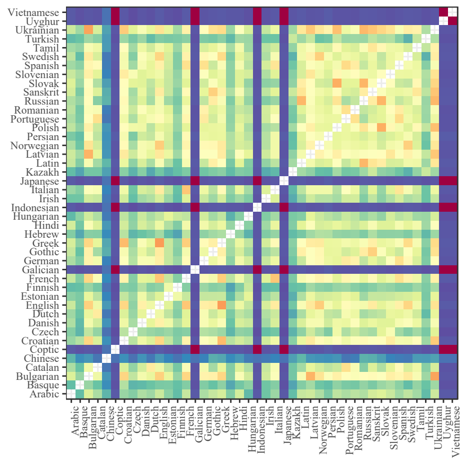
(1)

Crimen	*er-it*	*super-is*	*et*
crime.NOM	be-FUT.3SG	god-DAT.PL	also

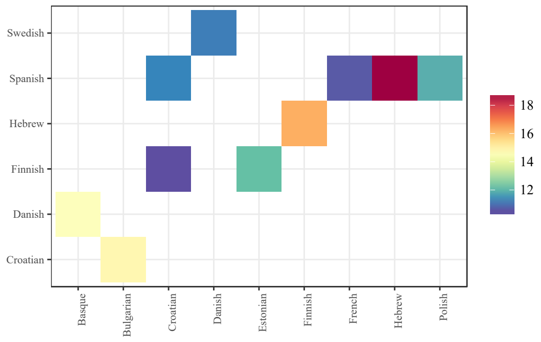
me	*fec-isse*	*nocent-em.*
me	make-INF.PST	guilty-ACC

'It will be a reproach to the gods, that they have made even me guilty.'

The future tense is expressed by inflecting the verb *erit* in Latin, whereas English has the auxiliary verb *will*. In addition, Latin can express the English preposition *to* with the dative case *-is*. This variation has systematic impact on UD annotation. On one hand, Latin would display the attribute-value pairs TENSE=FUTURE and CASE=DATIVE among the features of *erit* and *superis*. On the other hand, in English the function words (*will* and *to*) add nodes to the dependency structure, modifying the equivalent words (*be* and *gods*). This pattern is not unique to English and Latin: there are similar correspondences between specific function words and morphological features in many other languages.



(a) Jaccard index of the morphological feature sets.



(b) Average tree edit distance.

Figure 1: Heatmaps of anisomorphism metrics for UD language pairs. The colours range from blue (low values) to red (high values).

The other source of anisomorphism are *construction strategies*: the same syntactic construction is expressed through different types of strategies (Croft et al., 2017), which results in different kinds of subtrees in UD. An example construction is *predicative possession*, which conveys the ownership of an item by a possessor through the predicate of a clause (Stassen, 2009). Consider these examples in Dutch and Arabic, respectively:

(2)

Ik	*heb*	*een*	*filmidee*
I	have.1SG	a	film+idea

'I have an idea for a movie.'

(3)

Laday-himā	*'ašyā-'u*	*muštarakat-un*
at-them	thing-NOM.PL	common-NOM.PL

'They have things in common.'

In Dutch (Example 2), the owner *Ik* is the subject and the item *filmidee* is the object of a transitive verb (*hab*). However, in Arabic (Example 3) the owner is a predicate with a locative prefix (*laday-himā*), the item *'ašyā'u* is the subject, and there is no verb. These are called transitive and locative strategies, respectively. Each strategy results in a different (delexicalised) subtree, as shown in Figure 2b: this simple example with one construction already suggests that the variation in syntactic constructions affects the compatibility of cross-lingual trees pervasively.[2]

[2]Other strategies for predicative possession include *topic*, *conjunctional* and *genitive*. More examples of constructions are available in the supplemental material.

2.2 Measures of Anisomorphism

How can the differences described in §2.1 translate into quantitative metrics of compatibility between sentences in different languages? As the first answer to this question, we propose to measure the affinity in morphological type by considering the sets of morphological features attested within each of the UD treebanks.[3] Particularly, for each pair of a source language set M_S and a target language set M_T, we estimate their Jaccard index, which is defined for two sets as the cardinality of their intersection divided by the cardinality of their union, as shown in Equation (4).

$$J(M_S, M_T) = \frac{||M_S \cap M_T||}{||M_S \cup M_T||} \quad (4)$$

The values of the Jaccard index lie in $[0, 1]$ A heatmap is displayed in Figure 1a: the morphological similarity between language pairs varies considerably, ranging from low (0.07 in Chinese-Uyghur), mild (0.48 in Latvian-Tamil), to high (0.72 in Bulgarian-Ukrainian). Note that the Jaccard index 1 is an artifact for languages with no expression of grammatical function (in Vietnamese, among others) or lacking morphological annotation (in Japanese). This metric exhibits other disadvantages: it does not take into account another source of variation, the construction strategies, and is based on general properties of a grammar rather

[3]The full list of features can be consulted at http://universaldependencies.org/u/feat/

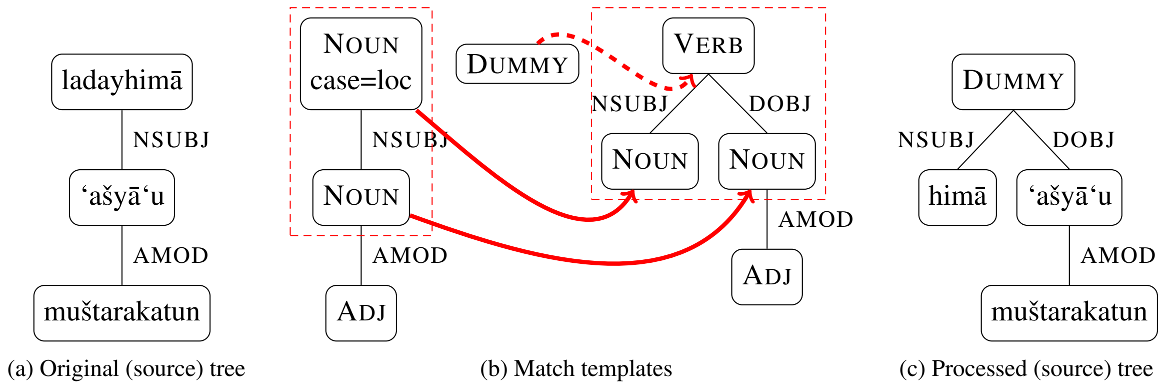


Figure 2: Tree processing steps that transform the locative strategy for predicative possession in an Arabic sentence into a transitive strategy. Tree processing is always applied on source language constructions.

than specific individual sentences.

Hence, we propose an approach to measure anisomorphism between individual sentences. We parse the texts of the multi-parallel Bible corpus (Christodouloupoulos and Steedman, 2015) with the SyntaxNet parser (see §4). The language pairs taken into account are limited to those present both in our UD sample and in the Bible corpus, and sentence-aligned by book, chapter and verse indices. For a given language pair, we estimate the tree edit distance between every corresponding pair of sentence trees S and T with the Zhang-Sasha algorithm (Zhang and Shasha, 1989) and then average over the number of trees.[4]

This tree edit distance operates on ordered trees with node (but not edge) labels, hence it is suited for delexicalised dependencies. In particular, it is defined over a map M, which is a list of node pairs where the former belongs to S or ϵ (empty node), and the latter belongs to T or ϵ. If both are non-empty, they trigger an operation of *relabeling*; if the latter is ϵ, it is *deletion*; if the former is ϵ, it is *addition*. The edit distance is the number of operations required for a complete transformation weighted by the factor γ.[5] The following equation summarises the tree edit distance measure:

$$\gamma(M,S,T) = \sum_{i,j \in M} \gamma(S_i \to T_j) + \gamma(S_i \to \epsilon) + \gamma(\epsilon \to T_j)$$

The possible values of this metric are non-negative real numbers. We opted for this metric in particular because it allows the insertion of internal nodes but not transpositions. The former criterion allows to capture complex transformations without rebuilding entire subtrees, the latter is aimed at taking into account also variations in word order. In order to evaluate pure syntactic isomorphism one should allow for transpositions and/or operate on unordered trees.[6]

A heatmap of tree edit distances is shown in Figure 1b. The values reflect the typological affinity of the language pairs: e.g., Spanish is very close to French (both are Romance languages), mildly similar to Polish (Slavic language, but still part of the Indo-European family), but remote from Hebrew (from a different family, Semitic). The values agree in part with the metrics of Figure 1a, where the Jaccard indices of Hebrew (0.26), Polish (0.46), and French (0.59) mirror the same relationships.

In §4, we show how these metrics can benefit the source selection for knowledge transfer, sometimes even outranking established criteria such as genealogical closeness. However, they have also weaknesses: the Jaccard index of feature sets is not reliable for languages with a limited number of morphologically expressed grammatical categories. On the other hand, the tree edit distance measure requires resources (such as treebanks and parallel corpora) that are not available for many languages.

2.3 Reduction of Anisomorphism

The measures of anisomorphism reveal which languages are structurally similar, which is directly useful for source selection. However, the data available for many tasks are often limited to distant languages. Hence, it is necessary to increase their affinity by gearing one towards the other. We propose to process source dependency trees with an algorithm inspired by the same rules of the tree edit

[4]We implement this algorithm with the *zs* Python package, available at `https://github.com/timtadh/zhang-shasha`.

[5]For simplicity, we set $\gamma = 1$.

[6]For a survey on tree edit distances, see Bille (2005).

distance described in §2.2.

We leverage the readily available documentation in typological databases (e.g., World Atlas of Language Structures: WALS) (Dryer and Haspelmath, 2013).[7] Given a source and a target language, the documentation informs about their respective strategies. For each strategy, we manually define a 'template', i.e. the subtree it corresponds to, in terms of morpho-syntactic features. For instance, see the dashed circles in Figure 2b: note that templates are limited to a head and its immediate dependents.

Then we explore source trees in a top-down breadth-first fashion, and if a template for a source strategy is identified, it is mapped to the corresponding target template. In order to preserve semantic information, contrary to Zhang and Shasha (1989), the mapping operates on lexicalised edge-labeled trees. Hence, ADD and CHANGE affect both words (nodes) and edges (dependency relations). The whole process is summarised in Algorithm 1.

Algorithm 1 Tree processing with rules

```
 1: strategies$_s$ ← WALS$_s$                          ▷ Define templates
 2: strategies$_t$ ← WALS$_t$
 3: function CHANGE($s, t^{(l)}$)                     ▷ Define operations
 4:     $s \leftarrow t^{(l)}$
 5: function DELETE($s$)
 6:     $s \leftarrow \epsilon$
 7: function ADD($t^{(l)}$)
 8:     $\epsilon \leftarrow t^{(l)}$
 9: function MAPPING($r_s$, strategies$_t$)           ▷ Define mapping
10:     assert($r_s \in$ strategies$_s$)
            return {CHANGE, DELETE, ADD}*
11: for subtree in tree$_s$ do                        ▷ Explore tree
12:     if subtree ∈ strategies$_s$ then
13:         list ← MAPPING(subtree)
14:         for $n_s, n_t$ in list do                 ▷ Perform operations
15:             if $n_s \neq \epsilon \wedge n_t \neq \epsilon$ then
16:                 CHANGE($n_s, n_t$)
17:             else if $n_t = \epsilon$ then
18:                 DELETE($n_s$)
19:             else if $n_s = \epsilon$ then
20:                 ADD($n_t$)
```

For instance, consider the transformation from the locative strategy for predicative possession in Arabic from Example 3 into a transitive strategy. By exploring its dependency graph (Figure 2a), the Algorithm identifies a subtree corresponding to one of the source strategies (left side of Figure 2b). This subtree is mapped to the target template (right side of Figure 2b) with the following operations: it CHANGEs the root noun *ladayhimā* (the possessor) with a dummy node (the verb). The same noun is re-ADDed as a dependent with a new label *nsubj*. Finally, the dependency relation of the other noun *'ašyā-'u* is CHANGEd from *nsubj* to *dobj*. The resulting tree uses the source language vocabulary, but target language construction strategies, as shown in Figure 2c.

3 Data

In order to validate the usefulness of anisomorphism reduction through guided source selection and tree processing, we experiment with three different cross-lingual tasks: *cross-lingual dependency parsing*, *neural machine translation (NMT)*, and *cross-lingual sentence similarity (STS)*. In this section, we present the data used in these tasks.

The data for dependency parsing are sourced from Universal Dependencies v1.4.[8] We sample a group of 21 treebanks ensuring their representativeness by balancing them by family. We filter out all languages but two belonging to same branches of the Indo-European family, and keep those of all the other families.[9] We take into account only the language-independent components of the annotation: coarse POS tags, morphological features, and dependency relations.

Regarding NMT data, English is ubiquitous in the current datasets, overshadowing the wide spectrum of existing morphological types and syntactic strategies. To address this limitation, we create a *new NMT dataset* that matches typologically distant languages directly without the need of a bridge/pivot language. We extract aligned sentences from the Open Subtitles 2016 tokenised corpus (Tiedemann, 2009)[10] for Arabic-Dutch and Indonesian-Portuguese. This choice was made based on their volume of parallel data in order to produce evaluation data similar in size to those of NMT datasets in shared tasks such as WMT16 (Bojar et al., 2016). Training and test sets consist of 3M and 5K sentences, respectively. These sentences come automatically annotated by SyntaxNet.

The data for cross-lingual STS are chosen to resemble a real-world scenario with a resource-poor target language. The training data (9,709 sentence

[7] In particular, we take into account the following relevant WALS features: 116 (polar questions), 122-123 (relativisation on subjects and obliques), 117 (predicative possession), 113-115 (negation), 107 (passive), 37-38 (articles), and 85 (prepositions).

[8] http://universaldependencies.org/

[9] Language names are substituted in this work by their corresponding ISO 639-1 codes. A table of names and codes is provided in the supplemental material.

[10] http://opus.nlpl.eu/OpenSubtitles.php

pairs) are in English, taken from the STS benchmark, the ensemble of all the datasets from SemEval 2012-2017 STS tasks. The test data (250 sentence pairs) come from Task 1 of SemEval 2017 (Cer et al., 2017); target language is Arabic.[11] All the sentence pairs are associated with a label ranging from 0 (dissimilarity) to 5 (equivalence).

4 Methodology

Cross-lingual Dependency Parsing. To assess if the anisomorphism metrics devised in §2.2 are reliable in finding compatible languages for knowledge transfer, we use the Jaccard index of the morphological feature sets as a criterion to choose source languages for cross-lingual parser transfer. We adopt the variant of delexicalised model transfer (Zeman and Resnik, 2008) for this task. This technique ignores lexicalised features and leverages only language-independent features instead.

For each language from a sample of 7 (typologically diverse) targets, we report LAS scores using three different source languages: (1) the highest-ranked source according to the Jaccard index; (2) a source sampled from the middle of the list ranked by the Jaccard indices; (3) a very dissimilar language sampled from the bottom of the ranked list. The total number of sentences used for training corresponds to the smallest of the three source language treebanks in order to isolate the effect of treebank size on the final transfer results.

We conduct experiments with two well-known transition-based parsers (Nivre, 2006): (1) *DeSR* (Attardi et al., 2007) and (2) *SyntaxNet* (Andor et al., 2016; Alberti et al., 2017). The two were selected as they represent two different architectures: the former is an SVM-based model with a polynomial kernel, whereas the latter is a feed-forward neural network with beam search based on conditional random fields. The results are evaluated in terms of LAS and UAS scores.

Neural Machine Translation. For NMT, we examine whether the tree processing procedure from §2.3 can reduce anisomorphism between source and target language syntactic structures. We thus run NMT models in two settings: with and without the anisomorphism reduction procedure.

For this experiment we rely on a state-of-the-art syntax-aware NMT architecture. We report its performance by BLEU scores (Papineni et al., 2002).

[11] http://alt.qcri.org/semeval2017/task1/

In particular, we use an attentional encoder-decoder network that jointly learns to translate and align words (Bahdanau et al., 2015) implemented in the Nematus suite[12] (Sennrich et al., 2017). The encoder is a bidirectional gated recurrent network. For each step *i*, the decoder predicts the next word in output by taking as input the current hidden state h_i, the previous word w_{i-1} and a context vector, i.e., a weighted sum of all the hidden states $\sum_{j=1}^{n} w_j \cdot h_1$. The weights are learned by a multi-layer perceptron that estimates the likelihood of the alignment between the predicted word and each of the input words: $w_{i,j} = P(a|y_i, x_j)$.

This model is enriched with additional linguistic features on input, as proposed by Sennrich and Haddow (2016). In particular, we select the following which are proven as useful in prior work, and also relevant to our experiment: word form, POS tag, and dependency relations. These features are concatenated and fed to the encoder. Tree processing from §2.3 affects these features (and consequently the sentence representation) by changing the initial tree structure. For instance, the original tree in Figure 2a and the processed one in Figure 2c would correspond to these feature sets:

Original	**Preprocessed**
ladayhimā ⊕ N ⊕ ROOT	*himā* ⊕ N ⊕ NSUBJ
	DUMMY ⊕ V ⊕ ROOT
ʻašyāʻu ⊕ N ⊕ NSUBJ	*ʻašyāʻu* ⊕ N ⊕ DOBJ
muštarakatun ⊕ A ⊕ Amod	*muštarakatun* ⊕ A ⊕ Amod

Cross-lingual STS. We use cross-lingual STS as another evaluation task to validate if the anisomorphism reduction algorithm from §2.3 generalises beyond the initial application in NMT. The state-of-art approach to this task in the monolingual setting encodes trees of sentence pairs with a TreeLSTM architecture (Tai et al., 2015). The hidden representations of the tree roots of both sentences in each pair are then concatenated and fed to a multi-layer perceptron classifier, which yields a probability distribution over the six classes (from 0=dissimilarity to 5=equivalence).

The following TreeLSTM has been implemented in PyTorch. The parameters of an LSTM model are the matrix weights W_q for inputs and U_q for hidden representations, and a bias b_q. q corresponds to an input gate i_t, a forget gate f_t, an output gate o_t, or a memory cell c_t at time step *t*. The hidden state h_t

[12] https://github.com/EdinburghNLP/nematus

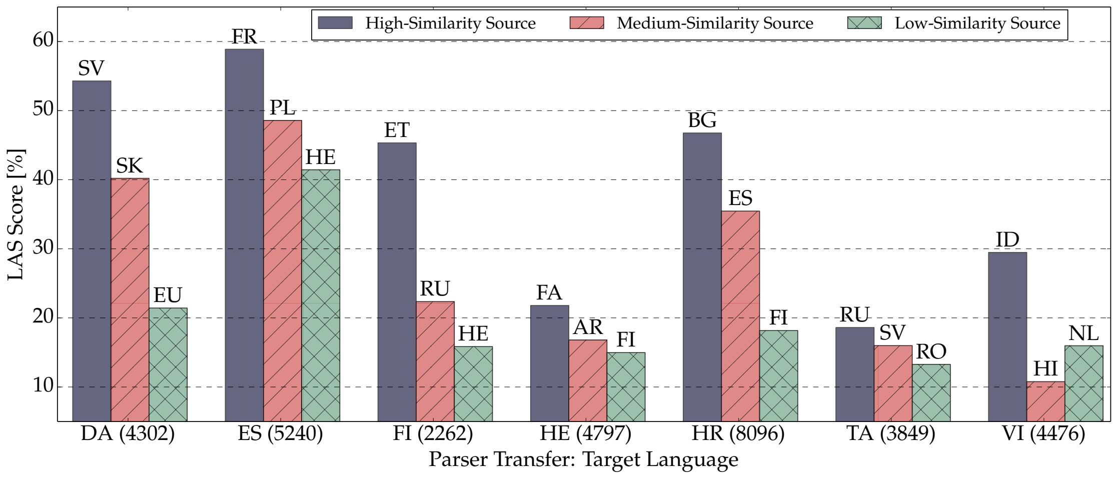


Figure 3: Results of delexicalised cross-lingual transfer using DeSR. Results with SyntaxNet are omitted as they show very similar patterns. The numbers in parentheses denote the amount of training sentences.

is derived from the equations below. To extend this model to dependency trees, we consider h_{t-1} to equal the sum of the hidden states of the children of a node $\sum_{k \in C(x_t)} h_k$, and provide a different forget gate f_{tk} for each child.

$$q_t = \sigma\left(W_q x_t + U_q h_{t-1} + b_q\right) \quad (5)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \quad (6)$$

$$h_t = o_t \odot \tanh(c_t) \quad (7)$$

In our resource-lean cross-lingual scenario the language of the training data (English) differs from that of the target (Arabic). Since TreeLSTM is a lexicalised model, we employ multilingual word embeddings, such that the words of both languages lie in the shared cross-lingual semantic space. In particular, we map English into Arabic through the iterative Procustes method devised by Artetxe et al. (2017). The results are evaluated through the Pearson correlation and the Mean Squared Error (MSE) between predicted and golden labels.

Hyperparameters. DeSR has degree 2, γ 0.18, C 0.4, $coef_0$ 0.4, and ϵ 1.0. The hyper-parameters for the deep models are shown in Table 1: we have followed the training setup suggestions from prior work for all the models used in our experiments.

	SyntaxNet (Parsing)	Nematus (NMT)	TreeLSTM (STS)
Hidden layers	2	2	1
Hidden size	512	1000	300
Input size	160	280	512
Batch size	256	80	25
Epochs	12 (greed); 10 (beam)	Early stopping	5
Learning rate	0.8	1^{-4}	1^{-2}
Optimiser	Adam	AdaDelta	SGD
Dropout	0.2 / 0.3	0.1 / 0.2	0

Table 1: Hyper-parameters of the models.

5 Results and Discussion

Source Selection. The results for cross-lingual parser transfer with the DeSR parser are provided in Figure 3, while the results with SyntaxNet are provided as supplemental material as they follow the same trends. The selection of the source for delexicalised cross-lingual parsing based on the proposed Jaccard index measure shows than selecting a source language with a lower degree of anisomorphism is crucial for knowledge transfer. The values for the selected languages are listed in Table 2.

Target	**High**	**Mid**	**Low**
Danish	0.49	0.39	0.19
Spanish	0.59	0.46	0.26
Finnish	0.44	0.23	0.15
Hebrew	0.31	0.24	0.15
Croatian	0.62	0.46	0.25
Tamil	0.48	0.43	0.38
Vietnamese	1.00	0.02	0.01

Table 2: Jaccard indices of source-target pairs.

The high-similarity source always outperforms the alternatives with both DeSR and SyntaxNet, and with respect to both LAS and UAS scores. For instance, Swedish is the best source for Danish, Estonian for Finnish, and Bulgarian for Croatian. Similarly, the preference for medium- over low-

	AR-NL	ID-PT
Baseline	7.01	14.79
+Syntax	14.40	23.70
++Preprocessing	**15.40**	**24.12**

Table 3: NMT results: BLEU scores of a joint translator and aligner *(Baseline)*, fed with linguistic features *(+Syntax)*, and with processed trees to reduce anisomorphism *(++Preprocessing)*.

	Pearson	MSE
Mono-lingual	77.9	0.94
Cross-lingual	44.7	1.82
+Preprocessing	**48.0**	**1.64**

Table 4: Cross-lingual STS results: Pearson and MSE scores of the TreeLSTM architecture with original and processed trees.

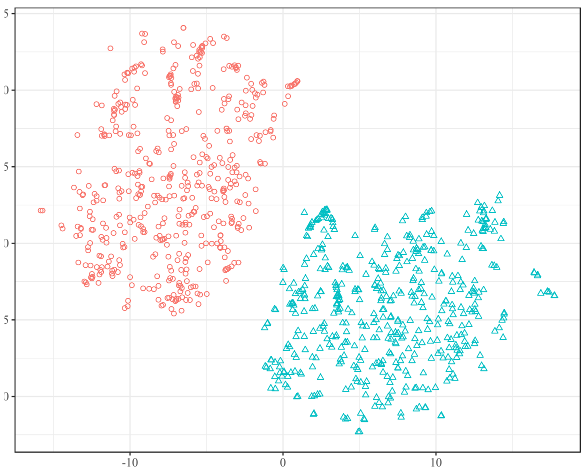


Figure 4: Hidden representations of original (red circles) and processed (blue triangles) sentences.

ranking languages is pronounced, too, as it holds for 6 groups out of 7. For instance, Slovak is a better source choice for Danish than Basque, Polish is a better source choice for Spanish than Hebrew.

Most notably, our findings generalise even to cases when the top-ranking language (e.g. Farsi) does not belong to the language family of the target (e.g. Hebrew) whereas the language with a medium overlap does (e.g. Arabic).

Tree Processing. The results of the experiments also corroborate the idea that tree harmonisation informed by linguistic typology, and implemented through our anisomorphism reduction procedure can assist model transfer in cross-lingual tasks. The BLEU scores for Neural Machine Translation, shown in Table 3, reveal consistent improvements. The model enriched with syntactic features outperforms the baseline with joint translation and alignment without syntactic features by 7.39 BLEU points in Arabic-Dutch and 8.91 BLEU points in Indonesian-Portuguese. Importantly, our extension which reduces anisomorphism by processing syntactic trees in the source language leads to further improvements for both language pairs: it surpasses the model with syntactic features by 1.0 BLEU points in Arabic-Dutch, and 0.42 BLEU points in Indonesian-Portuguese.

These results support our hypotheses: **a)** syntax is pivotal in NMT, confirming findings from prior work (Sennrich et al., 2017); **b)** the tree processing algorithm from §2.3 facilitates the alignment between source and target words, and also grants the encoder-decoder architecture a better leverage of dependency features. This lends support to our argument that anisomorphism limits the ability of models to generalise beyond single languages, and reducing it can help cross-lingual syntax-aware NLP tasks.

A similar conclusion can be reached by comparing the performance of TreeLSTM-based models on the cross-lingual STS task, reported in Table 4. In particular, the Pearson correlation score increases by 3.3 points and MSE decreases by 0.18 points when our tree processing algorithm is applied. We inspect the hidden representations of both original and processed sentences with t-SNE dimensionality reduction in Figure 4. The impact of the algorithm becomes evident as their clusters are completely separate. However, the comparison against the monolingual STS score obtained on the English test set shows that there is still a wide gap to be bridged by cross-lingual knowledge transfer.

Note that our tree processing algorithm is guided by typological knowledge in WALS. The results of the NMT and cross-lingual STS tasks suggest that existing knowledge in such large typological databases (O'Horan et al., 2016; Bender, 2016) can be readily used to support cross-lingual transfer tasks in NLP, as well as the interpretation of polyglot neural models (Ponti et al., 2017). We hope that our work will spark further research on the use of typology in cross-lingual NLP applications.

6 Related Work

The need to account for discrepancies in tree structures emerged early in the domain of Information Theory: in particular, the tree edit distance turned out to be useful for correcting programming scripts (Tai, 1979), evolution studies, and most notably accounting for transformations in constituency trees (Selkow, 1977). Although previous works were aware of the problem of anisomorphism in the context of syntax-based NLP applications (Ambati, 2008), to our knowledge we are the first to quantify it formally and to leverage it in cross-lingual NLP.

For source selection, similarity metrics from prior work mostly relied on information stored in typological databases (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Deri and Knight, 2016). Otherwise, the metrics were derived empirically: they mostly concerned linear-order properties such as part-of-speech n-grams (Rosa and Zabokrtsky, 2015; Agić, 2017). In domain adaptation, the selection also hinges upon topic models (Plank and Van Noord, 2011) or Bayesian Optimisation (Ruder and Plank, 2017). The metrics we defined in §2.2 are instead based on configurational properties of languages, and add another piece to the puzzle of source selection.

The idea of tree processing dates back to the attempts to steer source towards target syntactic structures in statistical MT, although they were mostly limited to simple reordering steps.

Gildea (2003) proposed cloning operations to relocate subtrees. Other works learned rewrite patterns in an automatic fashion to minimize differences in the order of chunks (Zhang et al., 2007) or labeled dependencies (Habash, 2007). Instead, Smith and Eisner (2009) proposed to learn jointly a translation and a loose alignment of nodes, in order to avoid enforcing the bias of the source structure. Reviving these approaches within the framework of deep learning seems crucial as far as state-of-art models depend on syntactic information (Eriguchi et al., 2016; Dyer et al., 2016).

In general, our approach aims at developing and evaluating models focused on specific constructions rather than languages as a whole (Rimell et al., 2009; Bender, 2011; Rimell et al., 2016). The gist is that current models have reached a plateau in performance because they excel with frequent and simple phenomena, but they still lag behind with respect to rarer or more complex constructions.

7 Conclusions and Future Work

We have demonstrated that syntactic structures differ across languages even in well-developed annotation schemes such as Universal Dependencies. This variation stems from morphological and syntactic differences across languages. This phenomenon, which we have labeled as anismorphism, can challenge the transfer of knowledge from one language to another. We have proposed novel methodology which reduces the degree of anisomorphism cross-lingually 1) by selecting the most compatible languages for transfer, and 2) by editing the syntactic structures (i.e., trees) themselves.

First, we have provided two measures of anisomorphism based on Jaccard distance of morphological feature sets, as well as average tree edit distance of parallel sentences. These can provide reliable indicators for language compatibility for source selection in cross-lingual parsing.

Second, we have proposed a new method for fine-tuning source dependency trees to resemble target language trees in order to reduce anisomorphism. The method does not depend on parallel data, and it leverages readily available information in typological databases. It boosts the performance of standard frameworks in two downstream applications, obtaining competitive or state-of-art results for 1) NMT on a new dataset of Arabic-Dutch and Indonesian-Portuguese and 2) cross-lingual sentence similarity.

Future work will look into automating the tree processing procedure. A parametrised model could be trained to imitate the operations performed by Zhang and Shasha (1989)'s algorithm on multi-parallel texts, conditioned on the tree features and previous operations. Another possible research direction is learning the mapping between structures from parallel texts jointly with a main task, in the spirit of quasi-synchronous grammars (Smith and Eisner, 2009). Finally, a wider range of syntactic constructions could be covered by inferring typological strategies from texts (Östling, 2015; Coke et al., 2016).

The data for NMT, and the code for our cross-lingual STS are available at the following link: `github.com/ducdauge/isotransf`.

Acknowledgements

This work is supported by the ERC Consolidator Grant LEXICAL (no 648909). The authors would like to thank the anonymous reviewers.

References

Željko Agić. 2017. Cross-lingual parser selection for low-resource languages. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 1–10.

Željko Agić, Jörg Tiedemann, Kaja Dobrovoljc, Simon Krek, Danijela Merkler, and Sara Može. 2014. Cross-lingual dependency parsing of related languages with rich morphosyntactic tagsets. In *Proceedings of the EMNLP 2014 Workshop on Language Technology for Closely Related Languages and Language Variants*, pages 13–24.

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of ACL*, pages 132–140.

Chris Alberti, Daniel Andor, Ivan Bogatyy, Michael Collins, Dan Gillick, Lingpeng Kong, Terry Koo, Ji Ma, Mark Omernick, Slav Petrov, et al. 2017. SyntaxNet models for the CoNLL 2017 shared task. *arXiv preprint arXiv:1703.04929*.

Vamshi Ambati. 2008. Dependency structure trees in syntax based machine translation. In *Adv. MT Seminar Course Report*, volume 137.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*, pages 2442—2452.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*, pages 451–462.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of EMNLP-CoNLL*, pages 1112–1118.

Giuseppe Attardi, Simone Saletti, and Maria Simi. 2015. Evolution of Italian treebank and dependency parsing towards Universal Dependencies. In *Proceedings of the Second Italian Conference in Computational Linguistics (CLiC-it)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR (Conference Papers)*.

Emily M Bender. 2011. On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology*, 6(3):1–26.

Emily M. Bender. 2016. Linguistic typology in natural language processing. *Linguistic Typology*, 20(3).

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of WMT*, volume 2, pages 131–198.

Joan L Bybee. 1985. *Morphology: A study of the relation between meaning and form*, volume 9. John Benjamins Publishing.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of SEMEVAL*, pages 1–14.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: The Bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.

Reed Coke, Ben King, and Dragomir R. Radev. 2016. Classifying syntactic regularities for hundreds of languages. *CoRR*, abs/1603.08016.

William Croft, Dawn Nordquist, Katherine Looney, and Michael Regan. 2017. Linguistic typology meets Universal Dependencies. In *Proceedings of the 15th International Workshop on Treebanks and Linguistic Theories (TLT15)*, pages 63–75.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *Proceedings of ACL*, pages 399–408.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 199–209.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of ACL*, pages 823—-833.

Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of ACL*, pages 80–87.

Thomas Groß and Timothy Osborne. 2015. The dependency status of function words: Auxiliaries. In *Proceedings of the International Conference on Dependency Linguistics (DepLing)*, pages 111–120.

Nizar Habash. 2007. Syntactic preprocessing for statistical machine translation. *Proceedings of MT SUMMIT*.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, pages 4585–4592.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of ACL*, pages 629–637.

Joakim Nivre. 2006. *Inductive dependency parsing*. Springer.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*, pages 1659–1666.

Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, and Anna Korhonen. 2016. Survey on the use of typological information in natural language processing. In *Proceedings of COLING*, pages 1297–1308.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *Proceedings of ACL*, pages 205–211.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Barbara Plank and Gertjan Van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of ACL*, pages 1566–1576.

Edoardo Maria Ponti. 2016. Divergence from syntax to linear order in Ancient Greek lexical networks. In *Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 541–547.

Edoardo Maria Ponti and Anna Korhonen. 2017. Event-related features in feedforward neural networks contribute to identifying causal relations in discourse. In *Proceedings of the EACL 2017 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 25–30.

Edoardo Maria Ponti, Ivan Vulić, and Anna Korhonen. 2017. Decoding sentiment from distributed representations of sentences. In *Proceedings of *SEM*, pages 22–32.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the ACL*, 5:279–293.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of EMNLP*, pages 813–821.

Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. 2016. RELPRON: A relative clause evaluation data set for compositional distributional semantics. *Computational Linguistics*, 42(4):661–701.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. KLcpos3 - a language similarity measure for delexicalized parser transfer. In *Proceedings of ACL*, pages 243–249.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *Proceedings of EMNLP*, pages 372–382.

Stanley M. Selkow. 1977. The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: A toolkit for neural machine translation. In *Proceedings of EACL*, pages 65–68.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of WMT*, pages 83–91.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*, pages 822–831.

Leon Stassen. 2009. *Predicative possession*. Oxford University Press.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL-HLT*, pages 1061–1071.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

Kuo-Chung Tai. 1979. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Proceedings of RANLP*, pages 237–248.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted POS labels. In *Proceedings of the International Conference on Dependency Linguistics (DepLing)*, pages 340–349.

David Vilares, Miguel A. Alonso, and Carlos Gómez-Rodríguez. 2016. One model, two languages: Training bilingual parsers with harmonized treebanks. In *Proceedings of ACL*, pages 425–431.

Veronika Vincze, Katalin Ilona Simkó, Zsolt Szántó, and Richárd Farkas. 2017. Universal Dependencies and morphology for Hungarian–and on the price of universality. In *Proceedings of EACL*, pages 356–365.

Ivan Vulić. 2017. Cross-lingual syntactically informed distributed word representations. In *Proceedings of EACL*, pages 408–414.

Ivan Vulić and Anna Korhonen. 2016. Is "universal syntax" universally useful for learning distributed word representations? In *Proceedings of ACL*, pages 518–524.

Dingquan Wang and Jason Eisner. 2016. The Galactic Dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the ACL*, 4:491–505.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of IJCNLP*, pages 35–42.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of EMNLP*, pages 1857–1867.

Yuqi Zhang, Richard Zens, and Hermann Ney. 2007. Chunk-level reordering of source language sentences with automatically learned rules for statistical machine translation. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 1–8.

Language Modeling for Code-Mixing: The Role of Linguistic Theory based Synthetic Data

Adithya Pratapa[1] **Gayatri Bhat**[2*] **Monojit Choudhury**[1] **Sunayana Sitaram**[1]
Sandipan Dandapat[3] **Kalika Bali**[1]
[1] Microsoft Research, Bangalore, India
[2] Language Technology Institute, Carnegie Mellon University
[3] Microsoft R&D, Hyderabad, India
[1]{t-pradi, monojitc, sunayana.sitaram, kalikab}@microsoft.com,
[2]gbhat@andrew.cmu.edu, [3]sadandap@microsoft.com

Abstract

Training language models for Code-mixed (CM) language is known to be a difficult problem because of lack of data compounded by the increased confusability due to the presence of more than one language. We present a computational technique for creation of grammatically valid artificial CM data based on the Equivalence Constraint Theory. We show that when training examples are sampled appropriately from this synthetic data and presented in certain order (aka training curriculum) along with monolingual and real CM data, it can significantly reduce the perplexity of an RNN-based language model. We also show that randomly generated CM data does not help in decreasing the perplexity of the LMs.

1 Introduction

Code-switching or *code-mixing* (CM) refers to the juxtaposition of linguistic units from two or more languages in a single conversation or sometimes even a single utterance.[1] It is quite commonly observed in speech conversations of multilingual societies across the world. Although, traditionally, CM has been associated with informal or casual speech, there is evidence that in several societies, such as urban India and Mexico, CM has become the default code of communication (Parshad et al., 2016), and it has also pervaded written text, especially in computer-mediated communication and social media (Rijhwani et al., 2017).

It is, therefore, imperative to build NLP technology for CM text and speech. There have been some efforts towards building of Automatic Speech Recognition Systems and TTS for CM speech (Li and Fung, 2013, 2014; Gebhardt, 2011; Sitaram et al., 2016), and tasks like language identification (Solorio et al., 2014; Barman et al., 2014), POS tagging (Vyas et al., 2014; Solorio and Liu, 2008), parsing and sentiment analysis (Sharma et al., 2016; Prabhu et al., 2016; Rudra et al., 2016) for CM text. Nevertheless, the accuracies of all these systems are much lower than their monolingual counterparts, primarily due to lack of enough data.

Intuitively, since CM happens between two (or more languages), one would typically need twice as much, if not more, data to train a CM system. Furthermore, any CM corpus will contain large chunks of monolingual fragments, and relatively far fewer code-switching points, which are extremely important to learn patterns of CM from data. This implies that the amount of data required would not just be twice, but probably 10 or 100 times more than that for training a monolingual system with similar accuracy. On the other hand, apart from user-generated content on the Web and social media, it is extremely difficult to gather large volumes of CM data because (a) CM is rare in formal text, and (b) speech data is hard to gather and even harder to transcribe.

In order to circumvent the data scarcity issue, in this paper we propose the use of linguistically-motivated synthetically generated CM data (as a supplement to real CM data) for development of CM NLP systems. In particular, we use the Equivalence Constraint Theory (Poplack, 1980; Sankoff, 1998) for generating linguistically valid CM sentences from a pair of parallel sentences in the two languages. We then use these generated sentences, along with monolingual and little

*Work done during author's internship at Microsoft Research

[1]According to some linguists, code-switching refers to inter-sentential mixing of languages, whereas code-mixing refers to intra-sentential mixing. Since the latter is more general, we will use code-mixing in this paper to mean both.

amount of real CM data to train a CM *Language Model* (LM). Our experiments show that, when trained following certain sampling strategies and training curriculum, the synthetic CM sentences are indeed able to improve the perplexity of the trained LM over a baseline model that uses only monolingual and real CM data.

LM is useful for a variety of downstream NLP tasks such as Speech Recognition and Machine Translation. By definition, it is a discriminator between natural and unnatural language data. The fact that linguistically constrained synthetic data can be used to develop better LM for CM text is, on one hand an indirect statistical and task-based validation of the linguistic theory used to generate the data, and on the other hand an indication that the approach in general is promising and can help solve the issue of data scarcity for a variety of NLP tasks for CM text and speech.

2 Generating Synthetic Code-mixed Data

There is a large and growing body of linguistic research regarding the occurrence, syntactic structure and pragmatic functions of code-mixing in multilingual communities across the world. This includes many attempts to explain the grammatical constraints on CM, with three of the most widely-accepted being the *Embedded-Matrix* (Joshi, 1985; Myers-Scotton, 1993, 1995), the *Equivalence Constraint* (EC) (Poplack, 1980; Sankoff, 1998) and the *Functional Head Constraint* (DiSciullo et al., 1986; Belazi et al., 1994) theories.

For our experiments, we generate CM sentences as per the EC theory, since it explains a range of interesting CM patterns beyond lexical substitution and is also suitable for computational modeling. Further, in a brief human-evaluation we conducted, we found that it is representative of real CM usage. In this section, we list the assumptions made by the EC theory, briefly explain the theory, and then describe how we generate CM sentences as per this theory.

2.1 Assumptions of the EC Theory

Consider two languages L_1 and L_2 that are being mixed. The EC Theory assumes that both languages are defined by context-free grammars G_1 and G_2. It also assumes that every non-terminal category X_1 in G_1 has a *corresponding* non-terminal category X_2 in G_2 and that every terminal symbol (or word) w_1 in G_1 has a *corresponding* terminal symbol w_2 in G_2. Finally, it assumes that every production rule in L_1 has a *corresponding* rule in L_2 - i.e, the non-terminal categories on the left-hand side of the two rules correspond to each other, and every category/symbol on the right-hand side of one rule corresponds to a category/symbol on the right-hand side of the other rule.

All these correspondences must also hold vice-versa (between languages L_2 and L_1), which implies that the two grammars can only differ in the ordering of categories/symbols on the right-hand side of any production rule. As a result, any sentence in L_1 has a corresponding translation in L_2, with their parse trees being equivalent except for the ordering of sibling nodes. Fig.1(a) and (b) illustrate one such sentence pair in English and Spanish and their parse-trees. The EC Theory describes a CM sentence as a constrained combination of two such equivalent sentences.

While the assumptions listed above are quite strong, they do not prevent the EC Theory from being applied to two natural languages whose grammars do not correspond as described above. We apply a simple but effective strategy to reconcile the structures of a sentence and its translation - if any corresponding subtrees of the two parse trees do not have equivalent structures, we collapse each of these subtrees to a single node. Accounting for the actual asymmetry between a pair of languages will certainly allow for the generation of more CM variants of any L_1-L_2 sentence pair. However, in our experiments, this strategy retains most of the structural information in the parse trees, and allows for the generation of up to thousands of CM variants of a single sentence pair.

2.2 The Equivalence Constraint Theory

Sentence production. Given two monolingual sentences (such as those introduced in Fig.1), a CM sentence is created by traversing all the leaf nodes in the parse tree of either of the two sentences. At each node, either the word at that node or at the corresponding node in the other sentence's parse is generated. While the traversal may start at any leaf node, once the production enters one constituent, it will exhaust all the lexical slots (leaf nodes) in that constituent or its equivalent constituent in the other language before entering into a higher level constituent or a sister

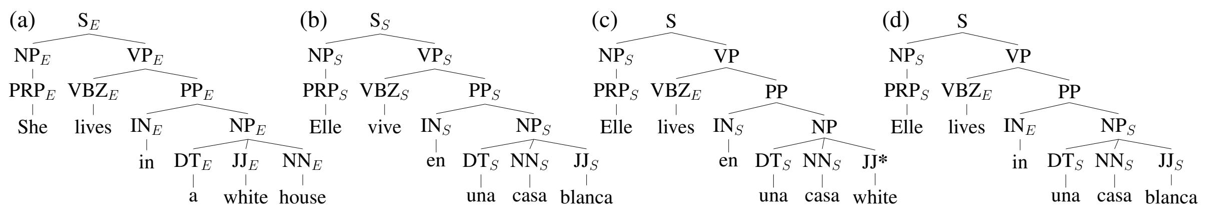


Figure 1: Parse trees of a pair of equivalent (a) English and (b) Spanish sentences, with corresponding hierarchical structure (due to production rules), internal nodes (non-terminal categories) and leaf nodes (terminal symbols), and parse trees of (c) incorrectly code-mixed and (d) correctly code-mixed variants of these sentences (as per the EC theory).

constituent. (Sankoff, 1998) This guarantees that the parse tree of a sentence so produced will have the same hierarchical structure as the two monolingual parse trees (Fig. 1(c) and (d)).

The EC theory also requires that any monolingual fragment that occurs in the CM sentence must occur in one of the monolingual sentences (in the running example, the fragment `una blanca` would be disallowed since it does not appear in the Spanish sentence).

Switch-point identification. To ensure that the CM sentence does not at any point deviate from *both* monolingual grammars, the EC theory imposes certain constraints on its parse tree. To this end and in order to identify the code-switching points in a generated sentence, nodes in its parse tree are assigned language labels according to the following rules: All leaf nodes are labeled by the languages of their symbols. If all the children of any internal node share a common label, the internal node is also labeled with that language. Any node that is out of rank-order among its siblings according to one language is labeled with the other language. (See labeling in Fig.1(c) and (d)) If any node acquires labels of both languages during this process (such as the node marked with an asterisk in Fig.1(c)), the sentence is disallowed as per the EC theory. In the labeled tree, any pair of adjacent sibling nodes with contrasting labels are said to be at a *switch-point* (SP).

Equivalence constraint. Every switch-point identified in the generated sentence must abide by the EC. Let $U \rightarrow U_1U_2...U_n$ and $V \rightarrow V_1V_2...V_n$ be corresponding rules applied in the two monolingual parse trees, and nodes U_i and V_{i+1} be adjacent in the CM parse tree. This pair of nodes is a switch-point, and it only abides by the EC if every node in $U_1...U_i$ has a corresponding node in $V_1...V_i$. This is true for the switch-point in Fig.1(d), and indicates that the two grammars are 'equivalent' at the code-switch point. More importantly, it shows that switching languages at this point does not require another switch later in the sentence. If every switch-point in the generated sentence abides by the EC, the generated sentence is allowed by the EC theory.

2.3 System Description

We assume that the input to the generation model is a pair of parallel sentences in L_1 and L_2, along with word level alignments. For our experiments, L_1 and L_2 are English and Spanish, and Sec 3.2 describes how we create the input set. We use the Stanford Parser (Klein and Manning, 2003) to parse the English sentence.

Projecting parses. We use the alignments to project the English parse tree onto the Spanish sentence in two steps: (1) We first replace every word in the English parse tree with its Spanish equivalent (2) We re-order the child nodes of each internal node in the tree such that their left-to-right order is as in the Spanish sentence. For instance, after replacing every English word in Fig.1(a) with its corresponding Spanish word, we interchange the positions of *casa* and *blanca* to arrive Fig.1(b). For a pair of parallel sentences that follow all the assumptions of the EC theory, these steps can be performed without exception and result in the creation of a Spanish parse tree with the same hierarchical structure as the English parse.

We use various techniques to address cases in which the grammatical structures of the two sentences deviate. English words that are unaligned to any Spanish words are replaced by empty strings. (See Fig.2 wherein the English word *she* has no Spanish counterpart, since this pronoun is dropped in the Spanish sentence.) Contiguous word sequences in one sentence that are aligned to the

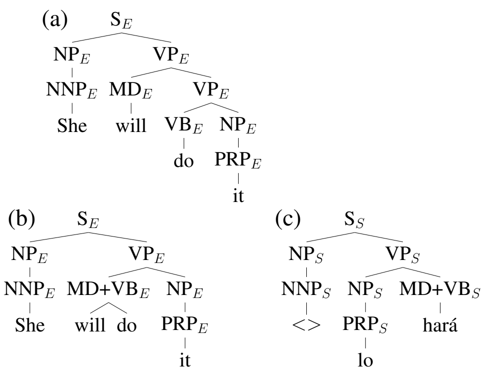


Figure 2: (a) The parse of an English sentence as per Stanford CoreNLP. This parse is projected onto the parallel Spanish sentence *Lo hará* and modified during this process, to produce corresponding (b) English and (c) Spanish parse trees.

same word(s) in the other language are collapsed into a single multi-word node, and the entire subtree between these collapsed nodes and their closest common ancestor is flattened to accommodate this change (example in Fig.2). While these changes do result in slightly unnatural or simplified parse trees, they are used very sparingly since English and Spanish have very compatible grammars.

Generating CS sentences. The number of CS sentences that can be produced by combining a corresponding pair of English and Spanish sentences increases exponentially with the length of the sentences. Instead of generating these sentences exhaustively, we use the parses to construct a finite-state automaton that succinctly captures the acceptable CS sentences. Since the CS sentence must have the same hierarchical structure as the monolingual sentences, we construct the automaton during a post-order traversal of the monolingual parses. An automaton is constructed at each node by (1) concatenating the automatons constructed at its child nodes, (2) splitting states and removing transitions to ensure that the EC theory is not violated. The last automaton to be constructed, which is associated with the root node, accepts all the CS sentences that can be generated using the monolingual parses. We do not provide the exact details of automaton construction here, but we plan to release our code in the near future.

3 Datasets

In this work, we use three types of language data: monolingual data in English and Spanish (*Mono*), real code-mixed data (*rCM*), and artificial or generated code-mixed data (*gCM*). In this section, we describe these datasets and their CM properties. We begin with description of some metrics that we shall use for quantification of the complexity of a CM dataset.

3.1 Measuring CM Complexity

The CM data, both real and artificial, can vary in the their relative usage and ordering of L1 and L2 words, and thereby, significantly affect downstream applications like language modeling. We use the following metrics to estimate the amount and complexity of code-mixing in the datasets.

Switch-point (SP): As defined in the last section, switch-points are points within a sentence where the languages of the words on the two sides are different. Intuitively, sentences that have more number of SPs are inherently more complex. We also define the metric **SP Fraction** (SPF) as the number of SP in a sentence divided by the total number of word boundaries in the sentence.

Code mixing index (CMI): Proposed by Gamback and Das (2014, 2016), CMI quantifies the amount of code mixing in a corpus by accounting for the language distribution as well as the switching between them. Let N be the number of language tokens, x an utterance; let t_{L_i} be the tokens in language L_i, P be the number of code switching points in x. Then, the *Code mixed index per utterance*, $C_u(x)$ for x computed as follows,

$$C_u(x) = \frac{(N(x) - \max_{L_i \in L}\{t_{L_i}\}(x)) + P(x)}{N(x)} \tag{1}$$

Note that all the metrics can be computed at the sentence level as well as at the corpus level by averaging the values for all the sentences in a corpus.

3.2 Real Datasets

We chose to conduct all our experiments on English-Spanish CM tweets because English-Spanish CM is well documented (Solorio and Liu, 2008), is one of the most commonly mixed language pairs on social media (Rijhwani et al., 2017), and a couple of CM tweet datasets are readily available (Solorio et al., 2014; Rijhwani et al., 2017).

Dataset	# Tweets	# Words	CMI	SPF
		Mono		
English	100K	850K (48K)	0	0
Spanish	100K	860K (61K)	0	0
		rCM		
Train	100K	1.4M (91K)	0.31	0.105
Validation	100K	1.4M (91K)	0.31	0.106
Test-17	83K	1.1M (82K)	0.31	0.104
Test-14	13K	138K (16K)	0.12	0.06
gCM	31M	463M (79K)	0.75	0.35

Table 1: Size of the datasets. Numbers in parenthesis show the vocabulary size, i.e., the no. of unique words.

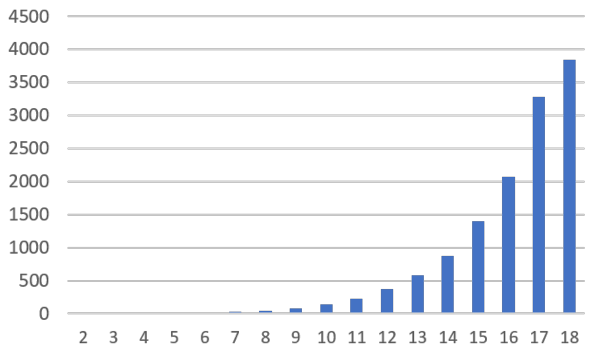


Figure 3: Average number of gCM sentences (y-axis) vs mean input sentence length (x-axis)

For our experiments, we use a subset of the tweets collected by Rijhwani et al. (2017) that were automatically identified as English, Spanish or English-Spanish CM. The authors provided us around 4.5M monolingual tweets per language, and 283K CM tweets. These were already deduplicated and tagged for hashtags, URLs, emoticons and language labels automatically through the method proposed in the paper. Table 1 shows the sizes of the various datasets, which are also described below.

Mono: 50K tweets were sampled for Spanish and English from the entire collection of monolingual tweets. The Spanish tweets were translated to English and vice versa, which gives us a total of 100K monolingual tweets in each language. We shall refer to this dataset as *Mono*. The sampling strategy and reason for generating translations will become apparent in Sec. 3.3.

rCM: We use two real CM datasets in our experiment. The 283K real CM tweets provided by Rijhwani et al. (2017) were randomly divided into training, validation and test sets of nearly equal sizes. Note that for most of our experiments, we will use a very small subset of the training set consisting of 5000 tweets as train data, because the fundamental assumption of this work is that very little amount of CM data is available for most language pairs (which is in fact true for most pairs beyond some very popularly mixed languages like English-Spanish). Nevertheless, the much larger training set is required for studying the effect of varying the amount of real CM data on our models. We shall refer to this training dataset as *rCM*. The test set with 83K tweets will be referred to as *Test-17*. We also use another dataset of English-Spanish CM tweets for testing our models which was released during the language labeling shared task at the Workshop on "Computational Approaches to Code-switching, EMNLP 2014" (Solorio et al., 2014). We mixed the training, validation and test datasets released during this shared task to construct a set of 13K tweets, which we shall refer to as *Test-14*. The two test datasets are tweets that were collected three years apart, and therefore, will help us estimate the robustness of the language models. As shown in Table 1, these datasets are quite different in terms of CMI and average number of SP per tweet. For computing the CMI and SP, we used a English-Spanish LID to language tag the words. In fact, 9500 tweets in the Test-14 dataset are monolingual, but we chose to retain them because it reflects the real distribution of CM data. Further, Test-14 also has manually annotated language labels, which will be helpful while conducting an in-depth analysis of the models.

3.3 Synthetic Code-Mixed Data

As described in the previous section, we use parallel monolingual sentences to generate grammatically valid code mixed sentences. The entire process involves the following four steps.

Step 1: We created the parallel corpus by generating translations for all the monolingual English and Spanish tweets (4.5M each) using the Bing Translator API.[2] We have found, that the translation quality varies widely across different sentences. Thus, we rank the translated sentences using *Pseudo Fuzzy-match Score* (PFS)

[2]https://www.microsoft.com/en-us/translator/translatorapi.aspx

(He et al., 2010). First, the forward translation engine (eg. English-to-Spanish) translates monolingual source sentence s into target t. Then the reverse translation system (eg. Spanish-English) translates target t into pseudo source s'. Equation 2 computes the PFS between s and s'.

$$PFS = \frac{EditDistance(s, s')}{max(|s|, |s'|)} \tag{2}$$

After manual inspection, we decided to select translation pairs whose $PFS \leq 0.7$. The edit distance is based on Wagner and Fischer (1974).

Step 2: We used the `fast_align` toolkit[3] (Dyer et al., 2013), to generate the word alignments from these parallel sentences.

Step 3: The constituency parses for all the English tweets were obtained using the Stanford PCFG parser (Klein and Manning, 2003).

Step 4: Using the parallel sentences, alignments and parse trees, we apply the Equivalent constraint theory (Sec 2.2) to generate all syntactically valid CM sentences while allowing for lexical substitution.

We randomly selected 50K monolingual Spanish and English tweets whose PFS $\leq$ 0.7. This gave us 200K monolingual tweets in all (*Mono* dataset) and the total amount of generated CM sentences from these 100K translation pairs was 31M, which we shall refer to as *gCM*. Note that even though we consider the *Mono* and *gCM* as two separate sets, in reality the EC model also generates the monolingual sentences; further, existence of *gCM* presumes existence of *Mono*. Hence, we also use *Mono* as part of all training experiments which use *gCM*.

We would also like to point out that the choice of experimenting with a much smaller set of tweets, only 50K per language, was made because the number of generated tweets even from this small set of monolingual tweet pairs is almost prohibitively large to allow experimentation with several models and their respective configurations.

4 Approach

Language modeling is a very widely researched topic (Rosenfeld, 2000; Bengio et al., 2003; Sundermeyer et al., 2015). In recent times, deep learning has been successfully employed to build efficient LMs (Mikolov et al., 2010; Sundermeyer et al., 2012; Arisoy et al., 2012; Che et al., 2017).

[3]https://github.com/clab/fast_align

Baheti et al. (2017) recently showed that there is significant effect of the training curriculum, that is the order in which data is presented to an RNN-based LM, on the perplexity of the learnt English-Spanish CM language model on tweets. Along similar lines, in this study we focus our experiments on training curriculum, especially regarding the use of gCM data during training, which is the primary contribution of this paper.

We do not attempt to innovate in terms of the architecture or computational structure of the LM, and use a standard LSTM-based RNN LM (Sundermeyer et al., 2012) for all our experiments. Indeed, there are enough reasons to believe that CM language is not fundamentally different from non-CM language, and therefore, should not require an altogether different LM architecture. Rather, the difference arises in terms of added complexity due to the presence of lexical items and syntactic structures from two linguistic systems that blows up the space of valid grammatical and lexical configurations, which makes it essential to train the models on large volumes of data.

4.1 Training Curricula

Baheti et al. (2017) showed that rather than randomly mixing the monolingual and CM data during training, the best performance is achieved when the LM is first trained with a mixture of monolingual texts from both languages in nearly equal proportions, and ending with CM data. Motivated by this finding, we define the following basic training curricula ("X | Y" indicates training the model first with data X and then data Y):

(1) rCM, **(2)** Mono, **(3)** Mono | rCM,
(4a) Mono | gCM, **(4b)** gCM | Mono,
(5a) Mono | gCM | rCM,
(5b) gCM | Mono | rCM

Curricula 1-3 are baselines, where gCM data is not used. Note that curriculum 3 is the best case according to Baheti et al. (2017). Curricula 4a and 4b help us examine how far generated data can substitute real data. Finally, curricula 5a and 5b use all the data, and we would expect them to perform the best.

Note that we do not experiment with other potential combinations (e.g., rCM | gCM | Mono) because it is known (and we also see this in our experiments) that adding rCM data at the end always leads to better models.

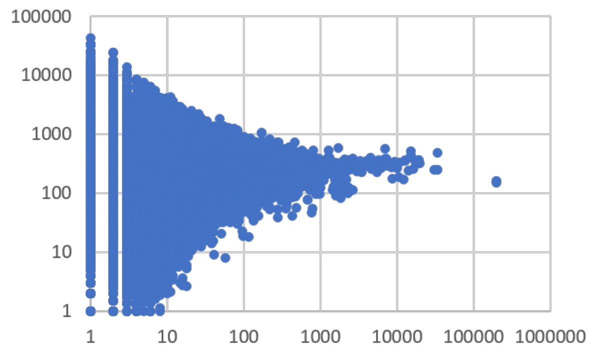


Figure 4: Scatter plot of fractional increase in word frequency in gCM (y-axis) vs original frequency (x-axis).

4.2 Sampling from gCM

As we have seen in Sec 3.3 (Fig. 3), in the EC model, a pair of monolingual parallel tweets gives rise to a large number (typically exponential in the length of the tweet) of CM tweets. On the other hand, in reality, only a few of those tweets would be observed. Further, if all the generated sentences are used to train an LM, it is not only computationally expensive, it also leads to undesirable results because the statistical properties of the distribution of the gCM corpus is very different from real data. We see this in our experiments (not reported in this paper for paucity of space), and also in Fig 4, where we plot the ratio of the frequencies of the words in *gCM* and *Mono* corpora (y-axis) against their original frequencies in *Mono* (x-axis). We can clearly see that the frequencies of the words are scaled up non-uniformly, the ratios varying between 1 and 500,000 for low frequency words.

In order to reduce this skew, instead of selecting the entire gCM data, we propose three sampling techniques for creating the training data from gCM:

Random: For each monolingual pair of parallel tweets, we randomly pick a fixed number, k, of CM tweets. We shall refer to the resultant training corpus as χ-***gCM***.

CMI-based: For each monolingual pair of parallel tweets, we randomly pick k CM tweets and bucket them using CMI (in 0.1 intervals). Thus, in this case we can define two different curricula, where we present the data in *increasing* or *decreasing* order of CMI during training, which will be represented by the notations $\uparrow$-***gCM*** and $\downarrow$-***gCM*** respectively.

SPF-based: For each monolingual pair of parallel tweets, we randomly pick k CM tweets such that the SPF distribution (section 3.1) of these tweets is similar to that of rCM data (as estimated from the validation set). This strategy will be referred to as ρ-***gCM***.

Thus, depending on the gCM sampling strategy used, curricula 4a-b and 5a-b can have three different versions each. Note that since CMI for *Mono* is 0, $\uparrow$-***gCM*** is not meaningful for 4b and 5b and similarly, $\downarrow$-***gCM*** not for 4a and 5a.

5 Experiments and Results

For all our experiments, we use a 2 layered RNN with LSTM units and hidden layer dimension of 100. While training, we use sampled softmax with 5000 samples instead of a full softmax to speed up the training process. The sampling is based on the word frequency in the training corpus. We use momentum SGD with a learning rate of 0.002. We have used the CNTK toolkit for building our models.[4] We use a fixed k=5 (from each monolingual pair) for sampling the gCM data. We observed the performance on $\uparrow$-gCM to be the best when trained till CMI 0.4 and similarly on $\downarrow$-gCM when trained from 1.0 to 0.6.

5.1 Results

Table 2 presents the perplexities on validation, Test-14 and Test-17 datasets for all the models (Col. 3, 4 and 5). We observe the following trends: (1) Model 5(b)-ρ has the least perplexity value (significantly different from the second lowest value in the column, $p < 0.00001$ for a paired t-test). (2) There is 55 and 90 point reduction in perplexity on Test-17 and Test-14 sets respectively from the baseline experiment 3, that does not use gCM data. Thus, addition of gCM data is helpful. (3) Only the 4a and 4b models are worse than 3, while 5a and 5b models are better. Hence, rCM is indispensable, even though gCM helps. (4) SPF based sampling performs significantly better (again $p < 0.00001$) than other sampling techniques.

To put these numbers in perspective, we also trained our model on 50k monolingual English data, which gave a PPL of 264. This shows that the high PPL values our models obtain are due to the inherent complexity of modeling CM language. This is further substantiated by the PPL

[4]https://www.microsoft.com/en-us/cognitive-toolkit/

ID	Training curriculum	Overall PPL			Avg. SP PPL		
		Valid	Test-17	Test-14	Valid	Test-17	Test-14
1	rCM	1995	2018	1822	5598	5670	8864
2	Mono	1588	1607	892	23378	23790	26901
3	Mono | rCM	1029	1041	861	4734	4824	7913
4(a)-χ	Mono | χ-gCM	1749	1771	1119	5752	5869	6065
4(a)-$\uparrow$	Mono | $\uparrow$-gCM	1852	1872	1208	9074	9167	8803
4(a)-ρ	Mono | ρ-gCM	1599	1618	1116	6534	6618	7293
4(b)-χ	χ-gCM | Mono	1659	1680	903	20634	21028	20300
4(b)-$\downarrow$	$\downarrow$-gCM | Mono	1900	1917	973	28422	28722	25006
4(b)-ρ	ρ-gCM | Mono	1622	1641	871	26191	26710	22557
5(a)-χ	Mono | χ-gCM | rCM	1026	1038	836	**4317**	**4386**	**5958**
5(a)-$\uparrow$	Mono | $\uparrow$-gCM | rCM	1045	1058	961	4983	5078	6861
5(a)-ρ	Mono | ρ-gCM | rCM	999	1011	830	4736	4829	6807
5(b)-χ	χ-gCM | Mono | rCM	1006	1019	790	4878	4987	7018
5(b)-$\downarrow$	$\downarrow$-gCM | Mono | rCM	1012	1025	800	5396	5489	7476
5(b)-ρ	ρ-gCM | Mono | rCM	**976**	**986**	**772**	4810	4912	6547

Table 2: Perplexity of the LM Models on all tweets and only on SP (right block).

RL	3	5(a)-χ	5(a)-ρ	5(a)-$\uparrow$	5(b)-$\downarrow$	5(b)-χ	5(b)-ρ
1	13222	**12815**	13717	14017	13761	13494	13077
2	2201	2120	**2064**	**2078**	2155	2256	2108
3	970	926	**902**	**896**	914	966	911
4	643	594	**567**	575	573	608	**571**
5	574	540	509	517	**502**	553	**503**
6	593	545	529	543	**520**	566	529
≥ 7	507	465	444	460	**431**	479	440

Table 3: Perplexities of minor language runs for various run lengths on Test-17.

# rCM	0.5K	1K	2.5K	5K	10K	50K
3	1238	1186	1120	1041	991	812
5(b)-ρ	1181	1141	1068	986	951	808

Table 4: Perplexity variation on Test-17 with changes in amount of rCM train data. Similar trends for other models (left for paucity of space)

values computed only at the code-switch points, which are shown in Table 2, col. 6, 7 and 8. Even for the best model, which in this case is **5(a)-χ**, PPL is four times higher than the overall PPL on Test-17.

Run length: The complexity of modeling CM is also apparent from Table 3, which reports the perplexity value of the 3 and 5 models for monolingual fragments of various *run lengths*. We define *run length* as the number of words in a maximal monolingual fragment or *run* within a tweet. In our analysis, we only consider runs of the *embedded language*, defined as the language that has fewer words. As one would expect, model 5(a)-χ performs the best for run length 1 (recall that it has lowest PPL at SP), but as the run length increases, the models sampling the gCM data using CMI (5(a)-$\uparrow$ and 5(b)-$\downarrow$) are better than the randomly sampled (χ) models. Run length 1 are typically cases of word borrowing and lexical substitution; higher run length segments are typically an indication of CM. Clearly, modeling the shorter runs of the embedded language seems to be one of the most challenging aspect of CM LM.

Sample size (k)	1	2	5	10
# tweets	93K	184K	497K	952K
5(b)-ρ	1081	1053	986	1019

Table 5: Variation of PPL on Test-17 with gCM sample size k. Similar trends for other models.

Significance of Linguistic Constraints: To understand the importance of the linguistic constraints imposed by EC on generation of gCM, we conducted an experiment where a synthetic CM corpus was created by combining random contiguous segments from the monolingual tweets such that the generated CM tweets' SPF distribution matched that of rCM. When we replaced gCM by this corpus in **5(b)-ρ**, the PPL on test-17 was 1060, which is worse than the baseline PPL.

Effect of rCM size: Table 4 shows the PPL values for models 3 and 5(b)-ρ when trained with different amounts of rCM data, keeping other parameters constant. As expected, the PPL drops for both models as rCM size increases. However, even with high rCM data, gCM does help in improving the LM until we have 50k rCM data (comparable to monolingual, and an unrealistic scenario in practice), where the returns of adding gCM starts diminishing. We also observe that in general, model 3 needs twice the amount of rCM data to perform as well as model 5(b)-ρ.

Effect of gCM size: In our sampling methods on gCM data, we fixed our sample size, k as 5 for consistency and feasibility of experiments. To understand the effect of k (and hence the size of the gCM data), we experimented with $k = 1, 2$, and 10 keeping everything else fixed. Table 5 reports the results for the models 3 and 5(b)-ρ. We observe that unlike rCM data, increasing gCM data or k does not necessarily decrease PPL after a point. We speculate that there is trade-off between k and the amount of rCM data, and also probably between these and the amount of monolingual data. We plan to explore this further in future.

6 Related Work

We briefly describe the various types of approaches used for building LM for CM text.

Bilingual models: These models combine data from monolingual data sources in both languages (Weng et al., 1997). **Factored models**: Gebhardt (2011) uses Factored Language Models for rescoring n-best lists during ASR decoding. The factors used include POS tags, CS point probability and LID. In Adel et al.(2014b; 2014a; 2013) RNNLMs are combined with n-gram based models, or converted to backoff models, giving improvements in perplexity and mixed error rate. **Models that incorporate linguistic constraints**: Li and Fung (2013) use inversion constraints to predict CS points and integrates this prediction into the ASR decoding process. Li and Fung (2014) integrates Functional Head constraints (FHC) for code-switching into the Language Model for Mandarin-English speech recognition. This work uses parsing techniques to restrict the lattice paths during decoding of speech to those permissible under the FHC theory. Our method instead imposes grammatical constraints (EC theory) to generate synthetic data, which can potentially be used to augment real CM data. This allows flexibility to deploy any sophisticated LM architecture and the synthetic data generated can also be used for CM tasks other than speech recognition. **Training curricula for CM**: Baheti et al. (2017) show that a training curriculum where an RNN-LM is trained first with interleaved monolingual data in both languages followed by CM data gives the best results for English-Spanish LM. The perplexity of this model is 4544, which then reduces to 298 after interpolation with a statistical n-gram LM. However, these numbers are not directly comparable to our work because the datasets are different. Our work is an extension of this approach showing that adding synthetic data further improves results.

We do not know of any work that uses synthetically generated CM data for training LMs.

7 Conclusion

In this paper, we presented a computational method for generating synthetic CM data based on the EC theory of code-mixing, and showed that sampling text from the synthetic corpus (according to the distribution of SPF found in real CM data) helps in reduction of PPL of the RNN-LM by an amount which is equivalently achieved by doubling the amount of real CM data. We also showed that randomly generated CM data doesn't improve the LM. Thus, the linguistic theory based generation is of crucial significance. There is no unanimous theory in linguistics on syntactic structure of CM language. Hence, as a future work, we would like to compare the usefulness of different linguistic theories and different constraints within each theory in our proposed LM framework. This can also provide an indirect validation of the theories. Further, we would like to study sampling techniques motivated by natural distributions of linguistic structures.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable suggestions.

References

Heike Adel, K. Kirchhoff, N. T. Vu, D. Telaar, and T. Schultz. 2014a. Combining recurrent neural networks and factored language models during decoding of code-switching speech. In *INTERSPEECH*, pages 1415–1419.

Heike Adel, K Kirchhoff, N T Vu, D Telaar, and T Schultz. 2014b. Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding. In *INTERSPEECH*, pages 651–655.

Heike Adel, N T Vu, and T Schultz. 2013. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *ACL (2)*, pages 206–211.

Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics.

Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. In *Proc. of ICON-2017, Kolkata, India*, pages 65–74.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *The 1st Workshop on Computational Approaches to Code Switching, EMNLP 2014*.

Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic inquiry*, pages 221–237.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.

A.-M. DiSciullo, Pieter Muysken, and R. Singh. 1986. Government and code-mixing. *Journal of Linguistics*, 22:1–24.

Chris Dyer, Victor Chahuneau, and N A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT 2013*, pages 644–648. Association for Computational Linguistics.

B. Gamback and A Das. 2014. On measuring the complexity of code-mixing. In *Proc. of the 1st Workshop on Language Technologies for Indian Social Media (Social-India)*.

B. Gamback and A Das. 2016. Comparing the level of code-switching in corpora. In *Proc. of the 10th International Conference on Language Resources and Evaluation (LREC)*.

Jan Gebhardt. 2011. Speech recognition on english-mandarin code-switching data using factored language models.

Yifan He, Yanjun Ma, Andy Way, and Josef Van Genabith. 2010. Integrating n-best smt outputs into a tm system. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 374–382. Association for Computational Linguistics.

A. K. Joshi. 1985. Processing of Sentences with Intrasentential Code Switching. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 190–205. Cambridge University Press, Cambridge.

D Klein and CD Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics*. Association of Computational Linguistics.

Ying Li and P Fung. 2013. Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints. In *ICASSP*, pages 7368–7372.

Ying Li and P Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *EMNLP*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Carol Myers-Scotton. 1993. *Duelling Languages:Grammatical structure in Code-switching*. Clarendon Press, Oxford.

Carol Myers-Scotton. 1995. A lexically based model of code-switching. In Lesley Milroy and Pieter Muysken, editors, *One Speaker, Two Languages: Cross-disciplinary Perspectives on Code-switching*, pages 233–256. Cambridge University Press, Cambridge.

Rana D. Parshad, Suman Bhowmick, Vineeta Chand, Nitu Kumari, and Neha Sinha. 2016. What is India speaking? Exploring the "Hinglish" invasion. *Physica A*, 449:375–389.

Shana Poplack. 1980. Sometimes Ill start a sentence in Spanish y termino en espaol. *Linguistics*, 18:581–618.

Ameya Prabhu, Aditya Joshi, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

Shruti Rijhwani, R Sequiera, M Choudhury, K Bali, and C S Maddila. 2017. Estimating code-switching on Twitter with a novel generalized word-level language identification technique. In *ACL*.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Koustav Rudra, S Rijhwani, R Begum, K Bali, M Choudhury, and N Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *EMNLP*, pages 1131–1141.

David Sankoff. 1998. A formal production-based explanation of the facts of code-switching. *Bilingualism: language and cognition*, 1(01):39–50.

A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Srivastava, R. Mamidi, and D.M Sharma. 2016. Shallow parsing pipeline for hindi-english code-mixed social media text. In *Proceedings of NAACL-HLT*.

Sunayana Sitaram, Sai Krishna Rallabandi, Shruti Rijhwani, and Alan W Black. 2016. Experiments with cross-lingual systems for synthesis of code-mixed text. In *9th ISCA Speech Synthesis Workshop*.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proc. of EMNLP*.

Thamar Solorio et al. 2014. Overview for the first shared task on language identification in code-switched data. *In 1st Workshop on Computational Approaches to Code Switching, EMNLP*, pages 62–72.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent lstm neural networks for language modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):517–529.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Yogarshi Vyas, S Gella, J Sharma, K Bali, and M Choudhury. 2014. POS Tagging of English-Hindi Code-Mixed Social Media Content. In *Proc. EMNLP*, pages 974–979.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Fuliang Weng, H Bratt, L Neumeyer, and A Stolcke. 1997. A study of multilingual speech recognition. In *EUROSPEECH*, volume 1997, pages 359–362. Citeseer.

Chinese NER Using Lattice LSTM

Yue Zhang[*] and **Jie Yang**[*]
Singapore University of Technology and Design
`yue_zhang@sutd.edu.sg`
`jie_yang@mymail.sutd.edu.sg`

Abstract

We investigate a lattice-structured LSTM model for Chinese NER, which encodes a sequence of input characters as well as all potential words that match a lexicon. Compared with character-based methods, our model explicitly leverages word and word sequence information. Compared with word-based methods, lattice LSTM does not suffer from segmentation errors. Gated recurrent cells allow our model to choose the most relevant characters and words from a sentence for better NER results. Experiments on various datasets show that lattice LSTM outperforms both word-based and character-based LSTM baselines, achieving the best results.

1 Introduction

As a fundamental task in information extraction, named entity recognition (NER) has received constant research attention over the recent years. The task has traditionally been solved as a sequence labeling problem, where entity boundary and category labels are jointly predicted. The current state-of-the-art for English NER has been achieved by using LSTM-CRF models (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Liu et al., 2018) with character information being integrated into word representations.

Chinese NER is correlated with word segmentation. In particular, named entity boundaries are also word boundaries. One intuitive way of performing Chinese NER is to perform word segmentation first, before applying word sequence labeling. The segmentation → NER pipeline, however, can suffer the potential issue of error propagation, since NEs are an important source of OOV in segmentation, and incorrectly segmented entity boundaries lead to NER errors. This problem can be severe in the open domain since cross-domain word segmentation remains an unsolved problem (Liu and Zhang, 2012; Jiang et al., 2013; Liu et al., 2014; Qiu and Zhang, 2015; Chen et al., 2017; Huang et al., 2017). It has been shown that character-based methods outperform word-based methods for Chinese NER (He and Wang, 2008; Liu et al., 2010; Li et al., 2014).

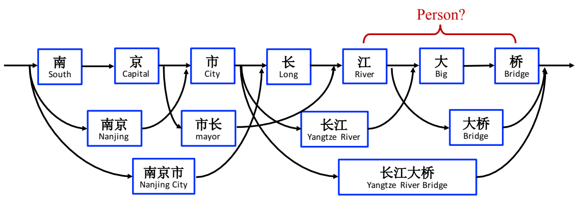


Figure 1: Word character lattice.

One drawback of character-based NER, however, is that explicit word and word sequence information is not fully exploited, which can be potentially useful. To address this issue, we integrate latent word information into character-based LSTM-CRF by representing lexicon words from the sentence using a lattice structure LSTM. As shown in Figure 1, we construct a word-character lattice by matching a sentence with a large automatically-obtained lexicon. As a result, word sequences such as "长江大桥 (Yangtze River Bridge)", "长江 (Yangtze River)" and "大桥 (Bridge)" can be used to disambiguate potential relevant named entities in a context, such as the person name "江大桥 (Daqiao Jiang)".

Since there are an exponential number of word-character paths in a lattice, we leverage a lattice LSTM structure for automatically controlling information flow from the beginning of the sentence to the end. As shown in Figure 2, gated cells are used to dynamically route information from

[*] Equal contribution.

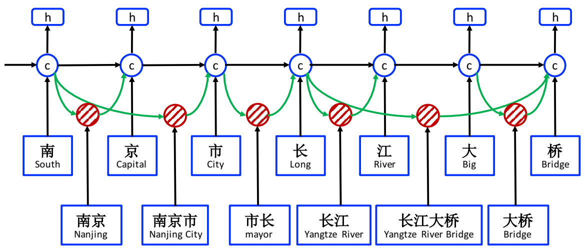


Figure 2: Lattice LSTM structure.

different paths to each character. Trained over NER data, the lattice LSTM can learn to find more useful words from context automatically for better NER performance. Compared with character-based and word-based NER methods, our model has the advantage of leveraging explicit word information over character sequence labeling without suffering from segmentation error.

Results show that our model significantly outperforms both character sequence labeling models and word sequence labeling models using LSTM-CRF, giving the best results over a variety of Chinese NER datasets across different domains. Our code and data are released at `https://github.com/jiesutd/LatticeLSTM`.

2 Related Work

Our work is in line with existing methods using neural network for NER. Hammerton (2003) attempted to solve the problem using a unidirectional LSTM, which was among the first neural models for NER. Collobert et al. (2011) used a CNN-CRF structure, obtaining competitive results to the best statistical models. dos Santos et al. (2015) used character CNN to augment a CNN-CRF model. Most recent work leverages an LSTM-CRF architecture. Huang et al. (2015) uses hand-crafted spelling features; Ma and Hovy (2016) and Chiu and Nichols (2016) use a character CNN to represent spelling characteristics; Lample et al. (2016) use a character LSTM instead. Our baseline word-based system takes a similar structure to this line of work.

Character sequence labeling has been the dominant approach for Chinese NER (Chen et al., 2006b; Lu et al., 2016; Dong et al., 2016). There have been explicit discussions comparing statistical word-based and character-based methods for the task, showing that the latter is empirically a superior choice (He and Wang, 2008; Liu et al., 2010; Li et al., 2014). We find that with proper representation settings, the same conclusion holds for neural NER. On the other hand, lattice LSTM is a better choice compared with both word LSTM and character LSTM.

How to better leverage word information for Chinese NER has received continued research attention (Gao et al., 2005), where segmentation information has been used as soft features for NER (Zhao and Kit, 2008; Peng and Dredze, 2015; He and Sun, 2017a), and joint segmentation and NER has been investigated using dual decomposition (Xu et al., 2014), multi-task learning (Peng and Dredze, 2016), etc. Our work is in line, focusing on neural representation learning. While the above methods can be affected by segmented training data and segmentation errors, our method does not require a word segmentor. The model is conceptually simpler by not considering multi-task settings.

External sources of information has been leveraged for NER. In particular, lexicon features have been widely used (Collobert et al., 2011; Passos et al., 2014; Huang et al., 2015; Luo et al., 2015). Rei (2017) uses a word-level language modeling objective to augment NER training, performing multi-task learning over large raw text. Peters et al. (2017) pretrain a character language model to enhance word representations. Yang et al. (2017b) exploit cross-domain and cross-lingual knowledge via multi-task learning. We leverage external data by pretraining word embedding lexicon over large automatically-segmented texts, while semi-supervised techniques such as language modeling are orthogonal to and can also be used for our lattice LSTM model.

Lattice structured RNNs can be viewed as a natural extension of tree-structured RNNs (Tai et al., 2015) to DAGs. They have been used to model motion dynamics (Sun et al., 2017), dependency-discourse DAGs (Peng et al., 2017), as well as speech tokenization lattice (Sperber et al., 2017) and multi-granularity segmentation outputs (Su et al., 2017) for NMT encoders. Compared with existing work, our lattice LSTM is different in both motivation and structure. For example, being designed for character-centric lattice-LSTM-CRF sequence labeling, it has recurrent cells but not hidden vectors for words. To our knowledge, we are the first to design a novel lattice LSTM representation for mixed characters and lexicon words, and the first to use a word-character lattice for segmentation-free Chinese NER.

3 Model

We follow the best English NER model (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016), using LSTM-CRF as the main network structure. Formally, denote an input sentence as $s = c_1, c_2, \ldots, c_m$, where c_j denotes the jth character. s can further be seen as a word sequence $s = w_1, w_2, \ldots, w_n$, where w_i denotes the ith word in the sentence, obtained using a Chinese segmentor. We use $t(i,k)$ to denote the index j for the kth character in the ith word in the sentence. Take the sentence in Figure 1 for example. If the segmentation is “南京市 长江大桥”, and indices are from 1, then $t(2,1) = 4$ (长) and $t(1,3) = 3$ (市). We use the BIOES tagging scheme (Ratinov and Roth, 2009) for both word-based and character-based NER tagging.

3.1 Character-Based Model

The character-based model is shown in Figure 3(a). It uses an LSTM-CRF model on the character sequence $c_1, c_2, \ldots, c_m$. Each character c_j is represented using

$$\mathbf{x}^c_j = \mathbf{e}^c(c_j) \tag{1}$$

$\mathbf{e}^c$ denotes a character embedding lookup table.

A bidirectional LSTM (same structurally as Eq. 11) is applied to $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ to obtain $\overrightarrow{\mathbf{h}}^c_1, \overrightarrow{\mathbf{h}}^c_2, \ldots, \overrightarrow{\mathbf{h}}^c_m$ and $\overleftarrow{\mathbf{h}}^c_1, \overleftarrow{\mathbf{h}}^c_2, \ldots, \overleftarrow{\mathbf{h}}^c_m$ in the left-to-right and right-to-left directions, respectively, with two distinct sets of parameters. The hidden vector representation of each character is:

$$\mathbf{h}^c_j = [\overrightarrow{\mathbf{h}}^c_j; \overleftarrow{\mathbf{h}}^c_j] \tag{2}$$

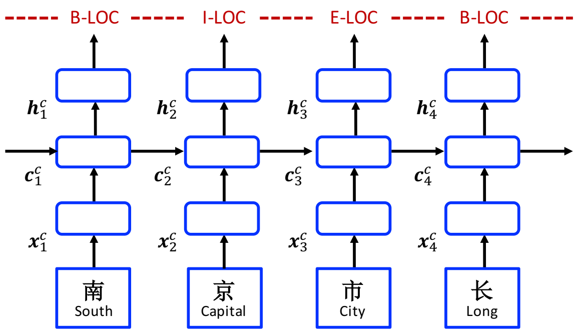
A standard CRF model (Eq. 17) is used on $\mathbf{h}^c_1, \mathbf{h}^c_2, \ldots, \mathbf{h}^c_m$ for sequence labelling.

• **Char + bichar**. Character bigrams have been shown useful for representing characters in word segmentation (Chen et al., 2015; Yang et al., 2017a). We augment the character-based model with bigram information by concatenating bigram embeddings with character embeddings:

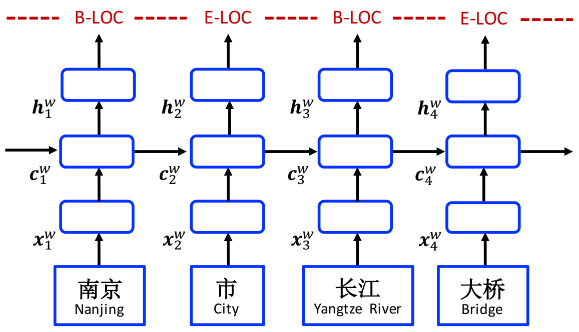
$$\mathbf{x}^c_j = [\mathbf{e}^c(c_j); \mathbf{e}^b(c_j, c_{j+1})], \tag{3}$$

where $\mathbf{e}^b$ denotes a charater bigram lookup table.

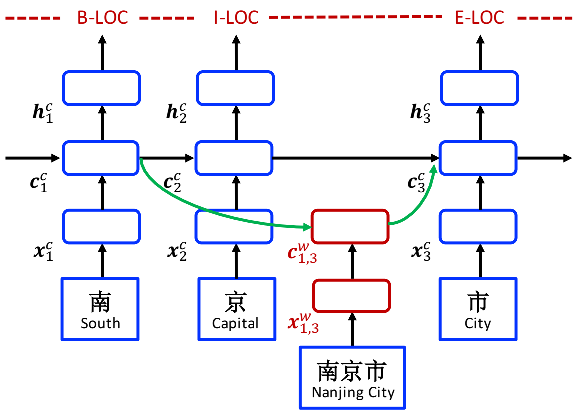
• **Char + softword**. It has been shown that using segmentation as soft features for character-based NER models can lead to improved performance (Zhao and Kit, 2008; Peng and Dredze, 2016).


B-LOC
I-LOC
E-LOC
B-LOC
h^c_1 h^c_2 h^c_3 h^c_4
c^c_1 c^c_2 c^c_3 c^c_4
x^c_1 x^c_2 x^c_3 x^c_4
南 South
京 Capital
市 City
长 Long

(a) Character-based model.


B-LOC
E-LOC
B-LOC
E-LOC
h^w_1 h^w_2 h^w_3 h^w_4
c^w_1 c^w_2 c^w_3 c^w_4
x^w_1 x^w_2 x^w_3 x^w_4
南京 Nanjing
市 City
长江 Yangtze River
大桥 Bridge

(b) Word-based model.



(c) Lattice model.

Figure 3: Models.[1]

We augment the character representation with segmentation information by concatenating segmentation label embeddings to character embeddings:

$$\mathbf{x}^c_j = [\mathbf{e}^c(c_j); \mathbf{e}^s(seg(c_j))], \tag{4}$$

where $\mathbf{e}^s$ represents a segmentation label embedding lookup table. $seg(c_j)$ denotes the segmentation label on the character c_j given by a word segmentor. We use the BMES scheme for repre-

[1]To keep the figure concise, we (i) do not show gate cells, which uses $\mathbf{h}_{t-1}$ for calculating $\mathbf{c}_t$; (ii) only show one direction.

senting segmentation (Xue, 2003).

$$\mathbf{h}_i^w = [\overrightarrow{\mathbf{h}_i^w}; \overleftarrow{\mathbf{h}_i^w}] \tag{5}$$

Similar to the character-based case, a standard CRF model (Eq. 17) is used on $\mathbf{h}_1^w, \mathbf{h}_2^w, \ldots, \mathbf{h}_m^w$ for sequence labelling.

3.2 Word-Based Model

The word-based model is shown in Figure 3(b). It takes the word embedding $\mathbf{e}^w(w_i)$ for representation each word w_i:

$$\mathbf{x}_i^w = \mathbf{e}^w(w_i), \tag{6}$$

where $\mathbf{e}^w$ denotes a word embedding lookup table. A bi-directioanl LSTM (Eq. 11) is used to obtain a left-to-right sequence of hidden states $\overrightarrow{\mathbf{h}_1^w}, \overrightarrow{\mathbf{h}_2^w}, \ldots, \overrightarrow{\mathbf{h}_n^w}$ and a right-to-left sequence of hidden states $\overleftarrow{\mathbf{h}_1^w}, \overleftarrow{\mathbf{h}_2^w}, \ldots, \overleftarrow{\mathbf{h}_n^w}$ for the words $w_1, w_2, \ldots, w_n$, respectively. Finally, for each word w_i, $\overrightarrow{\mathbf{h}_i^w}$ and $\overleftarrow{\mathbf{h}_i^w}$ are concatenated as its representation:

Integrating character representations

Both character CNN (Ma and Hovy, 2016) and LSTM (Lample et al., 2016) have been used for representing the character sequence within a word. We experiment with both for Chinese NER. Denoting the representation of characters within w_i as $\mathbf{x}_i^c$, a new word representation is obtained by concatenation of $\mathbf{e}^w(w_i)$ and $\mathbf{x}_i^c$:

$$\mathbf{x}_i^w = [\mathbf{e}^w(w_i); \mathbf{x}_i^c] \tag{7}$$

• **Word + char LSTM**. Denoting the embedding of each input character as $\mathbf{e}^c(c_j)$, we use a bi-directional LSTM (Eq. 11) to learn hidden states $\overrightarrow{\mathbf{h}}^c_{t(i,1)}, \ldots, \overrightarrow{\mathbf{h}}^c_{t(i,len(i))}$ and $\overleftarrow{\mathbf{h}}^c_{t(i,1)}, \ldots, \overleftarrow{\mathbf{h}}^c_{t(i,len(i))}$ for the characters $c_{t(i,1)}, \ldots, c_{t(i,len(i))}$ of w_i, where $len(i)$ denotes the number of characters in w_i. The final character representation for w_i is:

$$\mathbf{x}_i^c = [\overrightarrow{\mathbf{h}}^c_{t(i,len(i))}; \overleftarrow{\mathbf{h}}^c_{t(i,1)}] \tag{8}$$

• **Word + char LSTM′**. We investigate a variation of word + char LSTM model that uses a **single** LSTM to obtain $\overrightarrow{\mathbf{h}}^c_j$ and $\overleftarrow{\mathbf{h}}^c_j$ for each c_j. It is similar with the structure of Liu et al. (2018) but not uses the highway layer. The same LSTM structure as defined in Eq. 11 is used, and the same method as Eq. 8 is used to integrate character hidden states into word representations.

• **Word + char CNN**. A standard CNN (LeCun et al., 1989) structure is used on the character sequence of each word to obtain its character representation $\mathbf{x}_i^c$. Denoting the embedding of character c_j as $\mathbf{e}^c(c_j)$, the vector $\mathbf{x}_i^c$ is given by:

$$\mathbf{x}_i^c = \max_{t(i,1) \le j \le t(i,len(i))} (\mathbf{W}_{\text{CNN}}^\top \begin{bmatrix} \mathbf{e}^c(c_{j-\frac{ke-1}{2}}) \\ \ldots \\ \mathbf{e}^c(c_{j+\frac{ke-1}{2}}) \end{bmatrix} + \mathbf{b}_{\text{CNN}}), \tag{9}$$

where $\mathbf{W}_{\text{CNN}}$ and $\mathbf{b}_{\text{CNN}}$ are parameters, $ke = 3$ is the kernal size and max denotes max pooling.

3.3 Lattice Model

The overall structure of the word-character lattice model is shown in Figure 2, which can be viewed as an extension of the character-based model, integrating word-based cells and additional gates for controlling information flow.

Shown in Figure 3(c), the input to the model is a character sequence $c_1, c_2, \ldots, c_m$, together with all character subsequences that match words in a lexicon $\mathbb{D}$. As indicated in Section 2, we use automatically segmented large raw text for buinding $\mathbb{D}$. Using $w_{b,e}^d$ to denote such a subsequence that begins with character index b and ends with character index e, the segment $w_{1,2}^d$ in Figure 1 is "南京 (Nanjing)" and $w_{7,8}^d$ is "大桥 (Bridge)".

Four types of vectors are involved in the model, namely *input vectors*, *output hidden vectors*, *cell vectors* and *gate vectors*. As basic components, a character input vector is used to represent each chacracter c_j as in the character-based model:

$$\mathbf{x}_j^c = \mathbf{e}^c(c_j) \tag{10}$$

The basic recurrent structure of the model is constructed using a character cell vector $\mathbf{c}_j^c$ and a hidden vector $\mathbf{h}_j^c$ on each c_j, where $\mathbf{c}_j^c$ serves to record recurrent information flow from the beginning of the sentence to c_j and $\mathbf{h}_j^c$ is used for CRF sequence labelling using Eq. 17.

The basic recurrent LSTM functions are:

$$\begin{bmatrix} \mathbf{i}_j^c \\ \mathbf{o}_j^c \\ \mathbf{f}_j^c \\ \widetilde{\mathbf{c}}_j^c \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \left(\mathbf{W}^{c\top} \begin{bmatrix} \mathbf{x}_j^c \\ \mathbf{h}_{j-1}^c \end{bmatrix} + \mathbf{b}^c \right) \tag{11}$$
$$\mathbf{c}_j^c = \mathbf{f}_j^c \odot \mathbf{c}_{j-1}^c + \mathbf{i}_j^c \odot \widetilde{\mathbf{c}}_j^c$$
$$\mathbf{h}_j^c = \mathbf{o}_j^c \odot tanh(\mathbf{c}_j^c)$$

where $\mathbf{i}_j^c, \mathbf{f}_j^c$ and $\mathbf{o}_j^c$ denote a set of input, forget and output gates, respectively. $\mathbf{W}^{c\top}$ and $\mathbf{b}^c$ are model parameters. $\sigma()$ represents the sigmoid function.

Different from the character-based model, however, the computation of $\mathbf{c}^c_j$ now considers lexicon subsequences $w^d_{b,e}$ in the sentence. In particular, each subsequence $w^d_{b,e}$ is represented using

$$\mathbf{x}^w_{b,e} = \mathbf{e}^w(w^d_{b,e}), \tag{12}$$

where $\mathbf{e}^w$ denotes the same word embedding lookup table as in Section 3.2.

In addition, a word cell $\mathbf{c}^w_{b,e}$ is used to represent the recurrent state of $\mathbf{x}^w_{b,e}$ from the beginning of the sentence. The value of $\mathbf{c}^w_{b,e}$ is calculated by:

$$\begin{bmatrix} \mathbf{i}^w_{b,e} \\ \mathbf{f}^w_{b,e} \\ \widetilde{\mathbf{c}}^w_{b,e} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ tanh \end{bmatrix} \left(\mathbf{W}^{w\top} \begin{bmatrix} \mathbf{x}^w_{b,e} \\ \mathbf{h}^c_b \end{bmatrix} + \mathbf{b}^w \right) \tag{13}$$
$$\mathbf{c}^w_{b,e} = \mathbf{f}^w_{b,e} \odot \mathbf{c}^c_b + \mathbf{i}^w_{b,e} \odot \widetilde{\mathbf{c}}^w_{b,e}$$

where $\mathbf{i}^w_{b,e}$ and $\mathbf{f}^w_{b,e}$ are a set of input and forget gates. There is no output gate for word cells since labeling is performed only at the character level.

With $\mathbf{c}^w_{b,e}$, there are more recurrent paths for information flow into each $\mathbf{c}^c_j$. For example, in Figure 2, input sources for $\mathbf{c}^c_7$ include $\mathbf{x}^c_7$ (桥 Bridge), $\mathbf{c}^w_{6,7}$ (大桥 Bridge) and $\mathbf{c}^w_{4,7}$ (长江大桥 Yangtze River Bridge).[2] We link all $\mathbf{c}^w_{b,e}$ with $b \in \{b'|w^d_{b',e} \in \mathbb{D}\}$ to the cell $\mathbf{c}^c_e$. We use an additional gate $\mathbf{i}^c_{b,e}$ for each subsequence cell $\mathbf{c}^w_{b,e}$ for controlling its contribution into $\mathbf{c}^c_{b,e}$:

$$\mathbf{i}^c_{b,e} = \sigma\left(\mathbf{W}^{l\top} \begin{bmatrix} \mathbf{x}^c_e \\ \mathbf{c}^w_{b,e} \end{bmatrix} + \mathbf{b}^l\right) \tag{14}$$

The calculation of cell values $\mathbf{c}^c_j$ thus becomes

$$\mathbf{c}^c_j = \sum_{b \in \{b'|w^d_{b',j} \in \mathbb{D}\}} \boldsymbol{\alpha}^c_{b,j} \odot \mathbf{c}^w_{b,j} + \boldsymbol{\alpha}^c_j \odot \widetilde{\mathbf{c}}^c_j \tag{15}$$

In Eq. 15, the gate values $\mathbf{i}^c_{b,j}$ and $\mathbf{i}^c_j$ are normalised to $\boldsymbol{\alpha}^c_{b,j}$ and $\boldsymbol{\alpha}^c_j$ by setting the sum to $\mathbf{1}$.

$$\begin{aligned} \boldsymbol{\alpha}^c_{b,j} &= \frac{exp(\mathbf{i}^c_{b,j})}{exp(\mathbf{i}^c_j) + \sum_{b' \in \{b''|w^d_{b'',j} \in \mathbb{D}\}} exp(\mathbf{i}^c_{b',j})} \\ \boldsymbol{\alpha}^c_j &= \frac{exp(\mathbf{i}^c_j)}{exp(\mathbf{i}^c_j) + \sum_{b' \in \{b''|w^d_{b'',j} \in \mathbb{D}\}} exp(\mathbf{i}^c_{b',j})} \end{aligned} \tag{16}$$

The final hidden vectors $\mathbf{h}^c_j$ are still computed as described by Eq. 11. During NER training, loss values back-propagate to the parameters $\mathbf{W}^c, \mathbf{b}^c, \mathbf{W}^w, \mathbf{b}^w, \mathbf{W}^l$ and $\mathbf{b}^l$ allowing the model to dynamically focus on more relevant words during NER labelling.

[2]We experimented with alternative configurations on indexing word and character path links, finding that this configuration gives the best results in preliminary experiments. Single-character words are excluded; the final performance drops slightly after integrating single-character words.

Dataset	Type	Train	Dev	Test
OntoNotes	Sentence	15.7k	4.3k	4.3k
	Char	491.9k	200.5k	208.1k
MSRA	Sentence	46.4k	–	4.4k
	Char	2169.9k	–	172.6
Weibo	Sentence	1.4k	0.27k	0.27k
	Char	73.8k	14.5k	14.8k
resume	Sentence	3.8k	0.46k	0.48k
	Char	124.1k	13.9k	15.1k

Table 1: Statistics of datasets.

3.4 Decoding and Training

A standard CRF layer is used on top of $\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_\tau$, where τ is n for character-based and lattice-based models and m for word-based models. The probability of a label sequence $y = l_1, l_2, \dots, l_\tau$ is

$$P(y|s) = \frac{exp(\sum_i(\mathbf{W}^{l_i}_{\text{CRF}}\mathbf{h}_i + b^{(l_{i-1},l_i)}_{\text{CRF}}))}{\sum_{y'} exp(\sum_i(\mathbf{W}^{l'_i}_{\text{CRF}}\mathbf{h}_i + b^{(l'_{i-1},l'_i)}_{\text{CRF}}))} \tag{17}$$

Here y' represents an arbitary label sequence, and $\mathbf{W}^{l_i}_{\text{CRF}}$ is a model parameter specific to l_i, and $b^{(l_{i-1},l_i)}_{\text{CRF}}$ is a bias specific to l_{i-1} and l_i.

We use the first-order Viterbi algorithm to find the highest scored label sequence over a word-based or character-based input sequence. Given a set of manually labeled training data $\{(s_i, y_i)\}|^N_{i=1}$, sentence-level log-likelihood loss with L_2 regularization is used to train the model:

$$L = \sum^N_{i=1} log(P(y_i|s_i)) + \frac{\lambda}{2}||\Theta||^2, \tag{18}$$

where λ is the L_2 regularization parameter and Θ represents the parameter set.

4 Experiments

We carry out an extensive set of experiments to investigate the effectiveness of word-character lattice LSTMs across different domains. In addition, we aim to empirically compare word-based and character-based neural Chinese NER under different settings. Standard precision (P), recall (R) and F1-score (F1) are used as evaluation metrics.

4.1 Experimental Settings

Data. Four datasets are used in this paper, which include OntoNotes 4 (Weischedel et al., 2011), MSRA (Levow, 2006) Weibo NER (Peng and

Statistics	Train	Dev	Test
Country	260	33	28
Educational Institution	858	106	112
Location	47	2	6
Personal Name	952	110	112
Organization	4611	523	553
Profession	287	18	33
Ethnicity Background	115	15	14
Job Title	6308	690	772
Total Entity	13438	1497	1630

Table 2: Detailed statistics of resume NER.

Dredze, 2015; He and Sun, 2017a) and a Chinese resume dataset that we annotate. Statistics of the datasets are shown in Table 1. We take the same data split as Che et al. (2013) on OntoNotes. The development set of OntoNotes is used for reporting development experiments. While the OntoNotes and MSRA datasets are in the news domain, the Weibo NER dataset is drawn from the social media website Sina Weibo.[3]

For more variety in test domains, we collected a resume dataset from Sina Finance[4], which consists of resumes of senior executives from listed companies in the Chinese stock market. We randomly selected 1027 resume summaries and manually annotated 8 types of named entities. Statistics of the dataset is shown in Table 2. The inter-annotator agreement is 97.1%. We release this dataset as a resource for further research.

Segmentation. For the OntoNotes and MSRA datasets, gold-standard segmentation is available in the training sections. For OntoNotes, gold segmentation is also available for the development and test sections. On the other hand, no segmentation is available for the MSRA test sections, nor the Weibo / resume datasets. As a result, OntoNotes is leveraged for studying oracle situations where gold segmentation is given. We use the neural word segmentor of Yang et al. (2017a) to automatically segment the development and test sets for word-based NER. In particular, for the OntoNotes and MSRA datasets, we train the segmentor using gold segmentation on their respective training sets. For Weibo and resume, we take the best model of Yang et al. (2017a) off the shelf[5], which is trained using CTB 6.0 (Xue et al., 2005).

[3]`https://www.weibo.com/`

[4]`http://finance.sina.com.cn/stock/index.shtml`

[5]`https://github.com/jiesutd/RichWordSegmentor`

Parameter	Value	Parameter	Value
char emb size	50	bigram emb size	50
lattice emb size	50	LSTM hidden	200
char dropout	0.5	lattice dropout	0.5
LSTM layer	1	regularization λ	1e-8
learning rate lr	0.015	lr decay	0.05

Table 3: Hyper-parameter values.

Word Embeddings. We pretrain word embeddings using word2vec (Mikolov et al., 2013) over automatically segmented Chinese Giga-Word[6], obtaining 704.4k words in a final lexicon. In particular, the number of single-character, two-character and three-character words are 5.7k, 291.5k, 278.1k, respectively. The embedding lexicon is released alongside our code and models as a resource for further research. Word embeddings are fine-tuned during NER training. Character and character bigram embeddings are pretrained on Chinese Giga-Word using word2vec and fine-tuned at model training.

Hyper-parameter settings. Table 3 shows the values of hyper-parameters for our models, which as fixed according to previous work in the literature without grid-search adjustments for each individual dataset. In particular, the embedding sizes are set to 50 and the hidden size of LSTM models to 200. Dropout (Srivastava et al., 2014) is applied to both word and character embeddings with a rate of 0.5. Stochastic gradient descent (SGD) is used for optimization, with an initial learning rate of 0.015 and a decay rate of 0.05.

4.2 Development Experiments

We compare various model configurations on the OntoNotes development set, in order to select the best settings for word-based and character-based NER models, and to learn the influence of lattice word information on character-based models.

Character-based NER. As shown in Table 4, without using word segmentation, a character-based LSTM-CRF model gives a development F1-score of 62.47%. Adding character-bigram and softword representations as described in Section 3.1 increases the F1-score to 67.63% and 65.71%, respectively, demonstrating the usefulness of both sources of information. In addition, a combination of both gives a 69.64% F1-score, which is the best

[6]`https://catalog.ldc.upenn.edu/LDC2011T13`

Input	Models	P	R	F1
Auto seg	Word baseline	73.20	57.05	64.12
	+char LSTM	71.98	65.41	68.54
	+char LSTM′	71.08	65.83	68.35
	+char+bichar LSTM	72.63	67.60	70.03
	+char CNN	73.06	66.29	69.51
	+char+bichar CNN	72.01	65.50	68.60
No seg	Char baseline	67.12	58.42	62.47
	+softword	69.30	62.47	65.71
	+bichar	71.67	64.02	67.63
	+bichar+softword	72.64	66.89	69.64
	Lattice	**74.64**	**68.83**	**71.62**

Table 4: Development results.

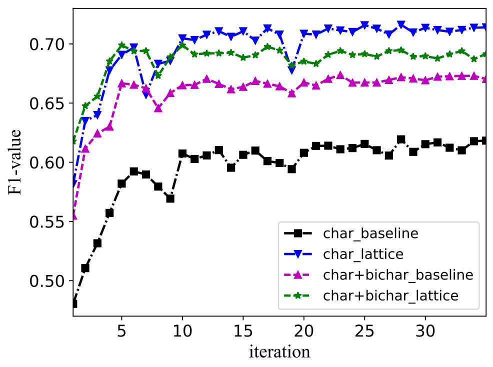


Figure 4: F1 against training iteration number.

among various character representations. We thus choose this model in the remaining experiments.

Word-based NER. Table 4 shows a variety of different settings for word-based Chinese NER. With automatic segmentation, a word-based LSTM CRF baseline gives a 64.12% F1-score, which is higher compared to the character-based baseline. This demonstrates that both word information and character information are useful for Chinese NER. The two methods of using character LSTM to enrich word representations in Section 3.2, namely word+char LSTM and word+char LSTM′, lead to similar improvements.

A CNN representation of character sequences gives a slightly higher F1-score compared to LSTM character representations. On the other hand, further using character bigram information leads to increased F1-score over word+char LSTM, but decreased F1-score over word+char CNN. A possible reason is that CNN inherently captures character n-gram information. As a result, we use word+char+bichar LSTM for word-based NER in the remaining experiments, which gives the best development results, and is structurally consistent with the state-of-the-art English NER models in the literature.

Lattice-based NER. Figure 4 shows the F1-score of character-based and lattice-based models against the number of training iterations. We include models that use concatenated character and character bigram embeddings, where bigrams can play a role in disambiguating characters. As can be seen from the figure, lattice word information is useful for improving character-based NER, improving the best development result from 62.5% to 71.6%. On the other hand, the bigram-enhanced lattice model does not lead to further improvements compared with the original lattice model.

Input	Models	P	R	F1
Gold seg	Yang et al. (2016)	65.59	71.84	68.57
	Yang et al. (2016)*†	72.98	**80.15**	**76.40**
	Che et al. (2013)*	77.71	72.51	75.02
	Wang et al. (2013)*	76.43	72.32	74.32
	Word baseline	76.66	63.60	69.52
	+char+bichar LSTM	**78.62**	73.13	75.77
Auto seg	Word baseline	72.84	59.72	65.63
	+char+bichar LSTM	73.36	70.12	71.70
No seg	Char baseline	68.79	60.35	64.30
	+bichar+softword	74.36	69.43	71.81
	Lattice	**76.35**	**71.56**	**73.88**

Table 5: Main results on OntoNotes.

This is likely because words are better sources of information for character disambiguation compared with bigrams, which are also ambiguous.

As shown in Table 4, the lattice LSTM-CRF model gives a development F1-score of 71.62%, which is significantly[7] higher compared with both the word-based and character-based methods, despite that it does not use character bigrams or word segmentation information. The fact that it significantly outperforms char+softword shows the advantage of lattice word information as compared with segmentor word information.

4.3 Final Results

OntoNotes. The OntoNotes test results are shown in Table 5[8]. With gold-standard segmentation, our word-based methods give competitive results to the state-of-the-art on the dataset (Che et al., 2013; Wang et al., 2013), which leverage bilingual data. This demonstrates that LSTM-CRF is a competitive choice for word-based Chinese NER, as it is for other languages. In addition, the results show

[7]We use a p-value of less than 0.01 from pairwise t-test to indicate statistical significance.

[8]In Table 5, 6 and 7, we use * to denote a model with external labeled data for semi-supervised learning. † means that the model also uses discrete features.

Models	P	R	F1
Chen et al. (2006a)	91.22	81.71	86.20
Zhang et al. (2006)*	92.20	90.18	91.18
Zhou et al. (2013)	91.86	88.75	90.28
Lu et al. (2016)	–	–	87.94
Dong et al. (2016)	91.28	90.62	90.95
Word baseline	90.57	83.06	86.65
+char+bichar LSTM	91.05	89.53	90.28
Char baseline	90.74	86.96	88.81
+bichar+softword	92.97	90.80	91.87
Lattice	**93.57**	**92.79**	**93.18**

Table 6: Main results on MSRA.

Models	NE	NM	Overall
Peng and Dredze (2015)	51.96	61.05	56.05
Peng and Dredze (2016)*	**55.28**	**62.97**	**58.99**
He and Sun (2017a)	50.60	59.32	54.82
He and Sun (2017b)*	54.50	62.17	58.23
Word baseline	36.02	59.38	47.33
+char+bichar LSTM	43.40	60.30	52.33
Char baseline	46.11	55.29	52.77
+bichar+softword	50.55	60.11	56.75
Lattice	**53.04**	**62.25**	**58.79**

Table 7: Weibo NER results.

Models	P	R	F1
Word baseline	93.72	93.44	93.58
+char+bichar LSTM	94.07	94.42	94.24
Char baseline	93.66	93.31	93.48
+bichar+softword	94.53	**94.29**	94.41
Lattice	**94.81**	94.11	**94.46**

Table 8: Main results on resume NER.

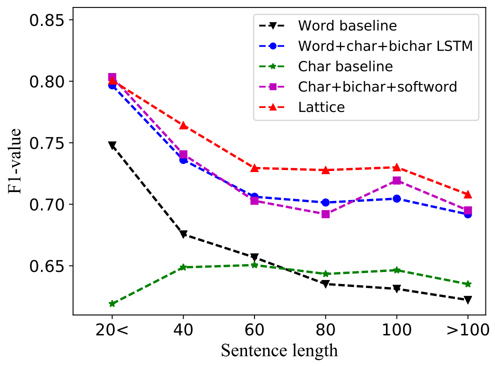


Figure 5: F1 against sentence length.

that our word-based models can serve as highly competitive baselines. With automatic segmentation, the F1-score of word+char+bichar LSTM decreases from 75.77% to 71.70%, showing the influence of segmentation to NER. Consistent with observations on the development set, adding lattice word information leads to an 88.81% $\rightarrow$ 93.18% increasement of F1-score over the character baseline, as compared with 88.81% $\rightarrow$ 91.87% by adding bichar+softword. The lattice model gives significantly the best F1-score on automatic segmentation.

MSRA. Results on the MSRA dataset are shown in Table 6. For this benchmark, no gold-standard segmentation is available on the test set. Our chosen segmentor gives 95.93% accuracy on 5-fold cross-validated training set. The best statistical models on the dataset leverage rich hand-crafted features (Chen et al., 2006a; Zhang et al., 2006; Zhou et al., 2013) and character embedding features (Lu et al., 2016). Dong et al. (2016) exploit neural LSTM-CRF with radical features.

Compared with the existing methods, our word-based and character-based LSTM-CRF models give competitive accuracies. The lattice model significantly outperforms both the best character-based and word-based models ($p < 0.01$), achieving the best result on this standard benchmark.

Weibo/resume. Results on the Weibo NER dataset are shown in Table 7, where NE, NM and Overall denote F1-scores for named entities, nominal entities (excluding named entities) and both, respectively. Gold-standard segmentation is not available for this dataset. Existing state-of-the-art systems include Peng and Dredze (2016) and He and Sun (2017b), who explore rich embedding features, cross-domain and semi-supervised data, some of which are orthogonal to our model[9].

Results on the resume NER test data are shown in Table 8. Consistent with observations on OntoNotes and MSRA, the lattice model significantly outperforms both the word-based mode and the character-based model for Weibo and resume ($p < 0.01$), giving state-of-the-art results.

4.4 Discussion

F1 against sentence length. Figure 5 shows the F1-scores of the baseline models and lattice LSTM-CRF on the OntoNotes dataset. The character-based baseline gives relatively stable F1-scores over different sentence lengths, although the performances are relatively low. The word-based baseline gives substantially higher F1-scores over short sentences, but lower F1-scores over long sentences, which can be because of lower segmentation accuracies over longer sentences. Both word+char+bichar and char+bichar+softword give better performances compared to their respective baselines, showing

[9]The results of Peng and Dredze (2015, 2016) are taken from Peng and Dredze (2017).

Sentence (truncated)	卸下东莞台协会长职务后 After stepping down as president of Taiwan Association in Dongguan.
Correct Segmentation	卸下 东莞 台 协 会长 职务 后 step down, Dongguan, Taiwan, association, president, role, after
Auto Segmentation	卸下 东莞 台 协会长 职务 后 step down, Dongguan, Taiwan, association president, role, after
Lattice words	卸下 下东 东莞 台协会 协会 会长 长职 职务 step down, incorrect word, Dongguan, Taiwan association, association, president, permanent job, role
Word+char+bichar LSTM	卸下 东莞 $_{GPE}$ 台 $_{GPE}$协会长职务后 . . . Dongguan $_{GPE}$ Taiwan $_{GPE}$ . . .
Char+bichar+softword	卸下 东莞台协会 $_{ORG}$长职务后 . . . Taiwan Association in Dongguan $_{ORG}$ . . . (ungrammatical)
Lattice	卸下 东莞台协 $_{ORG}$会长职务后 . . . Taiwan Association in Dongguan $_{ORG}$ . . .

Table 9: Example. Red and green represent incorrect and correct entities, respectively.

that word and character representations are complementary for NER. The accuracy of lattice also decreases as the sentence length increases, which can result from exponentially increasing number of word combinations in lattice. Compared with word+char+bichar and char+bichar+softword, the lattice model shows more robustness to increased sentence lengths, demonstrating the more effective use of word information.

F1 against sentence length. Table 9 shows a case study comparing char+bichar+softword, word+char+bichar and the lattice model. In the example, there is much ambiguity around the named entity "东莞台协 (Taiwan Association in Dongguan)". Word+char+bichar yields the entities "东莞 (Dongguan)" and "台 (Taiwan)" given that "东莞台协 (Taiwan Association in Dongguan)" is not in the segmentor output. Char+bichar+softword recognizes "东莞台协会 (Taiwan Association in Dongguan)", which is valid on its own, but leaves the phrase "长职务后" ungrammatical. In contrast, the lattice model detects the organization name correctly, thanks to the lattice words "东莞 (Dongguan)", "会长 (President)" and "职务 (role)". There are also irrelevant words such as "台协会 (Taiwan Association)" and "下东 (noisy word)" in the lexicon, which did not affect NER results.

Note that both word+char+bichar and lattice use the same source of word information, namely the same pretrained word embedding lexicon. However, word+char+bichar first uses the lexicon in the segmentor, which imposes hard constrains (i.e. fixed words) to its subsequence use in NER. In contrast, lattice LSTM has the freedom of considering all lexicon words.

Entities in lexicon. Table 10 shows the total number of entities and their respective match ratios in the lexicon. The error reductions (ER) of the final lattice model over the best character-based method (i.e. "+bichar+softword") are also shown. It can be seen that error reductions have a correlation between matched entities in the lexicon. In this respect, our automatic lexicon also played to some extent the role of a gazetteer (Ratinov and Roth, 2009; Chiu and Nichols, 2016), but not fully since there is no explicit knowledge in the lexicon which tokens are entities. The ultimate disambiguation power still lies in the lattice encoder and supervised learning.

Dataset	**Split**	**#Entity**	**#Match**	**Ratio** (%)	**ER** (%)
OntoNotes	Train	13.4k	9.5k	71.04	–
	Test	7.7k	6.0k	78.72	7.34
MSRA	Train	74.7k	54.3k	72.62	–
	Test	6.2k	4.6k	73.76	16.11
Weibo (all)	Train	1.9k	1.1k	58.83	–
	Test	414	259	62.56	4.72
resume	Train	13.4k	3.8k	28.55	–
	Test	1.6k	483	29.63	0.89

Table 10: Entities in lexicon.

The quality of the lexicon may affect the accuracy of our NER model since noise words can potentially confuse NER. On the other hand, our lattice model can potentially learn to select more correct words during NER training. We leave the investigation of such influence to future work.

5 Conclusion

We empirically investigated a lattice LSTM-CRF representations for Chinese NER, finding that it gives consistently superior performance compared to word-based and character-based LSTM-CRF across different domains. The lattice method is fully independent of word segmentation, yet more effective in using word information thanks to the freedom of choosing lexicon words in a context for NER disambiguation.

Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *HLT-NAACL*. pages 52–62.

Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. 2006a. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the*

Fifth SIGHAN Workshop on Chinese Language Processing. pages 173–176.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006b. Chinese named entity recognition with conditional random fields. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. pages 118–121.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*. Lisbon, Portugal, pages 1197–1206. http://aclweb.org/anthology/D15-1141.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for Chinese word segmentation. In *ACL*. volume 1, pages 1193–1203.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL* 4:357–370. https://transacl.org/ojs/index.php/tacl/article/view/792.

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

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In *International Conference on Computer Processing of Oriental Languages*. Springer, pages 239–250.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*. page 25.

Jianfeng Gao, Mu Li, Chang-Ning Huang, and Andi Wu. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics* 31(4):531–574.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *HLT-NAACL 2003-Volume 4*. pages 172–175.

Hangfeng He and Xu Sun. 2017a. F-score driven max margin neural network for named entity recognition in Chinese social media. In *EACL*. volume 2, pages 713–718.

Hangfeng He and Xu Sun. 2017b. A unified model for cross-domain and semi-supervised named entity recognition in Chinese social media. In *AAAI*. pages 3216–3222.

Jingzhou He and Houfeng Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Shen Huang, Xu Sun, and Houfeng Wang. 2017. Addressing domain adaptation for chinese word segmentation with global recurrent structure. In *IJCNLP*. volume 1, pages 184–193.

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

Wenbin Jiang, Meng Sun, Yajuan Lü, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *ACL*. volume 1, pages 761–769.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*. pages 260–270.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4):541–551.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. pages 108–117.

Haibo Li, Masato Hagiwara, Qi Li, and Heng Ji. 2014. Comparison of the impact of word segmentation on name tagging for Chinese and japanese. In *LREC*. pages 2532–2536.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *AAAI* .

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. *Proceedings of COLING 2012: Posters* pages 745–754.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based Chinese word segmentation using free annotations. In *EMNLP*. pages 864–874.

Zhangxun Liu, Conghui Zhu, and Tiejun Zhao. 2010. Chinese named entity recognition with a sequence labeling approach: based on characters, or based on words? In *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*, Springer, pages 634–640.

Yanan Lu, Yue Zhang, and Dong-Hong Ji. 2016. Multi-prototype Chinese character embedding. In *LREC*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *EMNLP*. pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. In *ACL*. volume 1, pages 1064–1074.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*. pages 78–86.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *EMNLP*. pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for Chinese social media with word segmentation representation learning. In *ACL*. volume 2, pages 149–155.

Nanyun Peng and Mark Dredze. 2017. Supplementary results for named entity recognition on chinese social media with an updated dataset .

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL* 5:101–115.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*. volume 1, pages 1756–1765.

Likun Qiu and Yue Zhang. 2015. Word segmentation for chinese novels. In *AAAI*. pages 2440–2446.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*. pages 147–155.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*. volume 1, pages 2121–2130.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *EMNLP*. pages 1380–1389. https://www.aclweb.org/anthology/D17-1145.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI*. pages 3302–3308.

Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E. Shi, and Silvio Savarese. 2017. Lattice long short-term memory for human action recognition. In *ICCV*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*. Beijing, China, pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *AAAI*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium* .

Y Xu, Y Wang, T Liu, J Liu, Y Fan, Y Qian, J Tsujii, and EI Chang. 2014. Joint segmentation and named entity recognition using dual decomposition in Chinese discharge summaries. *JAMIA* 21(e1):e84–92.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering* 11(02):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *International Conference on Computational Linguistics and Intelligent Text Processing*.

Jie Yang, Yue Zhang, and Fei Dong. 2017a. Neural word segmentation with rich pretraining. In *ACL*. Vancouver, Canada, pages 839–849. http://aclweb.org/anthology/P17-1078.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017b. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. pages 158–161.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese Journal of Electronics* 22(2):225–230.

Nugget Proposal Networks for Chinese Event Detection

Hongyu Lin[1,2], **Yaojie Lu**[1,2], **Xianpei Han**[1], **Le Sun**[1]
[1]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
`{hongyu2016,yaojie2017,xianpei,sunle}@iscas.ac.cn`

Abstract

Neural network based models commonly regard event detection as a word-wise classification task, which suffer from the mismatch problem between words and event triggers, especially in languages without natural word delimiters such as Chinese. In this paper, we propose Nugget Proposal Networks (NPNs), which can solve the word-trigger mismatch problem by directly proposing entire trigger nuggets centered at each character regardless of word boundaries. Specifically, NPNs perform event detection in a character-wise paradigm, where a hybrid representation for each character is first learned to capture both structural and semantic information from both characters and words. Then based on learned representations, trigger nuggets are proposed and categorized by exploiting character compositional structures of Chinese event triggers. Experiments on both ACE2005 and TAC KBP 2017 datasets show that NPNs significantly outperform the state-of-the-art methods.

1 Introduction

Automatic event extraction is a fundamental task of information extraction. Event detection, which aims to identify event triggers of specific types, is a key step of event extraction. For example, from the sentence "*Henry was injured, and then passed away soon*", an event detection system should detect an "*Injure*" event triggered by "*injured*", and a "*Die*" event triggered by "*passed away*".

Recently, neural network methods, which transform event detection into a word-wise classification paradigm, have achieved significant progress in event detection (Nguyen and Grishman, 2015; Chen et al., 2015b; Ghaeini et al., 2016). For instance, a model will detect events in sentence "*Henry was injured*" by successively classifying its three words into *NIL*, *NIL* and *Injure*. By automatically extracting features from raw texts, these methods rely little on prior knowledge and achieved promising results.

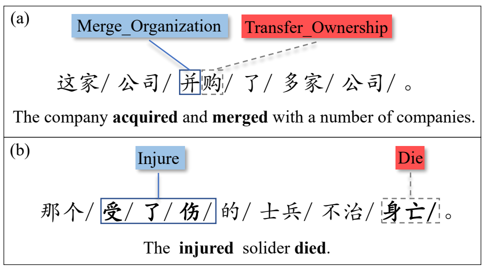

(a)
Merge_Organization
Transfer_Ownership
这家/ 公司/ 并购/ 了/ 多家/ 公司/ 。
The company **acquired** and **merged** with a number of companies.
(b)
Injure
Die
那个/ 受/ 了/ 伤/ 的/ 士兵/ 不治/ 身亡/ 。
The **injured** solider **died**.

Figure 1: Examples of word-trigger mismatch. Slashes in the figure indicate word boundaries.

Unfortunately, word-wise event detection models suffer from the word-trigger mismatch problem, because a number of triggers do not exactly match with a word. Specifically, a trigger can be part of a word or cross multiple words, which is impossible to detect using word-wise models. This problem is more severe in languages without natural word delimiters such as Chinese. Figure 1 (a) shows several examples of part-of-word triggers, where two characters in one word "并购"(acquire and merge) trigger two different events: a "*Merge_Org*" event triggered by "并"(merge) and a "*Transfer_Ownership*" event triggered by "购" (acquire). Figure 1 (b) shows a multi-word trigger, where three words "受"(is), "了" and "伤"(injured) trigger an *Injure* event together. Table 1 shows the statistics of different types of word-trigger match on two standard datasets. We can see that word-trigger mismatch is crucial for Chinese event detection since nearly 25% of triggers in RichERE and 15% of them in ACE2005 dataset don't exactly match with a word.

To resolve the word-trigger mismatch problem,

Match Type	Rich ERE	ACE2005
Exact Match	75.52%	85.39%
Part of Word	19.55%	11.67%
Cross words	4.93%	2.94%

Table 1: Percentages of different types of matches between words and triggers.

this paper proposes *Nugget Proposal Networks (NPNs)*, which identify triggers by modeling character compositional structures of trigger nuggets regardless of word boundaries. Given a sentence, NPNs regard characters as basic detecting units and are able to 1) directly propose the entire potential trigger nugget at each character by exploiting inner compositional structure of triggers; 2) effectively categorize proposed triggers by learning semantic representation from both characters and words. For example, at character "伤"(injured) in Figure 1 (b), NPNs are not only capable to detect it is part of an *Injure* event trigger, but also can propose the entire trigger nugget "受了伤"(is injured). The main idea behind NPNs is that most Chinese triggers have regular character compositional structure (Li et al., 2012). Concretely, most of Chinese event triggers have one central character which can indicate its event type, e.g. "杀"(kill) in "枪杀"(kill by shooting). Furthermore, characters are composed into a trigger based on regular compositional structures, e.g. "manner + verb" for "枪杀"(kill by shooting), "砍杀"(hack to death), as well as "verb + auxiliary + noun" for "受了伤"(is injured) and "挨了打"(beaten).

Figure 2 shows the architecture of NPNs. Given a character in sentence, a hybrid representation learning module is first used to learn its semantic representation from both characters and words in the sentence. This hybrid representation is then fed into two modules: one is *trigger nugget generator*, which proposes the entire potential trigger nugget by exploiting inner character compositional structure. Once a trigger is proposed, an *event type classifier* is applied to determine its event type. Compared with previous methods, NPNs mainly have following advantages:

1) **By directly proposing the entire trigger nugget centered at a character, trigger nugget generator can effectively resolve the word-trigger mismatch problem.** First, using characters as basic units, NPNs will not suffer from the word-trigger mismatch problem of word-wise methods. Furthermore, by modeling and exploiting character compositional structure

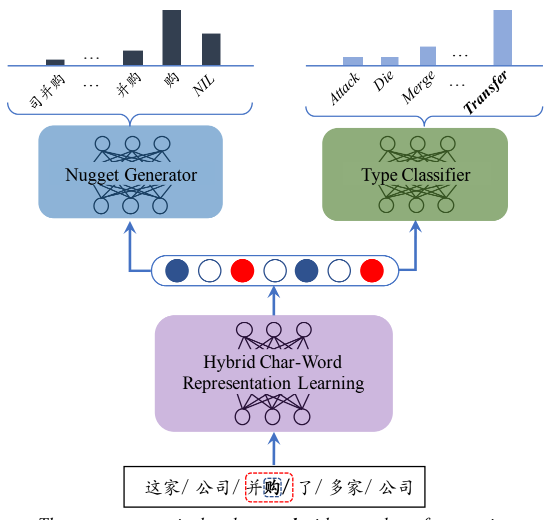

司并购
…
并购
购
NIL
Attack
Die
Merge
…
Transfer
Nugget Generator
Type Classifier
Hybrid Char-Word Representation Learning
这家/ 公司/ 并购/ 了/ 多家/ 公司
The company acquired and **merged** with a number of companies.

Figure 2: The overall architecture of Nugget Proposal Networks. The concerning character is "购".

of triggers, our model is more error-tolerant to character-wise classification errors than traditional character-based models, as shown in Section 4.4.

2) **By summarizing information from both characters and words, our hybrid representation can effectively capture information for both inner character composition and accurate event categorization.** For example, the inner compositional structure of trigger "枪杀"(kill by shooting) can be learned from the character-level sequence. Besides, characters are often ambiguous, therefore the accurate representations must take their word context into consideration. For example, the representation "杀"(kill) in " 枪杀"(kill by shooting) should be different from its representation in "杀青"(completed).

We conducted experiments on both the ACE2005 and the TAC KBP 2017 Event Nugget Detection datasets. Experiment results show that NPNs can effectively solve the word-mismatch problem, and therefore significantly outperform previous state-of-the-art methods[1].

2 Hybrid Representation Learning

Given a sentence, NPNs will first learn a representation for each character, then the representation is fed into downstream modules. We observe that both characters and words contain rich information for Chinese event detection: characters reveals the inner compositional structure of event

[1]Our source code, including all hyper-parameter settings and pre-trained word embeddings, is openly available at `github.com/sanmusunrise/NPNs`.

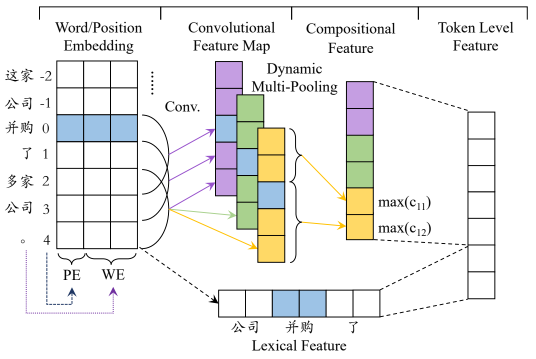


Figure 3: Token-level feature extractor, where PE is relative positional embeddings and WE is word embeddings. The concerning token is “并购”.

triggers (Li et al., 2012), while words can provide more accurate and less ambiguous semantics than characters (Chen et al., 2015a). For example, character-level information can tell us that “枪杀”(kill by shooting) is a trigger constructed of regular pattern “manner + verb”. While word-level sequences can provide more explicit information when we distinguish the semantics of “杀”(kill) in this context with that character in other words like “杀青”(completed).

Therefore, we propose to learn a hybrid representation which can summarize information from both characters and words. Specifically, we first learn two separate character-level and word-level representations using token-level neural networks. Then we design three kinds of hybrid paradigms to obtain the hybrid representation.

2.1 Token-level Representation Learning

Two token-level neural networks are used to extract features from characters and words respectively. The network architecture is similar to DMCNN (Chen et al., 2015b). Figure 3 shows a word-level example. Given n tokens $t_1, t_2, ..., t_n$ in the sentence and the concerning token t_c, let $\mathbf{x}_i$ be the concatenation of the word embedding of t_i and the embedding of t_i's relative position to t_c, a convolutional layer with window size as h is introduced to capture compositional semantics:

$$r_{ij} = \tanh(\mathbf{w}_i \cdot \mathbf{x}_{j:j+h-1} + b_i) \quad (1)$$

Here $\mathbf{x}_{i:i+j}$ refers to the concatenation of embeddings from $\mathbf{x}_i$ to $\mathbf{x}_{i+j}$, $\mathbf{w}_i$ is the i-th filter of the convolutional layer, $b_i \in R$ is a bias term. Then a dynamic multi-pooling layer is applied to preserve important signals of different parts of the sentence:

$$r_i^{left} = \max_{j<c} r_{ij}, \quad r_i^{right} = \max_{j \geq c} r_{ij} \quad (2)$$

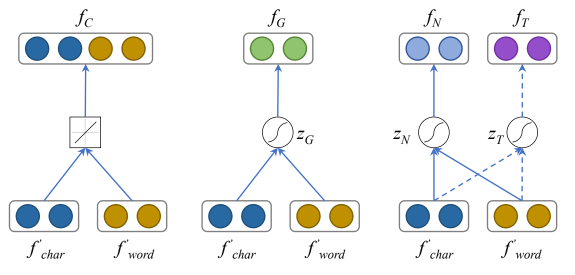


Figure 4: Three hybrid representation learning methods.

After that we concatenate r_i^{left} and r_i^{right} from all feature maps, as well as the embeddings of tokens nearing to t_c to obtain the word-level representation $\mathbf{f_{word}}$ of t_c. Using the same procedure to character sequences, we can obtain the character-level representation $\mathbf{f_{char}}$.

2.2 Hybrid Representation Learning

So far we have both character-level feature representation $\mathbf{f_{char}}$ and word-level feature representation $\mathbf{f_{word}}$. This section describes how we mix them up to obtain a hybrid representation. Before this, we first project $\mathbf{f_{char}}$ and $\mathbf{f_{word}}$ respectively into the same vector space using two dense layers, and we represent the projected d'-dimensional vectors as $\mathbf{f'_{char}}$ and $\mathbf{f'_{word}}$. Then we design three different paradigms to mix them up: Concat Hybrid, General Hybrid and Task-specific Hybrid, as illustrated in Figure 4.

Concat Hybrid is the most simple method, which simply concatenates character-level and word-level representations:

$$\mathbf{f_C} = \mathbf{f'_{char}} \oplus \mathbf{f'_{word}} \quad (3)$$

This simple approach doesn't introduce any additional parameter, but we find it very effective in our experiments.

General Hybrid aims to learn a shared hybrid representation for both trigger nugget proposal and event type classification. Specifically, we design a gated structure to model the information flow from $\mathbf{f'_{char}}$ and $\mathbf{f'_{word}}$ to the general hybrid feature representation $\mathbf{f_G}$:

$$\mathbf{z_G} = s(\mathbf{W_{GH}}\mathbf{f'_{char}} + \mathbf{U_{GH}}\mathbf{f'_{word}} + \mathbf{b_{GH}}) \quad (4)$$

$$\mathbf{f_G} = \mathbf{z_G}\mathbf{f'_{char}} + (\mathbf{1} - \mathbf{z_G})\mathbf{f'_{word}} \quad (5)$$

Here s is the sigmoid function, $\mathbf{W_{GH}} \in R^{d' \times d'}$ and $\mathbf{U_{GH}} \in R^{d' \times d'}$ are weight matrix, and

$\mathbf{b_{GH}} \in R^{d'}$ is the bias term. $\mathbf{z_G}$ is a d'-dimensional vector whose values represent the contribution of $\mathbf{f}'_{\mathbf{char}}$ and $\mathbf{f}'_{\mathbf{word}}$ to the final hybrid representation, which models the importance of individual features in the given contexts.

As two downstream modules of NPNs have individual functions, they might hold different requirements to the input features. Intuitively, trigger nugget generator depends more on fine-grained character-level features. In contrast, word-level features might play more important roles in the event type classifier since it is enriched with more explicit semantics. As a result, a unified representation may be insufficient and it is better to learn task-specific hybrid representations.

Task-specific Hybrid is proposed to tackle this problem, where two gates are introduced for two modules respectively. Formally, we learn one representation for the trigger nugget generator and one for event type classifier as:

$$\mathbf{z_N} = s(\mathbf{W_N f'_{char}} + \mathbf{U_N f'_{word}} + \mathbf{b_N}) \tag{6}$$

$$\mathbf{z_T} = s(\mathbf{W_T f'_{char}} + \mathbf{U_T f'_{word}} + \mathbf{b_T}) \tag{7}$$

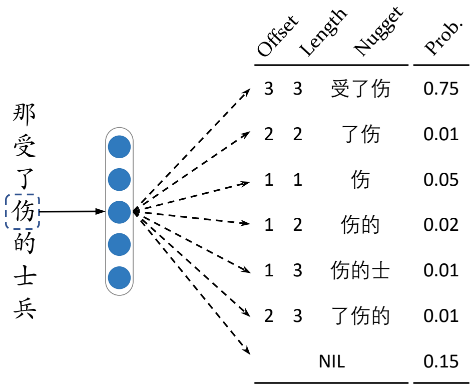
$$\mathbf{f_N} = \mathbf{z_N f'_{char}} + (\mathbf{1} - \mathbf{z_N})\mathbf{f'_{word}} \tag{8}$$

$$\mathbf{f_T} = \mathbf{z_T f'_{char}} + (\mathbf{1} - \mathbf{z_T})\mathbf{f'_{word}} \tag{9}$$

Here $\mathbf{f_N}$ and $\mathbf{f_T}$ are hybrid features for the trigger nugget generator and the event type classifier respectively and the meanings of other parameters are similar to the ones in Equation (4) and (5).

3 Nugget Proposal Networks

Given the hybrid representation of a character in a sentence, the goal of NPNs is to propose the potential trigger nugget, as well as to identify its corresponding event type at each character. For example in Figure 5, centered at the character "伤"(injured), NPNs need to propose "受了伤"(is injured) as the entire trigger nugget and identify its event type as "*Injure*". For this, NPNs are equipped with two modules: one is called *trigger nugget generator*, which is used to propose the potential trigger nugget containing the concerning character by exploiting character compositional structures of triggers. Another module, named as *event type classifier*, is used to determine the specific type of this event once a trigger nugget is detected.



Offset	Length	Nugget	Prob.
3	3	受了伤	0.75
2	2	了伤	0.01
1	1	伤	0.05
1	2	伤的	0.02
1	3	伤的士	0.01
2	3	了伤的	0.01
		NIL	0.15

Figure 5: Our trigger nugget generator. For each character, there are 7 candidate nuggets including "NIL" if the maximum length of nuggets is 3.

3.1 Trigger Nugget Generator

Chinese event triggers have regular inner compositional structures, e.g. "受了伤"(is injured) and "挨了打"(is beaten) have the same "verb + auxiliary + noun" structure, and "枪杀"(kill by shooting) and "射杀"(kill by shooting) share the same "manner + verb" pattern. If a model is able to learn this compositional structure regularity, it can effectively detect trigger nuggets at characters. Recent advances have presented that convolutional neural networks are effective at capturing and predicting the region information in object detection (Ren et al., 2015) and semantic segmentation (He et al., 2017), which reveals the strong ability of CNNs to learning spatial and positional information. Inspired by this, we propose a neural network based trigger nugget generator, which is expected to not only be able to predict whether a character belongs to a trigger nugget, but also can point out the entire trigger nugget.

Figure 5 is an illustration of our trigger nugget generator. Hybrid representation $\mathbf{f_N}$ for concerning character is first learned as described in Section 2, which is then fed into a fully-connected layer to compute the scores for different possible trigger nuggets containing that character:

$$\mathbf{O^G} = \mathbf{W_G f_N} + \mathbf{b_G} \tag{10}$$

where $\mathbf{O^G} \in R^{d^N}$ and d^N is the amount of candidate nuggets plus one *"NIL"* label indicating this character doesn't belong to an trigger. Given the maximum length L of trigger nuggets, there are $\frac{L^2+L}{2}$ possible nuggets containing a specific character, as we shown in Figure 5. In both ACE and Rich ERE corpus, more than 98.5% triggers contain no more than 3 characters, so for a specific character we consider 6 candidate nuggets and thus $d^N = 7$. We expect NPNs to give a high score to a nugget if it follows a regular compositional structure of triggers. For example in Figure 5, "受了伤"(is injured) follows the compositional pattern of "verb + auxiliary + noun", therefore a high score is given to the category where "伤" is at the 3rd place of a nugget with a length of 3. By contrast "了伤" does not match a regular pattern, then the score for "伤" at the 2nd place of a nugget with a length of 2 will be low in this context.

After obtaining the scores for each nugget, a softmax layer is applied to normalize the scores:

$$P(y_i^G|x;\theta) = \frac{e^{O_i^G}}{\sum_{j=1}^{d^N} e^{O_j^G}} \tag{11}$$

where O_i^G is the i-the element in $\mathbf{O^G}$ and θ is the model parameters.

3.2 Event Type Classifier

The event type classifier aims to identify whether the given character in the given context will exhibit an event type. Once we detect an event trigger nugget at one character, the hybrid feature $\mathbf{f_T}$ extracted previously is then feed into a neural network classifier, which further determines the specific type of this trigger. Following previous work (Chen and Ng, 2012), our event type classifier directly classifies nuggets into event subtypes, while ignores the hierarchy between event types.

Formally, given the hybrid feature vector $\mathbf{f_T}$ of input x, a fully-connected layer is applied to compute its scores assigned to each event subtype:

$$\mathbf{O^C} = \mathbf{W_C}\mathbf{f_T} + \mathbf{b_C} \tag{12}$$

where $\mathbf{O^C} \in R^{d^T}$ and d^T is the number of event subtypes. Then similar to the trigger nugget generator, a softmax layer is introduced:

$$P(y_i^C|x;\theta) = \frac{e^{O_i^C}}{\sum_{j=1}^{d^T} e^{O_j^C}} \tag{13}$$

where O_i^C is the i-th element in $\mathbf{O^C}$, representing the score for i-th subtype.

3.3 Dealing with Conflicts between Proposed Nuggets

While NPNs directly propose nugget at each character, there might exists conflicts between proposed nuggets at different characters. Generally speaking, there are two types of conflicts: (i) NIL/trigger conflict, which means NPNs propose a trigger nugget at one character, but classify other character in that nugget into "NIL" (e.g., proposing nugget "受了伤"(is injured) at "受" and output "NIL" at "了"); (ii) overlapped conflict, i.e., proposing two overlapped nuggets (e.g., proposing nugget "受了伤"(is injured) at "受" and nugget "伤" at "伤"). But we find that overlapped conflict is very rare because NPNs is very effective in capturing positional knowledge and the main challenge of event detection is to distinguish triggers from non-triggers.

Therefore in this paper, we employ a redundant prediction strategy by simply adding all proposed nuggets into results and ignoring "NIL" predictions. For example, if NPNs successively propose "受了伤"(is injured), "NIL", " 伤" from "受了伤", then we will ignore the "NIL" and add both two other nuggets into result. We found such a redundant prediction paradigm is an advantage of our model. Compared with conventional character-based models, even NPNs mistakenly classified character "了" into "NIL", we can still accurately detect trigger "受了伤"(is injured) if we can predict the entire nugget at character "受" or "伤". This redundant prediction makes our model more error-tolerant to character-wise classification errors, as verified in Section 4.4.

3.4 Model Learning

To train the trigger nugget generator, we regard all characters included in trigger nuggets as positive training instances, and randomly sample characters not in any trigger as negative instances and label them as "NIL". Suppose we have T^G training examples in $S^G = \{(x_k, y_k^G)|k = 1, 2, ...T^G\}$ to train the trigger nugget generator, as well as T^C examples in $S^C = \{(x_k, y_k^C)|k = 1, 2, ...T^C\}$ to train the event type classifier, we can define the loss function $\mathcal{L}(\theta)$ as follow:

$$\begin{aligned}\mathcal{L}(\theta) = &- \sum_{(x_k, y_k^G)\in S^G} \log P(y_k^G|x_k;\theta) \\ &- \sum_{(x_k, y_k^C)\in S^C} \log P(y_k^C|x_k;\theta)\end{aligned} \tag{14}$$

where θ is parameters in NPNs. Since all modules in NPNs are differentiable, any gradient-based algorithms can be applied to minimize $\mathcal{L}(\theta)$.

4 Experiments

4.1 Data Preparation and Evaluation

We conducted experiments on two standard datasets: ACE2005 and TAC KBP 2017 Even-

Model	ACE2005						KBPEval2017					
	Trigger Identification			Trigger Classification			Trigger Identification			Trigger Classification		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
FBRNN(Char)	61.3	45.6	52.3	57.5	42.8	49.1	57.97	36.92	45.11	51.71	32.94	40.24
DMCNN(Char)	60.1	61.6	60.9	57.1	58.5	57.8	53.67	49.92	51.73	50.03	46.53	48.22
C-BiLSTM*	65.6	66.7	66.1	60.0	60.9	60.4	-	-	-	-	-	-
FBRNN(Word)	64.1	63.7	63.9	59.9	59.6	59.7	65.10	46.86	54.50	60.05	43.22	50.27
DMCNN(Word)	66.6	63.6	65.1	61.6	58.8	60.2	60.43	51.64	55.69	54.81	46.84	50.51
HNN*	74.2	63.1	68.2	**77.1**	53.1	63.0	-	-	-	-	-	-
Rich-C*	62.2	71.9	66.7	58.9	68.1	63.2	-	-	-	-	-	-
KBP2017 Best*	-	-	-	-	-	-	**67.76**	45.92	54.74	**62.69**	42.48	50.64
NPN(Concat)	**76.5**	59.8	67.1	72.8	56.9	63.9	64.58	50.31	56.56	59.14	46.07	51.80
NPN(General)	71.5	63.2	67.1	67.3	59.6	63.2	63.67	51.32	56.83	57.78	46.58	51.57
NPN(Task-specific)	64.8	**73.8**	**69.0**	60.9	**69.3**	**64.8**	64.32	**53.16**	**58.21**	57.63	**47.63**	**52.15**

Table 2: Experiment results on ACE2005 and KBPEval2017. * indicates the result adapted from the original paper. For KBPEval2017, "Trigger Identification" corresponds to the "Span" metric and "Trigger Classification" corresponds to the "Type" metric reported in official evaluation.

t Nugget Detection Evaluation (KBPEval2017) datasets. For ACE2005 (LDC2006T06), we used the same setup as Chen and Ji (2009), Feng et al. (2016) and Zeng et al. (2016), in which 569/64/64 documents are used as training/development/test set. For KBPEval2017, we evaluated our model on the 2017 Chinese evaluation dataset(LDC2017E55), using previous RichERE annotated Chinese datasets (LDC2015E78, LDC2015E105, LDC2015E112, and LDC2017E02) as the training set except 20 randomly sampled documents reserved as development set. Finally, there were 506/20/167 documents for training/development/test set. We used Stanford CoreNLP toolkit (Manning et al., 2014) to preprocess all documents for sentence splitting and word segmentation. Adadelta update rule (Zeiler, 2012) is applied for optimization.

Models are evaluated by micro-averaged Precision(P), Recall(R) and F1-score. For ACE2005, we followed Chen and Ji (2009) to compute the above measures. For KBPEval2017, we used the official evaluation toolkit [2] to obtain these metrics.

4.2 Baselines

Three groups of baselines were compared:

Character-based NN models. This group of methods solve Chinese Event Detection in a character-level sequential labeling paradigm, which include Convolutional Bi-LSTM model (C-BiLSTM) proposed by Zeng et al. (2016), Forward-backward Recurrent Neural Networks (FBRNN) by Ghaeini et al. (2016), and a character-level DMCNN model with a classifier using IOB encoding (Sang and Veenstra, 1999).

Word-based NN models. This group of methods directly adopt currently NN models into word-level sequences, which includes word-based FBRNN, word-based DMCNN and Hybrid Neural Network proposed by Feng et al. (2016), which incorporates CNN with Bi-LSTM and achieves the SOTA NN based result on ACE2005. To alleviate OOV problem stemming from word-trigger mismatch, we also adopt errata table replacing (Han et al., 2017), which introduce an errata table extracted from the training data and replace those words that part of whom was a trigger nugget with that trigger directly.

Feature-enriched Methods. This group of methods includes Rich-C (Chen and Ng, 2012) and CLUZH (KBP2017 Best) (Makarov and Clematide, 2017). Rich-C developed several handcraft Chinese-specific features, which is one of the state-of-the-art on ACE2005. CLUZH incorporated many heuristic features into LSTM encoder, which achieved the best performance in TAC KBP2017 evaluation.

4.3 Overall Results

Table 2 shows the results on ACE2005 and KBPEval2017. From this table, we can see that:

1) **NPNs steadily outperform all baselines significantly.** Compared with baselines, NPN(Task-specific) gains at least 1.6 (2.5%) and 1.5 (3.0%) F1-score improvements on trigger classification task on ACE2005 and KBPEval2017 respectively.

[2] github.com/hunterhector/EvmEval/tarball/master

2) **By exploiting compositional structures of triggers, our trigger nugget generator can effectively resolve the word-trigger mismatch problem.** As shown in Table 2, NPN(Task-specific) achieved significant F1-score improvements on trigger identification task on both datasets. It is notable that our method achieved a remarkable high recall on both datasets, which indicates that NPNs do detect a number of triggers which previous methods can not identify.

3) **By summarizing information from both characters and words, the hybrid representation learning is effective for event detection.** Comparing with corresponding character-based methods[3], word-based methods achieved 2 to 3 F1-score improvements, which indicates that words can provide additional information for event detection. By combining character-level and word-level features, NPNs are able to perform character-based event detection meanwhile take word-level knowledge into consideration too.

4.4 Comparing with Conventional Character-based Methods

To further investigate the effects of the trigger nugget generator, we compared NPNs with other character-based methods and analyzed behaviors of them. We conducted a supplementary experiment by replacing our trigger nugget generator and event type classifier with an IOB encoding labeling layer. We call this system NPN(IOB). Besides, we also compared the result with F-BRNN(Char), which proposes candidate trigger nuggets according to an external trigger table.

Model	P	R	F1
FBRNN(Char)	57.97	36.92	45.11
NPN(IOB)	60.96	47.39	53.32
NPN(Task-specific)	**64.32**	**53.16**	**58.21**

Table 3: Performances of character-based methods on KBP2017Eval Trigger Identification task.

Table 3 shows the results on KBP2017Eval. We can see that NPN(Task-specific) outperforms other methods significantly. We believe this is because:

1) FBRNN(Char) only regards tokens in the candidate table as potential trigger nuggets, which limits the choice of possible trigger nuggets and results in a very low recall rate.

2) To accurately identify a trigger, NPN(IOB) and conventional character-based methods require all characters in a trigger being classified correctly, which is very challenging (Zeng et al., 2016): many characters appear in a trigger nugget will not serve as a part of a trigger nugget in the majority of contexts, thus they will be easily classified into “NIL”. For the first example in Table 5, NPN(IOB) was unable to fully recognize the trigger nugget “贺电”(congratulatory message) because character “贺”(congratulatory) doesn’t often serve as part of ”PhoneWrite” trigger. In fact, “贺” serves as a “NIL” in the majority of similar contexts, e.g., “贺喜”(congratulation) and “祝贺”(congratulation).

3) NPNs are able to handle above problems. First, NPNs doesn’t rely on candidate tables to generate potential triggers, which guarantees a good generalization ability. Second, NPNs propose the entire trigger nugget at each character, such a redundant prediction paradigm makes NPNs more error-tolerant to character-level errors. For example, even might mistakenly classify “贺” into “NIL”, NPNs can still identify the correct nugget “贺电” at character “电” because “电” is a common part of “PhoneWrite” event trigger.

4.5 Influence of Word-Trigger Mismatch

This subsection investigates the effects of resolving the word-trigger mismatch problem using different methods. According to different types of word-trigger match, we split KBP2017Eval test set into three parts: Exact, Part-of-Word, Cross-Words, which are as defined in Table 1.

Model	Exact	Part	Cross
NPN(IOB)	48.65	29.13	8.54
DMCNN(Word)	57.36	23.28	0.00
- w/o Errata replacing	59.03	0.00	0.00
NPN(Task-specific)	56.47	42.66	26.58

Table 4: Recall rates on three word-trigger match splits on KBP2017Eval Trigger Identification task.

Table 4 shows the recall of different methods on each split. NPN(Task-specific) significantly outperform other baselines when trigger-word mismatch exists. This verified that NPNs can resolve different cases of word-trigger mismatch problems robustly, meanwhile retain high performance on exact match cases. In contrast, NPN(IOB) can not

[3]C-BiLSTM and HNN are similar methods to some extent. They both use a hybrid representation from CNN and BiLSTM encoders.

Sentence	DMCNN	NPN(IOB)	NPN	Correct
贺电/全文/如下,... Full **congratulatory message**:...	(贺电,PhoneWrite)	(电,PhoneWrite)	(贺电,PhoneWrite)	(贺电,PhoneWrite)
死伤/的/所有/士兵... all soldiers **died and injured**...	None	(死,Die) (伤,Injure)	(死,Die) (伤,Injure)	(死,Die) (伤,Injure)

Table 5: System prediction examples. (X,Y) indicates a trigger nugget X is annotated with event type Y.

exactly detect boundaries of trigger nuggets, thus has a low recall on all splits. Conventional DMCNN regards words as potential triggers, which means it can only identify triggers that exactly match with words. As the second example in Table 5, word "死伤"(dead or injured) as a whole has never been annotated as a trigger, so DMCNN is unable to recognize it at all. Errata replacing can only solve some of the part-of-word mismatch problem, but it can not handle the cases where one word contains multiple triggers(e.g., "死伤" in Table 5) and the cases that a trigger crosses multiple words.

4.6 Effects of Hybrid Representation

This section analyzed the effect of feature hybrid in NPNs. First, from Table 2, we can see that Task-specific Hybrid method achieved the best performance in both datasets. Surprisingly, simple Concat Hybrid outperforms the General Hybrid approach. We believe this is because the trigger nugget generator and the event type classifier rely on different information, and therefore using one unified gate is not enough. And Task-specific Hybrid uses two different task-specific gates which can satisfy both sides, thus resulting in the best overall performance.

Furthermore, to investigate the necessary of using hybrid features, an auxiliary experiment, called NPN(Char), was conducted by removing word-level features from NPNs. Also, we compared with the model removing character-level features, which is the original DMCNN(Word).

Model	P	R	F1
DMCNN(Word)	54.81	46.84	50.51
NPN(Char)	56.19	43.88	49.28
NPN(Task-specific)	**57.63**	**47.63**	**52.15**

Table 6: Results of using different representation on Trigger Classification task on KBP2017Eval.

Table 6 shows the experiment results. We can see that neither character-level or word-level representation can achieve competitive results with the NPNs. This verified the necessity of hybrid representation. Besides, we can see that NPN(Char) outperforms other character-level methods in Table 2, which further confirms that our trigger nugget generator is still effective even only using character-level information.

5 Related Work

Event detection is an important task in information extraction and has attracted many attentions. Traditional methods (Ji and Grishman, 2008; Patwardhan and Riloff, 2009; Liao et al., 2010; McClosky et al., 2011; Hong et al., 2011; Huang and Riloff, 2012; Li et al., 2013a,b, 2014) rely heavily on hand-craft features, which are hard to transfer among languages and annotation standards.

Recently, deep learning methods, which automatically extract high-level features and perform token-level classification with neural networks (Chen et al., 2015b; Nguyen and Grishman, 2015), have achieved significant progress. Some improvements have been made by jointly predicting triggers and arguments (Nguyen et al., 2016) and introducing more complicated architectures to capture larger scale of contexts (Feng et al., 2016; Nguyen and Grishman, 2016; Ghaeini et al., 2016). These methods have achieved promising results in English event detection.

Unfortunately, the word-trigger mismatch problem significantly undermines the performance of word-level models in Chinese event detection (Chen and Ji, 2009). To resolve this problem, Chen and Ji (2009) proposed a feature-driven BIO tagging methods at character-level sequences. Qin et al. (2010) introduced a method which can automatically expand candidate Chinese trigger set. While Li et al. (2012) and Li and Zhou (2012) defined manually character compositional patterns for Chinese event triggers. However, their methods rely on hand-crafted features and patterns, which make them difficult to be integrated into recent Deep Learning models.

Recent advances have shown that neural networks can effectively capture spatial and positional information from raw inputs (Ren et al., 2015; He et al., 2017; Wang and Jiang, 2017).

This paper designs Nugget Proposal Networks to capture character compositional structure of event triggers, which is more robust and more effective than previous hand-crafted patterns or character-level sequential labeling methods.

6 Conclusions and Future Work

This paper proposes Nugget Proposal Networks for Chinese event detection, which can effectively resolve the word-trigger mismatch problem by modeling and exploiting character compositional structure of Chinese event triggers, using hybrid representation which can summarize information from both characters and words. Experiment results have shown that our method significantly outperforms conventional methods.

Because the mismatch between words and extraction units is a common problem in information extraction, we believe our method can also be applied to many other languages and tasks for exploiting inner composition structure during extraction, such as Named Entity Recognition.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505, and the Young Elite Scientists Sponsorship Program no. YESS20160177. Moreover, we sincerely thank all reviewers for their valuable comments.

References

Chen Chen and Vincent Ng. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *Proceedings of COLING 2012*.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015a. Joint learning of character and word embeddings. In *Proceedings of IJCAI 2015*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015b. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL 2015*.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in chinese event extraction. In *Proceedings of NAACL-HLT 2009*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of ACL 2016*.

Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of ACL 2016*.

Xianpei Han, Xiliang Song, Hongyu Lin, Qichen Zhu, Yaojie Lu, Le Sun, Jingfang Xu, Mingrong Liu, Ranxu Su, Sheng Shang, Chenwei Ran, and Feifei Xu. 2017. ISCAS_Sogou at TAC-KBP 2017. In *Proceedings of TAC 2017*.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. *arXiv preprint arXiv:1703.06870*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL-HLT 2011*.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of AAAI 2012*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL 2008*.

Peifeng Li and Guodong Zhou. 2012. Employing morphological structures and sememes for chinese event extraction. In *Proceedings of COLING 2012*.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in chinese event extraction. In *Proceedings of EMNLP-CoNLL 2012*.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013a. Argument inference from relevant event mentions in chinese argument extraction. In *Proceedings of ACL 2013*.

Qi Li, Heng Ji, Yu HONG, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of EMNLP 2014*.

Qi Li, Heng Ji, and Liang Huang. 2013b. Joint event extraction via structured prediction with global features. In *Proceedings of ACL 2013*.

Shasha Liao, New York, Ralph Grishman, and New York. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL 2010*.

Peter Makarov and Simon Clematide. 2017. UZH at TAC KBP 2017: Event nugget detection via joint learning with softmax-margin objective. In *Proceedings of TAC 2017*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL 2014*.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL-HLT 2011*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT 2016*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL 2015*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of EMNLP 2016*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of EMNLP 2009*.

Bing Qin, Yanyan Zhao, Xiao Ding, Ting Liu, and Guofu Zhai. 2010. Event type recognition based on trigger expansion. *Tsinghua Science and Technology*, 15(3):251–258.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of NIPS 2015*.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL 1999*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *Proceedings of ICLR 2017*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Ying Zeng, Honghui Yang, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2016. A convolution bilstm neural network model for chinese event extraction. In *Proceedings of NLPCC-ICCPOL 2016*.

Higher-order Relation Schema Induction using Tensor Factorization with Back-off and Aggregation

Madhav Nimishakavi
Indian Institute of Science
Bangalore
madhav@iisc.ac.in

Manish Gupta
Microsoft
Hyderabad
manishg@microsoft.com

Partha Talukdar
Indian Institute of Science
Bangalore
ppt@iisc.ac.in

Abstract

Relation Schema Induction (RSI) is the problem of identifying type signatures of arguments of relations from unlabeled text. Most of the previous work in this area have focused only on *binary* RSI, i.e., inducing only the subject and object type signatures per relation. However, in practice, many relations are *high-order*, i.e., they have more than two arguments and inducing type signatures of all arguments is necessary. For example, in the sports domain, inducing a schema *win(WinningPlayer, OpponentPlayer, Tournament, Location)* is more informative than inducing just *win(WinningPlayer, OpponentPlayer)*. We refer to this problem as Higher-order Relation Schema Induction (HRSI). In this paper, we propose Tensor Factorization with Back-off and Aggregation (TFBA), a novel framework for the HRSI problem. To the best of our knowledge, this is the first attempt at inducing higher-order relation schemata from unlabeled text. Using the experimental analysis on three real world datasets, we show how TFBA helps in dealing with sparsity and induce higher order schemata.

1 Introduction

Building Knowledge Graphs (KGs) out of unstructured data is an area of active research. Research in this has resulted in the construction of several large scale KGs, such as NELL (Mitchell et al., 2015), Google Knowledge Vault (Dong et al., 2014) and YAGO (Suchanek et al., 2007). These KGs consist of millions of entities and beliefs involving those entities. Such KG construction methods are schema-guided as they require the list of input relations and their schemata (e.g., *playerPlaysSport(Player, Sport)*). In other words, knowledge of schemata is an important first step towards building such KGs.

While beliefs in such KGs are usually binary (i.e., involving two entities), many beliefs of interest go beyond two entities. For example, in the sports domain, one may be interested in beliefs of the form *win(Roger Federer, Nadal, Wimbledon, London)*, which is an instance of the high-order (or n-ary) relation *win* whose schema is given by *win(WinningPlayer, OpponentPlayer, Tournament, Location)*. We refer to the problem of inducing such relation schemata involving multiple arguments as Higher-order Relation Schema Induction (HRSI). In spite of its importance, HRSI is mostly unexplored.

Recently, tensor factorization-based methods have been proposed for *binary* relation schema induction (Nimishakavi et al., 2016), with gains in both speed and accuracy over previously proposed generative models. To the best of our knowledge, tensor factorization methods have not been used for HRSI. We address this gap in this paper.

Due to data sparsity, straightforward adaptation of tensor factorization from (Nimishakavi et al., 2016) to HRSI is not feasible, as we shall see in Section 3.1. We overcome this challenge in this paper, and make the following contributions.

- We propose Tensor Factorization with Back-off and Aggregation (TFBA), a novel tensor factorization-based method for Higher-order RSI (HRSI). In order to overcome data sparsity, TFBA *backs-off* and jointly factorizes multiple lower-order tensors derived from an extremely sparse higher-order tensor.

- As an aggregation step, we propose a constrained clique mining step which constructs the higher-order schemata from multiple binary schemata.

- Through experiments on multiple real-world datasets, we show the effectiveness of TFBA for HRSI.

Source code of TFBA is available at `https://github.com/madhavcsa/TFBA`.

The remainder of the paper is organized as follows. We discuss related work in Section 2. In Section 3.1, we first motivate why a back-off strategy is needed for HRSI, rather than factorizing the higher-order tensor. Further, we discuss the proposed TFBA framework in Section 3.2. In Section 4, we demonstrate the effectiveness of the proposed approach using multiple real world datasets. We conclude with a brief summary in Section 5.

2 Related Work

In this section, we discuss related works in two broad areas: schema induction, and tensor and matrix factorizations.

Schema Induction: Most work on inducing schemata for relations has been in the binary setting (Mohamed et al., 2011; Movshovitz-Attias and Cohen, 2015; Nimishakavi et al., 2016). McDonald et al. (2005) and Peng et al. (2017) extract `n`-ary relations from Biomedical documents, but do not induce the schema, i.e., type signature of the `n`-ary relations. There has been significant amount of work on Semantic Role Labeling (Lang and Lapata, 2011; Titov and Khoddam, 2015; Roth and Lapata, 2016), which can be considered as `n`-ary relation extraction. However, we are interested in inducing the schemata, i.e., the type signature of these relations. Event Schema Induction is the problem of inducing schemata for events in the corpus (Balasubramanian et al., 2013; Chambers, 2013; Nguyen et al., 2015). Recently, a model for event representations is proposed in (Weber et al., 2018).

Cheung et al. (2013) propose a probabilistic model for inducing frames from text. Their notion of frame is closer to that of scripts (Schank and Abelson, 1977). Script learning is the process of automatically inferring sequence of events from text (Mooney and DeJong, 1985). There is a fair amount of recent work in statistical script learning (Pichotta and Mooney, 2016), (Pichotta and Mooney, 2014). While script learning deals with the sequence of events, we try to find the schemata of relations at a corpus level. Ferraro and Durme (2016) propose a unified Bayesian model for scripts, frames and events. Their model tries to capture all levels of Minsky Frame structure (Minsky, 1974), however we work with the surface semantic frames.

Tensor and Matrix Factorizations: Matrix factorization and joint tensor-matrix factorizations have been used for the problem of predicting links in the Universal Schema setting (Riedel et al., 2013; Singh et al., 2015). Chen et al. (2015) use matrix factorizations for the problem of finding semantic slots for unsupervised spoken language understanding. Tensor factorization methods are also used in factorizing knowledge graphs (Chang et al., 2014; Nickel et al., 2012). Joint matrix and tensor factorization frameworks, where the matrix provides additional information, is proposed in (Acar et al., 2013) and (Wang et al., 2015). These models are based on PARAFAC (Harshman, 1970), a tensor factorization model which approximates the given tensor as a sum of rank-1 tensors. A boolean Tucker decomposition for discovering facts is proposed in (Erdos and Miettinen, 2013). In this paper, we use a modified version (Tucker2) of Tucker decomposition (Tucker, 1963).

RESCAL (Nickel et al., 2011) is a simplified Tucker model suitable for relational learning. Recently, SICTF (Nimishakavi et al., 2016), a variant of RESCAL with side information, is used for the problem of schema induction for binary relations. SICTF cannot be directly used to induce higher order schemata, as the higher-order tensors involved in inducing such schemata tend to be extremely sparse. TFBA overcomes these challenges to induce higher-order relation schemata by performing Non-Negative Tucker-style factorization of sparse tensor while utilizing a back-off strategy, as explained in the next section.

3 Higher Order Relation Schema Induction using Back-off Factorization

In this section, we start by discussing the approach of factorizing a higher-order tensor and provide the motivation for back-off strategy. Next, we discuss the proposed TFBA approach in detail. Please refer to Table 1 for notations used in this paper.

Notation	Definition
$\mathbb{R}_+$	Set of non-negative reals.
$\boldsymbol{\mathcal{X}} \in \mathbb{R}_+^{n_1 \times n_2 \times \ldots \times n_N}$	N^{th} -order non-negative tensor.
$\boldsymbol{\mathcal{X}}_{(i)}$	mode-i matricization of tensor $\boldsymbol{\mathcal{X}}$. Please see (Kolda and Bader, 2009) for details.
$\mathbf{A} \in \mathbb{R}_+^{n \times r}$	Non-negative matrix of order $n \times r$.
$*$	Hadamard product: $(\mathbf{A} * \mathbf{B})_{i,j} = \mathbf{A}_{i,j} \times \mathbf{B}_{i,j}$.

Table 1: Notations used in the paper.

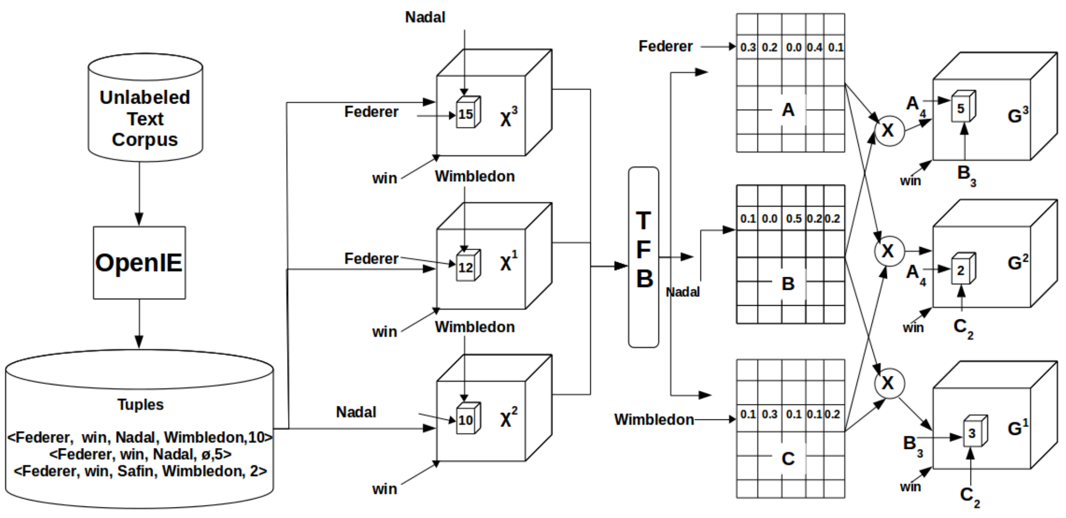


Figure 1: **Overview of Step 1 of TFBA**. Rather than factorizing the higher-order tensor $\boldsymbol{\mathcal{X}}$, TFBA performs joint Tucker decomposition of multiple 3-mode tensors, $\boldsymbol{\mathcal{X}}^1$, $\boldsymbol{\mathcal{X}}^2$, and $\boldsymbol{\mathcal{X}}^3$, derived out of $\boldsymbol{\mathcal{X}}$. This joint factorization is performed using shared latent factors **A**, **B**, and **C**. This results in binary schemata, each of which is stored as a cell in one of the core tensors $\mathcal{G}^1$, $\mathcal{G}^2$, and $\mathcal{G}^3$. Please see Section 3.2.1 for details.

3.1 Factorizing a Higher-order Tensor

Given a text corpus, we use OpenIEv5 (Mausam, 2016) to extract tuples. Consider the following sentence "*Federer won against Nadal at Wimbledon.*". Given this sentence, OpenIE extracts the 4-tuple *(Federer, won, against Nadal, at Wimbledon)*. We lemmatize the relations in the tuples and only consider the noun phrases as arguments. Let $\mathbb{T}$ represent the set of these 4-tuples. We can construct a 4-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}_+^{n_1 \times n_2 \times n_3 \times m}$ from $\mathbb{T}$. Here, n_1 is the number of *subject* noun phrases (NPs), n_2 is the number of *object* NPs, n_3 is the number of *other* NPs, and m is the number of relations in $\mathbb{T}$. Values in the tensor correspond to the frequency of the tuples. In case of 5-tuples of the form *(subject, relation, object, other-1, other-2)*, we split the 5-tuples into two 4-tuples of the form *(subject, relation, object, other-1)* and *(subject, relation, object, other-2)* and frequency of these 4-tuples is considered to be same as the original 5-tuple. Factorizing the tensor $\boldsymbol{\mathcal{X}}$ results in discovering latent categories of NPs, which help in inducing the schemata. We propose the following approach to factorize $\boldsymbol{\mathcal{X}}$.

$$\min_{\mathcal{G},\mathbf{A},\mathbf{B},\mathbf{C}} \|\boldsymbol{\mathcal{X}} - \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \times_4 \mathbf{I}\|_F^2$$
$$+ \lambda_a \|\mathbf{A}\|_F^2 + \lambda_b \|\mathbf{B}\|_F^2 + \lambda_c \|\mathbf{C}\|_F^2 ,$$

where,

$$\mathbf{A} \in \mathbb{R}_+^{n_1 \times r_1}, \mathbf{B} \in \mathbb{R}_+^{n_2 \times r_2}, \mathbf{C} \in \mathbb{R}_+^{n_3 \times r_3},$$
$$\mathcal{G} \in \mathbb{R}_+^{r_1 \times r_2 \times r_3 \times m}, \lambda_a \geq 0, \lambda_b \geq 0 \text{ and } \lambda_c \geq 0.$$

Here, **I** is the identity matrix. Non-negative updates for the variables can be obtained following (Lee and Seung, 2000). Similar to (Nimishakavi et al., 2016), schemata induced will be of the form *relation* $\langle \mathbf{A}_i, \mathbf{B}_j, \mathbf{C}_k \rangle$. Here, $\mathbf{P}_i$ represents the i^{th} column of a matrix **P**. **A** is the embedding matrix of subject NPs in $\mathbb{T}$ (i.e., mode-1 of $\boldsymbol{\mathcal{X}}$), r_1 is the embedding rank in mode-1 which is the number of latent categories of subject NPs. Similarly, **B** and

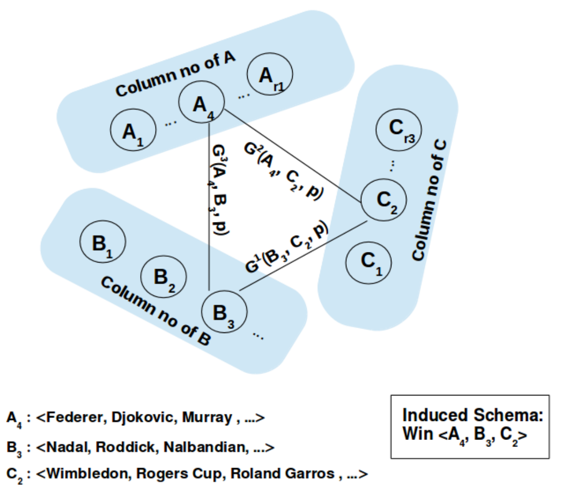

Column no of A
A1
A4
Ar1
Column no of B
B1
B2
B3
Column no of C
C1
C2
Cr3
G3(A4, B3, p)
G2(A4, C2, p)
G1(B3, C2, p)
A4 : <Federer, Djokovic, Murray , ...>
B3 : <Nadal, Roddick, Nalbandian, ...>
C2 : <Wimbledon, Rogers Cup, Roland Garros , ...>
Induced Schema: Win <A4, B3, C2>

Figure 2: **Overview of Step 2 of TFBA**. Induction of higher-order schemata from the tri-partite graph formed from the columns of matrices A, B, and C. Triangles in this graph (solid) represent a 3-ary schema, n-ary schemata for n $>$ 3 can be induced from the 3-ary schemata. Please refer to Section 3.2.2 for details.

$\mathbf{C}$ are the embedding matrices of object NPs and other NPs respectively. r_2 and r_3 are the number of latent categories of object NPs and other NPs respectively. $\boldsymbol{\mathcal{G}}$ is the core tensor. λ_a, λ_b and λ_c are the regularization weights.

However, the 4-order tensors are heavily sparse for all the datasets we consider in this work. The sparsity ratio of this 4-order tensor for all the datasets is of the order 1e-7. As a result of the extreme sparsity, this approach fails to learn any schemata. Therefore, we propose a more successful back-off strategy for higher-order RSI in the next section.

3.2 TFBA: Proposed Framework

To alleviate the problem of sparsity, we construct three tensors $\boldsymbol{\mathcal{X}}^3$, $\boldsymbol{\mathcal{X}}^2$, and $\boldsymbol{\mathcal{X}}^1$ from $\mathbb{T}$ as follows:

- $\boldsymbol{\mathcal{X}}^3 \in \mathbb{R}_+^{n_1 \times n_2 \times m}$ is constructed out of the tuples in $\mathbb{T}$ by dropping the *other* argument and aggregating resulting tuples, i.e., $\boldsymbol{\mathcal{X}}^3_{i,j,p} = \sum_{k=1}^{n_3} \boldsymbol{\mathcal{X}}_{i,j,k,p}$. For example, 4-tuples ⟨*(Federer, Win, Nadal, Wimbledon), 10*⟩ and ⟨*(Federer, Win, Nadal, Australian Open), 5*⟩ will be aggregated to form a triple ⟨*(Federer, Win, Nadal), 15*⟩.
- $\boldsymbol{\mathcal{X}}^2 \in \mathbb{R}_+^{n_1 \times n_3 \times m}$ is constructed out of the tuples in $\mathbb{T}$ by dropping the *object* argument and aggregating resulting tuples i.e., $\boldsymbol{\mathcal{X}}^2_{i,j,p} = \sum_{k=1}^{n_2} \boldsymbol{\mathcal{X}}_{i,k,j,p}$.
- $\boldsymbol{\mathcal{X}}^1 \in \mathbb{R}_+^{n_2 \times n_3 \times m}$ constructed out of the tuples in $\mathbb{T}$ by dropping the *subject* argument and aggregating resulting tuples i.e., $\boldsymbol{\mathcal{X}}^1_{i,j,p} = \sum_{k=1}^{n_1} \boldsymbol{\mathcal{X}}_{k,i,j,p}$.

The proposed framework TFBA for inducing higher order schemata involves the following two steps.

- **Step 1**: In this step, TFBA factorizes multiple lower-order overlapping tensors, $\boldsymbol{\mathcal{X}}^1$, $\boldsymbol{\mathcal{X}}^2$, and $\boldsymbol{\mathcal{X}}^3$, derived from $\boldsymbol{\mathcal{X}}$ to induce binary schemata. This step is illustrated in Figure 1 and we discuss details in Section 3.2.1.
- **Step 2**: In this step, TFBA connects multiple binary schemata identified above to induce higher-order schemata. The method accomplishes this by solving a constrained clique problem. This step is illustrated in Figure 2 and we discuss the details in Section 3.2.2.

3.2.1 Step 1: Back-off Tensor Factorization

A schematic overview of this step is shown in Figure 1. TFBA first preprocesses the corpus and extracts OpenIE tuple set $\mathbb{T}$ out of it. The 4-mode tensor $\boldsymbol{\mathcal{X}}$ is constructed out of $\mathbb{T}$. Instead of performing factorization of the higher-order tensor $\boldsymbol{\mathcal{X}}$ as in Section 3.1, TFBA creates three tensors out of $\boldsymbol{\mathcal{X}}$: $\boldsymbol{\mathcal{X}}^1_{n_2 \times n_3 \times m}$, $\boldsymbol{\mathcal{X}}^2_{n_1 \times n_3 \times m}$ and $\boldsymbol{\mathcal{X}}^3_{n_1 \times n_2 \times m}$.

TFBA performs a coupled non-negative Tucker factorization of the input tensors $\boldsymbol{\mathcal{X}}^1$, $\boldsymbol{\mathcal{X}}^2$ and $\boldsymbol{\mathcal{X}}^3$ by solving the following optimization problem.

$$\begin{aligned}\min_{\substack{\mathbf{A},\mathbf{B},\mathbf{C}\\ \boldsymbol{\mathcal{G}}^1,\boldsymbol{\mathcal{G}}^2,\boldsymbol{\mathcal{G}}^3}} \; & f(\boldsymbol{\mathcal{X}}^3, \boldsymbol{\mathcal{G}}^3, \mathbf{A}, \mathbf{B}) + f(\boldsymbol{\mathcal{X}}^2, \boldsymbol{\mathcal{G}}^2, \mathbf{A}, \mathbf{C}) \\ & + f(\boldsymbol{\mathcal{X}}^1, \boldsymbol{\mathcal{G}}^1, \mathbf{B}, \mathbf{C}) \\ & + \lambda_a \|\mathbf{A}\|_F^2 + \lambda_b \|\mathbf{B}\|_F^2 + \lambda_c \|\mathbf{C}\|_F^2 , \quad (1)\end{aligned}$$

where,

$$f(\boldsymbol{\mathcal{X}}^i, \boldsymbol{\mathcal{G}}^i, \mathbf{P}, \mathbf{Q}) = \left\| \boldsymbol{\mathcal{X}}^i - \boldsymbol{\mathcal{G}}^i \times_1 \mathbf{P} \times_2 \mathbf{Q} \times_3 \mathbf{I} \right\|_F^2$$

$$\mathbf{A} \in \mathbb{R}_+^{n_1 \times r_1}, \mathbf{B} \in \mathbb{R}_+^{n_2 \times r_2}, \mathbf{C} \in \mathbb{R}_+^{n_3 \times r_3}$$

$$\boldsymbol{\mathcal{G}}^1 \in \mathbb{R}_+^{r_2 \times r_3 \times m}, \boldsymbol{\mathcal{G}}^2 \in \mathbb{R}_+^{r_1 \times r_3 \times m}, \boldsymbol{\mathcal{G}}^3 \in \mathbb{R}_+^{r_1 \times r_2 \times m}.$$

We enforce non-negativity constraints on the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the core tensors $\boldsymbol{\mathcal{G}}^i$ ($i \in \{1, 2, 3\}$). Non-negativity is essential for learning interpretable latent factors (Murphy et al., 2012).

Each slice of the core tensor $\mathcal{G}^3$ corresponds to one of the m relations. Each cell in a slice corresponds to an induced schema in terms of the latent factors from matrices $\mathbf{A}$ and $\mathbf{B}$. In other words, $\mathcal{G}^3_{i,j,k}$ is an induced binary schema for relation k involving induced categories represented by columns $\mathbf{A}_i$ and $\mathbf{B}_j$. Cells in $\mathcal{G}^1$ and $\mathcal{G}^2$ may be interpreted accordingly.

We derive non-negative multiplicative updates for $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ following the NMF updating rules given in (Lee and Seung, 2000). For the update of $\mathbf{A}$, we consider the mode-1 matricization of first and the second term in Equation 1 along with the regularizer.

$$\mathbf{A} \leftarrow \mathbf{A} * \frac{\mathcal{X}^3_{(1)}\mathcal{G}_{B_A}^\top + \mathcal{X}^2_{(1)}\mathcal{G}_{C_A}^\top}{\mathbf{A}[\mathcal{G}_{B_A}\mathcal{G}_{B_A}^\top + \mathcal{G}_{C_A}\mathcal{G}_{C_A}^\top] + \lambda_a \mathbf{A}},$$

where,

$$\mathcal{G}_{B_A} = (\mathcal{G}^3 \times_2 \mathbf{B})_{(1)}, \;\; \mathcal{G}_{C_A} = (\mathcal{G}^2 \times_2 \mathbf{C})_{(1)}.$$

In order to estimate $\mathbf{B}$, we consider mode-2 matricization of first term and mode-1 matricization of third term in Equation 1, along with the regularization term. We get the following update rule for $\mathbf{B}$

$$\mathbf{B} \leftarrow \mathbf{B} * \frac{\mathcal{X}^3_{(2)}\mathcal{G}_{A_B}^\top + \mathcal{X}^1_{(1)}\mathcal{G}_{C_B}^\top}{\mathbf{B}[\mathcal{G}_{A_B}\mathcal{G}_{A_B}^\top + \mathcal{G}_{C_B}\mathcal{G}_{C_B}^\top] + \lambda_b \mathbf{B}},$$

where,

$$\mathcal{G}_{A_B} = (\mathcal{G}^3 \times_1 \mathbf{A})_{(2)}, \;\; \mathcal{G}_{C_B} = (\mathcal{G}^1 \times_2 \mathbf{C})_{(1)}.$$

For updating $\mathbf{C}$, we consider mode-2 matricization of second and third terms in Equation 1 along with the regularization term, and we get

$$\mathbf{C} \leftarrow \mathbf{C} * \frac{\mathcal{X}^3_{(2)}\mathcal{G}_{B_C}^\top + \mathcal{X}^2_{(2)}\mathcal{G}_{A_C}^\top}{\mathbf{C}[\mathcal{G}_{A_C}\mathcal{G}_{A_C}^\top + \mathcal{G}_{B_C}\mathcal{G}_{B_C}^\top] + \lambda_c \mathbf{C}},$$

where,

$$\mathcal{G}_{A_C} = (\mathcal{G}^3 \times_1 \mathbf{B})_{(2)}, \;\; \mathcal{G}_{B_C} = (\mathcal{G}^2 \times_1 \mathbf{A})_{(2)}.$$

Finally, we update the three core tensors in Equation 1 following (Kim and Choi, 2007) as follows,

$$\mathcal{G}^1 \leftarrow \mathcal{G}^1 * \frac{\mathcal{X}^1 \times_1 \mathbf{B}^\top \times_2 \mathbf{C}^\top}{\mathcal{G}^1 \times_1 \mathbf{B}^\top\mathbf{B} \times_2 \mathbf{C}^\top\mathbf{C}},$$

$$\mathcal{G}^2 \leftarrow \mathcal{G}^2 * \frac{\mathcal{X}^2 \times_1 \mathbf{A}^\top \times_2 \mathbf{C}^\top}{\mathcal{G}^2 \times_1 \mathbf{A}^\top\mathbf{A} \times_2 \mathbf{C}^\top\mathbf{C}},$$

$$\mathcal{G}^3 \leftarrow \mathcal{G}^3 * \frac{\mathcal{X}^3 \times_1 \mathbf{A}^\top \times_2 \mathbf{B}^\top}{\mathcal{G}^3 \times_1 \mathbf{A}^\top\mathbf{A} \times_2 \mathbf{B}^\top\mathbf{B}}.$$

In all the above updates, $\frac{\mathbf{P}}{\mathbf{Q}}$ represents element-wise division and $\mathbf{I}$ is the identity matrix.

Initialization: For initializing the component matrices $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, we first perform a non-negative Tucker2 Decomposition of the individual input tensors $\mathcal{X}^1, \mathcal{X}^2$, and $\mathcal{X}^3$. Then compute the average of component matrices obtained from each individual decomposition for initialization. We initialize the core tensors $\mathcal{G}^1, \mathcal{G}^2$, and $\mathcal{G}^3$ with the core tensors obtained from the individual decompositions.

3.2.2 Step 2: Binary to Higher-Order Schema Induction

In this section, we describe how a higher-order schema is constructed from the factorization described in the previous sub-section. Each relation k has three representations given by the slices $\mathcal{G}^1_k$, $\mathcal{G}^2_k$ and $\mathcal{G}^3_k$ from each core tensor. We need a principled way to produce a joint schema from these representations. For a relation, we select top-n indices (i, j) with highest values from each matrix. The indices i and j from $\mathcal{G}^3_k$ correspond to column numbers of $\mathbf{A}$ and $\mathbf{B}$ respectively, indices from $\mathcal{G}^2_k$ correspond to columns from $\mathbf{A}$ and $\mathbf{C}$ and columns from $\mathcal{G}^1_k$ correspond to columns from $\mathbf{B}$ and $\mathbf{C}$.

We construct a tri-partite graph with the column numbers from each of the component matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ as the vertices belonging to independent sets, the top-n indices selected are the edges between these vertices. From this tri-partite graph, we find all the triangles which will give schema with three arguments for a relation, illustrated in Figure 2. We find higher order schemata, i.e., schemata with more than three arguments by merging two third order schemata with same column number from $\mathbf{A}$ and $\mathbf{B}$. For example, if we find two schemata $(\mathbf{A}_2, \mathbf{B}_4, \mathbf{C}_{10})$ and $(\mathbf{A}_2, \mathbf{B}_4, \mathbf{C}_8)$ then we merge these two to give $(\mathbf{A}_2, \mathbf{B}_4, \mathbf{C}_{10}, \mathbf{C}_8)$ as a higher order schema. This can be continued further for even higher order schemata. This process may be thought of as finding a **constrained**

clique over the tri-partite graph. Here the constraint is that in the maximal clique, there can only be one edge between sets corresponding to columns of **A** and columns of **B**.

The procedure above is inspired by (McDonald et al., 2005). However, we note that (McDonald et al., 2005) solved a different problem, viz., n-ary relation instance extraction, while our focus is on inducing schemata. Though we discuss the case of back-off from 4-order to 3-order, ideas presented above can be extended for even higher orders depending on the sparsity of the tensors.

4 Experiments

In this section, we evaluate the performance of TFBA for the task of HRSI. We also propose a baseline model for HRSI called HardClust.

HardClust: We propose a baseline model called the Hard Clustering Baseline (HardClust) for the task of higher order relation schema induction. This model induces schemata by grouping per-relation NP arguments from OpenIE extractions. In other words, for each relation, all the Noun Phrases (NPs) in first argument form a cluster that represents the subject of the relation, all the NPs in the second argument form a cluster that represents object and so on. Then from each cluster, the top most frequent NPs are chosen as the representative NPs for the argument type. We note that this method is only able to induce one schema per relation.

Datasets: We run our experiments on three datasets. The first dataset (Shootings) is a collection of 1,335 documents constructed from a publicly available database of mass shootings in the United States. The second is New York Times Sports (NYT Sports) dataset which is a collection of 20,940 sports documents from the period 2005 and 2007. And the third dataset (MUC) is a set of 1300 Latin American newswire documents about terrorism events. After performing the processing steps described in Section 3, we obtained 357,914 unique OpenIE extractions from the NYT Sports dataset, 10,847 from Shootings dataset, and 8,318 from the MUC dataset. However, in order to properly analyze and evaluate the model, we consider only the 50 most frequent relations in the datasets and their corresponding OpenIE extractions. This is done to avoid noisy OpenIE extractions to yield better data quality and to aid subsequent manual evaluation of the data. We construct input tensors following the procedure described in Section 3.2. Details on the dimensions of tensors obtained are given in Table 2.

Model Selection: In order to select appropriate TFBA parameters, we perform a grid search over the space of hyper-parameters, and select the set of hyper-parameters that give best Average FIT score (AvgFIT).

$$\begin{aligned}
&\text{AvgFIT}(\boldsymbol{\mathcal{G}}^1, \boldsymbol{\mathcal{G}}^2, \boldsymbol{\mathcal{G}}^3, \mathbf{A}, \mathbf{B}, \mathbf{C}, \boldsymbol{\mathcal{X}}^1, \boldsymbol{\mathcal{X}}^2, \boldsymbol{\mathcal{X}}^3) = \\
&\frac{1}{3}\{\text{FIT}(\boldsymbol{\mathcal{X}}^1, \boldsymbol{\mathcal{G}}^1, \mathbf{B}, \mathbf{C}) + \text{FIT}(\boldsymbol{\mathcal{X}}^2, \boldsymbol{\mathcal{G}}^2, \mathbf{A}, \mathbf{C}) \\
&\qquad + \text{FIT}(\boldsymbol{\mathcal{X}}^3, \boldsymbol{\mathcal{G}}^3, \mathbf{A}, \mathbf{B})\},
\end{aligned}$$

where,

$$\text{FIT}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{G}}, \mathbf{P}, \mathbf{Q}) = 1 - \frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_1 \mathbf{P} \times_2 \mathbf{Q}\|_F}{\|\boldsymbol{\mathcal{X}}\|_F}.$$

We perform a grid search for the rank parameters between 5 and 20, for the regularization weights we perform a grid search over 0 and 1. Table 3 provides the details of hyper-parameters set for different datasets.

Evaluation Protocol: For TFBA, we follow the protocol mentioned in Section 3.2.2 for constructing higher order schemata. For every relation, we consider top 5 binary schemata from the factorization of each tensor. We construct a tri-partite graph, as explained in Section 3.2.2, and mine constrained maximal cliques from the tripartite graphs for schemata. Table 4 provides some qualitative examples of higher-order schemata induced by TFBA. Accuracy of the schemata induced by the model is evaluated by human evaluators. In our experiments, we use human judgments from three evaluators. For every relation, the first and second columns given in Table 4 are presented to the evaluators and they are asked to validate the schema. We present top 50 schemata based on the score of the constrained maximal clique induced by TFBA to the evaluators. This evaluation protocol was also used in (Movshovitz-Attias and Cohen, 2015) for evaluating ontology induction. All evaluations were blind, i.e., the evaluators were not aware of the model they were evaluating.

Difficulty with Computing Recall: Even though recall is a desirable measure, due to the lack of availability of gold higher-order schema annotated corpus, it is not possible to compute recall. Although the MUC dataset has gold annotations for some predefined list of events, it does not have annotations for the relations.

Dataset	$\mathcal{X}^1 shape$	$\mathcal{X}^2 shape$	$\mathcal{X}^3 shape$
Shootings	$3365 \times 1295 \times 50$	$2569 \times 1295 \times 50$	$2569 \times 3365 \times 50$
NYT Sports	$57,820 \times 20,356 \times 50$	$49,659 \times 20,356 \times 50$	$49,659 \times 57,820 \times 50$
MUC	$2825 \times 962 \times 50$	$2555 \times 962 \times 50$	$2555 \times 2825 \times 50$

Table 2: Details of dimensions of tensors constructed for each dataset used in the experiments.

Dataset	(r_1, r_2, r_3)	$(\lambda_a, \lambda_b, \lambda_c)$
Shootings	(10, 20,15)	(0.3, 0.1, 0.7)
NYT Sports	(20, 15, 15)	(0.9, 0.5, 0.7)
MUC	(15, 12, 12)	(0.7, 0.7, 0.4)

Table 3: Details of hyper-parameters set for different datasets.

Experimental results comparing performance of various models for the task of HRSI are given in Table 5. We present evaluation results from three evaluators represented as E1, E2 and E3. As can be observed from Table 5, TFBA achieves better results than HardClust for the Shootings and NYT Sports datasets, however HardClust achieves better results for the MUC dataset. Percentage agreement of the evaluators for TFBA is 72%, 70% and 60% for Shootings, NYT Sports and MUC datasets respectively.

HardClust Limitations: Even though HardClust gives better induction for MUC corpus, this approach has some serious drawbacks. HardClust can only induce one schema per relation. This is a restrictive constraint as multiple senses can exist for a relation. For example, consider the schemata induced for the relation *shoot* as shown in Table 4. TFBA induces two senses for the relation, but HardClust can induce only one schema. For a set of 4-tuples, HardClust can only induce ternary schemata; the dimensionality of the schemata cannot be varied. Since the latent factors induced by HardClust are entirely based on frequency, the latent categories induced by HardClust are dominated by only a fixed set of noun phrases. For example, in NYT Sports dataset, subject category induced by HardClust for all the relations is $\langle$*team, yankees, mets*$\rangle$. In addition to inducing only one schema per relation, most of the times HardClust only induces a fixed set of categories. Whereas for TFBA, the number of categories depends on the rank of factorization, which is a user provided parameter, thus providing more flexibility to choose the latent categories.

4.1 Using Event Schema Induction for HRSI

Event schema induction is defined as the task of learning high-level representations of events, like a tournament, and their entity roles, like winning-player etc, from unlabeled text. Even though the main focus of event schema induction is to induce the important roles of the events, as a side result most of the algorithms also provide schemata for the relations. In this section, we investigate the effectiveness of these schemata compared to the ones induced by TFBA.

Event schemata are represented as a set of *(Actor, Rel, Actor)* triples in (Balasubramanian et al., 2013). *Actors* represent groups of noun phrases and *Rel*s represent relations. From this style of representation, however, the n-ary schemata for relations cannot be induced. Event schemata generated in (Weber et al., 2018) are similar to that in (Balasubramanian et al., 2013). Event schema induction algorithm proposed in (Nguyen et al., 2015) doesn't induce schemata for relations, but rather induces the roles for the events. For this investigation we experiment with the following algorithm.

Chambers-13 (Chambers, 2013): This model learns event templates from text documents. Each event template provides a distribution over slots, where slots are clusters of NPs. Each event template also provides a cluster of relations, which is most likely to appear in the context of the aforementioned slots. We evaluate the schemata of these relation clusters.

As can be observed from Table 5, the proposed TFBA performs much better than Chambers-13. HardClust also performs better than Chambers-13 on all the datasets. From this analysis we infer that there is a need for algorithms which induce higher-order schemata for relations, a gap we fill in this paper. Please note that the experimental results provided in (Chambers, 2013) for MUC dataset are for the task of event schema induction, but in this work we evaluate the relation schemata. Hence the results in (Chambers, 2013) and results in this paper are not comparable. Example

Relation Schema	NPs from the induced categories	Evaluator Judgment	(Human) Suggested Label
	Shootings		
leave$\langle A_6, B_0, C_7\rangle$	A_6: *shooting, shooting incident, double shooting* B_0: *one person, two people, three people* C_7: *dead, injured, on edge*	valid	< shooting > < people > <injured >
identify$\langle A_1, B_1, C_5, C_6\rangle$	A_1: *police, officers, huntsville police* B_1: *man, victims, four victims* C_5: *sunday, shooting staurday, wednesday afternoon* C_6: *apartment, bedroom, building in the neighborhood*	valid	< police > < victim(s)> <day/time > <place >
shoot$\langle A_7, B_6, C_1\rangle$	A_7: *gunman, shooter, smith* B_6: *freeman, slain woman, victims* C_1: *friday, friday night, early monday morning*	valid	< perpetrator > <victim > < time>
shoot$\langle A_4, B_2, C_{13}\rangle$	A_4: *<num>-year-old man, <num>-year-old george reavis, <num>-year-old brockton man* B_2: *in the leg, in the head, in the neck* C_{13}: *in macon, in chicago, in an alley*	valid	< victim> < body part> < location >
say$\langle A_1, B_1, C_5\rangle$	A_1: *police, officers, huntsville police* B_1: *man, victims, four victims* C_5: *sunday, shooting staurday, wednesday afternoon*	invalid	–
	NYT sports		
spend$\langle A_0, B_{16}, C_3\rangle$	A_0: *yankees, mets, jets* B_{14}: *$ <num> million, $ <num>, $ <num> billion* C_3: *<num>, year, last season*	valid	< team > < money > < year >
win$\langle A_2, B_{10}, C_3\rangle$	A_2: *red sox, team, yankees* B_{10}: *world series, title, world cup* C_3: *<num>, year, last season*	valid	< team > < championship > < year >
get$\langle A_4, B_4, C_1\rangle$	A_4: *umpire, mike cameron, andre agassi* B_4: *ball, lives, grounder* C_1: *back, forward, <num>-yard line*	invalid	–
	MUC		
tell$\langle A_7, B_2, C_0\rangle$	A_7: *medardo gomez, jose azcona, gregorio roza chavez* B_2: *media, reporters, newsmen* C_0: *today, at <num>, tonight*	valid	< politician > <media > < day/time >
occur$\langle A_9, B_5, C_{10}\rangle$	A_9: *bomb, blast, explosion* B_5: *near san salvador, here in madrid, in the same office* C_{10}: *at <num>, this time, simultaneously*	valid	< bombing > < place > < time >
suffer$\langle A_5, B_4, C_4\rangle$	A_5: *justice maria elena diaz, vargas escobar, judge sofia de roldan* B_4: *casualties , car bomb, grenade* C_4: *settlement of refugees, in san roman, now*	invalid	–

Table 4: Examples of schemata induced by TFBA. Please note that some of them are 3-ary while others are 4-ary. For details about schema induction, please refer to Section 3.2.

	Shootings				NYT Sports				MUC			
	E1	E2	E3	Avg	E1	E2	E3	Avg	E1	E2	E3	Avg
HardClust	0.64	0.70	0.64	0.66	0.42	0.28	0.52	0.46	0.64	0.58	0.52	**0.58**
Chambers-13	0.32	0.42	0.28	0.34	0.08	0.02	0.04	0.07	0.28	0.34	0.30	0.30
TFBA	0.82	0.78	0.68	**0.76**	0.86	0.6	0.64	**0.70**	0.58	0.38	0.48	0.48

Table 5: Higher-order RSI accuracies of various methods on the three datasets. Induced schemata for each dataset and method are evaluated by three human evaluators, E1, E2, and E3. TFBA performs better than HardClust for Shootings and NYT Sports datasets. Even though HardClust achieves better accuracy on MUC dataset, it has several limitations, see Section 4 for more details. Chambers-13 solves a slightly different problem called event schema induction, for more details about the comparison with Chambers-13 see Section 4.1.

schemata induced by TFBA and (Chambers-13) are provided as part of the supplementary material.

5 Conclusion

Higher order Relation Schema Induction (HRSI) is an important first step towards building domain-specific Knowledge Graphs (KGs). In this paper, we proposed TFBA, a tensor factorization-based method for higher-order RSI. To the best of our knowledge, this is the first attempt at inducing higher-order (n-ary) schemata for relations from unlabeled text. Rather than factorizing a severely sparse higher-order tensor directly, TFBA performs *back-off* and jointly factorizes multiple lower-order tensors derived out of the higher-order tensor. In the second step, TFBA solves a constrained clique problem to induce schemata out of multiple binary schemata. We are hopeful that the backoff-based factorization idea exploited in TFBA will be useful in other sparse factorization settings.

Acknowledgment

We thank the anonymous reviewers for their insightful comments and suggestions. This research has been supported in part by the Ministry of Human Resource Development (Government of India), Accenture, and Google.

References

Evrim Acar, Morten Arendt Rasmussen, Francesco Savorani, Tormod Ns, and Rasmus Bro. 2013. Understanding data fusion within the framework of coupled matrix and tensor factorizations. *Chemometrics and Intelligent Laboratory Systems* 129:53–63.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*.

Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*.

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky. 2015. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *ACL*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *NAACL-HLT*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*.

Dora Erdos and Pauli Miettinen. 2013. Discovering facts with boolean tensor tucker decomposition. In *CIKM*.

Francis Ferraro and Benjamin Van Durme. 2016. A unified bayesian model of scripts, frames and language. In *AAAI*.

R. A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics* 16(1):84.

Yong-Deok Kim and Seungjin Choi. 2007. Nonnegative tucker decomposition. In *CVPR*.

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.

Joel Lang and Mirella Lapata. 2011. Unsupervised semantic role induction via split-merge clustering. In *NAACL-HLT*.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *NIPS*.

Mausam. 2016. Open information extraction systems and downstream applications. In *IJCAI*.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical ie. In *ACL*.

Marvin Minsky. 1974. A framework for representing knowledge. Technical report.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *AAAI*.

Thahir P. Mohamed, Jr. Estevam R. Hruschka, and Tom M. Mitchell. 2011. Discovering relations between noun categories. In *EMNLP*.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *IJCAI*.

Dana Movshovitz-Attias and William W. Cohen. 2015. Kb-lda: Jointly learning a knowledge base of hierarchy, relations, and facts. In *ACL*.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *COLING*.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *ACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: Scalable machine learning for linked data. In *WWW*.

Madhav Nimishakavi, Uday Singh Saini, and Partha Talukdar. 2016. Relation schema induction using tensor factorization with side information. In *EMNLP*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL* 5:101–115.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *ACL*.

R. Schank and R. Abelson. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Sameer Singh, Tim Rocktäschel, and Sebastian Riedel. 2015. Towards Combined Matrix and Tensor Factorization for Universal Schema Relation Extraction. In *NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *NAACL-HLT*.

L. R. Tucker. 1963. Implications of factor analysis of three-way matrices for measurement of change. In *Problems in measuring change.*, University of Wisconsin Press, Madison WI, pages 122–137.

Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C. Denny, Abel N. Kho, You Chen, Bradley A. Malin, and Jimeng Sun. 2015. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *KDD*.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Event representations with tensor-based compositions. In *AAAI*.

Discovering Implicit Knowledge with Unary Relations

Michael Glass
IBM Research AI
Knowledge Induction and Reasoning
mrglass@us.ibm.com

Alfio Gliozzo
IBM Research AI
Knowledge Induction and Reasoning
gliozzo@us.ibm.com

Abstract

State-of-the-art relation extraction approaches are only able to recognize relationships between mentions of entity arguments stated explicitly in the text and typically localized to the same sentence. However, the vast majority of relations are either implicit or not sententially localized. This is a major problem for Knowledge Base Population, severely limiting recall. In this paper we propose a new methodology to identify relations between two entities, consisting of detecting a very large number of unary relations, and using them to infer missing entities. We describe a deep learning architecture able to learn thousands of such relations very efficiently by using a common deep learning based representation. Our approach largely outperforms state of the art relation extraction technology on a newly introduced web scale knowledge base population benchmark, that we release to the research community.

1 Introduction

Knowledge Base Population (KBP) from text is the problem of extracting relations between entities with respect to a given schema, usually defined by a set of types and relations. The facts added to the KB are triples, consisting of two entities connected by a relation. Although providing explicit provenance for the triples is often a subgoal in KBP, we focus on the case where correct triples are gathered from text without necessarily annotating any particular text with a relation. Humans are able to perform very well on the task of understanding relations in text. For example, if the target relation is presidentOf, anyone will be able to detect an occurrence of this relation between the entities TRUMP and UNITED_STATES from both the sentences "Trump issued a presidential memorandum for the United States" and "The Houston Astros will visit President Donald Trump and the White House on Monday". However, the first example expresses an explicit relation between the two entities, while the second states the same relation implicitly and requires some background knowledge and inference to identify it properly. In fact, the entity UNITED_STATES is not even mentioned explicitly in the text, and it is up to the reader to recall that US presidents live in the White House, and therefore people visiting it are visiting the US president.

Very often, relations expressed in text are implicit. This reflects in the low recall of the current KBP relation extraction methods, that are mostly based on recognizing lexical-syntactic connections between two entities within the same sentence. The state-of-the-art systems are affected by very low performance, close to 16.6% F1, as shown in the latest TAC-KBP evaluation campaigns and in the open KBP evaluation benchmark[1]. Existing approaches to dealing with implicit information such as textual entailment depend on unsolved problems like inducing entailment rules from text.

In this paper, we address the problem of identifying implicit relations in text using a radically different approach, consisting of reducing the problem of identifying binary relations into a much larger set of simpler unary relations. For example, to build a Knowledge Base (KB) about presidents in the G8 countries, the *presidentOf* relation can be expanded to *presidentOf*:UNITED_STATES, *presidentOf*:GERMANY, *presidentOf*:JAPAN, and so

[1]https://kbpo.stanford.edu

on. For all these unary relations, we train a multi-class (and in other cases, multi-label) classifier from all the available training data. This classifier takes textual evidence where only one entity is identified (e.g. ANGELA_MERKEL) and predicts a confidence score for each unary relation. In this way, ANGELA_MERKEL will be assigned to the unary relation *presidentOf*:GERMANY, which in turn generates the triple ⟨ANGELA_MERKEL *presidentOf* GERMANY⟩.

To implement the idea above, we explore the use of knowledge-level supervision, sometimes called distant supervision, to train a deep learning based approach. The training data in this approach is a knowledge base and an unannotated corpus. A pre-existing Entity Detection and Linking system first identifies and links mentions of entities in the corpus. For each entity, the system gathers its *context set*, the contexts (e.g. sentences or token windows) where it is mentioned. The context set forms the textual evidence for a multi-class, multi-label deep network. The final layer of the network is vector of unary relation predictions and the intermediate layers are shared. This architecture allows us to efficiently train thousands of unary relations, while reusing the feature representations in the intermediate layers across relations as a form of transfer learning. The predictions of this network represent the probability for the input entity to belong to each unary relation.

To demonstrate the effectiveness of our approach we developed a new KBP benchmark, consisting of extracting unseen DBPedia triples from the text of a web crawl, using a portion of DBpedia to train the model. As part of the contributions for this paper, we release the benchmark to the research community providing the software needed to generate it from Common Crawl and DBpedia as an open source project[2].

As a baseline, we adapt a state of the art deep learning based approach for relation extraction (Lin et al., 2016). Our experiments clearly show that using unary relations to generate new triples greatly complements traditional binary approaches. An analysis of the data shows that our approach is able to capture implicit information from textual mentions and to highlight the reasons why the assignments have been made.

The paper is structured as follows. In section 2 we describe the state of the art in distantly supervised KBP methodologies, with a focus on knowledge induction applications. Section 3 introduces the use of Unary Relations for KBP and section 4 outlines the process for producing and training them. Section 5 describes a deep learning architecture able to recognize unary relations from textual evidence. In section 6 we describe the benchmark for evaluation. Section 7 provides an extensive evaluation of unary relations, and a saliency map exploration of what the deep learning model has learned. Section 8 concludes the paper highlighting research directions for future work.

2 Related Work

Binary relation extraction using distant supervision has a long history (Surdeanu et al., 2012; Mintz et al., 2009). Mentions of entities from the knowledge base are located in text. When two entities are mentioned in the same sentence that sentence becomes part of the evidence for the relation (if any) between those entities. The set of sentences mentioning an entity pair is used in a machine learning model to predict how the entities are related, if at all.

Deep learning has been applied to binary relation extraction. CNN-based (Zeng et al., 2014), LSTM-based (Xu et al., 2015), attention based (Wang et al., 2016) and compositional embedding based (Gormley et al., 2015) models have been trained successfully using a sentence as the unit of context. Recently, cross sentence approaches have been explored by building paths connecting the two identified arguments through related entities (Peng et al., 2017; Zeng et al., 2016). These approaches are limited by requiring both entities to be mentioned in a textual context. The context aggregation approaches of state-of-the-art neural models, max-pooling (Zeng et al., 2015) and attention (Lin et al., 2016), do not consider that different contexts may contribute to the prediction in different ways. Instead, the context pooling only determines the degree of a sentence's contribution to the relation prediction.

TAC-KBP is a long running challenge for knowledge base population. Effective systems in these competitions combine many approaches such as rule-based relation extraction, directly supervised linear and neural network extractors, distantly supervised neural network models (Zhang et al., 2016) and tensor factorization approaches to relation prediction. Compositional Universal

[2]https://github.com/IBM/cc-dbp

Schema is an approach based on combining the matrix factorization approach of universal schema (Riedel et al., 2013), with repesentations of textual relations produced by an LSTM (Chang et al., 2016). The rows of the universal schema matrix are entity pairs, and will only be supported by a textual relation if they occur in a sentence together.

Other approaches to relational knowledge induction have used distributed representations for words or entities and used a model to predict the relation between two terms based on their semantic vectors (Drozd et al., 2016). This enables the discovery of relations between terms that do not co-occur in the same sentence. However, the distributed representation of the entities is developed from the corpus without any ability to focus on the relations of interest. One example of such work is LexNET, which developed a model using the distributional word vectors of two terms to predict lexical relations between them (DS_h). The term vectors are concatenated and used as input to a single hidden layer neural network. Unlike our approach to unary relations the term vectors are produced by a standard relation-independent model of the term's contexts such as word2vec (Mikolov et al., 2013).

Unary relations can be considered to be similar to types. Work on ontology population has considered the general distribution of a term in text to predict its type (Cimiano and Völker, 2005). Like the method of DS_h, this does not customize the representation of an entity to a set of target relations.

3 Unary vs Binary Relations

The basic idea presented in this paper is that in many cases relation extraction problems can be reduced to sets of simpler and inter-related unary relation extraction problems. This is possible by providing a specific value to one of the two arguments, transforming the relations into a set of categories. For example, the *livesIn* relation between persons and countries can be decomposed into 195 relations (one relation for each country), including *livesIn*:UNITED_STATES, *livesIn*:CANADA, and so on. The argument that is combined with the binary relation to produce the unary relation is called the *fixed argument* while the other argument is the *filler argument*. The *KB extension* of a unary relation is the set of all filler arguments in the KB, and the *corpus extension* is the subset of the KB extension that occurs in the corpus.

A requisite for a unary relation is that in the training KB there should exist many triples that share a relation and a particular entity as one argument, thus providing enough training for each unary classifier. Therefore, in the example above, we will not likely be able to generate predicates for all the 195 countries, because some of them will either not occur at all in the training data or they will be very infrequent. However, even in cases where arguments tend to follow a long tail distribution, it makes sense to generate unary predicates for the most frequent ones.

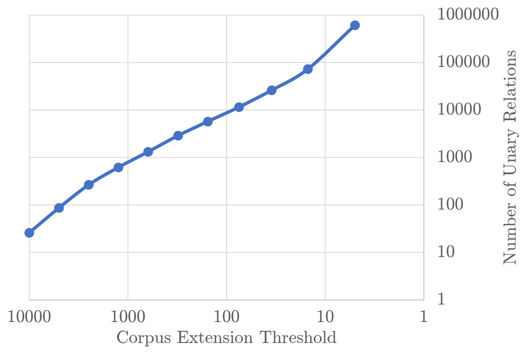


Figure 1: Minimum Corpus Extension to Number of Unary Relations

Figure 1 shows the relationship between the threshold for the size of the corpus extension of a unary relation and the number of different unary relations that can be found in our dataset. The relationship is approximately linear on a log-log scale. There are 26 unary relations with a corpus extension of at least 10,000. These relations include:

- *hasLocation*:UNITED_STATES
- *background*:GROUP_OR_BAND
- *kingdom*:ANIMAL
- *language*:ENGLISH_LANGUAGE

Lowering the threshold to 100 we have 8711 unary relations and we get close to 1M unary relations with more than 10 entities.

In a traditional binary KBP task a triple has a *relevant context set* if the two entities occur at least once together in the corpus - where the notion of 'together' is typically intra-sentential (within a single sentence). In KBP based on unary relations, a triple ⟨FILLER *rel* FIXED⟩ has a relevant context set if the unary relation *rel*:FIXED has the filler argument in its corpus extension, i.e. the filler occurs in the corpus.

Both approaches are limited in different important respects. KBP with unary relations can only produce triples when fixing a relation and argument provides a relatively large corpus extension. Triples such as ⟨BARACK_OBAMA *spouse* MICHELLE_OBAMA⟩ cannot be extracted in this way, since neither Barack nor Michelle Obama have a large set of spouses. The limitation of binary relation extraction is that the arguments must occur together. But for many triples, such as those relating to a person's occupation, a film's genre or a company's product type, the second argument is often not given explicitly.

In both cases, a relevant context set is a necessary but not sufficient condition for extracting the triple from text, since the context set may not express (even implicitly) the relation. Figure 2 shows the number of triples in our dataset that have a relevant context set with unary relations exclusively, binary relations exclusively and both unary and binary. The corpus extension threshold for the unary relations is 100.

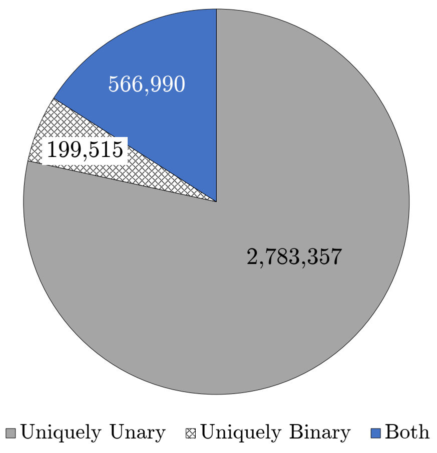


Figure 2: Triples with Relevant Context Sets Per-Relation Style

Although unary relations could also be viewed as types, we argue that it is preferable to consider them as relations. For example, if the unary relation *lives_in*:UNITED_STATES is represented as the type US-PERSON, it has no structured relationship to the type US-COMPANY (*based_in*:UNITED_STATES). So the inference rule that companies tend to employ people who live in the countries they are based in $(\langle company\ employs\ person\rangle \wedge \langle company\ based\_in\ country\rangle \Rightarrow \langle person\ lives\_in\ country\rangle)$ is not representable.

4 Training and Using Unary Relation Classifiers

A unary relation extraction system is a multi-class, multi-label classifier that takes an entity as input and returns its probability as a slot filler for each relation. In this paper, we represent each entity by the set of contexts (sentences in our experiments) where their mentions have been located; we call them context sets.

The process of training and applying a KBP system using unary relations is outlined step-by-step below.

- Build a set of unary relations that have a corpus extension above some threshold.
- Locate the entities from the knowledge graph in text.
- Create a context set for each entity from all the sentences that mention the entity.
- Label the context set with the unary relations (if any) for the entity. The negatives for each unary relation will be all the entities where that unary relation is not true.
- Train a model to determine the unary relations for any given entity from its context set.
- Apply the model to all the entities in the corpus, including those that do not exist in the knowledge graph.
- Convert the extracted unary relations back to binary relations and add to the knowledge graph as new edges. Any new entities are added to the knowledge graph as new nodes.

A closer look to the generated training data can provide insight in the value of unary relations for distant supervision.

Below are example binary contexts relating an organization to a country. The two arguments are shown in bold. Some contexts where two entities occur together (relevant contexts) will imply a relation between them, while others will not. In the first context, **Philippines** and **Eagle Cement** are not textually related. While in the second context, **Dyna Management Services** is explicitly stated to be located in **Bermuda**.

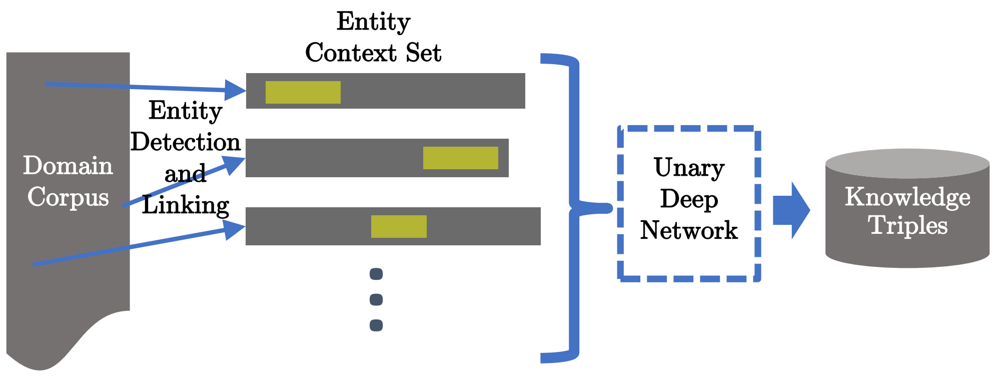


Figure 3: Unary Relational Knowledge Induction Architecture Overview

> The company competes with Holcim **Philippines**, the local unit of Swiss company LafargeHolcim, and **Eagle Cement**, a company backed by diversified local conglomerate San Miguel which is aggressively expanding into infrastructure.
>
> ... said Richmond, who is vice president of **Dyna Management Services**, a **Bermuda**-based insurance management company.

On the other hand, there are many triples that have no relevant context using binary extraction, but can be supported with unary extraction. **JB Hi-Fi** is a company located in **Australia**, (unary relation *hasLocation*:AUSTRALIA). Although "JB Hi-Fi" never occurs together with "Australia" in our corpus, we can gather implicit textual evidence for this relation from its unary relation context sets. Furthermore, even cases where there is a relevant binary context set, the contexts may not provide enough or any textual support for the relation, while the unary context sets might.

> Woolworths, Coles owner Wesfarmers, **JB Hi-Fi** and Harvey Norman were also trading higher.
>
> **JB Hi-Fi** in talks to buy The Good Guys
>
> In equities news, protective glove and condom maker Ansell and **JB Hi-Fi** are slated to post half year results, while Bitcoin group is expected to list on ASX.

The key indicators are: "ASX", which is an Australian stock exchange, and the other Australian businesses mentioned, such as Woolworths, Wesfarmers, Harvey Norman, The Good Guys, Ansell and Bitcoin group. There is no strict logical entailment, indicating **JB Hi-Fi** is located in Australia, instead there is textual evidence that makes it probable.

5 Architecture for Unary Relations

Figure 3 illustrates the overall architecture. First an Entity Detection and Linking system identifies occurrences in text of entities that are or should be in the knowledge base. Second, the contexts (here we use a sentence as the unit of context) for each entity are then gathered into an entity context set. This context set provides all the sentences that contain a mention of a particular entity and is the textual evidence for what triples are true for the entity. Third, the context set is then fed into a deep neural network, given in Figure 4. The output of the network is a set of predicted triples that can be added to the knowledge base.

Figure 4 shows the architecture of the deep learning model for unary relation based KBP. From an entity context set, each sentence is projected into a vector space using a piecewise convolutional neural network (Zeng et al., 2015). The sentence vectors are then aggregated using a Network-in-Network layer (NiN) (Lin et al., 2013).

The sentence-to-vector portion of the neural architecture begins by looking up the words in a word embedding table. The word embeddings are initialized with word2vec (Mikolov et al., 2013) and updated during training. The position of each word relative to the entity is also looked up in a position embedding table. Each word vector is concatenated with its position vector to produce each word representation vector. A piecewise max-pooled convolution (PCNN) is applied over

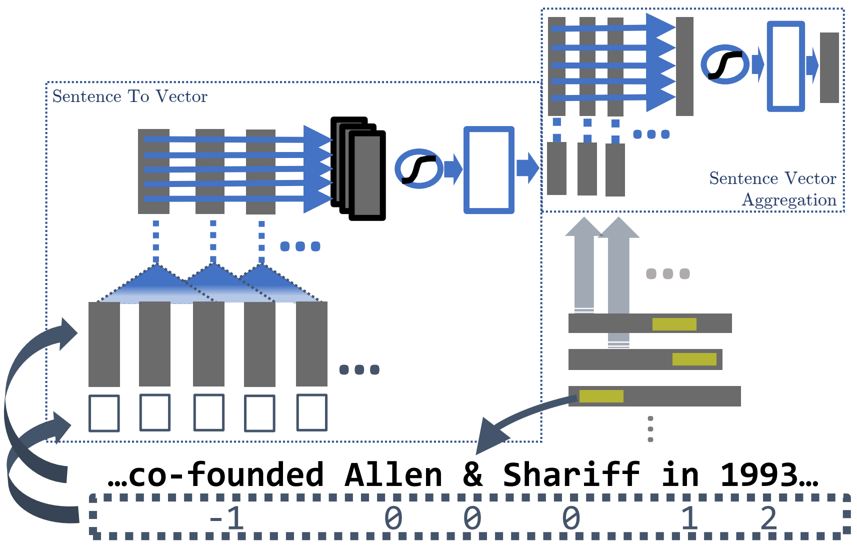

Sentence To Vector
Sentence Vector Aggregation
…co-founded Allen & Shariff in 1993…
-1 0 0 0 1 2

Figure 4: Deep Learning Architecture for Unary Relations

the resulting sentence matrix, with the pieces defined by the position of the entity argument: before the entity, the entity, and after the entity. A fully connected layer then produces the sentence vector representation. This is a refinement of the Neural Relation Extraction (NRE) (Lin et al., 2016) approach to sentence-to-vector mapping. The presence of only a single argument simply reduces from two position encoding vectors to one. The fully connected layer over the PCNN is an addition.

The sentence vector aggregation portion of the neural architecture uses a Network-in-Network over the sentence vectors. Network-in-Network (NiN) (Lin et al., 2013) is an approach of 1x1 CNNs to image processing. The width-1 CNN we use for mention aggregation is an adaptation to a set of sentence vectors. The result is max-pooled and put through a fully connected layer to produce the score for each unary relation. Unlike a maximum aggregation used in many previous works (Riedel et al., 2010; Zeng et al., 2015) for binary relation extraction the evidence from many contexts can be combined to produce a prediction. Unlike attention-based pooling also used previously for binary relation extraction (Lin et al., 2016), the different contexts can contribute to different aspects, not just different degrees. For example, a prediction that a city is in France might depend on the conjunction of several facets of textual evidence linking the city to the French language, the Euro, and Norman history.

In contrast, the common maximum aggregation approach is to move the final prediction layer to the sentence-to-vector modules and then aggregate by max-pooling the sentence level predictions. This aggregation strategy means that only the sentence most strongly indicating the relation contributes to its prediction. We measured the impact of the Network-in-Network sentence vector aggregation approach on the validation set. Relative to Network-in-Network aggregation and using the same hyperparameters, a maximum aggregation strategy gets two percent lower precision at one thousand: 66.55% compared to 68.49%.

There are 790 unary relations with at least one thousand positives in our benchmark. To speed the training, we divided these into eight sets of approximately 100 relations each and trained the models for them in parallel. Unary relations based on the same binary relation were grouped together to share useful learned representations. The resulting split also put similar numbers of positive examples in the training set for each model.

Training continued until no improvement was found on the validation set. This occurred at between five and nine epochs. All eight models were trained with the hyperparameters in Table 1. Dropout was applied on the penultimate layer, the max-pooled NiN.

Based on validation set performance, we found that when larger numbers of relations are trained together the NiN filters and sentence vector dimension must be increased. Of all the hyperparameters, the training time is most sensitive to the

Hyperparameter	Value
word embedding	50
position embedding	5
PCNN filters	1000
PCNN filter width	3
sentence vector	400
NiN filters	400
dropout	0.5
learnRate	0.003
decay multiplier	0.95
batch size	16
optimizer	SGD

Table 1: Hyperparameters used

number of PCNN filters, since these are applied to every sentence in a context set. We found major improvements moving from the 230 filters used for NRE to 1000 filters, but less improvement or no improvement to increases beyond that.

6 Benchmark

Large KBs and corpora are needed to train KBP systems in order to collect enough mentions for each relation. However, most of the existing Knowledge Base Population tasks are small in size (e.g. NYT-FB (Riedel et al., 2010) and TAC-KBP[3]) or focused on title-oriented-documents which are not available for most domains (e.g. WikiReading (Hewlett et al., 2016)). Therefore, we needed to create a new web-scale knowledge base population benchmark that we called *CC-DBP*[4]. It combines the text of Common Crawl[5] with the triples from 298 frequent relations in DBpedia (Auer et al., 2007). Mentions of DBpedia entities are located in text by gazetteer matching of the preferred label. We use the October 2017 Common Crawl and the most recent (2016-10) version of DBpedia, in both cases limited to English.

We divided the entity pairs into training, validation and test sets with a 80%, 10%, 10% split. All triples for a given entity pair are in one of the three splits. This split increases the challenge, since many relations could be used to predict others (such as birthPlace implying nationality). The task is to generate new triples for each relation and rank them according to their probability. We show the precision / recall curves and focus on the relative area under the curves to evaluate the quality of different systems.

Figure 5 shows the distribution of triples with relevant unary context sets per relation type. The relations giving rise to the most triples are high level relations such as hasLocation, a super-relation comprised of the sub-relations: country, state, city, headquarter, hometown, birthPlace, deathPlace, and others. Interestingly there are 165 years with enough people born in them to produce unary relations. While these all will have at least 100 relevant context sets, typically the context sets do not have textual evidence for any birth year. Perhaps most importantly, there are a large number of diverse relations that are suitable for a unary KBP approach. This indicates the broad applicability of our method.

To test what improvement can be found by incorporating unary relations into KBP, we combine the output of a state-of-the-art binary relation extraction system with our unary relation extraction system. For binary relation extraction, we use a slightly altered version of the PCNN model from NRE (Lin et al., 2016), with the addition of a fully connected layer for each sentence representation before the max-pooled aggregation over relation predictions. We found this refinement to perform slightly better in NYT-FB (Riedel et al., 2010), a standard dataset for distantly supervised relation extraction.

The binary and unary systems are trained from their relevant context sets to predict the triples in train. The validation set is used to tune hyper-parameters and choose a stopping point for training. We combine the output of the two systems by, for each triple, taking the highest confidence from each system.

7 Evaluation

Figure 6 shows the precision-recall curves for unary only, binary only and the combined system. The unary and binary systems alone achieve similar performance. But they are effective at very different triples. This is shown in the large gains from combining these complementary approaches. For example, at 0.5 precision, the combined approach has a recall of more than double (15,750 vs 7,400) compared to binary alone, which represents over 100% relative improvement.

The recall is given as a triple count rather than

[3]https://tac.nist.gov/
[4]https://github.com/IBM/cc-dbp
[5]http://commoncrawl.org

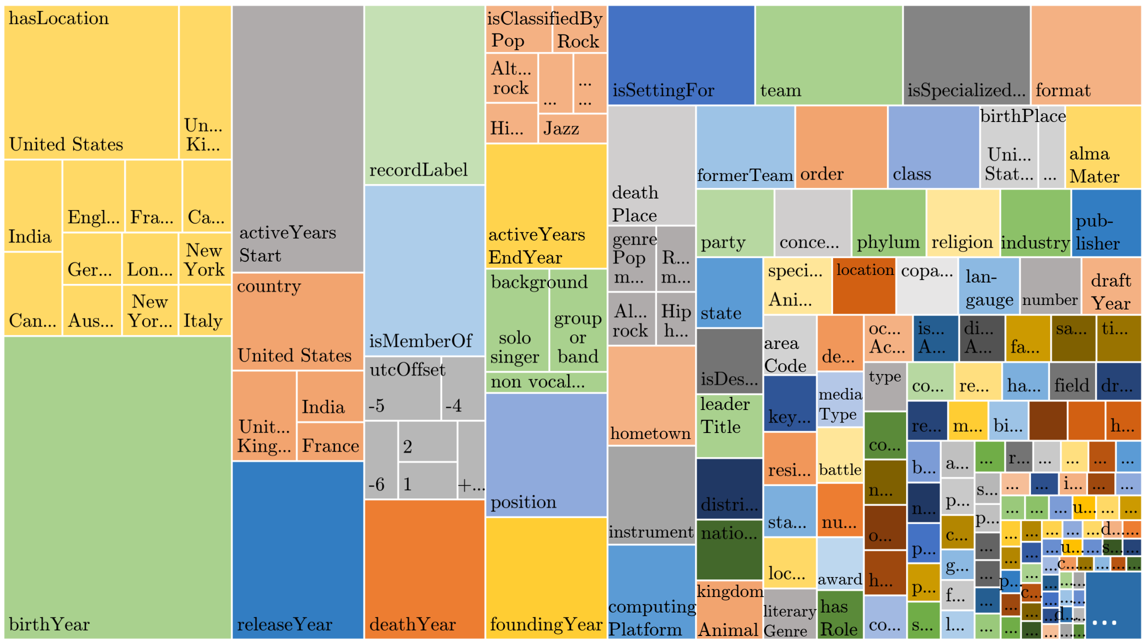


Figure 5: Distribution of Unary Relation Counts

a percentage. Traditional attempts to measure the recall of KBP systems use the set of all triples explicitly stated in text for the denominator of recall. This is unsuitable for evaluating our approach because the system is able to make probabilistic predictions based on implicit and partial textual evidence, thus producing correct triples outside the classic recall basis.

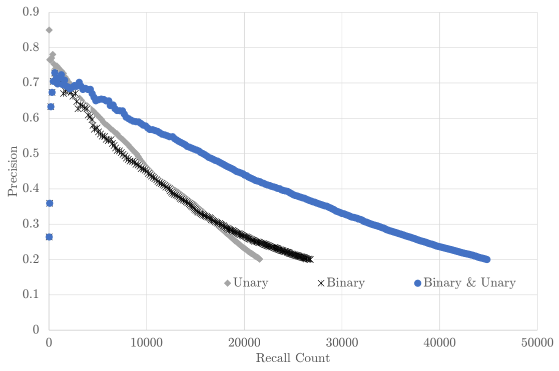


Figure 6: Precision Recall Curves for KBP

7.1 Saliency Maps

To gain some insight into how the unary KBP system is able to extract implicit knowledge we turn to saliency maps (Simonyan et al., 2014). By finding the derivative of a particular prediction with respect to the inputs, we can discover a locally linear approximation of how much each part of the input contributed (Zeiler and Fergus, 2014).

Cold Lake Provincial Park (Alberta, Canada) is mentioned in two sentences in the Common Crawl English text. The unary relational knowledge induction system predicts *hasLocation*:CANADA with the highest confidence (over 90%). Both sentences contribute to the decision. We see high weight from words including "cold", "provincial" and "french". A handful of countries have "provincial parks" including Argentina, Belgium, South Africa and Canada. Belgium and Canada have substantial French speaking populations and Canada has by far the coldest climate.

- located within 10 minutes of cold lake with quick access to OOV ridge ski hill , **cold** **lake** **provincial** **park** and french bay .
- welcome to **cold** **lake** **provincial** **park** on average 4.00 pages are viewed each , by the estimated 959 daily visitors .

Rock Kills Kid is a band mentioned twice in the corpus. From this context set, the relation *background*:GROUP_OR_BAND is predicted with high confidence. The fact that "Kid" occurs in the name of the entity seems to be important in identifying it as a musical group. The first sentence also draws focus to the band's connection to rock and pop.

While the second sentence seems to recognize the *band* - *song* (*year*) pattern as well as the comparison to Duran Duran.

- the latest stylish pop synth band is **rock kills kid** .
- **rock kills kid** - are you nervous ? (2006) who ever thought duran duran would become so influential ?

The Japanese singer-songwriter Masaki Haruna, aka Klaha is mentioned twice in the corpus. From this context set, the relation *background*:SOLO_SINGER is predicted with high confidence. The first sentence clearly establishes the connection to music while the second indicates that Klaha is a solo artist. The conjunction of these two facets, accomplished through the context vector aggregation using NiN permits the conclusion of SOLO_SINGER.

- tvk music chat interview **klaha** parade .
- **klaha** tvk music chat OOV red scarf interview the tv - k folks did after **klaha** went solo .

8 Conclusions

In this paper we presented a new methodology to identify relations between entities in text. Our approach, focusing on unary relations, can greatly improve the recall in automatic construction and updating of knowledge bases by making use of implicit and partial textual markers. Our method is extremely effective and complement very nicely existing binary relation extraction methods for KBP.

This is just the first step in our wider research program on KBP, whose goal is to improve recall by identifying implicit information from texts. First of all, we plan to explore the use of more advanced forms of entity detection and linking, including propagating features from the EDL system forward for both unary and binary deep models. In addition we plan to exploit unary and binary relations as source of evidence to bootstrap a probabilistic reasoning approach, with the goal of leveraging constraints from the KB schema such as domain, range and taxonomies. We will also integrate the new triples gathered from textual evidence with new triples predicted from existing KB relationships by knowledge base completion.

References

Sren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *In 6th Intl Semantic Web Conference, Busan, Korea*. Springer, pages 11–15.

H Chang, M Abdurrahman, Ao Liu, J Tian-Zheng Wei, Aaron Traylor, Ajay Nagesh, Nicholas Monath, Patrick Verga, Emma Strubell, and Andrew McCallum. 2016. Extracting multilingual relations under limited resources: Tac 2016 cold-start kb construction and slot-filling using compositional universal schema. *Proceedings of TAC* .

Philipp Cimiano and Johanna Völker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*. pages 3519–3530.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1774–1784.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *Proceedings of the Conference on Association for Computational Linguistics*.

Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* .

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 1003–1011. http://dl.acm.org/citation.cfm?id=1690219.1690287.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics* 5:101–115.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases* pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 74–84.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, pages 455–465.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1298–1307.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1785–1794.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, pages 818–833.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*. pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. pages 2335–2344.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Incorporating relation paths in neural relation extraction. In *CoRR*.

Yuhao Zhang, Arun Chaganty, Ashwin Paranjape, Danqi Chen, Jason Bolton, Peng Qi, and Christopher D Manning. 2016. Stanford at tac kbp 2016: Sealing pipeline leaks and understanding chinese. *Proceedings of TAC* .

Improving Entity Linking by Modeling Latent Relations between Mentions

Phong Le[1] **and Ivan Titov**[1,2]
[1]University of Edinburgh [2]University of Amsterdam
{ple,ititov}@inf.ed.ac.uk

Abstract

Entity linking involves aligning textual mentions of named entities to their corresponding entries in a knowledge base. Entity linking systems often exploit relations between textual mentions in a document (e.g., coreference) to decide if the linking decisions are compatible. Unlike previous approaches, which relied on supervised systems or heuristics to predict these relations, we treat relations as latent variables in our neural entity-linking model. We induce the relations without any supervision while optimizing the entity-linking system in an end-to-end fashion. Our multi-relational model achieves the best reported scores on the standard benchmark (AIDA-CoNLL) and substantially outperforms its relation-agnostic version. Its training also converges much faster, suggesting that the injected structural bias helps to explain regularities in the training data.

1 Introduction

Named entity linking (NEL) is the task of assigning entity mentions in a text to corresponding entries in a knowledge base (KB). For example, consider Figure 1 where a mention "World Cup" refers to a KB entity FIFA_WORLD_CUP. NEL is often regarded as crucial for natural language understanding and commonly used as preprocessing for tasks such as information extraction (Hoffmann et al., 2011) and question answering (Yih et al., 2015).

Potential assignments of mentions to entities are regulated by semantic and discourse constraints. For example, the second and third occurrences of mention "England" in Figure 1 are coreferent and thus should be assigned to the same entity. Besides coreference, there are many other relations between entities which constrain or favor certain alignment configurations. For example, consider relation participant_in in Figure 1: if "World Cup" is aligned to the entity FIFA_WORLD_CUP then we expect the second "England" to refer to a football team rather than a basketball one.

NEL methods typically consider only coreference, relying either on off-the-shelf systems or some simple heuristics (Lazic et al., 2015), and exploit them in a pipeline fashion, though some (e.g., Cheng and Roth (2013); Ren et al. (2017)) additionally exploit a range of syntactic-semantic relations such as apposition and possessives. Another line of work ignores relations altogether and models the predicted sequence of KB entities as a bag (Globerson et al., 2016; Yamada et al., 2016; Ganea and Hofmann, 2017). Though they are able to capture some degree of coherence (e.g., preference towards entities from the same general domain) and are generally empirically successful, the underlying assumption is too coarse. For example, they would favor assigning all the occurrences of "England" in Figure 1 to the same entity.

We hypothesize that relations useful for NEL can be induced without (or only with little) domain expertise. In order to prove this, we encode relations as latent variables and induce them by optimizing the entity-linking model in an end-to-end fashion. In this way, relations between mentions in documents will be induced in such a way as to be beneficial for NEL. As with other recent approaches to NEL (Yamada et al., 2017; Ganea and Hofmann, 2017), we rely on representation learning and learn embeddings of mentions, contexts and relations. This further reduces the amount of human expertise required to construct the system and, in principle, may make it more portable across languages and domains.

Our multi-relational neural model achieves an

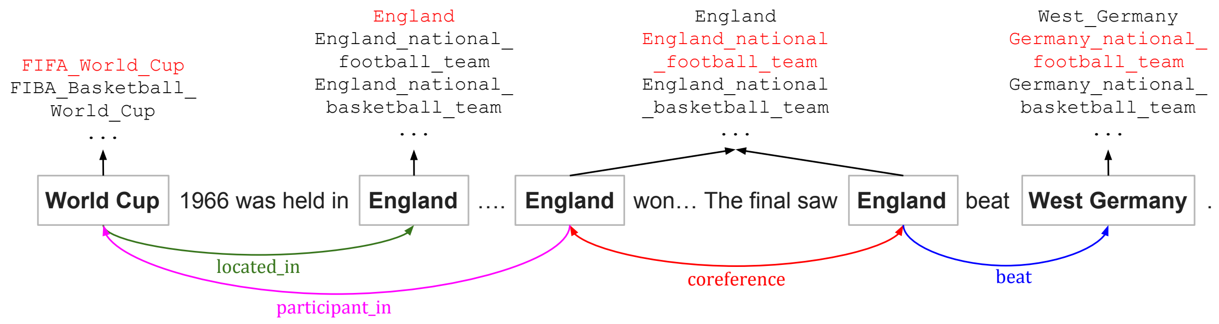

FIFA_World_Cup
FIBA_Basketball_World_Cup
...
England
England_national_football_team
England_national_basketball_team
...
England
England_national_football_team
England_national_basketball_team
...
West_Germany
Germany_national_football_team
Germany_national_basketball_team
...
World Cup
1966 was held in
England
....
England
won... The final saw
England
beat
West Germany
.
located_in
participant_in
coreference
beat

Figure 1: Example for NEL, linking each mention to an entity in a KB (e.g. "World Cup" to FIFA_WORLD_CUP rather than FIBA_BASKETBALL_WORLD_CUP). Note that the first and the second "England" are in different relations to "World Cup".

improvement of 0.85% F1 over the best reported scores on the standard AIDA-CoNLL dataset (Ganea and Hofmann, 2017). Substantial improvements over the relation-agnostic version show that the induced relations are indeed beneficial for NEL. Surprisingly its training also converges much faster: training of the full model requires ten times shorter wall-clock time than what is needed for estimating the simpler relation-agnostic version. This may suggest that the injected structural bias helps to explain regularities in the training data, making the optimization task easier. We qualitatively examine induced relations. Though we do not observe direct counterparts of linguistic relations, we, for example, see that some of the induced relations are closely related to coreference whereas others encode forms of semantic relatedness between the mentions.

2 Background and Related work

2.1 Named entity linking problem

Formally, given a document D containing a list of mentions $m_1, ..., m_n$, an entity linker assigns to each m_i an KB entity e_i or predicts that there is no corresponding entry in the KB (i.e., $e_i = \text{NILL}$).

Because a KB can be very large, it is standard to use an heuristic to choose potential candidates, eliminating options which are highly unlikely. This preprocessing step is called *candidate selection*. The task of a statistical model is thus reduced to choosing the best option among a smaller list of candidates $C_i = (e_{i1}, ..., e_{il_i})$. In what follows, we will discuss two classes of approaches tackling this problem: local and global modeling.

2.2 Local and global models

Local models rely only on local contexts of mentions and completely ignore interdependencies between the linking decisions in the document (these interdependencies are usually referred to as *coherence*). Let c_i be a local context of mention m_i and $\Psi(e_i, c_i)$ be a local score function. A local model then tackles the problem by searching for

$$e_i^* = \arg\max_{e_i \in C_i} \Psi(e_i, c_i) \quad (1)$$

for each $i \in \{1, ..., n\}$ (Bunescu and Paşca, 2006; Lazic et al., 2015; Yamada et al., 2017).

A *global* model, besides using local context within $\Psi(e_i, c_i)$, takes into account entity coherency. It is captured by a coherence score function $\Phi(E, D)$:

$$E^* = \arg\max_{E \in C_1 \times ... \times C_n} \sum_{i=1}^{n} \Psi(e_i, c_i) + \Phi(E, D)$$

where $E = (e_1, ..., e_n)$. The coherence score function, in the simplest form, is a sum over all pairwise scores $\Phi(e_i, e_j, D)$ (Ratinov et al., 2011; Huang et al., 2015; Chisholm and Hachey, 2015; Ganea et al., 2016; Guo and Barbosa, 2016; Globerson et al., 2016; Yamada et al., 2016), resulting in:

$$E^* = \arg\max_{E \in C_1 \times ... \times C_n} \sum_{i=1}^{n} \Psi(e_i, c_i) + \sum_{i \neq j} \Phi(e_i, e_j, D) \quad (2)$$

A disadvantage of global models is that exact decoding (Equation 2) is NP-hard (Wainwright et al., 2008). Ganea and Hofmann (2017) overcome this using loopy belief propagation (LBP), an approximate inference method based on message passing (Murphy et al., 1999). Globerson et al. (2016) propose a *star model* which approximates the decoding problem in Equation 2 by approximately decomposing it into n decoding problems, one per each e_i.

2.3 Related work

Our work focuses on modeling pairwise score functions Φ and is related to previous approaches in the two following aspects.

Relations between mentions

A relation widely used by NEL systems is *coreference*: two mentions are coreferent if they refer to the same entity. Though, as we discussed in Section 1, other linguistic relations constrain entity assignments, only a few approaches (e.g., Cheng and Roth (2013); Ren et al. (2017)), exploit any relations other than coreference. We believe that the reason for this is that predicting and selecting relevant (often semantic) relations is in itself a challenging problem.

In Cheng and Roth (2013), relations between mentions are extracted using a labor-intensive approach, requiring a set of hand-crafted rules and a KB containing relations between entities. This approach is difficult to generalize to languages and domains which do not have such KBs or the settings where no experts are available to design the rules. We, in contrast, focus on automating the process using representation learning.

Most of these methods relied on relations predicted by external tools, usually a coreference system. One notable exception is Durrett and Klein (2014): they use a joint model of entity linking and coreference resolution. Nevertheless their coreference component is still supervised, whereas our relations are latent even at training time.

Representation learning

How can we define local score functions Ψ and pairwise score functions Φ? Previous approaches employ a wide spectrum of techniques.

At one extreme, extensive feature engineering was used to define useful features. For example, Ratinov et al. (2011) use cosine similarities between Wikipedia titles and local contexts as a feature when computing the local scores. For pairwise scores they exploit information about links between Wikipedia pages.

At the other extreme, feature engineering is almost completely replaced by representation learning. These approaches rely on pretrained embeddings of words (Mikolov et al., 2013; Pennington et al., 2014) and entities (He et al., 2013; Yamada et al., 2017; Ganea and Hofmann, 2017) and often do not use virtually any other hand-crafted features. Ganea and Hofmann (2017) showed that such an approach can yield SOTA accuracy on a standard benchmark (AIDA-CoNLL dataset). Their local and pairwise score functions are

$$\Psi(e_i, c_i) = \mathbf{e}_i^T \mathbf{B} f(c_i)$$
$$\Phi(e_i, e_j, D) = \frac{1}{n-1} \mathbf{e}_i^T \mathbf{R} \mathbf{e}_j \qquad (3)$$

where $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^d$ are the embeddings of entity e_i, e_j, $\mathbf{B}, \mathbf{R} \in \mathbb{R}^{d \times d}$ are diagonal matrices. The mapping $f(c_i)$ applies an attention mechanism to context words in c_i to obtain a feature representations of context ($f(c_i) \in \mathbb{R}^d$).

Note that the global component (the pairwise scores) is agnostic to any relations between entities or even to their ordering: it models $e_1, ..., e_n$ simply as a bag of entities. Our work is in line with Ganea and Hofmann (2017) in the sense that feature engineering plays no role in computing local and pair-wise scores. Furthermore, we argue that pair-wise scores should take into account relations between mentions which are represented by *relation embeddings*.

3 Multi-relational models

3.1 General form

We assume that there are K latent relations. Each relation k is assigned to a mention pair (m_i, m_j) with a non-negative weight ('confidence') α_{ijk}. The pairwise score (m_i, m_j) is computed as a weighted sum of relation-specific pairwise scores (see Figure 2, top):

$$\Phi(e_i, e_j, D) = \sum_{k=1}^{K} \alpha_{ijk} \Phi_k(e_i, e_j, D)$$

$\Phi_k(e_i, e_j, D)$ can be any pairwise score function, but here we adopt the one from Equation 3. Namely, we represent each relation k by a diagonal matrix $\mathbf{R}_k \in \mathbb{R}^{d \times d}$, and

$$\Phi_k(e_i, e_j, D) = \mathbf{e}_i^T \mathbf{R}_k \mathbf{e}_j$$

The weights α_{ijk} are normalized scores:

$$\alpha_{ijk} = \frac{1}{Z_{ijk}} \exp\left\{ \frac{f^T(m_i, c_i)\mathbf{D}_k f(m_j, c_j)}{\sqrt{d}} \right\} \quad (4)$$

where Z_{ijk} is a normalization factor, $f(m_i, c_i)$ is a function mapping (m_i, c_i) onto $\mathbb{R}^d$, and $\mathbf{D}_k \in \mathbb{R}^{d\times d}$ is a diagonal matrix.

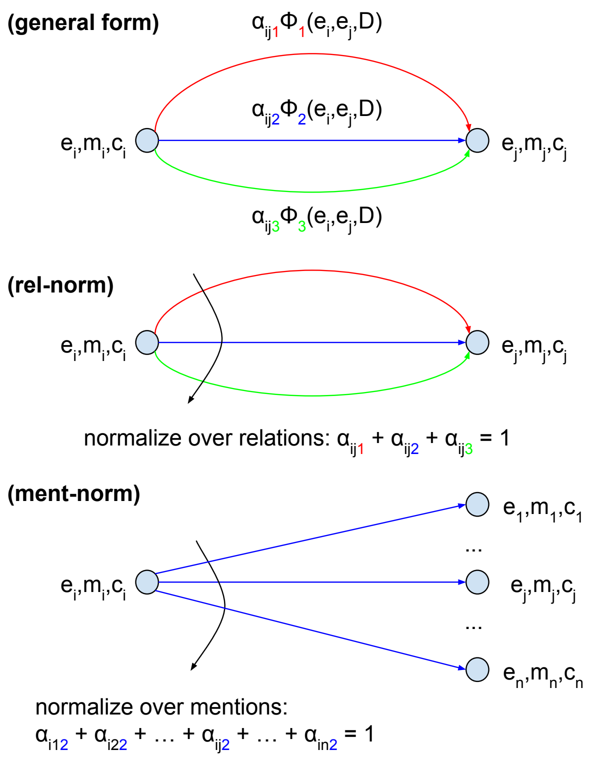


Figure 2: Multi-relational models: general form (top), rel-norm (middle) and ment-norm (bottom). Each color corresponds to one relation.

In our experiments, we use a single-layer neural network as f (see Figure 3) where c_i is a concatenation of the average embedding of words in the left context with the average embedding of words in the right context of the mention.[1]

As α_{ijk} is indexed both by mention index j and relation index k, we have two choices for Z_{ijk}: normalization over relations and normalization over mentions. We consider both versions of the model.

[1]We also experimented with LSTMs but we could not prevent them from severely overfitting, and the results were poor.

3.2 Rel-norm: Relation-wise normalization

For rel-norm, coefficients α_{ijk} are normalized over relations k, in other words,

$$Z_{ijk} = \sum_{k'=1}^{K} \exp\left\{ \frac{f^T(m_i, c_i)\mathbf{D}_{k'} f(m_j, c_j)}{\sqrt{d}} \right\}$$

so that $\sum_{k=1}^K \alpha_{ijk} = 1$ (see Figure 2, middle). We can also re-write the pairwise scores as

$$\Phi(e_i, e_j, D) = \mathbf{e}_i^T \mathbf{R}_{ij} \mathbf{e}_j \quad (5)$$

where $\mathbf{R}_{ij} = \sum_{k=1}^K \alpha_{ijk}\mathbf{R}_k$.

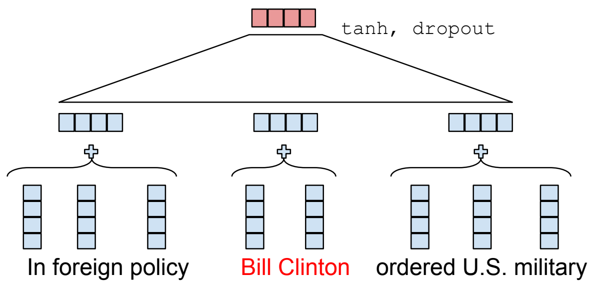


Figure 3: Function $f(m_i, c_i)$ is a single-layer neural network, with tanh activation function and a layer of dropout on top.

Intuitively, α_{ijk} is the probability of assigning a k-th relation to a mention pair (m_i, m_j). For every pair rel-norm uses these probabilities to choose one relation from the pool and relies on the corresponding relation embedding $\mathbf{R}_k$ to compute the compatibility score.

For $K = 1$ rel-norm reduces (up to a scaling factor) to the bag-of-entities model defined in Equation 3.

In principle, instead of relying on the linear combination of relation embeddings matrices $\mathbf{R}_k$, we could directly predict a context-specific relation embedding $\mathbf{R}_{ij} = diag\{g(m_i, c_i, m_j, c_j)\}$ where g is a neural network. However, in preliminary experiments we observed that this resulted in overfitting and poor performance. Instead, we choose to use a small fixed number of relations as a way to constrain the model and improve generalization.

3.3 Ment-norm: Mention-wise normalization

We can also normalize α_{ijk} over j:

$$Z_{ijk} = \sum_{\substack{j'=1 \\ j'\neq i}}^{n} \exp\left\{ \frac{f^T(m_i, c_i)\mathbf{D}_k f(m_{j'}, c_{j'})}{\sqrt{d}} \right\}$$

This implies that $\sum_{j=1,j\neq i}^{n} \alpha_{ijk} = 1$ (see Figure 2, bottom). If we rewrite the pairwise scores as

$$\Phi(e_i, e_j, D) = \sum_{k=1}^{K} \alpha_{ijk} \mathbf{e}_i^T \mathbf{R}_k \mathbf{e}_j, \qquad (6)$$

we can see that Equation 3 is a special case of ment-norm when $K = 1$ and $\mathbf{D}_1 = \mathbf{0}$. In other words, Ganea and Hofmann (2017) is our mono-relational ment-norm with uniform α.

The intuition behind ment-norm is that for each relation k and mention m_i, we are looking for mentions related to m_i with relation k. For each pair of m_i and m_j we can distinguish two cases: (i) α_{ijk} is small for all k: m_i and m_j are not related under any relation, (ii) α_{ijk} is large for one or more k: there are one or more relations which are predicted for m_i and m_j.

In principle, rel-norm can also indirectly handle both these cases. For example, it can master (i) by dedicating a distinct 'none' relation to represent lack of relation between the two mentions (with the corresponding matrix $\mathbf{R}_k$ set to $\mathbf{0}$). Though it cannot assign large weights (i.e., close to 1) to multiple relations (as needed for (ii)), it can in principle use the 'none' relation to vary the probability mass assigned to the rest of relations across mention pairs, thus achieving the same effect (up to a multiplicative factor). Nevertheless, in contrast to ment-norm, we do not observe this behavior for rel-norm in our experiments: the inductive basis seems to disfavor such configurations.

Ment-norm is in line with the current trend of using the attention mechanism in deep learning (Bahdanau et al., 2014), and especially related to multi-head attention of Vaswani et al. (2017). For each mention m_i and for each k, we can interpret α_{ijk} as the probability of choosing a mention m_j among the set of mentions in the document. Because here we have K relations, each mention m_i will have maximally K mentions (i.e. heads in terminology of Vaswani et al. (2017)) to focus on. Note though that they use multi-head attention for choosing input features in each layer, whereas we rely on this mechanism to compute pairwise scoring functions for the structured output (i.e. to compute potential functions in the corresponding undirected graphical model, see Section 3.4).

Mention padding

A potentially serious drawback of ment-norm is that the model uses all K relations even in cases where some relations are inapplicable. For example, consider applying relation *coreference* to mention "West Germany" in Figure 1. The mention is non-anaphoric: there are no mentions co-referent with it. Still the ment-norm model has to distribute the weight across the mentions. This problem occurs because of the normalization $\sum_{j=1,j\neq i}^{n} \alpha_{ijk} = 1$. Note that this issue does not affect standard applications of attention: normally the attention-weighted signal is input to another transformation (e.g., a flexible neural model) which can then disregard this signal when it is useless. This is not possible within our model, as it simply uses α_{ijk} to weight the bilinear terms without any extra transformation.

Luckily, there is an easy way to circumvent this problem. We add to each document a padding mention m_{pad} linked to a padding entity e_{pad}. In this way, the model can use the padding mention to damp the probability mass that the other mentions receive. This method is similar to the way some mention-ranking coreference models deal with non-anaphoric mentions (e.g. Wiseman et al. (2015)).

3.4 Implementation

Following Ganea and Hofmann (2017) we use Equation 2 to define a conditional random field (CRF). We use the local score function identical to theirs and the pairwise scores are defined as explained above:

$$q(E|D) \propto \exp\left\{\sum_{i=1}^{n} \Psi(e_i, c_i) + \sum_{i\neq j} \Phi(e_i, e_j, D)\right\}$$

We also use max-product loopy belief propagation (LBP) to estimate the max-marginal probability

$$\hat{q}_i(e_i|D) \approx \max_{\substack{e_1,\ldots,e_{i-1}\\ e_{i+1},\ldots,e_n}} q(E|D)$$

for each mention m_i. The final score function for m_i is given by:

$$\rho_i(e) = g(\hat{q}_i(e|D), \hat{p}(e|m_i))$$

where g is a two-layer neural network and $\hat{p}(e|m_i)$ is the probability of selecting e conditioned only on m_i. This probability is computed by mixing mention-entity hyperlink count statistics from Wikipedia, a large Web corpus and YAGO.[2]

[2]See Ganea and Hofmann (2017, Section 6).

We minimize the following ranking loss:

$$L(\theta) = \sum_{D \in \mathcal{D}} \sum_{m_i \in D} \sum_{e \in C_i} h(m_i, e) \tag{7}$$

$$h(m_i, e) = \max\big(0, \gamma - \rho_i(e_i^*) + \rho_i(e)\big)$$

where θ are the model parameters, $\mathcal{D}$ is a training dataset, and e_i^* is the ground-truth entity. Adam (Kingma and Ba, 2014) is used as an optimizer.

For ment-norm, the padding mention is treated like any other mentions. We add $\mathbf{f}_{pad} = f(m_{pad}, c_{pad})$ and $\mathbf{e}_{pad} \in \mathbb{R}^d$, an embedding of e_{pad}, to the model parameter list, and tune them while training the model.

In order to encourage the models to explore different relations, we add the following regularization term to the loss function in Equation 7:

$$\lambda_1 \sum_{i,j} \text{dist}(\mathbf{R}_i, \mathbf{R}_j) + \lambda_2 \sum_{i,j} \text{dist}(\mathbf{D}_i, \mathbf{D}_j)$$

where λ_1, λ_2 are set to -10^{-7} in our experiments, $\text{dist}(\mathbf{x}, \mathbf{y})$ can be any distance metric. We use:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \left\| \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \right\|_2$$

Using this regularization to favor diversity is important as otherwise relations tend to collapse: their relation embeddings $\mathbf{R}_k$ end up being very similar to each other.

4 Experiments

We evaluated four models: (i) *rel-norm* proposed in Section 3.2; (ii) *ment-norm* proposed in Section 3.3; (iii) *ment-norm ($K = 1$)*: the mono-relational version of ment-norm; and (iv) *ment-norm (no pad)*: the ment-norm without using mention padding. Recall also that our mono-relational (i.e. $K = 1$) rel-norm is equivalent to the relation-agnostic baseline of Ganea and Hofmann (2017).

We implemented our models in PyTorch and run experiments on a Titan X GPU. The source code and trained models will be publicly available at `https://github.com/lephong/mulrel-nel`.

4.1 Setup

We set up our experiments similarly to those of Ganea and Hofmann (2017), run each model 5 times, and report average and 95% confidence interval of the standard micro F1 score (aggregates over all mentions).

Datasets

For *in-domain* scenario, we used AIDA-CoNLL dataset[3] (Hoffart et al., 2011). This dataset contains AIDA-train for training, AIDA-A for dev, and AIDA-B for testing, having respectively 946, 216, and 231 documents. For *out-domain* scenario, we evaluated the models trained on AIDA-train, on five popular test sets: MSNBC, AQUAINT, ACE2004, which were cleaned and updated by Guo and Barbosa (2016); WNED-CWEB (CWEB), WNED-WIKI (WIKI), which were automatically extracted from ClueWeb and Wikipedia (Guo and Barbosa, 2016; Gabrilovich et al., 2013). The first three are small with 20, 50, and 36 documents whereas the last two are much larger with 320 documents each. Following previous works (Yamada et al., 2016; Ganea and Hofmann, 2017), we considered only mentions that have entities in the KB (i.e., Wikipedia).

Candidate selection

For each mention m_i, we selected 30 top candidates using $\hat{p}(e|m_i)$. We then kept 4 candidates with the highest $\hat{p}(e|m_i)$ and 3 candidates with the highest scores $\mathbf{e}^T \left(\sum_{w \in d_i} \mathbf{w}\right)$, where $\mathbf{e}, \mathbf{w} \in \mathbb{R}^d$ are entity and word embeddings, d_i is the 50-word window context around m_i.

Hyper-parameter setting

We set $d = 300$ and used GloVe (Pennington et al., 2014) word embeddings trained on 840B tokens for computing f in Equation 4, and entity embeddings from Ganea and Hofmann (2017).[4] We use the following parameter values: $\gamma = 0.01$ (see Equation 7), the number of LBP loops is 10, the dropout rate for f was set to 0.3, the window size of local contexts c_i (for the pairwise score functions) is 6. For rel-norm, we initialized $\text{diag}(\mathbf{R}_k)$ and $\text{diag}(\mathbf{D}_k)$ by sampling from $\mathcal{N}(0, 0.1)$ for all k. For ment-norm, we did the same except that $\text{diag}(\mathbf{R}_1)$ was sampled from $\mathcal{N}(1, 0.1)$.

To select the best number of relations K, we considered all values of $K \leq 7$ ($K > 7$ would not fit in our GPU memory, as some of the documents are large). We selected the best ones based on the development scores: 6 for rel-norm, 3 for ment-norm, and 3 for ment-norm (no pad).

When training the models, we applied early stopping. For rel-norm, when the model reached

[3]TAC KBP datasets are no longer available.

[4]`https://github.com/dalab/deep-ed`

91% F1 on the dev set, [5] we reduced the learning rate from 10^{-4} to 10^{-5}. We then stopped the training when F1 was not improved after 20 epochs. We did the same for ment-norm except that the learning rate was changed at 91.5% F1.

Note that all the hyper-parameters except K and the turning point for early stopping were set to the values used by Ganea and Hofmann (2017). Systematic tuning is expensive though may have further increased the result of our models.

4.2 Results

Methods	Aida-B
Chisholm and Hachey (2015)	88.7
Guo and Barbosa (2016)	89.0
Globerson et al. (2016)	91.0
Yamada et al. (2016)	91.5
Ganea and Hofmann (2017)	92.22 ± 0.14
rel-norm	92.41 ± 0.19
ment-norm	$\mathbf{93.07} \pm 0.27$
ment-norm ($K = 1$)	92.89 ± 0.21
ment-norm (no pad)	92.37 ± 0.26

Table 1: F1 scores on AIDA-B (test set).

Table 1 shows micro F1 scores on AIDA-B of the SOTA methods and ours, which all use Wikipedia and YAGO mention-entity index. To our knowledge, ours are the only (unsupervisedly) inducing and employing more than one relations on this dataset. The others use only one relation, coreference, which is given by simple heuristics or supervised third-party resolvers. All four our models outperform any previous method, with ment-norm achieving the best results, 0.85% higher than that of Ganea and Hofmann (2017).

Table 2 shows micro F1 scores on 5 out-domain test sets. Besides ours, only Cheng and Roth (2013) employs several mention relations. Ment-norm achieves the highest F1 scores on MSNBC and ACE2004. On average, ment-norm's F1 score is 0.3% higher than that of Ganea and Hofmann (2017), but 0.2% lower than Guo and Barbosa (2016)'s. It is worth noting that Guo and Barbosa (2016) performs exceptionally well on WIKI, but substantially worse than ment-norm on all other datasets. Our other three models, however, have lower average F1 scores compared to the best previous model.

The experimental results show that ment-norm outperforms rel-norm, and that mention padding plays an important role.

[5]We chose the highest F1 that rel-norm always achieved without the learning rate reduction.

4.3 Analysis

Mono-relational v.s. multi-relational

For rel-norm, the mono-relational version (i.e., Ganea and Hofmann (2017)) is outperformed by the multi-relational one on AIDA-CoNLL, but performs significantly better on all five out-domain datasets. This implies that multi-relational rel-norm does not generalize well across domains.

For ment-norm, the mono-relational version performs worse than the multi-relational one on all test sets except AQUAINT. We speculate that this is due to multi-relational ment-norm being less sensitive to prediction errors. Since it can rely on multiple factors more easily, a single mistake in assignment is unlikely to have large influence on its predictions.

Oracle

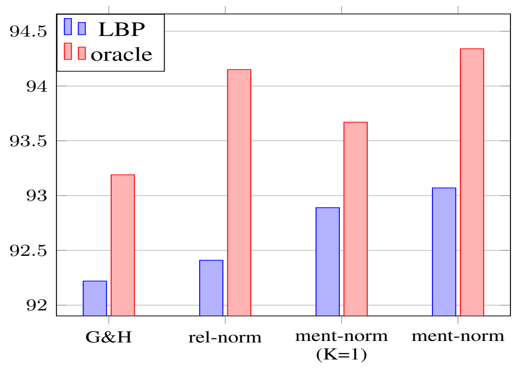


Figure 4: F1 on AIDA-B when using LBP and the oracle. G&H is Ganea and Hofmann (2017).

In order to examine learned relations in a more transparant setting, we consider an idealistic scenario where imperfection of LBP, as well as mistakes in predicting other entities, are taken out of the equation using an oracle. This oracle, when we make a prediction for mention m_i, will tell us the correct entity e^*_j for every other mentions $m_j, j \neq i$. We also used AIDA-A (development set) for selecting the numbers of relations for rel-norm and ment-norm. They are set to 6 and 3, respectively. Figure 4 shows the micro F1 scores.

Surprisingly, the performance of oracle rel-norm is close to that of oracle ment-norm, although without using the oracle the difference was substantial. This suggests that rel-norm is more sensitive to prediction errors than ment-norm. Ganea and Hofmann (2017), even with the help of the oracle, can only perform slightly better than LBP (i.e. non-oracle) ment-norm. This

Methods	MSNBC	AQUAINT	ACE2004	CWEB	WIKI	Avg
Milne and Witten (2008)	78	85	81	64.1	81.7	77.96
Hoffart et al. (2011)	79	56	80	58.6	63	67.32
Ratinov et al. (2011)	75	83	82	56.2	67.2	72.68
Cheng and Roth (2013)	90	**90**	86	67.5	73.4	81.38
Guo and Barbosa (2016)	92	87	88	77	**84.5**	**85.7**
Ganea and Hofmann (2017)	93.7 ± 0.1	88.5 ± 0.4	88.5 ± 0.3	**77.9** ± 0.1	77.5 ± 0.1	85.22
rel-norm	92.2 ± 0.3	86.7 ± 0.7	87.9 ± 0.3	75.2 ± 0.5	76.4 ± 0.3	83.67
ment-norm	**93.9** ± 0.2	88.3 ± 0.6	**89.9** ± 0.8	77.5 ± 0.1	<u>78.0</u> ± 0.1	<u>85.51</u>
ment-norm ($K = 1$)	93.2 ± 0.3	88.4 ± 0.4	88.9 ± 1.0	77.0 ± 0.2	77.2 ± 0.1	84.94
ment-norm (no pad)	93.6 ± 0.3	87.8 ± 0.5	**90.0** ± 0.3	77.0 ± 0.2	77.3 ± 0.3	85.13

Table 2: F1 scores on five out-domain test sets. Underlined scores show cases where the corresponding model outperforms the baseline.

suggests that its global coherence scoring component is indeed too simplistic. Also note that both multi-relational oracle models substantially outperform the two mono-relational oracle models. This shows the benefit of using more than one relations, and the potential of achieving higher accuracy with more accurate inference methods.

Relations

In this section we qualitatively examine relations that the models learned by looking at the probabilities α_{ijk}. See Figure 5 for an example. In that example we focus on mention "Liege" in the sentence at the top and study which mentions are related to it under two versions of our model: rel-norm (leftmost column) and ment-norm (rightmost column).

For rel-norm it is difficult to interpret the meaning of the relations. It seems that the first relation dominates the other two, with very high weights for most of the mentions. Nevertheless, the fact that rel-norm outperforms the baseline suggests that those learned relations encode some useful information.

For ment-norm, the first relation is similar to coreference: the relation prefers those mentions that potentially refer to the same entity (and/or have semantically similar mentions): see Figure 5 (left, third column). The second and third relations behave differently from the first relation as they prefer mentions having more distant meanings and are complementary to the first relation. They assign large weights to (1) "Belgium" and (2) "Brussels" but small weights to (4) and (6) "Liege". The two relations look similar in this example, however they are not identical in general. See a histogram of bucketed values of their weights in Figure 5 (right): their α have quite different distributions.

Complexity

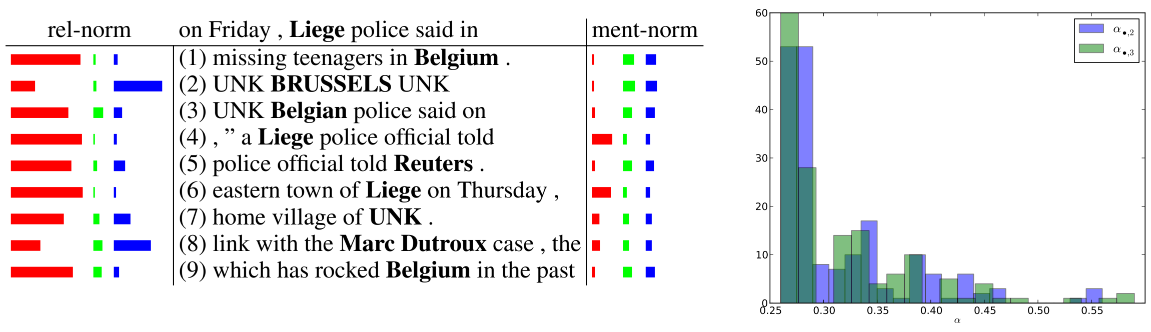
The complexity of rel-norm and ment-norm is linear in K, so in principle our models should be considerably more expensive than Ganea and Hofmann (2017). However, our models converge much faster than their relation-agnostic model: on average ours needs 120 epochs, compared to theirs 1250 epochs. We believe that the structural bias helps the model to capture necessary regularities more easily. In terms of wall-clock time, our model requires just under 1.5 hours to train, that is ten times faster than the relation agnostic model (Ganea and Hofmann, 2017). In addition, the difference in testing time is negligible when using a GPU.

5 Conclusion and Future work

We have shown the benefits of using relations in NEL. Our models consider relations as latent variables, thus do not require any extra supervision. Representation learning was used to learn relation embeddings, eliminating the need for extensive feature engineering. The experimental results show that our best model achieves the best reported F1 on AIDA-CoNLL with an improvement of 0.85% F1 over the best previous results.

Conceptually, modeling multiple relations is substantially different from simply modeling coherence (as in Ganea and Hofmann (2017)). In this way we also hope it will lead to interesting follow-up work, as individual relations can be informed by injecting prior knowledge (e.g., by training jointly with relation extraction models).

In future work, we would like to use syntactic and discourse structures (e.g., syntactic dependency paths between mentions) to encourage the models to discover a richer set of relations. We also would like to combine ment-norm and rel-norm. Besides, we would like to examine whether


rel-norm | on Friday , **Liege** police said in | ment-norm

(1) missing teenagers in **Belgium** .
(2) UNK **BRUSSELS** UNK
(3) UNK **Belgian** police said on
(4) , " a **Liege** police official told
(5) police official told **Reuters** .
(6) eastern town of **Liege** on Thursday ,
(7) home village of **UNK** .
(8) link with the **Marc Dutroux** case , the
(9) which has rocked **Belgium** in the past

Figure 5: (Left) Examples of α. The first and third columns show α_{ijk} for oracle rel-norm and oracle ment-norm, respectively. (Right) Histograms of $\alpha_{\bullet k}$ for $k = 2, 3$, corresponding to the second and third relations from oracle ment-norm. Only $\alpha > 0.25$ (i.e. high attentions) are shown.

the induced latent relations could be helpful for relation extract.

Acknowledgments

We would like to thank anonymous reviewers for their suggestions and comments. The project was supported by the European Research Council (ERC StG BroadSem 678254), the Dutch National Science Foundation (NWO VIDI 639.022.518), and an Amazon Web Services (AWS) grant.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Seattle, Washington, USA. Association for Computational Linguistics.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association of Computational Linguistics*, 3:145–156.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*, pages 927–938. International World Wide Web Conferences Steering Committee.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2609–2619. Association for Computational Linguistics.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631. Association for Computational Linguistics.

Zhaochen Guo and Denilson Barbosa. 2016. Robust named entity disambiguation with random walks. *Semantic Web*, (Preprint).

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–34, Sofia, Bulgaria. Association for Computational Linguistics.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011.

Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc.

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

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384. Association for Computational Linguistics.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024. International World Wide Web Conferences Steering Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416–1426. Association for Computational Linguistics.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259. Association for Computational Linguistics.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *arXiv preprint arXiv:1705.02494*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Dating Documents using Graph Convolution Networks

Shikhar Vashishth
IISc Bangalore
shikhar@iisc.ac.in

Shib Sankar Dasgupta
IISc Bangalore
shibd@iisc.ac.in

Swayambhu Nath Ray
IISc Bangalore
swayambhuray@iisc.ac.in

Partha Talukdar
IISc Bangalore
ppt@iisc.ac.in

Abstract

Document date is essential for many important tasks, such as document retrieval, summarization, event detection, etc. While existing approaches for these tasks assume accurate knowledge of the document date, this is not always available, especially for arbitrary documents from the Web. Document Dating is a challenging problem which requires inference over the temporal structure of the document. Prior document dating systems have largely relied on handcrafted features while ignoring such document-internal structures. In this paper, we propose NeuralDater, a Graph Convolutional Network (GCN) based document dating approach which jointly exploits syntactic and temporal graph structures of document in a principled way. To the best of our knowledge, this is the first application of deep learning for the problem of document dating. Through extensive experiments on real-world datasets, we find that NeuralDater significantly outperforms state-of-the-art baseline by 19% absolute (45% relative) accuracy points.

1 Introduction

Date of a document, also referred to as the Document Creation Time (DCT), is at the core of many important tasks, such as, information retrieval (Olson et al., 1999; Li and Croft, 2003; Dakka et al., 2008), temporal reasoning (Mani and Wilson, 2000; Llidó et al., 2001), text summarization (Wan, 2007), event detection (Allan et al., 1998), and analysis of historical text (de Jong et al., 2005a), among others. In all such tasks, the document date is assumed to be available and also

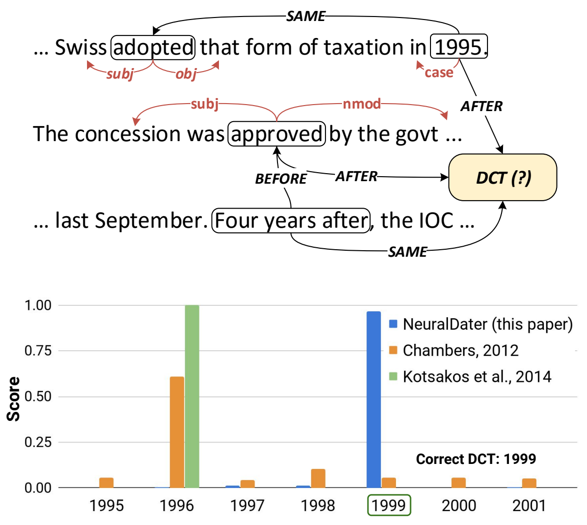


Figure 1: **Top:** An example document annotated with syntactic and temporal dependencies. In order to predict the right value of 1999 for the Document Creation Time (DCT), inference over these document structures is necessary. **Bottom:** Document date prediction by two state-of-the-art-baselines and NeuralDater, the method proposed in this paper. While the two previous methods are getting misled by the temporal expression (*1995*) in the document, NeuralDater is able to use the syntactic and temporal structure of the document to predict the right value (*1999*).

accurate – a strong assumption, especially for arbitrary documents from the Web. Thus, there is a need to automatically predict the date of a document based on its content. This problem is referred to as *Document Dating*.

Initial attempts on automatic document dating started with generative models by (de Jong et al., 2005b). This model is later improved by (Kanhabua and Nørvåg, 2008a) who incorporate additional features such as POS tags, collocations, etc. Chambers (2012) shows significant improvement over these prior efforts through their discriminative models using handcrafted temporal features. Kotsakos et al. (2014) propose a statistical approach for document dating exploiting term bursti-

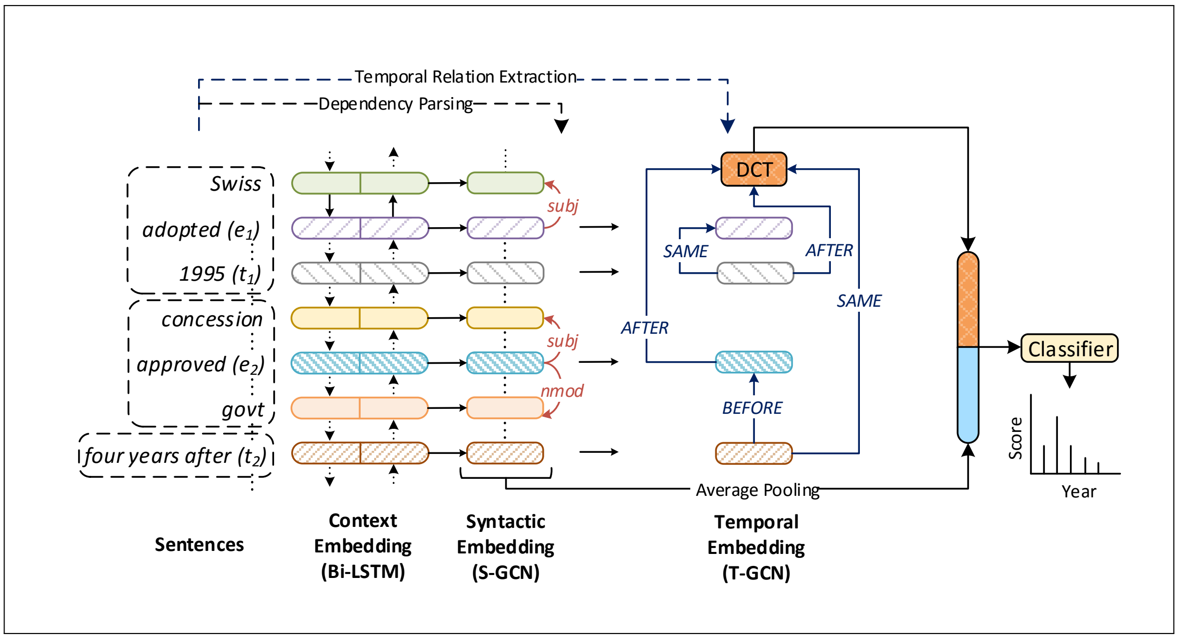


Figure 2: Overview of NeuralDater. NeuralDater exploits syntactic and temporal structure in a document to learn effective representation, which in turn are used to predict the document time. NeuralDater uses a Bi-directional LSTM (Bi-LSTM), two Graph Convolution Networks (GCN) – one over the dependency tree and the other over the document's temporal graph – along with a softmax classifier, all trained end-to-end jointly. Please see Section 4 for more details.

ness (Lappas et al., 2009).

Document dating is a challenging problem which requires extensive reasoning over the temporal structure of the document. Let us motivate this through an example shown in Figure 1. In the document, *four years after* plays a crucial role in identifying the creation time of the document. The existing approaches give higher confidence for timestamp immediate to the year mention *1995*. NeuralDater exploits the syntactic and temporal structure of the document to predict the right timestamp (1999) for the document. With the exception of (Chambers, 2012), all prior works on the document dating problem ignore such informative temporal structure within the document.

Research in document event extraction and ordering have made it possible to extract such temporal structures involving events, temporal expressions, and the (unknown) document date in a document (Mirza and Tonelli, 2016; Chambers et al., 2014). While methods to perform reasoning over such structures exist (Verhagen et al., 2007, 2010; UzZaman et al., 2013; Llorens et al., 2015; Pustejovsky et al., 2003), none of them have exploited advances in deep learning (Krizhevsky et al., 2012; Hinton et al., 2012; Goodfellow et al., 2016). In particular, recently proposed Graph Convolution Networks (GCN) (Defferrard et al., 2016; Kipf and Welling, 2017) have emerged as a way to learn graph representation while encoding structural information and constraints represented by the graph. We adapt GCNs for the document dating problem and make the following contributions:

- We propose NeuralDater, a Graph Convolution Network (GCN)-based approach for document dating. To the best of our knowledge, this is the first application of GCNs, and more broadly deep neural network-based methods, for the document dating problem.
- NeuralDater is the first document dating approach which exploits syntactic as well temporal structure of the document, all within a principled joint model.
- Through extensive experiments on multiple real-world datasets, we demonstrate NeuralDater's effectiveness over state-of-the-art baselines.

NeuralDater's source code and datasets used in the paper are available at `http://github.com/malllabiisc/NeuralDater`.

2 Related Work

Automatic Document Dating: de Jong et al. (2005b) propose the first approach for automating document dating through a statistical language model. Kanhabua and Nørvåg (2008a) further extend this work by incorporating semantic-based preprocessing and temporal entropy (Kanhabua and Nørvåg, 2008b) based term-weighting. Chambers (2012) proposes a MaxEnt based discriminative model trained on hand-crafted temporal features. He also proposes a model to learn probabilistic constraints between year mentions and the actual creation time of the document. We draw inspiration from his work for exploiting temporal reasoning for document dating. Kotsakos et al. (2014) propose a purely statistical method which considers lexical similarity alongside burstiness (Lappas et al., 2009) of terms for dating documents. To the best of our knowledge, NeuralDater, our proposed method, is the first method to utilize deep learning techniques for the document dating problem.

Event Ordering Systems: Temporal ordering of events is a vast research topic in NLP. The problem is posed as a temporal relation classification between two given temporal entities. Machine Learned classifiers and well crafted linguistic features for this task are used in (Chambers et al., 2007; Mirza and Tonelli, 2014). D'Souza and Ng (2013) use a hybrid approach by adding 437 hand-crafted rules. Chambers and Jurafsky (2008); Yoshikawa et al. (2009) try to classify with many more temporal constraints, while utilizing integer linear programming and Markov logic.

CAEVO, a CAscading EVent Ordering architecture (Chambers et al., 2014) use sieve-based architecture (Lee et al., 2013) for temporal event ordering for the first time. They mix multiple learners according to their precision based ranks and use transitive closure for maintaining consistency of temporal graph. Mirza and Tonelli (2016) recently propose CATENA (CAusal and TEmporal relation extraction from NAtural language texts), the first integrated system for the temporal and causal relations extraction between pre-annotated events and time expressions. They also incorporate sieve-based architecture which outperforms existing methods in temporal relation classification domain. We make use of CATENA for temporal graph construction in our work.

Graph Convolutional Networks (GCN): GCNs generalize Convolutional Neural Network (CNN) over graphs. GCN is introduced by (Bruna et al., 2014), and later extended by (Defferrard et al., 2016) with efficient localized filter approximation in spectral domain. Kipf and Welling (2017) propose a first-order approximation of localized filters through layer-wise propagation rule. GCNs over syntactic dependency trees have been recently exploited in the field of semantic-role labeling (Marcheggiani and Titov, 2017), neural machine translation (Bastings et al., 2017a), event detection (Bastings et al., 2017b). In our work, we successfully use GCNs for document dating.

3 Background: Graph Convolution Networks (GCN)

In this section, we provide an overview of Graph Convolution Networks (GCN) (Kipf and Welling, 2017). GCN learns an embedding for each node of the graph it is applied over. We first present GCN for undirected graphs and then move onto GCN for directed graph setting.

3.1 GCN on Undirected Graph

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ is a set of n vertices and $\mathcal{E}$ the set of edges. The input feature matrix $\mathcal{X} \in \mathbb{R}^{n \times m}$ whose rows are input representation of node u, $x_u \in \mathbb{R}^m, \forall u \in \mathcal{V}$. The output hidden representation $h_v \in \mathbb{R}^d$ of a node v after a single layer of graph convolution operation can be obtained by considering only the immediate neighbors of v. This can be formulated as:

$$h_v = f\left(\sum_{u \in \mathcal{N}(v)} (W x_u + b)\right), \quad \forall v \in \mathcal{V}.$$

Here, model parameters $W \in \mathbb{R}^{d \times m}$ and $b \in \mathbb{R}^d$ are learned in a task-specific setting using first-order gradient optimization. $\mathcal{N}(v)$ refers to the set of neighbors of v and f is any non-linear activation function. We have used ReLU as the activation function in this paper[1].

In order to capture nodes many hops away, multiple GCN layers may be stacked one on top of another. In particular, h_v^{k+1}, representation of node v after k^{th} GCN layer can be formulated as:

$$h_v^{k+1} = f\left(\sum_{u \in \mathcal{N}(v)} \left(W^k h_u^k + b^k\right)\right), \forall v \in \mathcal{V}.$$

where h_u^k is the input to the k^{th} layer.

[1]ReLU: $f(x) = \max(0, x)$

3.2 GCN on Labeled and Directed Graph

In this section, we consider GCN formulation over graphs where each edge is labeled as well as directed. In this setting, an edge from node u to v with label $l(u,v)$ is denoted as $(u,v,l(u,v))$. While a few recent works focus on GCN over directed graphs (Yasunaga et al., 2017; Marcheggiani and Titov, 2017), none of them consider labeled edges. We handle both direction and label by incorporating label and direction specific filters.

Based on the assumption that the information in a directed edge need not only propagate along its direction, following (Marcheggiani and Titov, 2017) we define an updated edge set $\mathcal{E}'$ which expands the original set $\mathcal{E}$ by incorporating inverse, as well self-loop edges.

$$\mathcal{E}' = \mathcal{E} \cup \{(v,u,l(u,v)^{-1}) \mid (u,v,l(u,v)) \in \mathcal{E}\} \cup \{(u,u,\top) \mid u \in \mathcal{V})\}. \quad (1)$$

Here, $l(u,v)^{-1}$ is the inverse edge label corresponding to label $l(u,v)$, and $\top$ is a special empty relation symbol for self-loop edges. We now define h_v^{k+1} as the embedding of node v after k^{th} GCN layer applied over the directed and labeled graph as:

$$h_v^{k+1} = f\left(\sum_{u\in\mathcal{N}(v)} \left(W_{l(u,v)}^k h_u^k + b_{l(u,v)}^k\right)\right). \quad (2)$$

We note that the parameters $W_{l(u,v)}^k$ and $b_{l(u,v)}^k$ in this case are edge label specific.

3.3 Incorporating Edge Importance

In many practical settings, we may not want to give equal importance to all the edges. For example, in case of automatically constructed graphs, some of the edges may be erroneous and we may want to automatically learn to discard them. Edge-wise gating may be used in a GCN to give importance to relevant edges and subdue the noisy ones. Bastings et al. (2017b); Marcheggiani and Titov (2017) used gating for similar reasons and obtained high performance gain. At k^{th} layer, we compute gating value for a particular edge $(u,v,l(u,v))$ as:

$$g_{u,v}^k = \sigma\left(h_u^k \cdot \hat{w}_{l(u,v)}^k + \hat{b}_{l(u,v)}^k\right),$$

where, $\sigma(\cdot)$ is the sigmoid function, $\hat{w}_{l(u,v)}^k$ and $\hat{b}_{l(u,v)}^k$ are label specific gating parameters. Thus, gating helps to make the model robust to the noisy labels and directions of the input graphs. GCN embedding of a node while incorporating edge gating may be computed as follows.

$$h_v^{k+1} = f\left(\sum_{u\in\mathcal{N}(v)} g_{u,v}^k \times \left(W_{l(u,v)}^k h_u^k + b_{l(u,v)}^k\right)\right).$$

4 NeuralDater Overview

The Documents Dating problem may be cast as a multi-class classification problem (Kotsakos et al., 2014; Chambers, 2012). In this section, we present an overview of NeuralDater, the document dating system proposed in this paper. Architectural overview of NeuralDater is shown in Figure 2.

NeuralDater is a deep learning-based multi-class classification system. It takes in a document as input and returns its predicted date as output by exploiting the syntactic and temporal structure of document.

NeuralDater network consists of three layers which learns an embedding for the Document Creation Time (DCT) node corresponding to the document. This embedding is then fed to a softmax classifier which produces a distribution over timestamps. Following prior research (Chambers, 2012; Kotsakos et al., 2014), we work with year granularity for the experiments in this paper. We however note that NeuralDater can be trained for finer granularity with appropriate training data. The NeuralDater network is trained end-to-end using training data. We briefly present NeuralDater's various components below. Each component is described in greater detail in subsequent sections.

- **Context Embedding**: In this layer, NeuralDater uses a Bi-directional LSTM (Bi-LSTM) to learn embedding for each token in the document. Bi-LSTMs have been shown to be quite effective in capturing local context inside token embeddings (Sutskever et al., 2014).

- **Syntactic Embedding**: In this step, NeuralDater revises token embeddings from previous step by running a GCN over the dependency parses of sentences in the document. We refer to this GCN as **Syntactic GCN** or **S-GCN**. While the Bi-LSTM captures immediate local context in token embeddings, S-

GCN augments them by capturing syntactic context.

- **Temporal Embedding**: In this step, NeuralDater further refines embeddings learned by S-GCN to incorporate cues from temporal structure of event and times in the document. NeuralDater uses state-of-the-art causal and temporal relation extraction algorithm (Mirza and Tonelli, 2016) for extracting temporal graph for each document. A GCN is then run over this temporal graph to refine the embeddings from previous layer. We refer to this GCN as **Temporal GCN** or **T-GCN**. In this step, a special DCT node is introduced whose embedding is also learned by the T-GCN.

- **Classifier**: Embedding of the DCT node along with average pooled embeddings learned by S-GCN are fed to a fully connected softmax classifier which makes the final prediction about the date of the document.

Even though the previous discussion is presented in a sequential manner, the whole network is trained in a joint end-to-end manner using backpropagation.

5 NeuralDater Details

In this section, we present detailed description of various components of NeuralDater.

5.1 Context Embedding (Bi-LSTM)

Let us consider a document D with n tokens $w_1, w_2, ..., w_n$. We first represent each token by a k-dimensional word embedding. For the experiments in this paper, we use GloVe (Pennington et al., 2014) embeddings. These token embeddings are stacked together to get the document representation $\mathcal{X} \in \mathbb{R}^{n \times k}$. We then employ a Bi-directional LSTM (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) on the input matrix $\mathcal{X}$ to obtain contextual embedding for each token. After stacking contextual embedding of all these tokens, we get the new document representation matrix $\mathcal{H}^{cntx} \in \mathbb{R}^{n \times r_{cntx}}$. In this new representation, each token is represented in a r_{cntx}-dimensional space. Our choice of LSTMs for learning contextual embeddings for tokens is motivated by the previous success of LSTMs in this task (Sutskever et al., 2014).

5.2 Syntactic Embedding (S-GCN)

While the Bi-LSTM is effective at capturing immediate local context of a token, it may not be as effective in capturing longer range dependencies among words in a sentence. For example, in Figure 1, we would like the embedding of token *approved* to be directly affected by *govt*, even though they are not immediate neighbors. A dependency parse may be used to capture such longer-range connections. In fact, similar features were exploited by (Chambers, 2012) for the document dating problem. NeuralDater captures such longer-range information by using another GCN run over the syntactic structure of the document. We describe this in detail below.

The context embedding, $\mathcal{H}^{cntx} \in \mathbb{R}^{n \times r_{cntx}}$ learned in the previous step is used as input to this layer. For a given document, we first extract its syntactic dependency structure by applying the Stanford CoreNLP's dependency parser (Manning et al., 2014) on each sentence in the document individually. We now employ the Graph Convolution Network (GCN) over this dependency graph using the GCN formulation presented in Section 3.2. We call this GCN the Syntactic GCN or S-GCN, as mentioned in Section 4.

Since S-GCN operates over the dependency graph and uses Equation 2 for updating embeddings, the number of parameters in S-GCN is directly proportional to the number of dependency edge types. Stanford CoreNLP's dependency parser returns 55 different dependency edge types. This large number of edge types is going to significantly over-parameterize S-GCN, thereby increasing the possibility of overfitting. In order to address this, we use only three edge types in S-GCN. For each edge connecting nodes w_i and w_j in $\mathcal{E}'$ (see Equation 1), we determine its new type $L(w_i, w_j)$ as follows:

- $L(w_i, w_j) = \rightarrow$ if $(w_i, w_j, l(w_i, w_j)) \in \mathcal{E}'$, i.e., if the edge is an original dependency parse edge

- $L(w_i, w_j) = \leftarrow$ if $(w_i, w_j, l(w_i, w_j)^{-1}) \in \mathcal{E}'$, i.e., if the edges is an inverse edge

- $L(w_i, w_j) = \top$ if $(w_i, w_j, \top) \in \mathcal{E}'$, i.e., if the edge is a self-loop with $w_i = w_j$

S-GCN now estimates embedding $h^{syn}_{w_i} \in \mathbb{R}^{r_{syn}}$ for each token w_i in the document using the formulation shown below.

$$h_{w_i}^{syn} = f\left(\sum_{w_j \in \mathcal{N}(w_i)} \left(W_{L(w_i,w_j)} h_{w_j}^{cntx} + b_{L(w_i,w_j)} \right) \right)$$

Please note S-GCN's use of the new edge types $L(w_i, w_j)$ above, instead of the $l(w_i, w_j)$ types used in Equation 2. By stacking embeddings for all the tokens together, we get the new embedding matrix $\mathcal{H}^{syn} \in \mathbb{R}^{n \times r_{syn}}$ representing the document.

AveragePooling: We obtain an embedding h_D^{avg} for the whole document by average pooling of every token representation.

$$h_D^{avg} = \frac{1}{n} \sum_{i=1}^{n} h_{w_i}^{syn}. \tag{3}$$

5.3 Temporal Embedding (T-GCN)

In this layer, NeuralDater exploits temporal structure of the document to learn an embedding for the Document Creation Time (DCT) node of the document. First, we describe the construction of temporal graph, followed by GCN-based embedding learning over this graph.

Temporal Graph Construction: NeuralDater uses Stanford's SUTime tagger (Chang and Manning, 2012) for date normalization and the event extraction classifier of (Chambers et al., 2014) for event detection. The annotated document is then passed to CATENA (Mirza and Tonelli, 2016), current state-of-the-art temporal and causal relation extraction algorithm, to obtain a temporal graph for each document. Since our task is to predict the creation time of a given document, we supply DCT as unknown to CATENA. We hypothesize that the temporal relations extracted in absence of DCT are helpful for document dating and we indeed find this to be true, as shown in Section 7. Temporal graph is a directed graph, where nodes correspond to events, time mentions, and the Document Creation Time (DCT). Edges in this graph represent causal and temporal relationships between them. Each edge is attributed with a label representing the type of the temporal relation. CATENA outputs 9 different types of temporal relations, out of which we selected five types, viz., *AFTER*, *BEFORE*, *SAME*, *INCLUDES*, and *IS_INCLUDED*. The remaining four types were ignored as they were substantially infrequent.

Please note that the temporal graph may involve only a small number of tokens in the document.

Datasets	# Docs	Start Year	End Year
APW	675k	1995	2010
NYT	647k	1987	1996

Table 1: Details of datasets used. Please see Section 6 for details.

For example, in the temporal graph in Figure 2, there are a total of 5 nodes: two temporal expression nodes (*1995* and *four years after*), two event nodes (*adopted* and *approved*), and a special DCT node. This graph also consists of temporal relation edges such as (*four years after*, *approved*, *BEFORE*).

Temporal Graph Convolution: NeuralDater employs a GCN over the temporal graph constructed above. We refer to this GCN as the Temporal GCN or T-GCN, as mentioned in Section 4. T-GCN is based on the GCN formulation presented in Section 3.2. Unlike S-GCN, here we consider label and direction specific parameters as the temporal graph consists of only five types of edges.

Let n_T be the number of nodes in the temporal graph. Starting with $\mathcal{H}^{syn}$ (Section 5.2), T-GCN learns a r_{temp}-dimensional embedding for each node in the temporal graph. Stacking all these embeddings together, we get the embedding matrix $\mathcal{H}^{temp} \in \mathbb{R}^{n_T \times r_{temp}}$. T-GCN embeds the temporal constraints induced by the temporal graph in $h_{DCT}^{temp} \in \mathbb{R}^{r_{temp}}$, embedding of the DCT node of the document.

5.4 Classifier

Finally, the DCT embedding h_{DCT}^{temp} and average-pooled syntactic representation h_D^{avg} (see Equation 3) of document D are concatenated and fed to a fully connected feed forward network followed by a softmax. This allows the NeuralDater to exploit context, syntactic, and temporal structure of the document to predict the final document date y.

$$\begin{aligned} h_D^{avg+temp} &= [h_{DCT}^{temp} \,;\, h_D^{avg}] \\ p(y|D) &= \text{Softmax}(W \cdot h_D^{avg+temp} + b). \end{aligned}$$

6 Experimental Setup

Datasets: We experiment on Associated Press Worldstream (APW) and New York Times (NYT) sections of Gigaword corpus (Parker et al., 2011). The original dataset contains around 3 million documents of APW and 2 million documents of NYT from span of multiple years. From both sections, we randomly sample around 650k documents while maintaining balance among years. Documents belonging to years with substantially fewer documents are omitted. Details of the dataset can be found in Table 1. For train, test and validation splits, the dataset was randomly divided in 80:10:10 ratio.

Evaluation Criteria: Given a document, the model needs to predict the year in which the document was published. We measure performance in terms of overall accuracy of the model.

Baselines: For evaluating NeuralDater, we compared against the following methods:

- **BurstySimDater** Kotsakos et al. (2014): This is a purely statistical method which uses lexical similarity and term burstiness (Lappas et al., 2009) for dating documents in arbitrary length time frame. For our experiments, we took the time frame length as 1 year. Please refer to (Kotsakos et al., 2014) for more details.

- **MaxEnt-Time-NER**: Maximum Entropy (MaxEnt) based classifier trained on hand-crafted temporal and Named Entity Recognizer (NER) based features. More details in (Chambers, 2012).

- **MaxEnt-Joint**: Refers to MaxEnt-Time-NER combined with year mention classifier as described in (Chambers, 2012).

- **MaxEnt-Uni-Time:** MaxEnt based discriminative model which takes bag-of-words representation of input document with normalized time expression as its features.

- **CNN:** A Convolution Neural Network (CNN) (LeCun et al., 1999) based text classification model proposed by (Kim, 2014), which attained state-of-the-art results in several domains.

- **NeuralDater**: Our proposed method, refer Section 4.

Hyperparameters: By default, edge gating (Section 3.3) is used in all GCNs. The parameter K represents the number of layers in T-GCN (Section 5.3). We use 300-dimensional GloVe embeddings and 128-dimensional hidden state for both GCNs and BiLSTM with 0.8 dropout. We used Adam (Kingma and Ba, 2014) with 0.001 learning rate for training.

Method	APW	NYT
BurstySimDater	45.9	38.5
MaxEnt-Time+NER	52.5	42.3
MaxEnt-Joint	52.5	42.5
MaxEnt-Uni-Time	57.5	50.5
CNN	56.3	50.4
NeuralDater	**64.1**	**58.9**

Table 2: Accuracies of different methods on APW and NYT datasets for the document dating problem (higher is better). NeuralDater significantly outperforms all other competitive baselines. This is our main result. Please see Section 7.1 for more details.

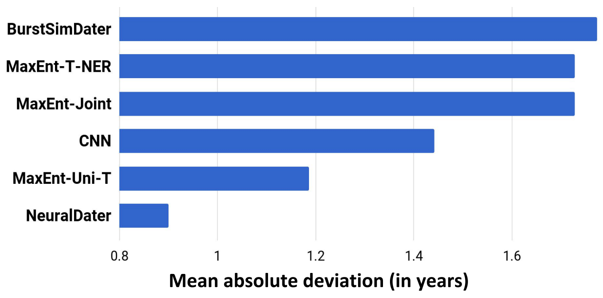


Figure 3: Mean absolute deviation (in years; lower is better) between a model's top prediction and the true year in the APW dataset. We find that NeuralDater, the proposed method, achieves the least deviation. Please see Section 7.1 for details.

Method	Accuracy
T-GCN	57.3
S-GCN + T-GCN ($K = 1$)	57.8
S-GCN + T-GCN ($K = 2$)	58.8
S-GCN + T-GCN ($K = 3$)	**59.1**
Bi-LSTM	58.6
Bi-LSTM + CNN	59.0
Bi-LSTM + T-GCN	60.5
Bi-LSTM + S-GCN + T-GCN (no gate)	62.7
Bi-LSTM + S-GCN + T-GCN ($K = 1$)	**64.1**
Bi-LSTM + S-GCN + T-GCN ($K = 2$)	63.8
Bi-LSTM + S-GCN + T-GCN ($K = 3$)	63.3

Table 3: Accuracies of different ablated methods on the APW dataset. Overall, we observe that incorporation of context (Bi-LSTM), syntactic structure (S-GCN) and temporal structure (T-GCN) in NeuralDater achieves the best performance. Please see Section 7.1 for details.

7 Results

7.1 Performance Comparison

In order to evaluate the effectiveness of NeuralDater, our proposed method, we compare it against existing document dating systems and text classification models. The final results are summarized in Table 2. Overall, we find that NeuralDater outperforms all other methods with a significant margin on both datasets. Compared to the previous state-of-the-art in document dating, BurstySimDater (Kotsakos et al., 2014), we get 19% average absolute improvement in accuracy across both datasets. We observe only a slight gain in the performance of MaxEnt-based model (MaxEnt-Time+NER) of (Chambers, 2012) on combining with temporal constraint reasoner (MaxEnt-Joint). This may be attributed to the fact that the model utilizes only year mentions in the document, thus ignoring other relevant signals which might be relevant to the task. BurstySimDater performs considerably better in terms of precision compared to the other baselines, although it significantly underperforms in accuracy. We note that NeuralDater outperforms all these prior models both in terms of precision and accuracy. We find that even generic deep-learning based text classification models, such as CNN (Kim, 2014), are quite effective for the problem. However, since such a model doesn't give specific attention to temporal features in the document, its performance remains limited. From Figure 3, we observe that NeuralDater's top prediction achieves on average the lowest deviation from the true year.

7.2 Ablation Comparisons

For demonstrating the efficacy of GCNs and BiLSTM for the problem, we evaluate different ablated variants of NeuralDater on the APW dataset. Specifically, we validate the importance of using syntactic and temporal GCNs and the effect of eliminating BiLSTM from the model. Overall results are summarized in Table 3. The first block of rows in the table corresponds to the case when BiLSTM layer is excluded from NeuralDater, while the second block denotes the case when BiLSTM is included. We also experiment with multiple stacked layers of T-GCN (denoted by K) to observe its effect on the performance of the model.

We observe that embeddings from Syntactic GCN (S-GCN) are much better than plain GloVe embeddings for T-GCN as S-GCN encodes the syntactic neighborhood information in event and time embeddings which makes them more relevant for document dating task.

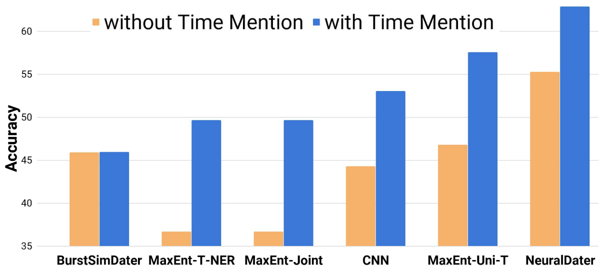


Figure 4: Evaluating performance of different methods on dating documents with and without time mentions. Please see Section 7.3 for details.

Overall, we observe that including BiLSTM in the model improves performance significantly. Single BiLSTM model outperforms all the models listed in the first block of Table 3. Also, some gain in performance is observed on increasing the number of T-GCN layers (K) in absence of BiLSTM, although the same does not follow when BiLSTM is included in the model. This observation is consistent with (Marcheggiani and Titov, 2017), as multiple GCN layers become redundant in the presence of BiLSTM. We also find that eliminating edge gating from our best model deteriorates its overall performance.

In summary, these results validate our thesis that joint incorporation of syntactic and temporal structure of a document in NeuralDater results in improved performance.

7.3 Discussion and Error Analysis

In this section, we list some of our observations while trying to identify pros and cons of NeuralDater, our proposed method. We divided the development split of the APW dataset into two sets – those with and without any mention of time expressions (year). We apply NeuralDater and other methods to these two sets of documents and report accuracies in Figure 4. We find that overall, NeuralDater performs better in comparison to the existing baselines in both scenarios. Even though the performance of NeuralDater degrades in the absence of time mentions, its performance is still the best relatively. Based on other analysis, we find that NeuralDater fails to identify timestamp of documents reporting local infrequent incidents without explicit time mention. NeuralDater becomes confused in the presence of multiple misleading time mentions; it also loses out on documents discussing events which are outside the time range of the text on which the model was trained. In future, we plan to eliminate these pitfalls by incorporating additional signals from Knowledge Graphs about entities mentioned in the document. We also plan to utilize free text temporal expression (Kuzey et al., 2016) in documents for improving performance on this problem.

8 Conclusion

We propose NeuralDater, a Graph Convolutional Network (GCN) based method for document dating which exploits syntactic and temporal structures in the document in a principled way. To the best of our knowledge, this is the first application of deep learning techniques for the problem of document dating. Through extensive experiments on real-world datasets, we demonstrate the effectiveness of NeuralDater over existing state-of-the-art approaches. We are hopeful that the representation learning techniques explored in this paper will inspire further development and adoption of such techniques in the temporal information processing research community.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This work is supported in part by the Ministry of Human Resource Development (Government of India) and by a gift from Google.

References

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '98, pages 37–45. https://doi.org/10.1145/290941.290954.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017a. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1957–1967. https://www.aclweb.org/anthology/D17-1209.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017b. Graph convolutional encoders for syntax-aware neural machine translation. *CoRR* abs/1704.04675. http://arxiv.org/abs/1704.04675.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Nathanael Chambers. 2012. Labeling documents with timestamps: Learning from their time expressions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '12, pages 98–106. http://dl.acm.org/citation.cfm?id=2390524.2390539.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association of Computational Linguistics* 2:273–284. http://www.aclweb.org/anthology/Q14-1022.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '08, pages 698–706. http://dl.acm.org/citation.cfm?id=1613715.1613803.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '07, pages 173–176. http://dl.acm.org/citation.cfm?id=1557769.1557820.

Angel X. Chang and Christopher Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA). http://www.aclweb.org/anthology/L12-1122.

Wisam Dakka, Luis Gravano, and Panagiotis G. Ipeirotis. 2008. Answering general time sensitive queries. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '08, pages 1437–1438. https://doi.org/10.1145/1458082.1458320.

Franciska M.G. de Jong, H. Rode, and Djoerd Hiemstra. 2005a. *Temporal Language Models for the Disclosure of Historical Text*, KNAW, pages 161–168. Imported from EWI/DB PMS [db-utwente:inpr:0000003683].

Franciska M.G. de Jong, H. Rode, and Djoerd Hiemstra. 2005b. *Temporal Language Models for the Disclosure of Historical Text*, KNAW, pages 161–168. Imported from EWI/DB PMS [db-utwente:inpr:0000003683].

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In

Proceedings of the 30th International Conference on Neural Information Processing Systems. Curran Associates Inc., USA, NIPS'16, pages 3844–3852. http://dl.acm.org/citation.cfm?id=3157382.3157527.

Jennifer D'Souza and Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 918–927. http://www.aclweb.org/anthology/N13-1112.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29(6):82–97. https://doi.org/10.1109/MSP.2012.2205597.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Nattiya Kanhabua and Kjetil Nørvåg. 2008a. Improving temporal language models for determining time of non-timestamped documents. In *Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries*. Springer-Verlag, Berlin, Heidelberg, ECDL '08, pages 358–370.

Nattiya Kanhabua and Kjetil Nørvåg. 2008b. Improving temporal language models for determining time of non-timestamped documents. In *International Conference on Theory and Practice of Digital Libraries*. Springer, pages 358–370.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1746–1751. https://doi.org/10.3115/v1/D14-1181.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Dimitrios Kotsakos, Theodoros Lappas, Dimitrios Kotzias, Dimitrios Gunopulos, Nattiya Kanhabua, and Kjetil Nørvåg. 2014. A burstiness-aware approach for document dating. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '14, pages 1003–1006. https://doi.org/10.1145/2600428.2609495.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Curran Associates Inc., USA, NIPS'12, pages 1097–1105. http://dl.acm.org/citation.cfm?id=2999134.2999257.

Erdal Kuzey, Vinay Setty, Jannik Strötgen, and Gerhard Weikum. 2016. As time goes by: Comprehensive tagging of textual phrases with temporal scopes. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, WWW '16, pages 915–925. https://doi.org/10.1145/2872427.2883055.

Theodoros Lappas, Benjamin Arai, Manolis Platakis, Dimitrios Kotsakos, and Dimitrios Gunopulos. 2009. On burstiness-aware search for document sequences. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '09, pages 477–486. https://doi.org/10.1145/1557019.1557075.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*. Springer-Verlag, London, UK, UK, pages 319–. http://dl.acm.org/citation.cfm?id=646469.691875.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Comput. Linguist.* 39(4):885–916.

Xiaoyan Li and W. Bruce Croft. 2003. Time-based language models. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '03, pages 469–475. https://doi.org/10.1145/956863.956951.

D. Llidó, R. Berlanga, and M. J. Aramburu. 2001. Extracting temporal references to assign document event-time periods*. In Heinrich C. Mayr, Jiri Lazansky, Gerald Quirchmayr, and Pavel Vogel, editors, *Database and Expert Systems Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 62–71.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. Semeval-2015 task 5:

Qa tempeval-evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 792–800.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '00, pages 69–76. https://doi.org/10.3115/1075218.1075228.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *CoRR* abs/1703.04826. http://arxiv.org/abs/1703.04826.

Paramita Mirza and Sara Tonelli. 2014. Classifying temporal relations with simple features. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 308–317. https://doi.org/10.3115/v1/E14-1033.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 64–75. http://www.aclweb.org/anthology/C16-1007.

MA Olson, K Bostic, MI Seltzer, and DB Berkeley. 1999. Usenix annual technical conference, freenix track.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition ldc2011t07. dvd. *Philadelphia: Linguistic Data Consortium* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*. Lancaster, UK., volume 2003, page 40.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'14, pages 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. volume 2, pages 1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th international workshop on semantic evaluations*. Association for Computational Linguistics, pages 75–80.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 57–62.

Xiaojun Wan. 2007. Timedtextrank: Adding the temporal dimension to multi-document summarization. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '07, pages 867–868. https://doi.org/10.1145/1277741.1277949.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir R. Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of CoNLL-2017*. Association for Computational Linguistics.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 405–413. http://www.aclweb.org/anthology/P09-1046.

A Graph-to-Sequence Model for AMR-to-Text Generation

Linfeng Song[1], **Yue Zhang**[3], **Zhiguo Wang**[2] and **Daniel Gildea**[1]
[1]Department of Computer Science, University of Rochester, Rochester, NY 14627
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
[3]Singapore University of Technology and Design

Abstract

The problem of AMR-to-text generation is to recover a text representing the same meaning as an input AMR graph. The current state-of-the-art method uses a sequence-to-sequence model, leveraging LSTM for encoding a linearized AMR structure. Although it is able to model non-local semantic information, a sequence LSTM can lose information from the AMR graph structure, and thus faces challenges with large graphs, which result in long sequences. We introduce a neural graph-to-sequence model, using a novel LSTM structure for directly encoding graph-level semantics. On a standard benchmark, our model shows superior results to existing methods in the literature.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism that encodes the meaning of a sentence as a rooted, directed graph. Figure 1 shows an AMR graph in which the nodes (such as "describe-01" and "person") represent the concepts, and edges (such as ":ARG0" and ":name") represent the relations between concepts they connect. AMR has been proven helpful on other NLP tasks, such as machine translation (Jones et al., 2012; Tamchyna et al., 2015), question answering (Mitra and Baral, 2015), summarization (Takase et al., 2016) and event detection (Li et al., 2015).

The task of AMR-to-text generation is to produce a text with the same meaning as a given input AMR graph. The task is challenging as word tenses and function words are abstracted away when constructing AMR graphs from texts. The translation from AMR nodes to text phrases can be far from literal. For example, shown in Figure 1, "Ryan" is represented as "(p / person :name (n / name :op1 "Ryan"))", and "description of" is represented as "(d / describe-01 :ARG1)".

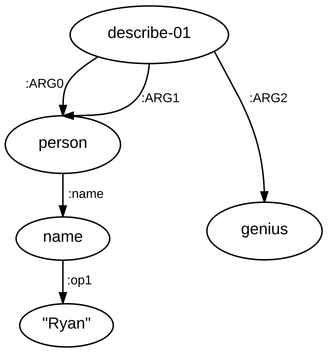


Figure 1: An example of AMR graph meaning "Ryan's description of himself: a genius."

While initial work used statistical approaches (Flanigan et al., 2016b; Pourdamghani et al., 2016; Song et al., 2017; Lampouras and Vlachos, 2017; Mille et al., 2017; Gruzitis et al., 2017), recent research has demonstrated the success of deep learning, and in particular the sequence-to-sequence model (Sutskever et al., 2014), which has achieved the state-of-the-art results on AMR-to-text generation (Konstas et al., 2017). One limitation of sequence-to-sequence models, however, is that they require serialization of input AMR graphs, which adds to the challenge of representing graph structure information, especially when the graph is large. In particular, closely-related nodes, such as parents, children and siblings can be far away after serialization. It can be difficult for a linear recurrent neural network to automatically induce their original connections from bracketed string forms.

To address this issue, we introduce a novel graph-to-sequence model, where a graph-state LSTM is used to encode AMR structures directly.

To capture non-local information, the encoder performs graph state transition by information exchange between connected nodes, with a graph state consisting of all node states. Multiple recurrent transition steps are taken so that information can propagate non-locally, and LSTM (Hochreiter and Schmidhuber, 1997) is used to avoid gradient diminishing and bursting in the recurrent process. The decoder is an attention-based LSTM model with a copy mechanism (Gu et al., 2016; Gulcehre et al., 2016), which helps copy sparse tokens (such as numbers and named entities) from the input.

Trained on a standard dataset (LDC2015E86), our model surpasses a strong sequence-to-sequence baseline by 2.3 BLEU points, demonstrating the advantage of graph-to-sequence models for AMR-to-text generation compared to sequence-to-sequence models. Our final model achieves a BLEU score of 23.3 on the test set, which is 1.3 points higher than the existing state of the art (Konstas et al., 2017) trained on the same dataset. When using gigaword sentences as additional training data, our model is consistently better than Konstas et al. (2017) using the same amount of gigaword data, showing the effectiveness of our model on large-scale training set.

We release our code and models at `https://github.com/freesunshine0316/neural-graph-to-seq-mp`.

2 Baseline: a seq-to-seq model

Our baseline is a sequence-to-sequence model, which follows the encoder-decoder framework of Konstas et al. (2017).

2.1 Input representation

Given an AMR graph $G = (V, E)$, where V and E denote the sets of nodes and edges, respectively, we use the depth-first traversal of Konstas et al. (2017) to linearize it to obtain a sequence of tokens $v_1, \dots, v_N$, where N is the number of tokens. For example, the AMR graph in Figure 1 is serialized as "describe :arg0 (person :name (name :op1 ryan)) :arg1 person :arg2 genius". We can see that the distance between "describe" and "genius", which are directly connected in the original AMR, becomes 14 in the serialization result.

A simple way to calculate the representation for each token v_j is using its word embedding e_j:

$$x_j = W_1 e_j + b_1, \tag{1}$$

where W_1 and b_1 are model parameters for compressing the input vector size.

To alleviate the data sparsity problem and obtain better word representation as the input, we also adopt a forward LSTM over the characters of the token, and concatenate the last hidden state h_j^c with the word embedding:

$$x_j = W_1\Big([e_j; h_j^c]\Big) + b_1 \tag{2}$$

2.2 Encoder

The encoder is a bi-directional LSTM applied on the linearized graph by depth-first traversal, as in Konstas et al. (2017). At each step j, the current states $\overleftarrow{h_j}$ and $\overrightarrow{h_j}$ are generated given the previous states $\overleftarrow{h_{j+1}}$ and $\overrightarrow{h_{j-1}}$ and the current input x_j:

$$\overleftarrow{h_j} = \text{LSTM}(\overleftarrow{h_{j+1}}, x_j)$$
$$\overrightarrow{h_j} = \text{LSTM}(\overrightarrow{h_{j-1}}, x_j)$$

2.3 Decoder

We use an attention-based LSTM decoder (Bahdanau et al., 2015), where the attention memory (A) is the concatenation of the attention vectors among all input words. Each attention vector a_j is the concatenation of the encoder states of an input token in both directions ($\overleftarrow{h_j}$ and $\overrightarrow{h_j}$) and its input vector (x_j):

$$a_j = [\overleftarrow{h_j}; \overrightarrow{h_j}; x_j] \tag{3}$$
$$A = [a_1; a_2; \dots; a_N] \tag{4}$$

where N is the number of input tokens.

The decoder yields an output sequence $w_1, w_2, \dots, w_M$ by calculating a sequence of hidden states $s_1, s_2 \dots, s_M$ recurrently. While generating the t-th word, the decoder considers five factors: (1) the attention memory A; (2) the previous hidden state of the LSTM model s_{t-1}; (3) the embedding of the current input (previously generated word) e_t; (4) the previous context vector μ_{t-1}, which is calculated with attention from A; and (5) the previous coverage vector γ_{t-1}, which is the accumulation of all attention distributions so far (Tu et al., 2016). When $t = 1$, we initialize μ_0 and γ_0 as zero vectors, set e_1 to the embedding of the start token "<s>", and s_0 as the average of all encoder states.

For each time-step t, the decoder feeds the concatenation of the embedding of the current input e_t and the previous context vector μ_{t-1} into the

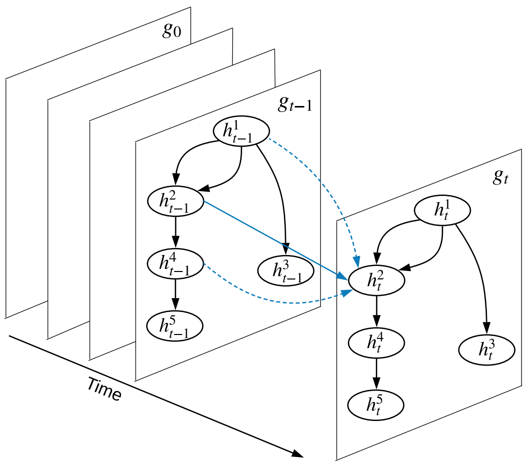


Figure 2: Graph state LSTM.

LSTM model to update its hidden state. Then the attention probability $\alpha_{t,i}$ on the attention vector $a_i \in A$ for the time-step is calculated as:

$$\epsilon_{t,i} = v_2^T \tanh(W_a a_i + W_s s_t + W_\gamma \gamma_{t-1} + b_2)$$
$$\alpha_{t,i} = \frac{\exp(\epsilon_{t,i})}{\sum_{j=1}^N \exp(\epsilon_{t,j})}$$

where W_a, W_s, W_γ, v_2 and b_2 are model parameters. The coverage vector γ_t is updated by $\gamma_t = \gamma_{t-1} + \alpha_t$, and the new context vector μ_t is calculated via $\mu_t = \sum_{i=1}^N \alpha_{t,i} a_i$.

The output probability distribution over a vocabulary at the current state is calculated by:

$$P_{vocab} = \text{softmax}(V_3[s_t, \mu_t] + b_3), \qquad (5)$$

where V_3 and b_3 are learnable parameters, and the number of rows in V_3 represents the number of words in the vocabulary.

3 The graph-to-sequence model

Unlike the baseline sequence-to-sequence model, we leverage a recurrent graph encoder to represent each input AMR, which directly models the graph structure without serialization.

3.1 The graph encoder

Figure 2 shows the overall structure of our graph encoder. Formally, given a graph $G = (V, E)$, we use a hidden state vector h^j to represent each node $v_j \in V$. The state of the graph can thus be represented as:

$$g = \{h^j\}|_{v_j \in V}$$

In order to capture non-local interaction between nodes, we allow information exchange between nodes through a sequence of state transitions, leading to a sequence of states $g_0, g_1, \dots, g_t, \dots$, where $g_t = \{h_t^j\}|_{v_j \in V}$. The initial state g_0 consists of a set of initial node states $h_0^j = h_0$, where h_0 is a hyperparameter of the model.

State transition A recurrent neural network is used to model the state transition process. In particular, the transition from g_{t-1} to g_t consists of a hidden state transition for each node, as shown in Figure 2. At each state transition step t, we allow direct communication between a node and all nodes that are directly connected to the node. To avoid gradient diminishing or bursting, LSTM (Hochreiter and Schmidhuber, 1997) is adopted, where a cell c_t^j is taken to record memory for h_t^j. We use an input gate i_t^j, an output gate o_t^j and a forget gate f_t^j to control information flow from the inputs and to the output h_t^j.

The inputs include representations of edges that are connected to v_j, where v_j can be either the source or the target of the edge. We define each edge as a triple (i, j, l), where i and j are indices of the source and target nodes, respectively, and l is the edge label. $x_{i,j}^l$ is the representation of edge (i, j, l), detailed in Section 3.3. The inputs for v_j are distinguished by incoming and outgoing edges, before being summed up:

$$x_j^i = \sum_{(i,j,l) \in E_{in}(j)} x_{i,j}^l$$
$$x_j^o = \sum_{(j,k,l) \in E_{out}(j)} x_{j,k}^l,$$

where $E_{in}(j)$ and $E_{out}(j)$ denote the sets of incoming and outgoing edges of v_j, respectively.

In addition to edge inputs, a cell also takes the hidden states of its incoming nodes and outgoing nodes during a state transition. In particular, the states of all incoming nodes and outgoing nodes are summed up before being passed to the cell and gate nodes:

$$h_j^i = \sum_{(i,j,l) \in E_{in}(j)} h_{t-1}^i$$
$$h_j^o = \sum_{(j,k,l) \in E_{out}(j)} h_{t-1}^k,$$

Based on the above definitions of x_j^i, x_j^o, h_j^i and h_j^o, the state transition from g_{t-1} to g_t, as represented by h_t^j, can be defined as:

$$\begin{aligned}
i_t^j &= \sigma(W_i x_j^i + \hat{W}_i x_j^o + U_i h_j^i + \hat{U}_i h_j^o + b_i),\\
o_t^j &= \sigma(W_o x_j^i + \hat{W}_o x_j^o + U_o h_j^i + \hat{U}_o h_j^o + b_o),\\
f_t^j &= \sigma(W_f x_j^i + \hat{W}_f x_j^o + U_f h_j^i + \hat{U}_f h_j^o + b_f),\\
u_t^j &= \sigma(W_u x_j^i + \hat{W}_u x_j^o + U_u h_j^i + \hat{U}_u h_j^o + b_u),\\
c_t^j &= f_t^j \odot c_{t-1}^j + i_t^j \odot u_t^j,\\
h_t^j &= o_t^j \odot \tanh(c_t^j),
\end{aligned}$$

where i_t^j, o_t^j and f_t^j are the input, output and forget gates mentioned earlier. W_x, $\hat{W}_x$, U_x, $\hat{U}_x$, b_x, where $x \in \{i, o, f, u\}$, are model parameters.

3.2 Recurrent steps

Using the above state transition mechanism, information from each node propagates to all its neighboring nodes after each step. Therefore, for the worst case where the input graph is a chain of nodes, the maximum number of steps necessary for information from one arbitrary node to reach another is equal to the size of the graph. We experiment with different transition steps to study the effectiveness of global encoding.

Note that unlike the sequence LSTM encoder, our graph encoder allows parallelization in node-state updates, and thus can be highly efficient using a GPU. It is general and can be potentially applied to other tasks, including sequences, syntactic trees and cyclic structures.

3.3 Input Representation

Different from sequences, the edges of an AMR graph contain labels, which represent relations between the nodes they connect, and are thus important for modeling the graphs. Similar with Section 2, we adopt two different ways for calculating the representation for each edge (i, j, l):

$$x_{i,j}^l = W_4\Big([e_l; e_i]\Big) + b_4 \tag{6}$$

$$x_{i,j}^l = W_4\Big([e_l; e_i; h_i^c]\Big) + b_4, \tag{7}$$

where e_l and e_i are the embeddings of edge label l and source node v_i, h_i^c denotes the last hidden state of the character LSTM over v_i, and W_4 and b_4 are trainable parameters. The equations correspond to Equations 1 and 2 in Section 2.1, respectively.

3.4 Decoder

We adopt the attention-based LSTM decoder as described in Section 2.3. Since our graph encoder generates a sequence of graph states, only the last graph state is adopted in the decoder. In particular, we make the following changes to the decoder. First, each attention vector becomes $a_j = [h_T^j; x_j]$, where h_T^j is the last state for node v_j. Second, the decoder initial state s_{-1} is the average of the last states of all nodes.

3.5 Integrating the copy mechanism

Open-class tokens, such as dates, numbers and named entities, account for a large portion in the AMR corpus. Most appear only a few times, resulting in a data sparsity problem. To address this issue, Konstas et al. (2017) adopt anonymization for dealing with the data sparsity problem. In particular, they first replace the subgraphs that represent dates, numbers and named entities (such as "(q / quantity :quant 3)" and "(p / person :name (n / name :op1 "Ryan"))") with predefined placeholders (such as "num_0" and "person_name_0") before decoding, and then recover the corresponding surface tokens (such as "3" and "Ryan") after decoding. This method involves hand-crafted rules, which can be costly.

Copy We find that most of the open-class tokens in a graph also appear in the corresponding sentence, and thus adopt the copy mechanism (Gulcehre et al., 2016; Gu et al., 2016) to solve this problem. The mechanism works on top of an attention-based RNN decoder by integrating the attention distribution into the final vocabulary distribution. The final probability distribution is defined as the interpolation between two probability distributions:

$$P_{final} = \theta_t P_{vocab} + (1 - \theta_t) P_{attn}, \tag{8}$$

where θ_t is a switch for controlling generating a word from the vocabulary or directly copying it from the input graph. P_{vocab} is the probability distribution of directly generating the word, as defined in Equation 5, and P_{attn} is calculated based on the attention distribution α_t by summing the probabilities of the graph nodes that contain identical concept. Intuitively, θ_t is relevant to the current decoder input e_t and state s_t, and the context vector μ_t. Therefore, we define it as:

$$\theta_t = \sigma(w_\mu^T \mu_t + w_s^T s_t + w_e^T e_t + b_5), \tag{9}$$

where vectors w_μ, w_s, w_e and scalar b_5 are model parameters. The copy mechanism favors generating words that appear in the input. For AMR-to-text generation, it facilitates the generation of dates, numbers, and named entities that appear in AMR graphs.

Copying vs anonymization Both copying and anonymization alleviate the data sparsity problem by handling the open-class tokens. However, the copy mechanism has the following advantages over anonymization: (1) anonymization requires significant manual work to define the placeholders and heuristic rules both from subgraphs to placeholders and from placeholders to the surface tokens, (2) the copy mechanism automatically learns what to copy, while anonymization relies on hard rules to cover all types of the open-class tokens, and (3) the copy mechanism is easier to adapt to new domains and languages than anonymization.

4 Training and decoding

We train our models using the cross-entropy loss over each gold-standard output sequence $W^* = w_1^*, \ldots, w_t^*, \ldots, w_M^*$:

$$l = -\sum_{t=1}^{M} \log p(w_t^* | w_{t-1}^*, \ldots, w_1^*, X; \theta), \quad (10)$$

where X is the input graph, and θ is the model parameters. Adam (Kingma and Ba, 2014) with a learning rate of 0.001 is used as the optimizer, and the model that yields the best devset performance is selected to evaluate on the test set. Dropout with rate 0.1 is used during training. Beam search with beam size to 5 is used for decoding. Both training and decoding use Tesla K80 GPUs.

5 Experiments

5.1 Data

We use a standard AMR corpus (LDC2015E86) as our experimental dataset, which contains 16,833 instances for training, 1368 for development and 1371 for test. Each instance contains a sentence and an AMR graph.

Following Konstas et al. (2017), we supplement the gold data with large-scale automatic data. We take Gigaword as the external data to sample raw sentences, and train our model on both the sampled data and LDC2015E86. We adopt Konstas et al. (2017)'s strategy for sampling sentences from Gigaword, and choose JAMR (Flanigan et al., 2016a) to parse selected sentences into AMRs, as the AMR parser of Konstas et al. (2017) only works on the anonymized data. For training on both sampled data and LDC2015E86, we also follow the method of Konstas et al. (2017), which is fine-tuning the model on the AMR corpus after every epoch of pretraining on the gigaword data.

Model	BLEU	Time
Seq2seq	18.8	35.4s
Seq2seq+copy	19.9	37.4s
Seq2seq+charLSTM+copy	20.6	39.7s
Graph2seq	20.4	11.2s
Graph2seq+copy	22.2	11.1s
Graph2seq+Anon	22.1	9.2s
Graph2seq+charLSTM+copy	**22.8**	16.3s

Table 1: DEV BLEU scores and decoding times.

5.2 Settings

We extract a vocabulary from the training set, which is shared by both the encoder and the decoder. The word embeddings are initialized from Glove pretrained word embeddings (Pennington et al., 2014) on Common Crawl, and are not updated during training. Following existing work, we evaluate the results with the BLEU metric (Papineni et al., 2002).

For model hyperparameters, we set the graph state transition number as 9 according to development experiments. Each node takes information from at most 10 neighbors. The hidden vector sizes for both encoder and decoder are set to 300 (They are set to 600 for experiments using large-scale automatic data). Both character embeddings and hidden layer sizes for character LSTMs are set 100, and at most 20 characters are taken for each graph node or linearized token.

5.3 Development experiments

As shown in Table 1, we compare our model with a set of baselines on the AMR devset to demonstrate how the graph encoder and the copy mechanism can be useful when training instances are not sufficient. *Seq2seq* is the sequence-to-sequence baseline described in Section 2. *Seq2seq+copy* extends *Seq2seq* with the copy mechanism, and *Seq2seq+charLSTM+copy* further extends *Seq2seq+copy* with character LSTM. *Graph2seq* is our graph-to-sequence model, *Graph2seq+copy* extends *Graph2seq* with the copy mechanism, and *Graph2seq+charLSTM+copy* further extends

Graph2seq+copy with the character LSTM. We also try *Graph2seq+Anon*, which applies our graph-to-sequence model on the anonymized data from Konstas et al. (2017).

The graph encoder As can be seen from Table 1, the performance of *Graph2seq* is 1.6 BLEU points higher than *Seq2seq*, which shows that our graph encoder is effective when applied alone. Adding the copy mechanism (*Graph2seq+copy* vs *Seq2seq+copy*), the gap becomes 2.3. This shows that the graph encoder learns better node representations compared to the sequence encoder, which allows attention and copying to function better.

Applying the graph encoder together with the copy mechanism gives a gain of 3.4 BLEU points over the baseline (*Graph2seq+copy* vs *Seq2seq*). The graph encoder is consistently better than the sequence encoder no matter whether character LSTMs are used.

We also list the encoding part of decoding times on the devset, as the decoders of the seq2seq and the graph2seq models are similar, so the time differences reflect efficiencies of the encoders. Our graph encoder gives consistently better efficiency compared with the sequence encoder, showing the advantage of parallelization.

The copy mechanism Table 1 shows that the copy mechanism is effective on both the graph-to-sequence and the sequence-to-sequence models. Anonymization gives comparable overall performance gains on our graph-to-sequence model as the copy mechanism (comparing *Graph2seq+Anon* with *Graph2seq+copy*). However, the copy mechanism has several advantages over anonymization as discussed in Section 3.5.

Character LSTM Character LSTM helps to increase the performances of both systems by roughly 0.6 BLEU points. This is largely because it further alleviates the data sparsity problem by handling unseen words, which may share common substrings with in-vocabulary words.

5.4 Effectiveness on graph state transitions

We report a set of development experiments for understanding the graph LSTM encoder.

Number of iterations We analyze the influence of the number of state transitions to the model performance on the devset. Figure 3 shows the BLEU scores of different state transition numbers,

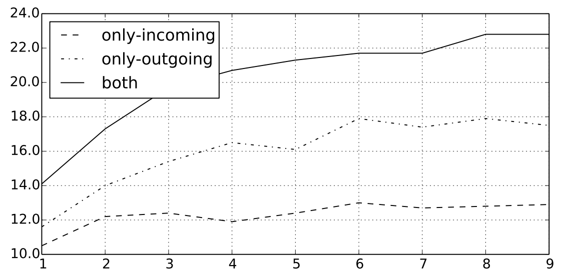


Figure 3: DEV BLEU scores against transition steps for the graph encoder.

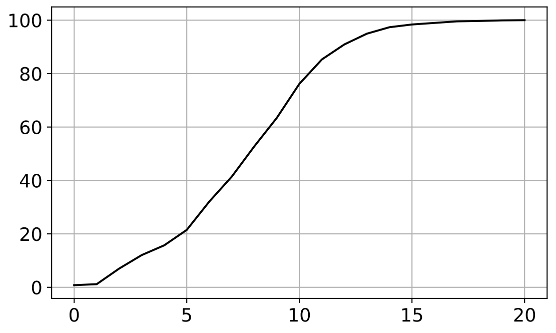


Figure 4: Percentage of DEV AMRs with different diameters.

when both incoming and outgoing edges are taken for calculating the next state (as shown in Figure 2). The system is *Graph2seq+charLSTM+copy*. Executing only 1 iteration results in a poor BLEU score of 14.1. In this case the state for each node only contains information about immediately adjacent nodes. The performance goes up dramatically to 21.5 when increasing the iteration number to 5. In this case, the state for each node contains information of all nodes within a distance of 5. The performance further goes up to 22.8 when increasing the iteration number from 5 to 9, where all nodes with a distance of less than 10 are incorporated in the state for each node.

Graph diameter We analyze the percentage of the AMR graphs in the devset with different graph diameters and show the cumulative distribution in Figure 4. The diameter of an AMR graph is defined as the longest distance between two AMR nodes.[1] Even though the diameters for less than 80% of the AMR graphs are less or equal than 10, our development experiments show that it is not necessary to incorporate the whole-graph information for each node. Further increasing state transition number may lead to additional improvement.

[1]The diameter of single-node graphs is 0.

Model	BLEU
PBMT	26.9
SNRG	25.6
Tree2Str	23.0
MSeq2seq+Anon	22.0
Graph2seq+copy	22.7
Graph2seq+charLSTM+copy	23.3
MSeq2seq+Anon (200K)	27.4
MSeq2seq+Anon (2M)	32.3
Seq2seq+charLSTM+copy (200K)	27.4
Seq2seq+charLSTM+copy (2M)	31.7
Graph2seq+charLSTM+copy (200K)	28.2
Graph2seq+charLSTM+copy (2M)	**33.0**

Table 2: TEST results. "(200K)", "(2M)" and "(20M)" represent training with the corresponding number of additional sentences from Gigaword.

We do not perform exhaustive search for finding the optimal state transition number.

Incoming and outgoing edges As shown in Figure 3, we analyze the efficiency of state transition when only incoming or outgoing edges are used. From the results, we can see that there is a huge drop when state transition is performed only with incoming or outgoing edges. Using edges of one direction, the node states only contain information of ancestors or descendants. On the other hand, node states contain information of ancestors, descendants, and siblings if edges of both directions are used. From the results, we can conclude that not only the ancestors and descendants, but also the siblings are important for modeling the AMR graphs. This is similar to observations on syntactic parsing tasks (McDonald et al., 2005), where sibling features are adopted.

We perform a similar experiment for the *Seq2seq+copy* baseline by only executing single-directional LSTM for the encoder. We observe BLEU scores of 11.8 and 12.7 using only forward or backward LSTM, respectively. This is consistent with our graph model in that execution using only one direction leads to a huge performance drop. The contrast is also reminiscent of using the normal input versus the reversed input in neural machine translation (Sutskever et al., 2014).

5.5 Results

Table 2 compares our final results with existing work. *MSeq2seq+Anon* (Konstas et al., 2017) is an attentional multi-layer sequence-to-sequence model trained with the anonymized data. *PBMT* (Pourdamghani et al., 2016) adopts a phrase-based model for machine translation (Koehn et al., 2003) on the input of linearized AMR graph, *SNRG* (Song et al., 2017) uses synchronous node replacement grammar for parsing the AMR graph while generating the text, and *Tree2Str* (Flanigan et al., 2016b) converts AMR graphs into trees by splitting the re-entrances before using a tree transducer to generate the results.

Graph2seq+charLSTM+copy achieves a BLEU score of 23.3, which is 1.3 points better than *MSeq2seq+Anon* trained on the same AMR corpus. In addition, our model without character LSTM is still 0.7 BLEU points higher than *MSeq2seq+Anon*. Note that *MSeq2seq+Anon* relies on anonymization, which requires additional manual work for defining mapping rules, thus limiting its usability on other languages and domains. The neural models tend to underperform statistical models when trained on limited (16K) gold data, but performs better with scaled silver data (Konstas et al., 2017).

Following Konstas et al. (2017), we also evaluate our model using both the AMR corpus and sampled sentences from Gigaword. Using additional 200K or 2M gigaword sentences, *Graph2seq+charLSTM+copy* achieves BLEU scores of 28.2 and 33.0, respectively, which are 0.8 and 0.7 BLEU points better than *MSeq2seq+Anon* using the same amount of data, respectively. The BLEU scores are 5.3 and 10.1 points better than the result when it is only trained with the AMR corpus, respectively. This shows that our model can benefit from scaled data with automatically generated AMR graphs, and it is more effective than *MSeq2seq+Anon* using the same amount of data. Using 2M gigaword data, our model is better than all existing methods. Konstas et al. (2017) also experimented with 20M external data, obtaining a BLEU of 33.8. We did not try this setting due to hardware limitations. The *Seq2seq+charLSTM+copy* baseline trained on the large-scale data is close to *MSeq2seq+Anon* using the same amount of training data, yet is much worse than our model.

5.6 Case study

We conduct case studies for better understanding the model performances. Table 3 shows example outputs of sequence-to-sequence (*S2S*), graph-to-sequence (*G2S*) and graph-to-sequence with copy mechanism (*G2S+CP*). *Ref* denotes the reference output sentence, and *Lin* shows the serialization results of input AMRs. The best hyperparameter configuration is chosen for each model.

For the first example, *S2S* fails to recognize the concept “a / account” as a noun and loses the concept “o / old” (both are underlined). The fact that “a / account” is a noun is implied by “a / account :mod (o / old)” in the original AMR graph. Though directly connected in the original graph, their distance in the serialization result (the input of *S2S*) is 26, which may be why *S2S* makes these mistakes. In contrast, *G2S* handles “a / account” and “o / old” correctly. In addition, the copy mechanism helps to copy “look-over” from the input, which rarely appears in the training set. In this case, *G2S+CP* is incorrect only on hyphens and literal reference to “anti-japanese war”, although the meaning is fully understandable.

For the second case, both *G2S* and *G2S+CP* correctly generate the noun “agreement” for “a / agree” in the input AMR, while *S2S* fails to. The fact that “a / agree” represents a noun can be determined by the original graph segment “p / provide :ARG0 (a / agree)”, which indicates that “a / agree” is the subject of “p / provide”. In the serialization output, the two nodes are close to each other. Nevertheless, *S2S* still failed to capture this structural relation, which reflects the fact that a sequence encoder is not designed to explicitly model hierarchical information encoded in the serialized graph. In the training instances, serialized nodes that are close to each other can originate from neighboring graph nodes, or distant graph nodes, which prevents the decoder from confidently deciding the correct relation between them. In contrast, *G2S* sends the node “p / provide” simultaneously with relation “ARG0” when calculating hidden states for “a / agree”, which facilitates the yielding of “the agreement provides”.

(p / possible-01 :polarity -
  :ARG1 (l / look-over-06
    :ARG0 (w / we)
    :ARG1 (a / account-01
      :ARG1 (w2 / war-01
        :ARG1 (c2 / country :wiki “Japan”
          :name (n2 / name :op1 “Japan”))
        :time (p2 / previous)
        :ARG1-of (c / call-01
          :mod (s / so)))
      :mod (o / old))))

Lin: possible :polarity - :arg1 (look-over :arg0 we :arg1 (account :arg1 (war :arg1 (country :wiki japan :name (name :op1 japan)) :time previous :arg1-of (call :mod so)) :mod old))
Ref: we can n’t look over the old accounts of the previous so-called anti-japanese war .
S2S: we can n’t be able to account the past drawn out of japan ’s entire war .
G2S: we can n’t be able to do old accounts of the previous and so called japan war.
G2S+CP: we can n’t look-over the old accounts of the previous so called war on japan .

(p / provide-01
  :ARG0 (a / agree-01)
  :ARG1 (a2 / and
    :op1 (s / staff
      :prep-for (c / center
        :mod (r / research-01)))
    :op2 (f / fund-01
      :prep-for c)))

Lin: provide :arg0 agree :arg1 (and :op1 (staff :prep-for (center :mod research)) :op2 (fund :prep-for center))
Ref: the agreement will provide staff and funding for the research center .
S2S: agreed to provide research and institutes in the center .
G2S: the agreement provides the staff of research centers and funding .
G2S+CP: the agreement provides the staff of the research center and the funding .

Table 3: Example system outputs.

6 Related work

Among early statistical methods for AMR-to-text generation, Flanigan et al. (2016b) convert input graphs to trees by splitting re-entrances, and then translate the trees into sentences with a tree-to-string transducer. Song et al. (2017) use a synchronous node replacement grammar to parse input AMRs and generate sentences at the same time. Pourdamghani et al. (2016) linearize input graphs by breadth-first traversal, and then use a phrase-based machine translation system[2] to generate results by translating linearized sequences.

Prior work using graph neural networks for NLP include the use graph convolutional networks (GCN) (Kipf and Welling, 2017) for semantic role labeling (Marcheggiani and Titov, 2017) and neural machine translation (Bastings et al., 2017). Both GCN and the graph LSTM update node states by exchanging information between neighboring nodes within each iteration. However, our graph state LSTM adopts gated operations for making updates, while GCN uses a linear transformation. Intuitively, the former has better learning power than the later. Another major difference is that our graph state LSTM keeps a cell vector for each node to remember all history. The contrast

[2]http://www.statmt.org/moses/

between our model with GCN is reminiscent of the contrast between RNN and CNN. We leave empirical comparison of their effectiveness to future work. In this work our main goal is to show that graph LSTM encoding of AMR is superior compared with sequence LSTM.

Closest to our work, Peng et al. (2017) modeled syntactic and discourse structures using DAG LSTM, which can be viewed as extensions to tree LSTMs (Tai et al., 2015). The state update follows the sentence order for each node, and has sequential nature. Our state update is in parallel. In addition, Peng et al. (2017) split input graphs into separate DAGs before their method can be used. To our knowledge, we are the first to apply an LSTM structure to encode AMR graphs.

The recurrent information exchange mechanism in our state transition process is remotely related to the idea of loopy belief propagation (LBP) (Murphy et al., 1999). However, there are two major differences. First, messages between LSTM states are gated neural node values, rather than probabilities in LBP. Second, while the goal of LBP is to estimate marginal probabilities, the goal of information exchange between graph states in our LSTM is to find neural representation features, which are directly optimized by a task objective.

In addition to NMT (Gulcehre et al., 2016), the copy mechanism has been shown effective on tasks such as dialogue (Gu et al., 2016), summarization (See et al., 2017) and question generation (Song et al., 2018). We investigate the copy mechanism on AMR-to-text generation.

7 Conclusion

We introduced a novel graph-to-sequence model for AMR-to-text generation. Compared to sequence-to-sequence models, which require linearization of AMR before decoding, a graph LSTM is leveraged to directly model full AMR structure. Allowing high parallelization, the graph encoder is more efficient than the sequence encoder. In our experiments, the graph model outperforms a strong sequence-to-sequence model, achieving the best performance.

Acknowledgments We thank the anonymized reviewers for the insightful comments, and the Center for Integrated Research Computing (CIRC) of University of Rochester for providing computation resources.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*, pages 1957–1967, Copenhagen, Denmark.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016a. CMU at semeval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016b. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-16)*, pages 731–739.

Normunds Gruzitis, Didzis Gosko, and Guntis Barzdins. 2017. RIGOTRIO at SemEval-2017 Task 9: Combining Machine Learning and Grammar Engineering for AMR Parsing and Generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 924–928, Vancouver, Canada.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 1631–1640, Berlin, Germany.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 140–149, Berlin, Germany.

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

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of* *the International Conference on Computational Linguistics (COLING-12)*, pages 1359–1376.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, pages 48–54.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 146–157, Vancouver, Canada.

Gerasimos Lampouras and Andreas Vlachos. 2017. Sheffield at semeval-2017 task 9: Transition-based language generation from amr. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 586–591, Vancouver, Canada.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of the First Workshop on Computing News Storylines*, pages 11–15, Beijing, China.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-17)*, pages 1506–1515, Copenhagen, Denmark.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 91–98, Ann Arbor, Michigan.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. Forge at semeval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 920–923, Vancouver, Canada.

Arindam Mitra and Chitta Baral. 2015. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-16)*.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 311–318.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 1532–1543.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *International Conference on Natural Language Generation (INLG-16)*, pages 21–25, Edinburgh, UK.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 1073–1083, Vancouver, Canada.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. AMR-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 7–13, Vancouver, Canada.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-18)*, New Orleans.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 1556–1566, Beijing, China.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 1054–1059, Austin, Texas.

Aleš Tamchyna, Chris Quirk, and Michel Galley. 2015. A discriminative model for semantics-to-string translation. In *Proceedings of the 1st Workshop on Semantics-Driven Statistical Machine Translation (S2MT 2015)*, pages 30–36, Beijing, China.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 76–85, Berlin, Germany.

GTR-LSTM: A Triple Encoder for Sentence Generation from RDF Data

Bayu Distiawan Trisedya[1], **Jianzhong Qi**[1], **Rui Zhang**[1*], **Wei Wang**[2]
[1] The University of Melbourne
[2] University of New South Wales
btrisedya@student.unimelb.edu.au
{jianzhong.qi,rui.zhang}@unimelb.edu.au
weiw@cse.unsw.edu.au

Abstract

A knowledge base is a large repository of facts that are mainly represented as RDF triples, each of which consists of a subject, a predicate (relationship), and an object. The RDF triple representation offers a simple interface for applications to access the facts. However, this representation is not in a natural language form, which is difficult for humans to understand. We address this problem by proposing a system to translate a set of RDF triples into natural sentences based on an encoder-decoder framework. To preserve as much information from RDF triples as possible, we propose a novel graph-based triple encoder. The proposed encoder encodes not only the elements of the triples but also the relationships both within a triple and between the triples. Experimental results show that the proposed encoder achieves a consistent improvement over the baseline models by up to 17.6%, 6.0%, and 16.4% in three common metrics BLEU, METEOR, and TER, respectively.

1 Introduction

Knowledge bases (KBs) are becoming an enabling resource for many applications including Q&A systems, recommender systems, and summarization tools. KBs are designed based on a W3C standard called the *Resource Description Framework* (RDF)[1]. An RDF triple consists of three elements in the form of $\langle subject, predicate\ (relationship), object\rangle$. It describes a relationship between an entity (the subject) and another entity or literal (the object) via the predicate. This representation allows easy data share between KBs. However, usually the elements of a triple are stored as *Uniform Resource Identifiers* (URIs), and many predicates (words or phrases) are not intuitive; this representation is difficult to comprehend by humans.

RDF triples	⟨John Doe,birth place,London⟩ ⟨John Doe,birth date,1967-01-10⟩ ⟨London,capital of,England⟩
Target sentence	John Doe was born on 1967-01-10 in London, the capital of England.

Table 1: RDF based sentence generation.

Translating RDF triples into natural sentences helps humans to comprehend the knowledge embedded in the triples, and building a natural language based user interface is an important task in user interaction studies (Damljanovic et al., 2010). This task has many applications, such as question answering (Bordes et al., 2014; Fader et al., 2014), profile summarizing (Lebret et al., 2016; Chisholm et al., 2017), and automatic weather forecasting (Mei et al., 2016). For example, the SPARQL inference of a Q&A system (Unger et al., 2012) returns a set of RDF triples which need to be translated into natural sentences to provide a more readable answer for the users. Table 1 illustrates such an example. Suppose a user is asking a question about "John Doe". By querying a KB, a Q&A system retrieves three triples "⟨John Doe,birth place,London⟩", "⟨John Doe,birth date,1967-01-10⟩", and "⟨London,capital of,England⟩." We aim to generate a natural sentence that incorporates the information of the triples and is easier to be understood by the user. In this example, the generated sentence is "John Doe was born on 1967-01-10 in London,

*Corresponding author
[1]https://www.w3.org/RDF/

`the capital of England.`"

Most existing studies for this task use domain specific rules. Bontcheva and Wilks (2004) create rules to generate sentences in the medical domain, while Cimiano et al. (2013) create rules to generate step by step cooking instructions. The problem of rule-based methods is that they need a lot of human efforts to create the rules, which mostly cannot deal with complex or novel cases.

Recent studies propose neural language generation systems. Lebret et al. (2016) generate the first sentence of a biography by a conditional neural language model. Mei et al. (2016) propose an encoder-aligner-decoder architecture to generate weather forecasts. The model does not need predefined rules and hence generalizes better to open domain data.

A straightforward adaptation of neural language generation system is to use the encoder-decoder model by first concatenating the elements of the RDF triples into a linear sequence and then feeding the sequence as the model input to learn the corresponding natural sentence. We implemented such a model (detailed in Section 3.2) that ranked top in the WebNLG Challenge 2017[2]. This Challenge has a primary objective of generating syntactically correct natural sentences from a set of RDF triples. Our model achieves the highest global scores on the automatic evaluation, outperforming competitors that use rule-based methods, statistical machine translation, and neural machine translation (Gardent et al., 2017b).

While our previous model achieves a good result, simply concatenating the elements in the RDF triples may lose the relationship between entities that affects the semantics of the resulting sentence (cf. Table 3). To address this issue, in this paper, we propose a novel graph-based triple encoder model that maintain the structure of RDF triples as a small knowledge graph named the *GTR-LSTM* model. This model computes the hidden state of each entity in a graph to preserve the relationships between entities in a triple (*intra-triple relationships*) and the relationships between entities in related triples (*inter-triple relationships*) that helps to achieve even more accurate sentences. This leads to two problems of preserving the relationships in a knowledge graph: (1) how to deal with a cycle in a knowledge graph; (2) how to deal with multiple non-predefined relationships between two entities in a knowledge graph. The proposed model differs from existing non-linear LSTM models such as Tree LSTM (Tai et al., 2015) and Graph LSTM (Liang et al., 2016) in addressing the mentioned problem. In particular, Tree LSTM does not allow cycles, while the proposed model handles cycles by first using a combination of topological sort and breadth-first traversal over a graph, and then using an attention model to capture the global information of the knowledge graph. Meanwhile, Graph LSTM only allows a predefined set of relationships between entities, while the proposed model allows any relationships by treating them as part of the input for the hidden state computation.

[2]http://talc1.loria.fr/webnlg/stories/challenge.html

To further enhance the capability of our model to handle unseen entities, we propose to use entity masking, which maps the entities in the model training pairs to their types, e.g., we map an entity (literal) "`1967-01-10`" to a type symbol "`DATE`" in the training pairs. This way, our model can learn to handle any date entities rather than just "`1967-01-10`". This is particularly helpful when there is a limited training dataset.

Our contributions are:

- We propose an end-to-end encoder-decoder based framework for the problem of translating RDF triples into natural sentences.
- We further propose a graph-based triple encoder to optimize the amount of information preserved in the input of the framework. The proposed model can handle cycles to capture the global information of a knowledge graph. The proposed model also handles non-predefined relationships between entities.
- We evaluate the proposed framework and model over two real datasets. The results show that our model outperforms the state-of-the-art models consistently.

The rest of this paper is organized as follows. Section 2 summarizes previous studies on sentence generation. Section 3 details the proposed model. Section 4 presents the experimental results. Section 5 concludes the paper.

2 Related Work

The studied problem falls in the area of *Natural Language Generation* (NLG) (Reiter and Dale, 2000). Bontcheva and Wilks (2004) follow a traditional NLG approach to generate sentences from RDF data in the medical domain. They start with filtering repetitive RDF data (document planning) and then group coherent triples (micro-planning). After that, they aggregate the sentences generated for coherent triples to produce the final sentences (aggregation and realization). Cimiano et al. (2013) generate cooking recipes from semantic web data. They focus on using a large corpus to extract lexicon in the cooking domain. The lexicon is then used with a traditional NLG approach to generate cooking recipes. Duma and Klein (2013) learn a sentence template from a parallel RDF data and text corpora. They first align entities in RDF triples with entities mentioned in sentences. Then, they extract templates from the aligned sentences by replacing the entity mention with a unique token. This method works well on RDF triples in a seen domain but fails on RDF triples in a previously unseen domain.

Recently, several methods using neural networks are proposed. Lebret et al. (2016) generate the first sentence of a biography using a conditional neural language model. This model is trained to predict the next word of a sentence not only based on previous words, but also by using features captured from Wikipedia infoboxes. Mei et al. (2016) propose an encoder-aligner-decoder model to generate weather forecasts. The aligner is used to filter the most relevant data to be used to predict the weather forecast. Both studies experiment on cross-domain datasets. The result shows that the neural language generation approach is more flexible to work in an open domain since it is not limited to handcrafted rules. This motivates us to use a neural network based framework.

The most similar system to ours is Neural Wikipedian (Vougiouklis et al., 2017), which generates a summary from RDF triples. It uses feed-forward neural networks to encode RDF triples and concatenate them as the input of the decoder. The decoder uses LSTM to predict a sequence of words as a summary. There are differences from our work. First, Neural Wikipedian only works with a set of RDF triples with a single entity point of view (i.e., the entity of interest must be in either the subject or the object of every triple). Our system does not have this constraint. Second, Neural Wikipedian uses standard feed-forward neural networks in the encoder. We design new triple encoder models to accommodate specific features of RDF triples. Experimental results show that our framework outperforms Neural Wikipedian.

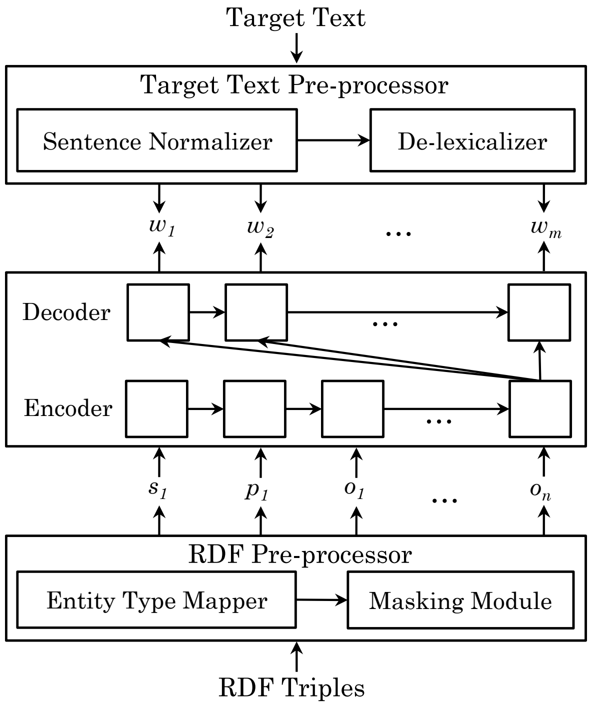


Figure 1: RDF sentence generation based on an encoder-decoder architecture.

3 Proposed Model

We start with the problem definition. We consider a set of RDF triples as the input, which is denoted by $T = [t_1, t_2, ..., t_n]$ where a triple t_i consists of three elements (subject s_i, predicate p_i, and object o_i), $t_i = \langle s_i, p_i, o_i \rangle$. Every element can contain multiple words. We aim to generate a set of sentences that consist of a sequence of words $S = \langle w_1, w_2, ..., w_m \rangle$, such that the relationships in the input triples are correctly represented in S while the sentences have a high quality. We use BLEU, METEOR, and TER to assess the quality of the sentence (detailed in Section 4). Table 1 illustrates our problem input and the target output.

This section is organized as follows. First we describe the overall framework (Section 3.1). Next, we describe three triple encoder models including the *adapted standard BLSTM* model (Section 3.2), the *adapted standard triple encoder* model (Section 3.3), and the *proposed GTR-LSTM* model (Section 3.4). The decoder which is used for all encoder models is described in Section 3.5. The entity masking is described in Section 3.6

3.1 Solution Framework

Our solution framework uses an encoder-decoder architecture as illustrated in Fig. 1. The framework consists of three components including an *RDF pre-processor*, a *target text pre-processor*, and an *encoder-decoder module*.

The RDF pre-processor consists of an entity type mapper and a masking module. The entity type mapper maps the subjects and objects in the triples to their types, such that the sentence patterns learned are based on entity types rather than entities. For example, the input entities in Table 1, "`John Doe`", "`London`", "`England`", and "`1967-01-10`" can be mapped to "`PERSON`", "`CITY`", "`COUNTRY`", and "`DATE`", respectively. The mapping has been shown in our experiments to be highly effective in improving the model output quality. The masking module converts each entity into an *entity identifier* (*eid*). The target text pre-processor consists of a text normalizer and a de-lexicalizer. The text normalizer converts abbreviations and dates into the same format as the corresponding entities in the triples. The de-lexicalizer replaces all entities in the target sentences by their eids. The RDF and target text pre-processors are detailed in Section 3.6. The replaced target sentences are combined with the original target sentences and the English Wikipedia articles is used as a corpus to learn the word embeddings of the vocabulary.

To accommodate the RDF data, in the encoder side, we consider three triple encoder models: (1) the adapted standard BLSTM encoder; (2) the adapted standard triple encoder; and (3) the proposed GTR-LSTM triple encoder. The adapted standard BLSTM encoder concatenates the tokens in RDF triples as an input sequence, while the standard triple encoder first encodes each RDF triple into a vector representation and then concatenates the vectors of different triples. The latter model better captures intra-triple relationships but suffers in capturing inter-triple relationships. Considering the native representation of RDF triples as a small knowledge graph, our graph-based GTR-LSTM triple encoder captures both intra-triple and inter-triple entity relationships.

3.2 Adapted Standard BLSTM Encoder

The standard encoder-decoder model with a BLSTM encoder is a sequence to sequence learning model (Cho et al., 2014). To adapt such a model for our problem, we transform a set of RDF triples input T into a sequence of elements (i.e., $T = [w_{1,1}, w_{1,2}, ..., w_{1,j}, ..., w_{n,j}]$), where $w_{n,j}$ is the word embedding of a word in the n-th triple. For example, following the triples in Table 1, $w_{1,1}$ is the word embedding of "John", $w_{1,2}$ is the word embedding of "Doe", etc. This sequence forms an input for the encoder. We use zero padding to ensure that each input has the same representation size. The rest of the model is the same as the standard encoder-decoder model with an attention mechanism (Bahdanau et al., 2015). We call this model the *adapted standard BLSTM encoder*.

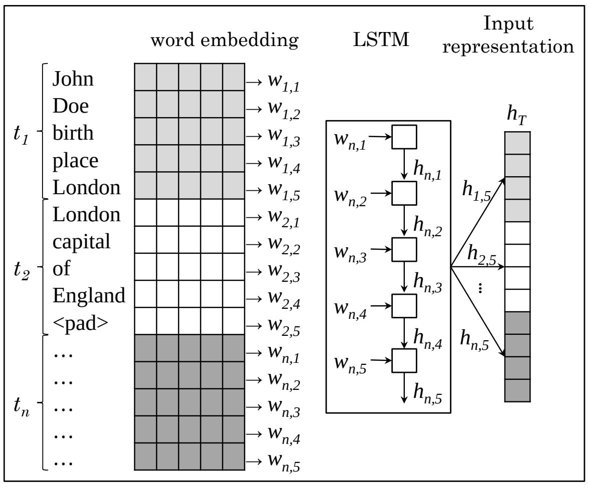


Figure 2: LSTM-based standard triple encoder.

3.3 Adapted Standard Triple Encoder

The standard BLSTM encoder suffers in capturing the element relationships as the elements are simply concatenated together. Next, we adapt the standard BLSTM encoder to aggregate the word embeddings of the elements of the same triple to retain the intra-triple relationship. We call this the *adapted standard triple encoder*.

The adaptation is done by grouping the elements of each triple, so the input is represented as $T = [\langle w_{1,1}, ..., w_{1,j}\rangle, ..., \langle w_{n,1}, ...w_{n,j}\rangle]$, where $w_{n,j}$ is the word embedding of a word in the n-th triple. We use zero padding to ensure that each triple has the same representation size. An LSTM network of the encoder computes a hidden state of each triple and concatenates them together to be the input for the decoder:

$$h_T = [f(t_1); f(t_2); ...; f(t_n)] \qquad (1)$$

where h_T is the input vector representation for the decoder and f is an LSTM network (cf. Fig. 2).

3.4 GTR-LSTM Triple Encoder

The adapted standard triple encoder has an advantage in preserving the intra-triple relationship. However, it has not considered the structural rela-

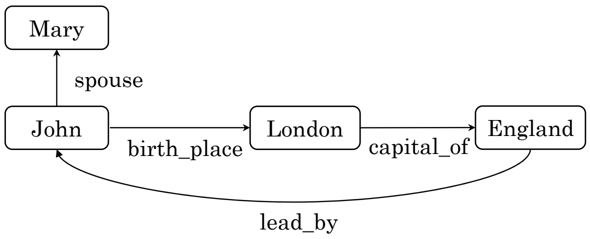


Figure 3: A small knowledge graph formed by a set of RDF triples.

tionships between the entities in different triples. To overcome this limitation, we propose a graph-based triple encoder. We call it the *GTR-LSTM triple encoder*. This encoder takes the input triples in the form of a graph, which preserves the natural structure of the triples (cf. Fig. 3).

GTR-LSTM differs from existing Graph LSTM (Liang et al., 2016) and Tree LSTM (Tai et al., 2015) models in the following aspects. Graph LSTM is proposed for image data. It constructs the graph based on the spatial relationships among super-pixels of an image. Tree LSTM uses the dependency tree as the structure of a sentence. Both models have a predefined relationship between the vertices (Graph LSTM uses spatial relationships: top, bottom, left, or right between super-pixels; Tree LSTM uses dependencies between words in a sentence as the relationship). In contrast, a KB has an open set of relationships between the vertices (i.e., the predicate defines the relationship between entities/vertices) which make our problem more difficult to model.

Our GTR-LSTM triple encoder overcomes the difficulty as follows. It receives a directed graph $G = \langle V, E \rangle$ as the input, where V is a set of vertices that represent entities or literals, and E is a set of directed edges that represent predicates. Since the graph can contain cycles, we use a combination of *topological sort* and *breadth-first traversal* algorithms to traverse the graph. The traversal is used to create an ordering of feeding the vertices into a GTR-LSTM unit to compute their hidden states. We start with running a topological sort to establish an order of the vertices until no further vertex has a zero in-degree. For the remaining vertices, they must be in strongly connected component(s). Then, we run a breadth-first traversal over the remaining vertices with a random starting vertex, since every vertex can be reached from all vertices of a strongly connected component. When a vertex v_i is visited, the hidden states of all adjacent vertices of v_i are computed (or updated if the hidden state of the vertex

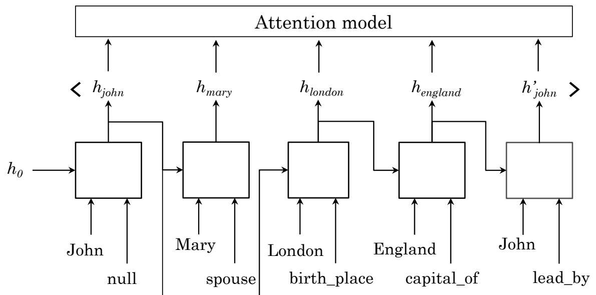


Figure 4: GTR-LSTM triple encoder.

is already computed in the previous step).

Following the graph in Fig. 3, the order of hidden state computation is as follows. The process starts with a vertex with zero in-degree. Because there is no such vertex, a vertex is randomly selected as the starting vertex. Assume we pick "`John`" as the starting vertex, then we compute h_{john} using h_0 as the previous hidden state. Next, following the *breadth-first traversal* algorithm, we visit vertex "`John`" and compute h_{mary} and h_{london} by passing h_{john} as the previous hidden state. Next step, vertex "`Mary`" is visited, but no hidden states are computed or updated since it does not have any adjacent vertices. In the last step, vertex "`England`" is visited and h_{john} is updated. Fig. 4 illustrates the overall process.

Different from the Graph LSTM, our GTR-LSTM model computes a hidden state by taking into account the processed entity and its edge (the edge pointing to the current entity from the previous entity) to handle non-predefined relationships (any relationships between entities in a knowledge graph). Thus, our GTR-LSTM unit (cf. Fig. 4) receives two inputs, i.e., the entity and its relationship. We propose the following model to compute the hidden state of each GTR-LSTM unit.

$$i_t = \sigma\left(\sum_e \left(U^{ie}x_{te} + W^{ie}h_{t-1}\right)\right) \quad (2)$$

$$f_{te} = \sigma\left(U^{f}x_{te} + W^{f}h_{t-1}\right) \quad (3)$$

$$o_t = \sigma\left(\sum_e \left(U^{oe}x_{te} + W^{oe}h_{t-1}\right)\right) \quad (4)$$

$$g_t = \tanh\left(\sum_e \left(U^{ge}x_{te} + W^{ge}h_{t-1}\right)\right) \quad (5)$$

$$c_t = \left(c_{t-1} * \sum_e f_{te}\right) + (g_t * i_t) \quad (6)$$

$$h_t = \tanh(c_t) * o_t \quad (7)$$

Here, U and W are learned parameter matrices, σ denotes the sigmoid function, $*$ denotes element-

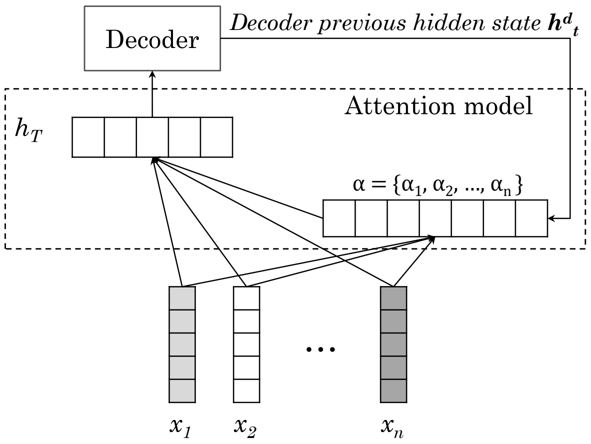

Decoder
Decoder previous hidden state h^d_t
Attention model
h_T
$\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$
...
x_1
x_2
x_n

Figure 5: Attention model of GTR-LSTM.

wise multiplication, and x is the input at the current time-step. The input gate i determines the weight of the current input. The forget gate f determines the weight of the previous state. The output gate o determines the weight of the cell state forwarded to the next time-step. The state g is the candidate hidden state used to compute the internal memory unit c based on the current input and the previous state. The subscript t is the time-step. The subscript/superscript e is the input element (an entity or a predicate). Following Tree LSTM (Tai et al., 2015) and Graph LSTM (Liang et al., 2016), we also use a separate forget gate for each input that allows the GTR-LSTM unit to incorporate information from each input selectively.

From Fig. 4, we can see that the traversal creates two branches, one ended in h_{mary} and the other ended in h'_{john}. After the encoder computes the hidden states of each vertex, h'_{john} does not include the information of h_{mary} and vice versa. Moreover, the graph can contain cycles that cause difficulty in determining the starting and ending vertices. Our traversal procedure ensures that the hidden states of all vertices are updated based on their adjacent vertices (local neighbors). To further capture the global information of the graph, we apply an attention model on the GTR-LSTM triple encoder. The attention model takes the hidden states of all vertices computed by the encoder and the previous hidden state of the decoder to compute the final input vector of each decoder time-step. Figure 5 illustrates the attention model of GTR-LSTM. Inspired by Luong et al. (2015), we adapt the following equation to compute the weight of each vertex.

$$\alpha_n = \frac{\exp({h^d_t}^T W x_n)}{\sum_{j=1}^{|X|} \exp({h^d_t}^T W x_j)} \tag{8}$$

Here, ${h^d}_t$ is the previous hidden state of the decoder, $|X|$ is the total number of entities in the triples, W is a learned parameter matrix, x_n and x_j are hidden states of vertices, and $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ is the weight vector of all vertices. Then the input of the decoder for each time-step can be computed as follows.

$$h_T = \sum_{n=1}^{|X|} \alpha_n x_n \tag{9}$$

3.5 Decoder

The decoder of the proposed framework is a standard LSTM. It is trained to generate the output sequence by predicting the next output word w_t conditioned on the hidden state ${h^d}_t$. The current hidden state ${h^d}_t$ is conditioned on the hidden state of the previous time-step ${h^d}_{t-1}$, the output of the previous time-step w_{t-1}, and input vector representation h_T. The hidden state and the output of the decoder at time-step t are computed as:

$$\begin{aligned} {h^d}_t &= f({h^d}_{t-1}, w_{t-1}, h_T) && (10) \\ w_t &= softmax(V h_t) && (11) \end{aligned}$$

Here, f is a single LSTM unit, and V is the hidden-to-output weight matrix. The encoder and the decoder are trained to maximize the conditional log-likelihood:

$$p(S_n \mid T_n) = \sum_{t=1}^{|S_n|} \log w_t \tag{12}$$

Hence, the training objective is to minimize the negative conditional log-likelihood:

$$J = \sum_{n=1}^{N} - \log p(S_n \mid T_n) \tag{13}$$

where (S_n, T_n) is a pair of output word sequence and input RDF triple set given for the training.

3.6 Entity Masking

Entity masking makes our framework generalizes better to unseen entities. This technique addresses the problem of a limited training set which is faced by many NLG problems.

Entity masking replaces entity mentions with *eids* and entity types in both the input triples and the target sentences. However, we do not want our model to be overly generalized either. Thus, we need to have general and specific entity types. For example, the entity "`John Doe`" is replaced by "`ENT-1 PERSON GOVERNOR`". To add the entity types, we use the DBpedia lookup API. The

API returns several entity types. The general and specific entity types are defined by the level of the word in the WordNet (Fellbaum, 1998) hierarchy.

In the encoder side, each element of the triple $t_n = \langle s_n, p_n, o_n \rangle$ is transformed into $s_n = \langle l_{s_n}, g_{s_n}, d_{s_n} \rangle$, $p_n = \langle l_{p_n} \rangle$, and $o_n = \langle l_{o_n}, g_{o_n}, d_{o_n} \rangle$, where l is the label of an element, g is the general entity type, and d is the specific entity type. The labels of the subject and the object are latter replaced by eids, while the label of the predicate is preserved, since it indicates the relationship between the subject and the object.

On the decoder side, the entities in the target text are also replaced by their corresponding eids. Entity matching is beyond the scope of our study. We simply use a combination of three string matching methods to find entity mentions in the sentence: exact matching, *n-gram* matching, and parse tree matching. The exact matching is used to find the exact mention; the *n-gram* matching is used to handle partial matching with the same token length; and parse tree matching is used to find a partial matching with different token length.

4 Experiments

We evaluate our framework on two datasets. The first is the dataset from Gardent et al. (2017a). We call it the *WebNLG* dataset. This dataset contains 25,298 RDF triple set-text pairs, with 9,674 unique sets of RDF triples. The dataset consists of a Train+Dev dataset and a Test Unseen dataset. We split Train+Dev into a training set (80%), a development set (10%), and a Seen testing set (10%). The Train+Dev dataset contains RDF triples in ten categories (topics, e.g., astronaut, monument, food, etc.), while the Test Unseen dataset has five other unseen categories. The maximum number of triples in each RDF triple set is seven. For the second dataset, we collected data from Wikipedia pages regarding landmarks. We call it the *GKB* dataset. We first extract RDF triples from Wikipedia infoboxes and sentences from the Wikipedia text that contain entities mentioned in the RDF triples. Human annotators then filter out false matches to obtain 1,000 RDF triple set-text pairs. This dataset is split into the training and development set (80%) and the testing set (20%). Table 1 illustrates an example of the data pairs of WebNLG and GKB dataset.

We implement the existing models, the adapted model, and the proposed model using Keras[3]. We use three common evaluation metrics including BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), and TER (Snover et al., 2006). For the metric computation and significance testing, we use MultEval (Clark et al., 2011).

4.1 Tested Models

We compare our proposed graph-based triple encoder (GTR-LSTM, Section 3.4) with three existing model including the adapted standard BLSTM encoder (BLSTM, Section 3.2), Neural Wikipedian (Vougiouklis et al., 2017) (TFF), and statistical machine translation (Hoang and Koehn, 2008) (SMT) trained on a *6-gram* language model. We also compare with the adapted standard triple encoder (TLSTM, Section 3.3).

4.2 Hyperparameters

We use grid search to find the best hyperparameters for the neural networks. We use GloVe (Pennington et al., 2014) trained on the GKB and WebNLG training data and full English Wikipedia data dump to get 300-dimension word embeddings. We use 512 hidden units for both encoder and decoder. We use a 0.5 dropout rate for regularization on both encoder and decoder to avoid overfitting. We train our model on NVIDIA Tesla K40c. We find that using adaptive learning rates for the optimization is efficient and leads the model to converge faster. Thus, we use Adam (Kingma and Ba, 2015) with a learning rate of 0.0002 instead of stochastic gradient descent. The update of parameters in training is computed using a mini batch of 64 instances. We further apply early stopping to detect the convergence.

4.3 Effect of Entity Masking

Table 2 shows the overall comparison of model performance. It shows that entity masking gives a consistent performance improvement for all models. Generalizing the input triples and target sentences helps the models to learn the relationships between entities from their types. This is particularly helpful when there is limited training data. We use a combination of exact matching, *n-gram* matching and parse tree matching to find the entity mentions in the sentence. The entity masking accuracy for WebNLG dataset is 87.15%, while for

[3]https://nmt-keras.readthedocs.io/en/latest/

Model		Metric/Dataset	BLEU↑			METEOR↑			TER↓		
			Seen	Unseen	GKB	Seen	Unseen	GKB	Seen	Unseen	GKB
Entity Unmasking	Existing models	BLSTM	42.7	23.0	28.0	34.4	28.7	27.5	55.7	69.9	67.7
		SMT	41.1	23.9	27.7	33.2	28.3	27.6	57.0	70.1	63.8
		TFF	44.6	26.4	26.4	33.9	29.4	27.2	52.4	62.6	60.1
	Adapted model	TLSTM	45.9	28.1	29.4	34.9	30.1	28.5	50.5	62.7	59.0
	Our proposed	GTR-LSTM	54.0	29.2	37.1	37.3	27.8	30.6	45.3	59.8	55.1
Entity Masking	Existing models	BLSTM	49.8	28.0	34.8	38.3	29.4	28.6	49.9	64.9	65.8
		SMT	46.5	24.8	32.0	37.1	29.1	28.5	52.3	62.2	67.8
		TFF	47.8	28.4	33.7	35.9	30.5	28.9	49.9	61.2	58.4
	Adapted Model	TLSTM	50.5	31.6	36.7	36.5	30.7	30.1	47.7	60.4	57.2
	Our proposed	GTR-LSTM	**58.6**	**34.1**	**40.1**	**40.6**	**32.0**	**34.6**	**41.7**	**57.9**	**50.6**

Table 2: Comparison of model performance.

RDF inputs	⟨`Elizabeth Tower, location, London`⟩, ⟨`Wembley Stadium, location, London`⟩, ⟨`London, capital of, England`⟩, ⟨`Theresa May, prime minister, England`⟩
Reference	`london , england is home to wembley stadium and the elizabeth tower. the name of the leader in england is theresa may.`
BLSTM	`england is lead by theresa may and` **`is located in the city of london .`** `the elizabeth tower is located` **`in the city of england`** `and is` **`located in the wembley stadium.`**
SMT	`wembley stadium is located in` **`london , elizabeth tower .`** `theresa may is the leader of` **`england , england.`**
TFF	`the elizabeth tower is located in london , england , where` **`wembley stadium is the leader`** `and theresa may is the leader.`
TLSTM	`the wembley stadium is located in london , england . the country is the location of elizabeth tower .` **`theresa may is the leader of london.`**
GTR-LSTM	`the wembley stadium and elizabeth tower are both located in london , england . theresa may is the leader of england.`

Table 3: Sample output of the system. The error is highlighted in bold.

the GKB dataset is 82.45%.

Entity masking improves the BLEU score of the proposed GTR-LSTM model by 8.5% (from 54.0 on the Entity Unmasking model to 58.6 on the Entity Masking model), 16.7%, and 8.0% on the WebNLG seen testing data (denoted by "**Seen**"), WebNLG unseen testing data (denoted by "**Unseen**"), and the GKB testing data (denoted by "**GKB**"). Using the entity masking not only improves the performance by recognizing the unknown vocabulary via eid masking but also improves the running time performance by requiring a smaller training vocabulary.

4.4 Effect of Models

Table 2 also shows that the proposed GTR-LSTM triple encoder achieves a consistent improvement over the baseline models, and the improvement is statistically significant, with $p < 0.01$ based on the t-test of all metrics. We use MultEval to compute the p value based on an approximate randomization (Clark et al., 2011). The improvement on the BLEU score indicates that the model reduces the errors in the generated sentence. Our manual inspection confirms this result. The better (lower) TER score suggests that the model generates a more compact output (i.e., better aggregation).

Table 3 shows a sample output of all models. From this table, we can see that all baseline models produce sentences that contain wrong relationships between entities (e.g., the BLSTM output contains a wrong relationship "`the elizabeth tower is located in the city of england`"). Moreover, the baseline models generate sentences with a weak aggregation (e.g., "`Elizabeth Tower`" and "`Wembley Stadium`" are in separate sentences for TLSTM). The proposed GTR-LSTM model successfully avoids these problems.

Model training time. GTR-LSM is slower in training than the baseline models, which is expected as it needs to encode more information. However, its training time is no more than twice as that of any baseline models tested, and the training can complete within one day which seems reasonable. Meanwhile, the number of parameters trained for GTR-LSTM is up to 59% smaller than those of the baseline models, which saves the space cost for model storage.

4.5 Human Evaluation

To complement the automatic evaluation, we conduct human evaluations for all of the masked models. We ask five human annotators. Each of them

Model \ Dataset/Metric		Seen			Unseen			GKB		
		Correctness	Grammar	Fluency	Correctness	Grammar	Fluency	Correctness	Grammar	Fluency
Existing Models	BLSTM	2.25	2.33	2.29	1.53	1.71	1.68	1.54	1.84	1.84
	SMT	2.03	2.11	2.07	1.36	1.48	1.44	1.81	1.99	1.89
	TFF	1.77	1.91	1.88	1.44	1.69	1.66	1.71	1.99	1.96
Adapted Model	TLSTM	2.53	2.61	2.55	1.75	1.93	1.86	2.21	2.38	2.35
Our Proposed	GTR-LSTM	**2.64**	**2.66**	**2.57**	**1.96**	**2.04**	**1.99**	**2.29**	**2.42**	**2.41**

Table 4: Human evaluation results.

has studied English for at least ten years and completed education in a full English environment for at least two years. We provide a website[4] that shows them the RDF triples and the generated text. The annotators are given training on the scoring criteria. We also provide scoring examples. We randomly selected 100 sets of triples along with the output of each model. We only select sets of triples that contain more than two triples. Following (Gardent et al., 2017b), we use three evaluation metrics including *correctness*, *grammaticality*, and *fluency*. For each pair of triple set and generated sentences, the annotators are asked to give a score between one to three for each metric.

Correctness is used to measure the semantics of the output sentence. A score of 3 is given to generated sentences that contain no errors in the relationships between entities; a score of 2 is given to generated sentences that contain one error in the relationship; and a score of 1 is given to generated sentences that contain more than one errors in the relationships. *Grammaticality* is used to rate the grammatical and spelling errors of the generated sentences. Similar to the *correctness* metric, a score of 3 is given to generated sentences with no grammatical and spelling errors; a score of 2 is given to generated sentences with one error; and a score of 1 for the others. The last metric, *fluency*, is used to measure the fluency of the sentence output. We ask the annotators to give a score based on the aggregation of the sentences and the existence of sentence repetition. Table 4 shows the results of the human evaluations. The results confirm the automatic evaluation in which our proposed model achieves the best scores.

Error analysis. We further perform a manual inspection of 100 randomly selected output sentences of GTR-LSTM and BLSTM on the Seen and Unseen test data. We find that 32% of BLSTM output contains wrong relationships between entities. In comparison, only 8% of GTR-LSTM output contains such errors. Besides, we find duplicate sub-sentences in the output of GTR-LSTM (15%). The following output is an example: "`beef kway teow is a dish from singapore, where english language is spoken and the leader is tony tan. the leader of singapore is tony tan.`" While the duplicate sentence is not wrong, it affects the reading experience. We conjecture that the LSTM in the decoder caused such an issue. We aim to solve this problem in future work.

[4]http://bit.ly/gkb-mappings

5 Conclusions

We proposed a novel graph-based triple encoder GTR-LSTM for sentence generation from RDF data. The proposed model maintains the structure of input RDF triples as a small knowledge graph to optimize the amount of information preserved in the input of the model. The proposed model can handle cycles to capture the global information of a knowledge graph and also handle non-predefined relationships between entities of a knowledge graph.

Our experiments show that GTR-LSTM offers a better performance than all the competitors. On the WebNLG dataset, our model outperforms the best existing model, the standard BLSTM model, by up to 17.6%, 6.0%, and 16.4% in terms of BLEU, METEOR, and TER scores, respectively. On the GKB dataset, our model outperforms the standard BLSTM model by up to 15.2%, 20.9%, and 23.1% in these three metrics, respectively.

Acknowledgments

Bayu Distiawan Trisedya is supported by the Indonesian Endowment Fund for Education (LPDP). This work is supported by Australian Research Council (ARC) Discovery Project DP180102050 and Future Fellowships Project FT120100832, and Google Faculty Research Award. This work is partly done while Jianzhong Qi is visiting the University of New South Wales. Wei Wang was partially supported by D2DCRC DC25002, DC25003, ARC DP 170103710 and 180103411.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*. http://arxiv.org/abs/1409.0473.

Kalina Bontcheva and Yorick Wilks. 2004. *Automatic Report Generation from Ontologies: The MIAKT Approach*, Springer, Berlin, Heidelberg, pages 324–335. https://doi.org/10.1007/978-3-540-27779-8_28.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 615–620. https://www.aclweb.org/anthology/D/D14/D14-1067.pdf.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. pages 633–642. http://aclweb.org/anthology/E17-1060.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Philipp Cimiano, Janna Lüker, David Nagel, and Christina Unger. 2013. Exploiting ontology lexica for generating natural language texts from rdf data. In *Proceedings of the 14th European Workshop on Natural Language Generation (ENLG)*. pages 10–19. http://www.aclweb.org/anthology/W13-2102.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*. pages 176–181. http://www.aclweb.org/anthology/P11-2031.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Natural language interfaces to ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on The Semantic Web (ISWC)*. pages 106–120. https://doi.org/10.1007/978-3-642-13486-9_8.

Michael J. Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT)*. pages 85–91. http://aclweb.org/anthology/W11-2107.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS)*. pages 83–94. http://www.aclweb.org/anthology/W13-0108.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. pages 1156–1165. https://doi.org/10.1145/2623330.2623677.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press. http://aclweb.org/anthology/J99-2008.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 179–188. http://aclweb.org/anthology/P17-1017.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation (INLG)*. pages 124–133. http://www.aclweb.org/anthology/W17-3518.

Hieu Hoang and Philipp Koehn. 2008. Design of the moses decoder for statistical machine translation. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP)*. pages 58–65. http://www.aclweb.org/anthology/W08-0510.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1203–1213. https://aclweb.org/anthology/D16-1128.

Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. 2016. Semantic object parsing with graph lstm. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*. pages 125–143. https://doi.org/10.1007/978-3-319-46448-0_8.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. pages 720–730. http://www.aclweb.org/anthology/N16-1086.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 311–318. http://aclweb.org/anthology/P02-1040.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. https://aclweb.org/anthology/D14-1162.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas (AMTA)*. pages 223–231. http://mt-archive.info/AMTA-2006-Snover.pdf.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing (IJCNLP)*. pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Christina Unger, Lorenz Bhmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web (WWW)*. pages 639–648. https://doi.org/10.1145/2187836.2187923.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aime Kaffee, Christoph Gravier, Frederique Laforest, Jonathon Hare, and Elena Simperl. 2017. Neural wikipedian: Generating textual summaries from knowledge base triples. *arXiv preprint arXiv:1711.00155* https://arxiv.org/pdf/1711.00155.pdf.

Learning to Write with Cooperative Discriminators

Ari Holtzman[†] **Jan Buys**[†] **Maxwell Forbes**[†]
Antoine Bosselut[†] **David Golub**[†] **Yejin Choi**[†‡]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[‡]Allen Institute for Artificial Intelligence
{ahai,jbuys,mbforbes,antoineb,golubd,yejin}@cs.washington.edu

Abstract

Despite their local fluency, long-form text generated from RNNs is often generic, repetitive, and even self-contradictory. We propose a unified learning framework that collectively addresses all the above issues by composing a committee of discriminators that can guide a base RNN generator towards more globally coherent generations. More concretely, discriminators each specialize in a different principle of communication, such as Grice's maxims, and are collectively combined with the base RNN generator through a composite decoding objective. Human evaluation demonstrates that text generated by our model is preferred over that of baselines by a large margin, significantly enhancing the overall coherence, style, and information of the generations.

1 Introduction

Language models based on Recurrent Neural Networks (RNNs) have brought substantial advancements across a wide range of language tasks (Jozefowicz et al., 2016; Bahdanau et al., 2015; Chopra et al., 2016). However, when used for long-form text generation, RNNs often lead to degenerate text that is repetitive, self-contradictory, and overly generic, as shown in Figure 1.

We propose a unified learning framework that can address several challenges of long-form text generation by composing a committee of discriminators each specializing in a different principle of communication. Starting with an RNN language model, our framework learns to construct a more powerful generator by training a number of discriminative models that can collectively address limitations of the base RNN generator, and

Context: The two guards thudded into the door on the other side and began pounding on it, shouting furiously. Scious raised the large bunch of keys then placed them in a large pocket in his coat. "Come, we have to go," he whispered and moved up the corridor, the guards still hammering on the door. Jared looked around. The table next to the room they had been in was bare.

LM: There was a small table set into the center of the table, and a table with two chairs and a table. A chair sat next to the couch. A large man stood in the doorway of the room, his arms crossed over his chest. "What 's up?" He asked.

L2W: Empty. It looked like some kind of *prison*. There were no windows, no doors, no windows. *No sign of anyone in the house.* No one had ever been here before.

Reference: Only a flagon of wine and a couple of mugs plus a crude dice game. Light flickered in the stone corridor from lanterns on the walls. The place was dank and gloomy, moss in scattered patches on the floor and walls. The corridor ended just beyond the door to their former *prison*. *No one else was about.*

Figure 1: Sample generations from an RNN language model (LM) and our system (L2W) conditioning on the context shown on the top. The red, underlined text highlights repetitions, while the *blue, italicized* text highlights details that have a direct semantic parallel in the reference text.

then learns how to weigh these discriminators to form the final decoding objective. These "cooperative" discriminators complement each other and the base language model to form a stronger, more global decoding objective.

The design of our discriminators are inspired by Grice's maxims (Grice et al., 1975) of quantity, quality, relation, and manner. The discriminators learn to encode these qualities through the selection of training data (e.g. distinguishing a true continuation from a randomly sampled one as in §3.2 Relevance Model), which includes generations from partial models (e.g. distinguishing a true continuation from one generated by a language model as in §3.2 Style Model). The system then learns to balance these discriminators by initially weighing them uniformly, then continually updating its weights by comparing the scores the system gives to its own generated continuations and to the reference continuation.

Empirical results (§5) demonstrate that our learning framework is highly effective in converting a generic RNN language model into a substantially stronger generator. Human evaluation confirms that language generated by our model is preferred over that of competitive baselines by a large margin in two distinct domains, and significantly enhances the overall coherence, style, and information content of the generated text. Automatic evaluation shows that our system is both less repetitive and more diverse than baselines.

2 Background

RNN language models learn the conditional probability $P(x_t|x_1, ..., x_{t-1})$ of generating the next word x_t given all previous words. This conditional probability learned by RNNs often assigns higher probability to repetitive, overly generic sentences, as shown in Figure 1 and also in Table 3. Even gated RNNs such as LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014) have difficulties in properly incorporating long-term context due to explaining-away effects (Yu et al., 2017b), diminishing gradients (Pascanu et al., 2013), and lack of inductive bias for the network to learn discourse structure or global coherence beyond local patterns.

Several methods in the literature attempt to address these issues. Overly simple and generic generation can be improved by length-normalizing the sentence probability (Wu et al., 2016), future cost estimation (Schmaltz et al., 2016), or a diversity-boosting objective function (Shao et al., 2017; Vijayakumar et al., 2016). Repetition can be reduced by prohibiting recurrence of the trigrams as a hard rule (Paulus et al., 2018). However, such hard constraints do not stop RNNs from repeating through paraphrasing while preventing occasional intentional repetition.

We propose a unified framework to address all these related challenges of long-form text generation by learning to construct a better decoding objective, generalizing over various existing modifications to the decoding objective.

3 The Learning Framework

We propose a general learning framework for conditional language generation of a sequence $\mathbf{y}$ given a fixed context $\mathbf{x}$. The decoding objective for generation takes the general form

$$f_\lambda(\mathbf{x}, \mathbf{y}) = \log(P_{\text{lm}}(\mathbf{y}|\mathbf{x})) + \sum_k \lambda_k s_k(\mathbf{x}, \mathbf{y}), \quad (1)$$

where every s_k is a scoring function. The proposed objective combines the RNN language model probability P_{lm} (§3.1) with a set of additional scores $s_k(\mathbf{x}, \mathbf{y})$ produced by discriminatively trained communication models (§3.2), which are weighted with learned mixture coefficients λ_k (§3.3). When the scores s_k are log probabilities, this corresponds to a Product of Experts (PoE) model (Hinton, 2002).

Generation is performed using beam search (§3.4), scoring incomplete candidate generations $\mathbf{y}_{1:i}$ at each time step i. The RNN language model decomposes into per-word probabilities via the chain rule. However, in order to allow for more expressivity over long range context we do not require the discriminative model scores to factorize over the elements of $\mathbf{y}$, addressing a key limitation of RNNs. More specifically, we use an estimated score $s'_k(\mathbf{x}, \mathbf{y}_{1:i})$ that can be computed for any prefix of $\mathbf{y} = \mathbf{y}_{1:n}$ to approximate the objective during beam search, such that $s'_k(\mathbf{x}, \mathbf{y}_{1:n}) = s_k(\mathbf{x}, \mathbf{y})$. To ensure that the training method matches this approximation as closely as possible, scorers are trained to discriminate prefixes of the same length (chosen from a predetermined set of prefix lengths), rather than complete continuations, except for the entailment module as described in §3.2 Entailment Model. The prefix scores are re-estimated at each time-step, rather than accumulated over beam search.

3.1 Base Language Model

The RNN language model treats the context $\mathbf{x}$ and the continuation $\mathbf{y}$ as a single sequence $\mathbf{s}$:

$$\log P_{\text{lm}}(\mathbf{s}) = \sum_i \log P_{\text{lm}}(\mathbf{s}_i|\mathbf{s}_{1:i-1}). \quad (2)$$

3.2 Cooperative Communication Models

We introduce a set of discriminators, each of which encodes an aspect of proper writing that RNNs usually fail to capture. Each model is trained to discriminate between good and bad generations; we vary the model parameterization and training examples to guide each model to focus on a different aspect of Grice's Maxims. The discriminator scores are interpreted as classification probabilities (scaled with the logistic function where necessary) and interpolated in the objective function as log probabilities.

Let $D = \{(\mathbf{x_1}, \mathbf{y_1}), \ldots (\mathbf{x_n}, \mathbf{y_n})\}$ be the set of training examples for conditional generation. $D_\mathbf{x}$ denote all contexts and $D_\mathbf{y}$ all continuations. The scoring functions are trained on prefixes of $\mathbf{y}$ to simulate their application to partial continuations at inference time.

In all models the first layer embeds each word w into a 300-dimensional vector $e(w)$ initialized with GloVe (Pennington et al., 2014) pretrained-embeddings.

Repetition Model

This model addresses the maxim of Quantity by biasing the generator to avoid repetitions. The goal of the repetition discriminator is to learn to distinguish between RNN-generated and gold continuations by exploiting our empirical observation that repetitions are more common in completions generated by RNN language models. However, we do not want to completely eliminate repetition, as words do recur in English.

In order to model natural levels of repetition, a score d_i is computed for each position in the continuation $\mathbf{y}$ based on pairwise cosine similarity between word embeddings within a fixed window of the previous k words, where

$$d_i = \max_{j=i-k \ldots i-1} (\text{CosSim}(e(y_j), e(y_i))), \quad (3)$$

such that $d_i = 1$ if y_i is repeated in the window.

The score of the continuation is then defined as

$$s_{\text{rep}}(\mathbf{y}) = \sigma(\mathbf{w}_r^\top \text{RNN}_{\text{rep}}(\mathbf{d})), \quad (4)$$

where $\text{RNN}_{\text{rep}}(\mathbf{d})$ is the final state of a unidirectional RNN ran over the similarity scores $\mathbf{d} = d_1 \ldots d_n$ and $\mathbf{w}_r$ is a learned vector. The model is trained to maximize the ranking log likelihood

$$L_{\text{rep}} = \sum_{\substack{(\mathbf{x}, \mathbf{y}_g) \in D, \\ \mathbf{y}_s \sim \text{LM}(\mathbf{x})}} \log \sigma(s_{\text{rep}}(\mathbf{y}_g) - s_{\text{rep}}(\mathbf{y}_s)), \quad (5)$$

which corresponds to the probability of the gold ending $\mathbf{y}_g$ receiving a higher score than the ending sampled from the RNN language model.

Entailment Model

Judging textual quality can be related to the natural language inference (NLI) task of recognizing textual entailment (Dagan et al., 2006; Bowman et al., 2015): we would like to guide the generator to neither contradict its own past generation (the maxim of Quality) nor state something that readily follows from the context (the maxim of Quantity). The latter case is driven by the RNNs habit of paraphrasing itself during generation.

We train a classifier that takes two sentences a and b as input and predicts the relation between them as either *contradiction*, *entailment* or *neutral*. We use the *neutral* class probability of the sentence pair as discriminator score, in order to discourage both contradiction and entailment. As entailment classifier we use the decomposable attention model (Parikh et al., 2016), a competitive, parameter-efficient model for entailment classification.[1] The classifier is trained on two large entailment datasets, SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017), which together have more than 940,000 training examples. We train separate models based on the vocabularies of each of the datasets we use for evaluation.

In contrast to our other communication models, this classifier cannot be applied directly to the full context and continuation sequences it is scoring. Instead every completed sentence in the continuation should be scored against all preceding sentences in both the context and continuation.

Let $t(\mathbf{a}, \mathbf{b})$ be the log probability of the neutral class. Let $S(\mathbf{y})$ be the set of complete sentences in $\mathbf{y}$, $S_{\text{last}}(\mathbf{y})$ the last complete sentence, and $S_{\text{init}}(\mathbf{y})$ the sentences before the last complete sentence. We compute the entailment score of $S_{\text{last}}(\mathbf{y})$ against all preceding sentences in $\mathbf{x}$ and $\mathbf{y}$, and use the score of the sentence-pair for which we have the least confidence in a *neutral* classification:

$$s_{\text{entail}}(\mathbf{x}, \mathbf{y}) = min_{\mathbf{a} \in S(\mathbf{x}) \cup S_{\text{init}}(\mathbf{y})} t(\mathbf{a}, S_{\text{last}}(\mathbf{y})). \quad (6)$$

Intuitively, we only use complete sentences because the ending of a sentence can easily flip entailment. As a result, we carry over entailment score of the last complete sentence in a generation until the end of the next sentence, in order to maintain the presence of the entailment score in the objective. Note that we check that the current

[1]We use the version without intra-sentence attention.

```
Data: context $\mathbf{x}$, beam size $k$, sampling temperature $t$
Result: best continuation
best = None
beam = [$\mathbf{x}$]
for step = 0; step $<$ max_steps; step = step $+1$ do
    next_beam = []
    for candidate in beam do
        next_beam.extend(next_k(candidate))
        if termination_score(candidate) $>$ best.score then
            best = candidate.append(term)
        end
    end
    for candidate in next_beam do
                                        ▷ score with models
        candidate.score += $f_\lambda$(candidate)
    end
                                        ▷ sample k candidates by score
    beam = sample(next_beam, $k$, $t$)
end
if learning then
    update $\lambda$ with gradient descent by comparing best
      against the gold.
end
return best
```

Algorithm 1: Inference/Learning in the Learning to Write Framework.

sentence is not directly entailed or contradicted by a previous sentence and not the reverse. [2] In contrast to our other models, the score this model returns only corresponds to a subsequence of the given continuation, as the score is not accumulated across sentences during beam search. Instead the decoder is guided locally to continue complete sentences that are not entailed or contradicted by the previous text.

Relevance Model

The relevance model encodes the maxim of Relation by predicting whether the content of a candidate continuation is relevant to the given context. We train the model to distinguish between true continuations and random continuations sampled from other (human-written) endings in the corpus, conditioned on the given context.

First both the context and continuation sequences are passed through a convolutional layer, followed by maxpooling to obtain vector representations of the sequences:

$$a = \text{maxpool}(\text{conv}_a(e(\mathbf{x}))), \tag{7}$$

$$b = \text{maxpool}(\text{conv}_b(e(\mathbf{y}))). \tag{8}$$

[2]If the current sentence entails a previous one it may simply be adding more specific information, for instance: "He hated broccoli. Every time he ate broccoli he was reminded that it was the thing he hated most."

The goal of maxpooling is to obtain a vector representing the most important semantic information in each dimension.

The scoring function is then defined as

$$s_{\text{rel}} = \mathbf{w}_l^T \cdot (a \circ b), \tag{9}$$

where element-wise multiplication of the context and continuation vectors will amplify similarities.

We optimize the ranking log likelihood

$$L_{\text{rel}} = \sum_{\substack{(\mathbf{x},\mathbf{y}_g)\in D,\\ \mathbf{y}_r\sim D_{\mathbf{y}}}} \log \sigma(s_{\text{rel}}(\mathbf{x},\mathbf{y}_g) - s_{\text{rel}}(\mathbf{x},\mathbf{y}_r)), \tag{10}$$

where $\mathbf{y}_g$ is the gold ending and $\mathbf{y}_r$ is a randomly sampled ending.

Lexical Style Model

In practice RNNs generate text that exhibit much less lexical diversity than their training data. To counter this effect we introduce a simple discriminator based on observed lexical distributions which captures writing style as expressed through word choice. This classifier therefore encodes aspects of the maxim of Manner.

The scoring function is defined as

$$s_{\text{bow}}(\mathbf{y}) = \mathbf{w}_s^T \text{maxpool}(e(\mathbf{y})). \tag{11}$$

The model is trained with a ranking loss using negative examples sampled from the language model, similar to Equation 5.

3.3 Mixture Weight Learning

Once all the communication models have been trained, we learn the combined decoding objective. In particular we learn the weight coefficients λ_k in equation 1 to linearly combine the scoring functions, using a discriminative loss

$$L_{\text{mix}} = \sum_{(\mathbf{x},\mathbf{y})\in D} (f_\lambda(\mathbf{x},\mathbf{y}) - f_\lambda(\mathbf{x},\mathcal{A}(\mathbf{x}))^2, \tag{12}$$

where $\mathcal{A}$ is the inference algorithm for beam search decoding. The weight coefficients are thus optimized to minimize the difference between the scores assigned to the gold continuation and the continuation predicted by the current model.

Mixture weights are learned online: Each successive generation is performed based on the current values of λ, and a step of gradient descent is then performed based on the prediction. This has the effect that the objective function changes

	BookCorpus					TripAdvisor				
Model	BLEU	Meteor	Length	Vocab	Trigrams	BLEU	Meteor	Length	Vocab %	Trigrams
L2W	**0.52**	**6.8**	**43.6**	**73.8**	98.9	1.7	11.0	83.8	**64.1**	**96.2**
ADAPTIVELM	**0.52**	6.3	43.5	59.0	92.7	**1.94**	**11.2**	**94.1**	52.6	92.5
CACHELM	0.33	4.6	37.9	31.0	44.9	1.36	7.2	52.1	39.2	57.0
SEQ2SEQ	0.32	4.0	36.7	23.0	33.7	1.84	8.0	59.2	33.9	57.0
SEQGAN	0.18	5.0	28.4	73.4	**99.3**	0.73	6.7	47.0	57.6	93.4
REFERENCE	100.0	100.0	65.9	73.3	99.7	100.0	100.0	92.8	69.4	99.4

Table 1: Results for automatic evaluation metrics for all systems and domains, using the original continuation as the reference. The metrics are: Length - Average total length per example; Trigrams - % unique trigrams per example; Vocab - % unique words per example.

dynamically during training: As the current samples from the model are used to update the mixture weights, it creates its own learning signal by applying the generative model discriminatively. The SGD learning rate is tuned separately for each dataset.

3.4 Beam Search

Due to the limitations of greedy decoding and the fact that our scoring functions do not decompose across time steps, we perform generation with a beam search procedure, shown in Algorithm 1. The naive approach would be to perform beam search based only on the language model, and then rescore the k best candidate completions with our full model. We found that this approach leads to limited diversity in the beam and therefore cannot exploit the strengths of the full model.

Instead we score the current hypotheses in the beam with the full decoding objective: First, each hypothesis is expanded by selecting the k highest scoring next words according to the language model (we use beam size $k = 10$). Then k sequences are sampled from the k^2 candidates according to the (softmax normalized) distribution over the candidate scores given by the full decoding objective. Sampling is performed in order to increase diversity, using a temperature of 1.8, which was tuned by comparing the coherence of continuations on the validation set.

At each step, the discriminator scores are recomputed for all candidates, with the exception of the entailment score, which is only recomputed for hypotheses which end with a sentence terminating symbol. We terminate beam search when the *termination_score*, the maximum possible score achievable by terminating generation at the current position, is smaller than the current best score.

4 Experiments

4.1 Corpora

We use two English corpora for evaluation. The first is the TripAdvisor corpus (Wang et al., 2010), a collection of hotel reviews with a total of 330 million words.[3] The second is the BookCorpus (Zhu et al., 2015), a 980 million word collection of novels by unpublished authors.[4] In order to train the discriminators, mixing weights, and the SEQ2SEQ and SEQGAN baselines, we segment both corpora into sections of length ten sentences, and use the first 5 sentence as context and the second 5 as the continuation. See supplementary material for further details.

4.2 Baselines

ADAPTIVELM Our first baseline is the same Adaptive Softmax (Grave et al., 2016) language model used as base generator in our framework (§3.1). This enables us to evaluate the effect of our enhanced decoding objective directly. A 100k vocabulary is used and beam search with beam size of 5 is used at decoding time. ADAPTIVELM achieves perplexity of 37.46 and 18.81 on BookCorpus and TripAdvisor respectively.

CACHELM As another LM baseline we include a continuous cache language model (Grave et al., 2017) as implemented by Merity et al. (2018), which recently obtained state-of-the-art perplexity on the Penn Treebank corpus (Marcus et al., 1993). Due to memory constraints, we use a vocabulary size of 50k for CACHELM. To generate, beam search decoding is used with a beam size 5. CACHELM obtains perplexities of 70.9 and 29.71 on BookCorpus and TripAdvisor respectively.

[3] `http://times.cs.uiuc.edu/~wang296/Data/`

[4] `http://yknzhu.wixsite.com/mbweb`

BookCorpus	**Specific Criteria**				**Overall Quality**		
L2W vs.	Repetition	Contradiction	Relevance	Clarity	Better	Equal	Worse
ADAPTIVELM	+0.48	+0.18	+0.12	+0.11	**47%**	20%	32%
CACHELM	+1.61	+0.37	+1.23	+1.21	**86%**	6%	8%
SEQ2SEQ	+1.01	+0.54	+0.83	+0.83	**72%**	7%	21%
SEQGAN	+0.20	+0.32	+0.61	+0.62	**63%**	20%	17%
LM VS. REFERENCE	-0.10	-0.07	-0.18	-0.10	41%	7 %	**52%**
L2W VS. REFERENCE	+0.49	+0.37	+0.46	+0.55	**53%**	18%	29%

TripAdvisor	**Specific Criteria**				**Overall Quality**		
L2W vs.	Repetition	Contradiction	Relevance	Clarity	Better	Equal	Worse
ADAPTIVELM	+0.23	-0.02	+0.19	-0.03	**47%**	19%	34%
CACHELM	+1.25	+0.12	+0.94	+0.69	**77%**	9%	14%
SEQ2SEQ	+0.64	+0.04	+0.50	+0.41	**58%**	12%	30%
SEQGAN	+0.53	+0.01	+0.49	+0.06	**55%**	22%	22%
LM VS. REFERENCE	-0.10	-0.04	-0.15	-0.06	38%	10%	**52%**
L2W VS. REFERENCE	-0.49	-0.36	-0.47	-0.50	25%	18%	**57%**

Table 2: Results of crowd-sourced evaluation on different aspects of the generation quality as well as overall quality judgments. For each sub-criteria we report the average of comparative scores on a scale from -2 to 2. For the overall quality evaluation decisions are aggregated over 3 annotators per example.

SEQ2SEQ As our evaluation can be framed as sequence-to-sequence transduction, we compare against a seq2seq model directly trained to predict 5 sentence continuations from 5 sentences of context, using the OpenNMT attention-based seq2seq implementation (Klein et al., 2017). Similarly to CACHELM, a 50k vocabulary was used and beam search decoding was performed with a beam size of 5.

SEQGAN Finally, as our use of discriminators is related to Generative Adversarial Networks (GANs), we use SeqGAN (Yu et al., 2017a), a GAN for discrete sequences trained with policy gradients.[5] This model is trained on 10 sentence sequences, which is significantly longer than previous experiments with GANs for text; the vocabulary is restricted to 25k words to make training tractable. Greedy sampling was found to outperform beam search. For implementation details, see the supplementary material.

4.3 Evaluation Setup

We pose the evaluation of our model as the task of generating an appropriate continuation given an initial context. In our open-ended generation setting the continuation is not required to be a specific length, so we require our models and baselines to generate 5-sentence continuations, consistent with the way the discriminator and seq2seq baseline datasets are constructed.

Previous work has reported that automatic measures such as BLEU (Papineni et al., 2002) and Meteor (Denkowski and Lavie, 2010) do not lead to meaningful evaluation when used for long or creative text generation where there can be high variance among acceptable generation outputs (Wiseman et al., 2017; Vedantam et al., 2015). However, we still report these measures as one component of our evaluation. Additionally we report a number of custom metrics which capture important properties of the generated text: *Length* – Average sequence length per example; *Trigrams* – percentage of unique trigrams per example; *Vocab* – percentage of unique words per example. Endings generated by our model and the baselines are compared against the reference endings in the original text. Results are given in Table 1.

For open-ended generation tasks such as our own, human evaluation has been found to be the only reliable measure (Li et al., 2016b; Wiseman et al., 2017). For human evaluation, two possible endings are presented to a human, who assesses the text according to several criteria, which are closely inspired by Grice's Maxims: repetition, contradiction, relevance and clarity. See supplementary material for examples of the evaluation forms we used. For each criterion, the two continuations are compared using a 5-point Likert scale, to which we assign numerical values of -2 to 2. The scale measures whether one generation is strongly or somewhat preferred above the other, or whether they are equal. Finally, the human is asked to make a judgement about overall quality: which ending is better, or are they of equal qual-

[5]We use the implementation available at `https://github.com/nhynes/abc`.

CONTEXT	We thoroughly enjoyed staying at this hotel. We travelled with our kids aged 4 and 6. The rooms were clean, spacious and well maintained. The beds were great and all of us slept well. The hotel provided vouchers towards breakfast at the adjoining restaurant.
L2W	We didn't have breakfast, so can't comment on that. The concierge was very helpful, and gave us directions, etc. Overall, we would definitely recommend this hotel to others. We would stay here again in a heartbeat. If you're looking for a great location, and good value, I'd recommend this hotel.
ADAPTIVELM	The breakfast buffet was very good, and the staff was very friendly and helpful, and it was a great way to start our day! I would definitely stay here again, and would recommend it to my friends, and would stay there again if I ever go back to Montreal. The location was great, and the staff was friendly and helpful. I would definitely stay here again!!
CACHELM	The staff was very friendly and helpful. The hotel is located in a great location. It is right in the middle of the gaslamp district. I would recommend this hotel to anyone. I would definitely stay here again.
SEQ2SEQ	The staff were very friendly and helpful. The hotel is in a great location. The staff were very friendly and helpful. The hotel is in a great location. The staff were very friendly and helpful.
SEQGAN	We had a breakfast at Shula's & a delicious breakfast. The staff was very helpful and helpful. The breakfast was great as well. The staff was very helpful and friendly. We had a great service and the food was excellent.
REFERENCE	The restaurant was great and we used the vouchers towards whatever breakfast we ordered. The hotel had amazing grounds with a putting golf course that was fun for everyone. The pool was fantastic and we lucked out with great weather. We spent many hours in the pool, lounging, playing shuffleboard and snacking from the attached bar. The happy hour was great perk.

Table 3: Example continuations generated by our model (L2W) and various baselines (all given the same context from TripAdvisor) compared to the reference continuation. For more examples go to https://ari-holtzman.github.io/l2w-demo/.

ity?

The human evaluation is performed on 100 examples selected from the test set of each corpus, for every pair of generators that are compared. We present the examples to workers on Amazon Mechanical Turk, using three annotators for each example. The results are given in Table 2. For the Likert scale, we report the average scores for each criterion, while for the overall quality judgement we simply aggregate votes across all examples.

5 Results and Analysis

5.1 Quantitative Results

The absolute performance of all the evaluated systems on BLEU and Meteor is quite low (Table 1), as expected. However, in relative terms L2W is superior or competitive with all the baselines, of which ADAPTIVELM performs best. In terms of vocabulary and trigram diversity only SEQGAN is competitive with L2W, likely due to the fact that sampling based decoding was used. For generation length only L2W and ADAPTIVELM even approach human levels, with the former better on BookCorpus and the latter on TripAdvisor.

Under the crowd-sourced evaluation (Table 2), on BookCorpus our model is consistently favored over the baselines on all dimensions of comparison. In particular, our model tends to be much less repetitive, while being more clear and relevant than the baselines. ADAPTIVELM is the most competitive baseline owing partially to the robustness of language models and to greater vocabulary coverage through the adaptive softmax. SEQGAN, while failing to achieve strong coherency, is surprisingly diverse, but tended to produce far shorter sentences than the other models. CACHELM has trouble dealing with the complex vocabulary of our domains without the support of either a hierarchical vocabulary structure (as in ADAPTIVELM) or a structured training method (as with SEQGAN), leading to overall poor results. While the SEQ2SEQ model has low conditional perplexity, we found that in practice it is less able to leverage long-distance dependencies than the base language model, producing more generic output. This reflects our need for more complex evaluations for generation, as such models are rarely evaluated under metrics that inspect *characteristics* of the text, rather than ability to predict the gold or overlap with the gold.

For the TripAdvisor corpus, L2W is ranked higher than the baselines on overall quality, as well as on most individual metrics, with the exception that it fails to improve on contradiction and clarity over the ADAPTIVELM (which is again the most competitive baseline). Our model's strongest improvements over the baselines are on repetition and relevance.

Trip Advisor Ablation

Ablation vs. LM	Repetition	Contradiction	Relevance	Clarity	Better	Neither	Worse
REPETITION ONLY	+0.63	+0.30	+0.37	+0.42	**50%**	23%	27%
ENTAILMENT ONLY	+0.01	+0.02	+0.05	-0.10	39%	20%	**41%**
RELEVANCE ONLY	-0.19	+0.09	+0.10	+0.060	36%	22%	**42%**
LEXICAL STYLE ONLY	+0.11	+0.16	+0.20	+0.16	38%	25%	**38%**
ALL	+0.23	-0.02	+0.19	-0.03	**47%**	19%	34%

Table 4: Crowd-sourced ablation evaluation of generations on TripAdvisor. Each ablation uses only one discriminative communication model, and is compared to ADAPTIVELM.

Ablation

To investigate the effect of individual discriminators on the overall performance, we report the results of ablations of our model in Table 4. For each ablation we include only one of the communication modules, and train a single mixture coefficient for combining that module and the language model. The diagonal of Table 4 contains only positive numbers, indicating that each discriminator does help with the purpose it was designed for. Interestingly, most discriminators help with most aspects of writing, but all except repetition fail to actually improve the overall quality over ADAPTIVELM.

The repetition module gives the largest boost by far, consistent with the intuition that many of the deficiencies of RNN as a text generator lie in semantic repetition. The entailment module (which was intended to reduce contradiction) is the weakest, which we hypothesize is the combination of (a) mismatch between training and test data (since the entailment module was trained on SNLI and MultiNLI) and (b) the lack of smoothness in the entailment scorer, whose score could only be updated upon the completion of a sentence.

Crowd Sourcing

Surprisingly, L2W is even preferred over the *original* continuation of the initial text on BookCorpus. Qualitative analysis shows that L2W's continuation is often a straightforward continuation of the original text while the true continuation is more surprising and contains complex references to earlier parts of the book. While many of the issues of automatic metrics (Liu et al., 2016; Novikova et al., 2017) have been alleviated by crowd-sourcing, we found it difficult to incentivize crowd workers to spend significant time on any one datum, forcing them to rely on a shallower understanding of the text.

5.2 Qualitative Analysis

L2W generations are more topical and stylistically coherent with the context than the baselines. Table 3 shows that L2W, ADAPTIVELM, and SEQGAN all start similarly, commenting on the breakfast buffet, as breakfast was mentioned in the last sentence of the context. The language model immediately offers generic compliments about the breakfast and staff, whereas L2W chooses a reasonable but less obvious path, stating that the previously mentioned vouchers were not used. In fact, L2W is the only system not to use the line *"The staff was very friendly and helpful."*, despite this sentence appearing in less than 1% of reviews. The semantics of this sentence, however, is expressed in many different surface forms in the training data (e.g., *"The staff were kind and quick to respond."*).

The CACHELM begins by generating the same over-used sentence and only produce short, generic sentences throughout. Seq2Seq simply repeats sentences that occur often in the training set, repeating one sentence three times and another twice. This indicates that the encoded context is essentially being ignored as the model fails to align the context and continuation.

The SEQGAN system is more detailed, e.g. mentioning a specific location "Shula's" as would be expected given its highly diverse vocabulary (as seen in Table 1). Yet it repeats itself in the first sentence. (e.g. *"had a breakfast"*, *"and a delicious breakfast"*). Consequently SEQGAN quickly devolves into generic language, repeating the incredibly common sentence *"The staff was very helpful and friendly."*, similar to SEQ2SEQ.

The L2W models do not fix every degenerate characteristic of RNNs. The TripAdvisor L2W generation consists of meaningful but mostly disconnected sentences, whereas human text tends to build on previous sentences, as in the reference continuation. Furthermore, while L2W repeats itself less than any of our baselines, it still paraphrases itself, albeit more subtly: “we would definitely recommend this hotel to others.” compared to “I’d recommend this hotel.” This example also exposes a more fine-grained issue: L2W switches from using “we” to using “I” mid-generation. Such subtle distinctions are hard to capture during beam re-ranking and none of our models address the linguistic issues of this subtlety.

6 Related Work

Alternative Decoding Objectives A number of papers have proposed *alternative decoding objectives* for generation (Shao et al., 2017). Li et al. (2016a) proposed a *diversity-promoting objective* that interpolates the conditional probability score with negative marginal or reverse conditional probabilities. Yu et al. (2017b) also incorporate the reverse conditional probability through a noisy channel model in order to alleviate the *explaining-away* problem, but at the cost of significant decoding complexity, making it impractical for paragraph generation. Modified decoding objectives have long been a common practice in statistical machine translation (Koehn et al., 2003; Och, 2003; Watanabe et al., 2007; Chiang et al., 2009) and remain common with neural machine translation, even when an extremely large amount of data is available (Wu et al., 2016). Inspired by all the above approaches, our work presents a general learning framework together with a more comprehensive set of composite communication models.

Pragmatic Communication Models Models for pragmatic reasoning about communicative goals such as Grice’s maxims have been proposed in the context of referring expression generation (Frank and Goodman, 2012). Andreas and Klein (2016) proposed a neural model where candidate descriptions are sampled from a generatively trained *speaker*, which are then re-ranked by interpolating the score with that of the *listener*, a discriminator that predicts a distribution over choices given the speaker’s description. Similar to our work the generator and discriminator scores are combined to select utterances which follow Grice’s maxims. Yu et al. (2017c) proposed a model where the speaker consists of a convolutional encoder and an LSTM decoder, trained with a ranking loss on negative samples in addition to optimizing log-likelihood.

Generative Adversarial Networks GANs (Goodfellow et al., 2014) are another alternative to maximum likelihood estimation for generative models. However, backpropagating through discrete sequences and the inherent instability of the training objective (Che et al., 2017) both present significant challenges. While solutions have been proposed to make it possible to train GANs for language (Che et al., 2017; Yu et al., 2017a) they have not yet been shown to produce high quality long-form text, as our results confirm.

Generation with Long-term Context Several prior works studied paragraph generation using sequence-to-sequence models for image captions (Krause et al., 2017), product reviews (Lipton et al., 2015; Dong et al., 2017), sport reports (Wiseman et al., 2017), and recipes (Kiddon et al., 2016). While these prior works focus on developing neural architectures for learning domain specific discourse patterns, our work proposes a general framework for learning a generator that is more powerful than maximum likelihood decoding from an RNN language model for an arbitrary target domain.

7 Conclusion

We proposed a unified learning framework for the generation of long, coherent texts, which overcomes some of the common limitations of RNNs as text generation models. Our framework learns a decoding objective suitable for generation through a learned combination of sub-models that capture linguistically-motivated qualities of good writing. Human evaluation shows that the quality of the text produced by our model exceeds that of competitive baselines by a large margin.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback and Omer Levy for helpful discussions. This research was supported in part by NSF (IIS-1524371), DARPA CwC through ARO (W911NF-15-1-0543), Samsung AI Research, and gifts by Tencent, Google, and Facebook.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In

Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1173–1182. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *CoRR*, abs/1702.07983.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Michael Denkowski and Alon Lavie. 2010. Extending the METEOR Machine Translation Evaluation Metric to the Phrase Level. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–253.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632, Valencia, Spain. Association for Computational Linguistics.

Michael C. Frank and Noah D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2016. Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*.

H Paul Grice, Peter Cole, Jerry Morgan, et al. 1975. Logic and conversation. *1975*, pages 41–58.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

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

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the Association of Computational Linguistics*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54. Association for Computational Linguistics.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Zachary Chase Lipton, Sharad Vikram, and Julian McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *CoRR*, abs/1511.03683.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. *ICLR*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

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

Allen Schmaltz, Alexander M. Rush, and Stuart Shieber. 2016. Word ordering without syntax. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas. Association for Computational Linguistics.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219. Association for Computational Linguistics.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *SIGKDD Conference on Knowledge Discovery and Data Mining*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017a. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 2852–2858.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017b. The neural noisy channel. In *International Conference on Learning Representations*.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2017c. A joint speaker-listener-reinforcer model for referring expressions. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 2.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

A Neural Approach to Pun Generation

Zhiwei Yu and **Jiwei Tan** and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{yuzw,tanjiwei,wanxiaojun}@pku.edu.cn

Abstract

Automatic pun generation is an interesting and challenging text generation task. Previous efforts rely on templates or laboriously manually annotated pun datasets, which heavily constrains the quality and diversity of generated puns. Since sequence-to-sequence models provide an effective technique for text generation, it is promising to investigate these models on the pun generation task. In this paper, we propose neural network models for homographic pun generation, and they can generate puns without requiring any pun data for training. We first train a conditional neural language model from a general text corpus, and then generate puns from the language model with an elaborately designed decoding algorithm. Automatic and human evaluations show that our models are able to generate homographic puns of good readability and quality.

1 Introduction

Punning is an ingenious way to make conversation enjoyable and plays important role in entertainment, advertising and literature. A pun is a means of expression, the essence of which is in the given context the word or phrase can be understood in two meanings simultaneously (Mikhalkova and Karyakin, 2017). Puns can be classified according to various standards, and the most essential distinction for our research is between homographic and homophonic puns. A homographic pun exploits distinct meanings of the same written word while a homophonic pun exploits distinct meanings of the same spoken word. Puns can be homographic, homophonic, both, or neither (Miller and Gurevych, 2015).

Puns have the potential to combine novelty and familiarity appropriately, which can induce pleasing effect to advertisement (Valitutti et al., 2008). Using puns also contributes to elegancy in literary writing, as laborious manual counts revealed that puns are one of the most commonly used rhetoric of Shakespeare, with the frequency in certain of his plays ranging from 17 to 85 instances per thousand lines (Miller and Gurevych, 2015). It is not an overstatement to say that pun generation has significance in human society. However, as a special branch of humor, generating puns is not easy for humans, let alone automatically generating puns with artificial intelligence techniques. While text generation is a topic of interest in the natural language processing community, pun generation has received little attention.

Recent sequence-to-sequence (seq2seq) framework is proved effective on text generation tasks including machine translation (Sutskever et al., 2014), image captioning (Vinyals et al., 2015), and text summarization (Tan et al., 2017). The end-to-end framework has the potential to train a language model which can generate fluent and creative sentences from a large corpus. Great progress has achieved on the tasks with sufficient training data like machine translation, achieving state-of-the-art performance. Unfortunately, due to the limited puns which are deemed insufficient for training a language model, there has not been any research concentrated on generating puns based on the seq2seq framework as far as we know.

The inherent property of humor makes the pun generation task more challenging. Despite decades devoted to theories and algorithms for humor, computerized humor still lacks of creativity, sophistication of language, world knowledge, empathy and cognitive mechanisms compared to humans, which are extremely difficult to model (Hossain et al., 2017).

In this paper, we study the challenging task of generating puns with seq2seq models without using a pun corpus for training. We propose a brand-new method to generate homographic puns using normal text corpus which can result in good quality of language model and avoid considerable expense of human annotators on the limited pun resources. Our proposed method can generate puns according to the given two senses of a target word. We achieve this by first proposing an improved language model that is able to generate a sentence containing a given word with a specific sense. Based on the improved language model, we are able to generate a pun sentence that is suitable for two specified senses of a homographic word, using a novel joint beam search algorithm we propose. Moreover, based on the observed characteristics of human generated puns, we further enhance the model to generate puns highlighting intended word senses. The proposed method demonstrates the ability to generate homographic puns containing the assigned two senses of a target word.

Our approach only requires a general text corpus, and we use the Wikipedia corpus in our experiment. We introduce both manual ways and automatic metrics to evaluate the generated puns. Experimental results demonstrate that our methods are powerful and inspiring in generating homographic puns.

The contributions of our work are as follows:

- To our knowledge, our work is the first attempt to adopt neural language models on pun generation. And we do not use any templates or pun data sets in training the model.

- We propose a brand-new algorithm to generate sentences containing assigned distinct senses of a target word.

- We further ameliorate our model with associative words and multinomial sampling to produce better pun sentences.

- Our approach yields substantial results on generating homographic puns with high accuracy of assigned senses and low perplexity.

2 Related Work

2.1 Pun Generation

In recent decades, exploratory research into computational humor has developed to some extent, but seldom is research specifically concerned with puns. Miller and Gurevych (2015) found that most previous studies on puns tend to focus on phonological or syntactic pattern rather than semantic pattern. In this subsection we briefly review some prior work on pun generation.

Lessard and Levison (1992) devised a program to create Tom Swifty, a type of pun which is present in a quoted utterance followed by a punning adverb. Binsted and Ritchie (1994) came up with an early prototype of pun-generator Joke Analysis and Production Engine (JAPE). The model generates question-answer punning with two types of structures: schemata for determining relationships between key words in a joke, and templates for producing the surface form of the joke. Later its successor JAPE-2 (Binsted, 1996; Binsted et al., 1997) and STANDUP (Ritchie et al., 2007) introduced constructing descriptions. The Homonym Common Phrase Pun generator (Venour, 1999) could create two-utterance texts: a one-sentence set-up and a punch-line. Venour (1999) used schemata to specify the required lexical items and their intern relations, and used templates to indicate where to fit the lexical items in a skeleton text (Ritchie, 2004). McKay (2002) proposed WISCRAIC program which can produce puns in three forms: question-answer form, single sentence and a two-sentence sequence. The Template-Based Pun Extractor and Generator (Hong and Ong, 2009) utilized phonetic and semantic linguistic resources to extract word relationships in puns automatically. The system stores the extracted knowledge in template form and results in computer-generated puns.

Most previous research on pun generation is based on templates which is convenient but lacks linguistic subtlety and can be inflexible. None of the systems aimed to be creative as the skeletons of the sentences are fixed and the generation process based on lexical information rarely needs world knowledge or reasoning (Ritchie, 2004). Recently more and more work focuses on pun detection and interpretation (Miller et al., 2017; Miller and Gurevych, 2015; Doogan et al., 2017), rather than pun generation.

2.2 Natural Language Generation

Natural language generation is an important area of NLP and it is an essential foundation for the tasks like machine translation, dialogue response generation, summarization and of course pun generation.

In the past, text generation is usually based on the techniques like templates or rules, probabilistic models like n-gram or log-linear models. Those models are fairly interpretable and well-behaved but require infeasible amounts of hand-engineering to scale with the increasing training data (Xie, 2017). In most cases larger corpus reveals better what matters, so it is natural to tackle large scale modeling (Józefowicz et al., 2016).

Recently, neural network language models (Bengio et al., 2003) have shown the good ability to model language and fight the curse of dimensionality. Cho et al. (2014) propose the encoder-decoder structure which proves very efficient to generate text. The encoder produces a fixed-length vector representation of the input sequence and the decoder uses the representation to generate another sequence of symbols. Such model has a simple structure and maps the source to the target directly, which outperforms the prior models in text generation tasks.

3 Our Models

The goal of our pun generation model is to generate a sentence containing a given target word as homographic pun. Give two senses of the target word (a polyseme) as input, our model generates a sentence where both senses of the word are appropriate in the sentence. We adopt the encoder-decoder framework to train a conditional language model which can generate sentences containing each given sense of the target word. Then we propose a joint beam search algorithm to generate an appropriate sentence to convey both senses of the target word. We call this **Joint Model** whose basic structure is illustrated in Figure 1. We further propose an improved model to highlight the different senses of the target word in one sentence, by reminding people the specific senses of the target word, which may not easily come to mind. We achieve this by using Pointwise Mutual Information (PMI) to find the associative words of each sense of the target word and increase their probability of appearance while decoding. To improve the diversity of the generated sentence, we use multinomial sampling to decode words in the decoding process. The improved model is named the **Highlight Model**.

3.1 Joint Model

3.1.1 Conditional Language Model

For a given word as input, we would like to generate a natural sentence containing the target word with the specified sense. We improve the neural language model to achieve this goal, and name it conditional language model.

The conditional language model for pun generation is similar to the seq2seq model with an input of only one word. We use Long Short-Term Memory (LSTM) as encoder to map the input sequence (target word) to a vector of a fixed dimensionality, and then another LSTM network as decoder to decode the target sequence from the vector (Sutskever et al., 2014).

Our goal is to generate a sentence containing the target word. However, vanilla seq2seq model cannot guarantee the target word to appear in the generated sequence all the time. To solve this problem, we adopt the asynchronous forward/backward generation model proposed by Mou et al. (2015), which employs a mechanism to guarantee some word to appear in the output in seq2seq models. The model first generates the backward sequence starting from the target word w_t at position t of the sentence (i.e., the words before w_t), and ending up with "</s>" at the position 0 of the sentence. The probability of the backward sequence is denoted as $p(\boldsymbol{w}_t^1)$. Then we reverse the output of the backward sequence as the input to the forward model. In this process, the goal of the encoder is to map the generated half sentence to a vector representation and the decoder will generate the latter part accordingly. The probability of the forward sequence is denoted as $p(\boldsymbol{w}_t^n)$. Then the input and output of the forward model are concatenated to form the generated sentence. In the asynchronous forward/backward model, the probability of the output sentence can be decomposed as:

$$p(\overset{\boldsymbol{w}_t^1}{\underset{\boldsymbol{w}_t^n}{=}}) = p(w_t) \prod_{i=0}^{t} p^{(bw)}(w_{t-i}|\cdot) \prod_{i=0}^{m-t+1} p^{(\mathrm{fw})}(w_{t+i}|\cdot), \tag{1}$$

where p($\doteqdot$) denotes the probability of a particular backward/forward sequence (Mou et al., 2015). $p^{(\mathrm{bw})}(w_t|\cdot)$ or $p^{(\mathrm{fw})}(w_t|\cdot)$ denotes the probabil-

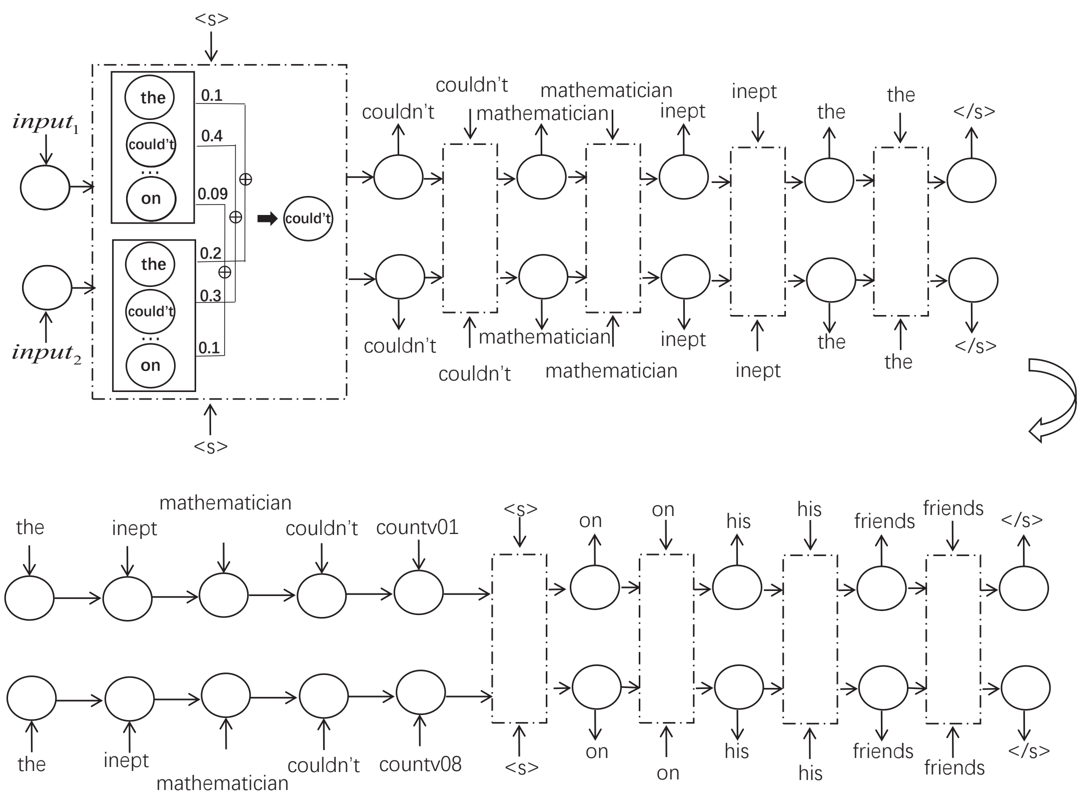


Figure 1: Framework of the proposed Joint Model. (*Top*) Two senses of the target word $input_1$ and $input_2$ (e.g. "*countv01*" and "*countv08*") are firstly provided to the backward model, to generate the backward sequence starting from the target senses and ending up with "</s>". (*Bottom*) Then the backward sequence are reversed and inputted to the forward model, to generate the forward sequence. The inputs and outputs of the forward model are concatenated to form the final output sentence. Joint beam search algorithm is used to generate each word that has the potential to make the generated sentence suitable for both input senses.

ity of w_t given previous sequence $\cdot$ in the backward or forward model respectively. The above model can only guarantee the target word to appear in the generated sentence. Since we hope to generate a sentence containing the specified word sense, we treat different senses of the same word as independent new pseudo-words. We label the senses of words with Word Sense Disambiguation (WSD) tools, and then we train the language model using the corpus with labeled senses so that for each word sense we can generate a sentence accordingly. We use the Python Implementations of WSD Technologies[1] for WSD. This tool can return the most possible sense for the target word based on WordNet (Miller, 1995). We attach the sense label to the word and form a new pseudo-word accordingly. Taking "*count*" for example, "*countv01*" means "*determine the number or amount of*", while "*countv08*" means "*have faith or confidence in*".

3.1.2 Decoding with Joint Beam Search Algorithm

Beam search is a frequently-used algorithm in the decoding stage of seq2seq models to generate the output sequence. It can be viewed as an adaptation of branch-and-bound search that uses an inadmissible pruning rule. In the beam search algorithm, only the most promising nodes at each level of the search graph are selected and the rest nodes are permanently removed. This strategy makes beam search able to find a solution within practical time or memory limits and work well in practical tasks (Zhou and Hansen, 2005; Freitag and Al-Onaizan, 2017).

We also use beam search in our pun generation model. According to the definition of homographic puns, at least two senses of the target word

[1]https://github.com/alvations/pywsd

should be interpreted in one sentence. We hope to generate a same sentence for distinct senses of the same word, and in this way the target word in the sentence can be interpreted as various senses. Provided with two senses of a target word as inputs to the encoder in the backward generation process, e.g. "*countv01*" as $input_1$ and "*countv08*" as $input_2$, we decode two output sentences in parallel, and the two sentences should be the same except for the input pseudo-words. Assume $\boldsymbol{h}_{t,i}^{(s)}$ denotes the hidden state of the i-th beam at time step t, when given the s-th pseudo-word as input (s =1 or 2). In the traditional beam search algorithm, softmax layer is applied on the hidden state to get the probability distribution on the vocabulary, and the log likelihood of the probability is used to get a word score distribution $\boldsymbol{d}_{t,i}^{(s)}$:

$$\boldsymbol{d}_{t,i}^{(s)} = log(softmax\_layer(\boldsymbol{h}_{t,i}^{(s)})). \quad (2)$$

The accumulated score distribution on the i-th beam is:

$$\boldsymbol{p}_{t,i}^{(s)} = \boldsymbol{u}_{t-1,i}^{(s)} + \boldsymbol{d}_{t,i}^{(s)}, \quad (3)$$

$|V|$ denotes the vocabulary size. $\boldsymbol{u}_{t-1,i}^{(s)}$ is a $|V|$-dimensional vector whose values are all equal to the accumulated score of the generated sequence till time step $t-1$. Assume the beam width is b, $\boldsymbol{p}_{t}^{(s)}$ is the concatenation of $\boldsymbol{p}_{t,i}^{(s)}$ on all beams and its dimension size is $|V|*b$. The beam search algorithm selects b candidate words at each time step according to $\boldsymbol{p}_{t}^{(s)}$(s =1 or 2). When decoding for $input_1$ and $input_2$ in parallel, at each time step there will be b candidates for each input according to $\boldsymbol{p}_{t}^{(1)}$ and $\boldsymbol{p}_{t}^{(2)}$ respectively. Since $input_1$ and $input_2$ are different, the candidates for two inputs will hardly be the same. However, our goal is to choose candidate words which have the potential to result in candidate sentences suitable for both senses. Our joint beam search algorithm selects b candidates while decoding for the two inputs according to the joint score distribution on all beams. The joint score distribution on the i-th beam is:

$$\boldsymbol{o}_{t,i} = \boldsymbol{p}_{t,i}^{(1)} + \boldsymbol{p}_{t,i}^{(2)}. \quad (4)$$

The summation of the log scores can be viewed as the product of original probabilities, which represents the joint probability if the two probability distributions are viewed independent. Given the b candidates selected according to the joint score distribution, our joint beam search algorithm

Algorithm 1 Joint Beam Search Algorithm

b denotes the beam width. l denotes the number of unfinished beams. $BeamId$ records which beams the candidates come from. $WordId$ records the indices of candidates in the vocabulary where 1 is the index of "<s>". $BEAM_t[i]$ denotes the i-th beam history till time step t. $|V|$ denotes the vocabulary size. $Copy(m, n)$ aims to make an n-dimensional vector by replicating m for n times. The initial states of the decoder ($\boldsymbol{h}_{-1,i}^{(1)}$,$\boldsymbol{h}_{-1,i}^{(2)}$) are equal to the final states of the encoder accordingly. $\boldsymbol{m} \uplus \boldsymbol{n}$ denotes appending $\boldsymbol{n}$ to $\boldsymbol{m}$.

```
$BEAM_{-1}[i]$= [], $i$=0, 1, ..., $b-1$
$\boldsymbol{u}_{-1,i}^{(1)} = \boldsymbol{u}_{-1,i}^{(2)} = Copy(0, |V|)$, $i = 0, 1, ..., b-1$
$BeamId = [0, 1, ..., b-1]$
$WordId = [1, .., 1] \in \mathbb{R}^b$
$Outputs = []$; $t = 0$; $l = b$
while $l > 0$ do
    $\boldsymbol{o}$=[]
    for $i$= 0 to $b-1$ do
        $\boldsymbol{x}_{t,i}$ is the word embedding corresponding to $WordId[i]$
        $\boldsymbol{h}_{t,i}^{(1)}$= LSTM($\boldsymbol{x}_{t,i}, \boldsymbol{h}_{t-1,i}^{(1)}$)
        $\boldsymbol{h}_{t,i}^{(2)}$= LSTM($\boldsymbol{x}_{t,i}, \boldsymbol{h}_{t-1,i}^{(2)}$)
        $\boldsymbol{p}_{t,i}^{(1)} = \boldsymbol{u}_{t-1,i}^{(1)} + \log(softmax\_layer(\boldsymbol{h}_{t,i}^{(1)}))$
        $\boldsymbol{p}_{t,i}^{(2)} = \boldsymbol{u}_{t-1,i}^{(2)} + \log(softmax\_layer(\boldsymbol{h}_{t,i}^{(2)}))$
        $\boldsymbol{o}_{t,i} = \boldsymbol{p}_{t,i}^{(1)} + \boldsymbol{p}_{t,i}^{(2)}$
        $\boldsymbol{o} \uplus \boldsymbol{o}_{t,i}$
    end for
    $WordId$ = the indices of words with the top $b$ scores in $\boldsymbol{o}$
    $BeamId$ = the indices of source beams w.r.t. $WordId$
    for $i$= 0 to $b-1$ do
        $BEAM_t[i] = BEAM_{t-1}[BeamId[i]] \uplus WordId[i]$
        $\boldsymbol{u}_{t,i}^{(1)} = \boldsymbol{u}_{t,i}^{(2)} = Copy(\boldsymbol{o}_{t,BeamId[i]}[WordId[i]], |V|)$
        if $WordId[i]$ represents "</s>"
            $l = l - 1$
            $Outputs = Outputs \uplus BEAM_t[i]$
        end if
    end for
    $t = t + 1$
return top $b$ items in $Outputs$
```

is similar to the vanilla beam search algorithm, which generates the candidate sequences step by step. If any beam selects "</s>" as the candidate, we regard this branch has finished decoding. The decoding process will be finished after all the beams have selected "</s>". The joint beam search algorithm is described in Algorithm 1.

3.2 Highlight Model

3.2.1 Word Association

The joint model we described above is able to generate sentences suitable for both given senses of the target word. But we found this model is prone to generate monotonous sentences, making it difficult to discover that the target word in the sentence can be understood in two ways. For example, in the sentence "*He couldn't count on his friends*", people can easily realize that the common meaning "*have faith or confidence in*" of the word "*count*", but may ignore other senses of the word. If we add some words and modify the sentence as "*The inept mathematician couldn't count on his friends*", people can also come up with the meaning "*determine the number or amount of*" due to the word "*mathematician*". Comparing the examples above, the two senses are proper in both sentences, but people may interpret "*count*" in the two sentences differently. Based on such observations, we improve the pun generation model by adding some keywords to the sentence which could remind people some special sense of the target word. We call those keywords associative words, and the improved model is named as Highlight Model.

To extract associative words of each sense of the target word, we first build word association norms in our corpus by using pointwise mutual information (PMI). As mutual information compares the probability of observing w_1 and w_2 together (the joint probability) with the probabilities of observing w_1 and w_2 independently (chance) (Church and Hanks, 1990), positive PMI scores indicate that the words occur together more than would be expected under an independence assumption, and negative scores indicate that one word tends to appear solely when the other does not (Islam and Inkpen, 2006). In this case we take top k associative words for each sense with relatively high positive PMI scores, which are calculated as follows:

$$PMI(w_1, w_2) = \log_2 \frac{p(w_1, w_2)}{p(w_1) \cdot p(w_2)}. \quad (5)$$

During decoding we increase the probability of the associative words to be chosen according to their PMI scores. For each sense of the target word, we normalize the PMI scores of the associative words as follows:

$$Asso(w_t^{(s)}, c_p) = \sigma\left(\frac{PMI(w_t^{(s)}, c_p)}{max_{c_j} PMI(w_t^{(s)}, c_j)}\right), \quad (6)$$

where $w_t^{(s)}$ represents the s-th sense of the target word w_t, and c_p is the p-th associative word for $w_t^{(s)}$. To smooth the PMI scores we use sigmoid function σ which is differentiable and widely used in the neural network models. The final PMI score for each associative word is denoted as $Asso(w_t^{(s)}, c_p)$. As we choose candidates according to a score distribution on the whole vocabulary, we need a PMI score distribution ($\boldsymbol{S}(w_t^{(s)})$) rather than single scores, and the value at position q is supposed to be:

$$\boldsymbol{S}\left(w_t^{(s)}\right)[q] = \begin{cases} Asso\left(w_t^{(s)}, v_q\right), v_q \in AssoTK(w_t^{(s)}); \\ 0, \qquad else, \end{cases} \quad (7)$$

where v_q denotes the q-th word in the vocabulary, and $AssoTK(w_t^{(s)})$ denotes the top k associative words of $w_t^{(s)}$.

3.2.2 Multinomial Sampling

In our highlight model, we add $\boldsymbol{S}(w_t^{(1)})$ and $\boldsymbol{S}(w_t^{(2)})$ to $\boldsymbol{o}_{t,i}$, as:

$$\widetilde{\boldsymbol{o}}_{t,i} = \boldsymbol{o}_{t,i} + \alpha_1 \cdot \boldsymbol{S}(w_t^{(1)}) + \alpha_2 \cdot \boldsymbol{S}(w_t^{(2)}), \quad (8)$$

where we use α_1 and α_2 as coefficient weights to balance the PMI scores of the two assigned senses and the joint score. In the Highlight Model, we first select $2b$ candidates according to the scores of words from Eq. 8. Then we use multinomial sampling to select the final b candidates. Sampling is useful in cases where we may want to get a variety of outputs for a particular input. One example of a situation where sampling is meaningful would be in a seq2seq model for a dialog system (Neubig, 2017). In our pun generation model we hope to produce relatively more creative sentences, so we use multinomial sampling to increase the uncertainty when generating the sentence. The multinomial distribution can be seen as a multivariate generalization of the binomial distribution and it is prone to choose the words with relatively high probabilities. If an associative word of one sense has been selected, we decay the scores for all associative words of this sense. In this way we can prevent the sentence obviously being prone to reflect one sense of the target word.

4 Experiments

4.1 Data Preprocessing

Most text generation tasks using seq2seq model require large amount of training data. However, for many tasks, like pun generation, it is difficult to get adequate data to train a seq2seq model. In this study, our pun generation model does not rely on training data of puns. We only require a text corpus to train the conditional language model, which is very cheap to get. In this paper, we use the English Wikipedia corpus to train the language model. The corpus texts are firstly lowercased and tokenized, and all numeric characters are replaced with "#". We split the texts into sentences and discard the sentences whose length is less than 5 words or more than 50 words. We then select polysemes appearing in the homographic pun data set (Miller et al., 2017) and pun websites. Those polysemes in the corpus are replaced by the labeled sense. We restrict that each sentence can be labeled with at most two polysemes in order to train a reliable language model. If there are more polysemes in one sentence, we keep the last two because in our observation we found pun words tend to occur near the end of a sentence. After labeling, we keep the 105,000 most frequently occurring words and other words are replaced with the "<unk>" token. We discard the sentences with two or more "<unk>" tokens. There are totally 3,974 distinct labeled senses corresponding to a total of 772 distinct polysemes. We assume those reserved senses are more likely to generate puns of good quality.

While training the language model we use 2,595,435 sentences as the training set, and 741,551 sentences as the development set to decide when to stop training.

4.2 Training Details

The number of LSTM layers we use in the seq2seq model is 2 and each layer has 128 units. To avoid overfitting, we set the dropout rate to 0.2. We use Stochastic Gradient Descent (SGD) with a decreasing learning rate schedule as optimizer. The initial learning rate is 1.0 and is halved every 1k steps after training for 8k steps, which is the same as Luong et al. (2017). We set beam size $b = 5$ while decoding. For each sense we select at most 30 associative words (k=30). To increase the probability of choosing the associative words, we set $\alpha_1 = 6.0$ and $\alpha_2 = 6.0$. If an associative word of some sense of a target word has been chosen, its corresponding α will be set to zero for all the associative words of this sense.

4.3 Baselines

Since there is no existing neural model applied on this special task, we implement two baseline models for comparison. We select 100 target words and two senses for each word to test the quality of those models.

Normal Language Model: It is trained with an encoder-decoder model and uses beam search while decoding. In the training process, inputs are unlabeled target words and outputs are sentences containing the target words.

Pun Language Model: We use the data set of homographic puns from Miller et al. (2017). The model is trained on the data set in asynchronous forward/backward way. As the pun data set is limited, the pun language model has no creativity, which means if we input a word appearing in the training data, then the output will usually be an existing sentence from the training data. Therefore, we remove the sentences which contain words in the 100 target words from the pun data set, and then train the model for test.

4.4 Automatic Evaluation

We select 100 target words and two senses for each word for test. We use the language modeling toolkit SRILM[2] to train a trigram model with another 7,746,703 sentences extracted from Wikipedia, which are different from the data set used before. The perplexity scores (PPL) of our models and baseline models are estimated based on the trained language model, as shown in Table 1. Normal Language Model has no constraint of generating sentences suitable for both senses. This means at each time step the beam search algorithm can select the candidates with highest probabilities. And thus it is natural that it obtains the lowest perplexity. Taking the constraint of senses into consideration, the perplexity scores of Joint Model and Highlight Model are still comparable to that of Normal Language Model. However, Pun Language Model could not be trained well considering the limit of the pun training data, so it gets the highest perplexity score. This result reveals that it is not feasible to build a homographic pun generation system based on the pun data set since pun data is far from enough. In the table, We further compare the diversity of the generated sentences of four models following Li et al. (2016). Distinct-1 (d.-1) and distinct-2 (d.-2) are the ratios of the distinct unigrams and bigrams in generated sentences, i.e., the number of distinct unigrams or bigrams divided by the total number of unigrams or bigrams. The results show our models are more creative than Normal Language and

[2]http://www.speech.sri.com/projects/srilm/

Model	PPL	d.-1(%)	d.-2(%)
Highlight	91.80	**27.13**	**62.85**
Joint	63.48	22.13	50.59
Normal Language	**62.66**	19.60	41.62
Pun Language	889.07	14.78	23.11

Table 1: Results of automatic evaluation.

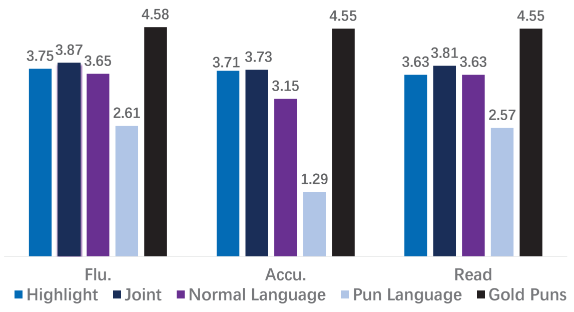


Figure 2: Results of human evaluation.

Pun Language models, and Highlight Model can generate sentences with the best diversity.

4.5 Human Evaluation

Because of the subtle and delicate structure of puns, automatic evaluation is not enough. So we sample one sentence for each word from four models mentioned above and then get 100 sentences of each model generated from the target words, together with 100 puns containing the same target words from homographic pun data set in Miller et al. (2017). We ask judges on Amazon Mechanical Turk to evaluate all the sentences and the rating score ranges from 1 to 5. Five native English speakers are asked to give a score on each sentence in three aspects with the following information: **Readability** indicates whether the sentence is easy to understand semantically; **Accuracy** indicates whether the given senses are suitable in a sentence; **Fluency** indicates whether the sentence is fluent and consistent with the rules of grammar.

The results in Figure 2 show that pun data is not enough to train an ideal language model, while Normal Language Model has enough corpus to train a good language model. But Normal Language Model is unable to make the given two senses appear in one sentence and in a few cases even can not assure the appearance of the target words. Joint Model and Highlight Model can generate fluent sentences for the assigned two senses. Although Highlight Model could remind people specific senses of the target words in most cases, in few cases sampled words make the whole sentence unsatisfactory and get a relatively lower score of accuracy. As to the Readability, the Joint Model performs better than other three models. Both Joint model and Highlight model outperform Normal Language Model and Pun Language Model.

Model	# sentences	avg. score
Highlight	15	0.98
Joint	12	0.87
Gold Puns	28	1.38

Table 2: Results of Soft Turing Test.

To test the potential of the sentences generated by our models to be homographic puns, we further design a Soft Turing Test. We select 30 sentences generated by Joint Model and 30 sentences generated by Highlight Model independently, together with 30 gold puns from the homographic pun data set. We mix them up, and give the definition of homographic pun and ask 10 people on Amazon Mechanical Turk to judge each sentence. People can judge each sentence as one of three categories: definitely by human, might by human and definitely by machine. The three categories correspond to the scores of 2, 1 and 0, respectively. If the average score of one sentence is equal or greater than 1, we regard it as judged to be generated by human. The number of sentences judged as by human for each model and the average score for each model are shown in Table 2.

Due to the flexible language structure of Highlight Model, the generated homographic puns outperform those generated by Joint Model in the Soft Turing Test, however still far from gold-standard puns. Our models are adept at generating homographic puns containing assigned senses but weak in making homographic puns humorous.

4.6 Examples

We show some examples generated by different models in Table 3. For the two senses of *"pitch"*, Highlight Model generates a sentence which uses *"high"* to remind readers the sense related to sound and uses *"player"* to highlight the sense related to throwing a baseball. Joint Model returns a sentence that can be understood in both way roughly only if we give the two senses in advance, otherwise readers may only think of the

Model	Sample
pitch: 1) the property of sound that arise with variation in the frequency of vibration; 2) the act of throwing a baseball by a pitcher to a batter.	
Highlight	in one that denotes player may have had a high pitch in the world
Joint	the object of the game is based on the pitch of the player
Normal Language	this is a list of high pitch plot
Pun Language	our bikinis are exciting they are simply the tops on the mouth
Gold Puns	if you sing while playing baseball you won't get a good pitch
square: 1) a plane rectangle with four equal sides and four right angles, a four-sided regular polygon; 2) someone who doesn't understand what is going on.	
Highlight	little is known when he goes back to the square of the football club
Joint	there is a square of the family
Normal Language	the population density was # people per square mile
Pun Language	when the pirate captain's ship ran aground he couldn't fathom why
Gold Puns	my advanced geometry class is full of squares
problem: 1) a source of difficulty; 2) a question raised for consideration or solution.	
Highlight	you do not know how to find a way to solve the problem which in the state
Joint	he is said to be able to solve the problem as he was a professor
Normal Language	in # he was appointed a member of the new york stock exchange
Pun Language	those who iron clothes have a lot of pressing veteran
Gold Puns	math teachers have lots of problems

Table 3: Examples of outputs by different models.

sense related to baseball. For Normal Language Model, it is difficult to be interpreted in two senses we assigned. Pun Language Model has no ability to return a sentence containing the assigned word at all. Observing the gold pun, the context describes a more vivid scene which we need to pay attention to. For "*square*", sentences generated by Highlight Model and Joint Model can be interpreted in two senses and Highlight Model results in a sentence with dexterity. Normal Language Model give a sentence where "*square*" means neither of the two given senses. Pun Language Model cannot return a sentence we need with no surprise. For "*problem*", both Highlight Model and Joint Model can generate sentences containing assigned two senses while Normal Language Model and Pun Language Model can not return sentences with the target word. Compare to our generated sentences, we find gold puns are more concise and accurate, which takes us into consideration on the delicate structure of puns and the conclusion is still in exploration.

5 Conclusion and Future Work

In this paper, we proposed two models for pun generation without using training data of puns. Joint Model makes use of conditional language model and the joint beam search algorithm, which can assure the assigned senses of target words suitable in one sentence. Highlight Model takes associative words into consideration, which makes the distinct senses more obvious in one sentence. The produced puns are evaluated using automatic evaluation and human evaluation, and they outperform the sentences generated by our baseline models.

For future work, we hope to improve the results by using the pun data and design a more proper way to select candidates from associative words.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61772036, 61331011) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155. http://www.jmlr.org/papers/v3/bengio03a.html.

Kim Binsted. 1996. Machine humour: An implemented model of puns .

Kim Binsted, Helen Pain, and Graeme D Ritchie. 1997. Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition* 5(2):305–354.

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*. pages 633–638. http://www.aaai.org/Library/AAAI/1994/aaai94-096.php.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint* arXiv:1406.1078. http://arxiv.org/abs/1406.1078.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics* 16(1):22–29.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*. pages 103–108. https://doi.org/10.18653/v1/S17-2011.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation, NMT@ACL 2017, Vancouver, Canada, August 4, 2017*. pages 56–60. https://aclanthology.info/papers/W17-3207/w17-3207.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*. Association for Computational Linguistics, Stroudsburg, PA, USA, CALC '09, pages 24–31. http://dl.acm.org/citation.cfm?id=1642011.1642015.

Nabil Hossain, John Krumm, Lucy Vanderwende, Eric Horvitz, and Henry A. Kautz. 2017. Filling the blanks (hint: plural noun) for mad libs humor. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 638–647. https://aclanthology.info/papers/D17-1067/d17-1067.

Aminul Islam and Diana Inkpen. 2006. Second order co-occurrence PMI for determining the semantic similarity of words. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006.*. pages 1033–1038.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint* arXiv:1602.02410. http://arxiv.org/abs/1602.02410.

Greg Lessard and Michael Levison. 1992. Computational modelling of linguistic humour: Tom swifties. In *In ALLC/ACH Joint Annual Conference, Oxford*. pages 175–178.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 110–119. http://aclweb.org/anthology/N/N16/N16-1014.pdf.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt* .

Justin McKay. 2002. Generation of idiom-based witticisms to aid second language learning. In *In Stock et al.*. pages 77–87.

Elena Mikhalkova and Yuri Karyakin. 2017. Punfields at semeval-2017 task 7: Employing roget's thesaurus in automatic pun recognition and interpretation. *arXiv preprint* arXiv:1707.05479. http://arxiv.org/abs/1707.05479.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 719–729. http://aclweb.org/anthology/P/P15/P15-1070.pdf.

Tristan Miller, Christian F. Hempelmann, and Iryna Gurevych. 2017. SemEval-2017 Task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 59–69.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015. Backbone language modeling for constrained natural language generation. *arXiv preprint* arXiv:1512.06612. http://arxiv.org/abs/1512.06612.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint* arXiv:1703.01619. http://arxiv.org/abs/1703.01619.

Graeme Ritchie. 2004. *The linguistic analysis of jokes*. Routledge.

Graeme Ritchie, Ruli Manurung, Helen Pain, Annalu Waller, Rolf Black, and Dave O'Mara. 2007. A practical application of computational humour. In *Proceedings of the 4th International Joint Conference on Computational Creativity*. pages 91–98.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics, pages 1171–1181. https://doi.org/10.18653/v1/P17-1108.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. 2008. Textual affect sensing for computational advertising. In *Creative Intelligent Systems, Papers from the 2008 AAAI Spring Symposium, Technical Report SS-08-03, Stanford, California, USA, March 26-28, 2008*. pages 117–122.

Chris Venour. 1999. The computational generation of a class of puns. In *Master's thesis, Queen's University,Kingston, Ontario*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, pages 3156–3164. https://doi.org/10.1109/CVPR.2015.7298935.

Ziang Xie. 2017. Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534* .

Rong Zhou and Eric A. Hansen. 2005. Beam-stack search: Integrating backtracking with beam search. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*. pages 90–98. http://www.aaai.org/Library/ICAPS/2005/icaps05-010.php.

Learning to Generate Move-by-Move Commentary for Chess Games from Large-Scale Social Forum Data

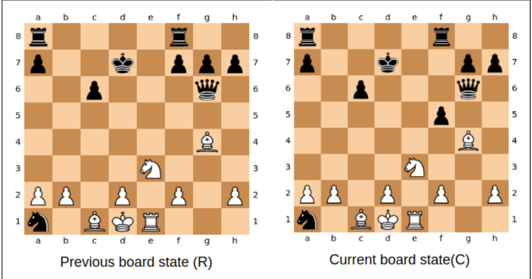
Harsh Jhamtani[*], Varun Gangal[*], Eduard Hovy, Graham Neubig, Taylor Berg-Kirkpatrick
Language Technologies Institute
Carnegie Mellon University
`{jharsh,vgangal,hovy,gneubig,tberg}@cs.cmu.edu`

Abstract

This paper examines the problem of generating natural language descriptions of chess games. We introduce a new large-scale chess commentary dataset and propose methods to generate commentary for individual moves in a chess game. The introduced dataset consists of more than 298K chess move-commentary pairs across 11K chess games. We highlight how this task poses unique research challenges in natural language generation: the data contain a large variety of styles of commentary and frequently depend on pragmatic context. We benchmark various baselines and propose an end-to-end trainable neural model which takes into account multiple pragmatic aspects of the game state that may be commented upon to describe a given chess move. Through a human study on predictions for a subset of the data which deals with direct move descriptions, we observe that outputs from our models are rated similar to ground truth commentary texts in terms of correctness and fluency.[1]

1 Introduction

A variety of work in NLP has sought to produce fluent natural language descriptions conditioned on a contextual grounding. For example, several lines of work explore methods for describing images of scenes and videos (Karpathy and Fei-Fei, 2015), while others have conditioned on structured sources like Wikipedia infoboxes (Lebret et al., 2016). In most cases, progress has been driven by the availability of large training corpora that pair natural language with examples from the grounding (Lin et al., 2014). One line of work has investigated methods for producing and interpreting language in the context of a game, a space that has rich pragmatic structure, but where training data has been hard to come by. In this paper, we introduce a new large-scale resource for learning to correlate natural language with individual moves in the game of chess. We collect a dataset of more than 298K chess move/commentary pairs across ≈ 11K chess games from online chess forums. To the best of our knowledge, this is the first such dataset of this scale for a game commentary generation task. We provide an analysis of the dataset and highlight the large variety in commentary texts by categorizing them into six different aspects of the game that they respectively discuss.



Template: Black moves the pawn from f7 to f5

Nearest Neighbor: Attacks the bishop , but allows black to gain the pawn back .

GAC(Our method): Black threatens white bishop with his pawn.

Ground Truth: Blocking the bishop out and eyeing for f4 and mate on c

Figure 1: Move commentary generated from our method (Game-aware neural commentary generation (GAC)) and some baseline methods for a sample move.

Automated game commentary generation can be a useful learning aid. Novices and experts alike can learn more about the game by hearing expla-

[*] HJ and VG contributed equally for this paper

[1]We will make the code-base (including data collection and processing) publicly available at `https://github.com/harsh19/ChessCommentaryGeneration`

nations of the motivations behind moves, or their quality. In fact, on sites for game aficionados, these commentaries are standard features, speaking to their interestingness and utility as complements to concrete descriptions of the game boards themselves.

Game commentary generation poses a number of interesting challenges for existing approaches to language generation. First, modeling human commentary is challenging because human commentators rely both on their prior knowledge of game rules as well as their knowledge of effective strategy when interpreting and referring to the game state. Secondly, there are multiple aspects of the game state that can be talked about for a given move — the commentator's choice depends on the pragmatic context of the game. For example, for the move shown in Figure 1, one can comment simply that the pawn was moved, or one may comment on how the check was blocked by that move. Both descriptions are true, but the latter is most salient given the player's goal. However, sometimes, none of the aspects may stand out as being most salient, and the most salient aspect may even change from commentator to commentator. Moreover, a human commentator may introduce variations in the aspects he or she chooses to talk about, in order to reduce monotony in the commentary. This makes the dataset a useful testbed not only for NLG but also for related work on modeling pragmatics in language (Liu et al., 2016).

Prior work has explored game commentary generation. Liao and Chang (1990); Sadikov et al. (2006) have explored chess commentary generation, but for lack of large-scale training data their methods have been mainly rule-based. Kameko et al. (2015) have explored commentary generation for the game of Shogi, proposing a two-step process where salient terms are generated from the game state and then composed in a language model. In contrast, given the larger amount of training data available to us, our proposed model uses an end-to-end trainable neural architecture to predict commentaries given the game state. Our model conditions on semantic and pragmatic information about the current state and explicitly learns to compose, conjoin, and select these features in a recurrent decoder module. We perform an experimental evaluation comparing against baselines and variants of our model that ablate various aspects of our proposed architecture. Outputs on the 'Move Description' subset of data from our final model were judged by humans to be as good as human written ground truth commentaries on measures of fluency and correctness.

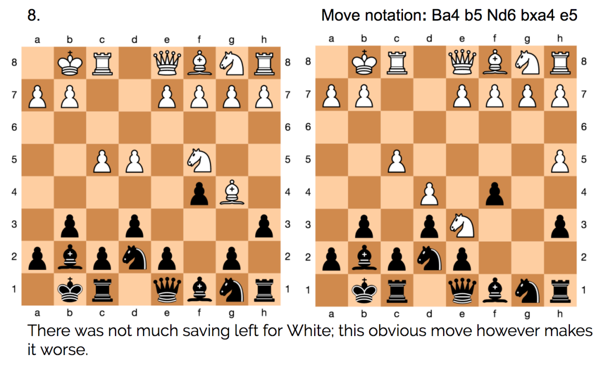

8.
Move notation: Ba4 b5 Nd6 bxa4 e5
There was not much saving left for White; this obvious move however makes it worse.

Figure 2: A multi-move, single commentary example from our data. Here, the sequence of moves Ba4 $\rightarrow$ b5 $\rightarrow$ Nd6 $\rightarrow$ bxa4 $\rightarrow$ e5 is commented upon.

Statistic	Value
Total Games	11,578
Total Moves	298,008
Average no. of recorded steps in a game	25.73
Frequent Word Types[2]	39,424
Rare Word Types	167,321
Word Tokens	6,125,921
Unigram Entropy	6.88
Average Comment Length (in #words)	20.55
Long Comments (#words > 5)	230745 (77%)

Table 1: Dataset and Vocabulary Statistics

2 Chess Commentary Dataset

In this section we introduce our new large-scale *Chess Commentary* dataset, share some statistics about the data, and discuss the variety in type of commentaries. The data is collected from the online chess discussion forum `gameknot.com`, which features multiple games self-annotated with move-by-move commentary.

The dataset consists of 298K aligned game move/commentary pairs. Some commentaries are written for a sequence of few moves (Figure 2) while others correspond to a single move. For the purpose of initial analysis and modeling, we limit ourselves to only those data points where commentary text corresponds to a single move. Additionally, we split the multi-sentence commentary texts to create multiple data points with the same chess board and move inputs.

What are commentaries about? We observe that there is a large variety in the commentary

Category	Example	% in data	Val acc.
Direct Move Description	An attack on the queen	31.4%	71%
Move Quality	A rook blunder.	8.0%	90%
Comparative	At this stage I figured I better move my knight.	3.7%	77.7%
Planning / Rationale	Trying to force a way to eliminate d5 and prevent Bb5.	31.2%	65%
Contextual Game Info	Somehow, the game I should have lost turned around in my favor .	12.6%	87%
General Comment	Protect Calvin , Hobbs	29.9%	78%

Table 2: Commentary texts have a large variety making the problem of content selection an important challenge in our dataset. We classify the commentaries into 6 different categories using a classifier trained on some hand-labelled data, a fraction of which is kept for validation. % data refers to the percentage of commentary sentences in the tagged data belonging to the respective category.

texts. To analyze this variety, we consider labelling the commentary texts in the data with a predefined set of categories. The choice of these categories is made based on a manual inspection of a sub-sample of data. We consider the following set of commentary categories (Also shown in Table 2):

- **Direct move description (MoveDesc[3]):** Explicitly or implicitly describe the current move.
- **Quality of move (Quality[4]):** Describe the quality of the current move.
- **Comparative:** Compare multiple possible moves.
- **Move Rationale or Planning (Planning):** Describe the rationale for the current move, in terms of the future gameplay, advantage over other potential moves etc.
- **Contextual game information:** Describe not the current move alone, but the overall game state – such as possibility of win/loss, overall aggression/defence, etc.
- **General information:** General idioms & advice about chess, information about players/tournament, emotional remarks, retorts, etc.

The examples in Table 2 illustrate these classes. Note that the commentary texts are not necessarily limited to one tag, though that is true for most of the data. A total of 1K comments are annotated by two annotators. A SVM classifier (Pedregosa et al., 2011a) is trained for each comment class, considering the annotation as ground truth and using word unigrams as features. This classifier is then used to predict tags for the train, validation and test sets. For "Comparative" category, we found that a classifier with manually defined rules such as presence of word "better" performs better than the classifier, perhaps due to the paucity of data, and thus we use this instead . As can be observed in Table 2, the classifiers used are able to generalize well on the held out dataset

[3]MoveDesc & 'Move Description' used interchangeably
[4]Quality and 'Move Quality' used interchangeably

3 Game Aware Neural Commentary Generations (GAC)

Our dataset D consists of data points of the form $(S_i, M_i, G_i), i \in \{1, 2, .., |D|\}$, where S_i is the commentary text for move M_i and G_i is the corresponding chess game. S_i is a sequence of m tokens $S_{i1}, S_{i2}, ..., S_{im}$ We want to model $P(S_i|M_i, G_i)$. For simplicity, we use only current board (C_i) and previous board (R_i) information from the game. $P(S_i|M_i, G_i) = P(S_i|M_i, C_i, R_i)$.

We model this using an end-to-end trainable neural model, which models conjunctions of features using feature encoders. Our model employs a selection mechanism to select the salient features for a given chess move. Finally a LSTM recurrent neural network (Hochreiter and Schmidhuber, 1997) is used to generate the commentary text based on selected features from encoder.

3.1 Incorporating Domain Knowledge

Past work shows that acquiring domain knowledge is critical for NLG systems (Reiter et al., 2003b; Mahamood and Reiter, 2012). Commentary texts cover a range of perspectives, including criticism or goodness of current move, possible alternate moves, quality of alternate moves, etc. To be able to make such comments, the model must learn about the quality of moves, as well as the set of valid moves for a given chess board state. We consider the following features to provide our model with necessary information to generate commentary texts (Figure 3):

Move features $f_{move}(M_i, C_i, R_i)$ encode the current move information such as which piece moved, the position of the moved piece before and after the move was made, the type and position

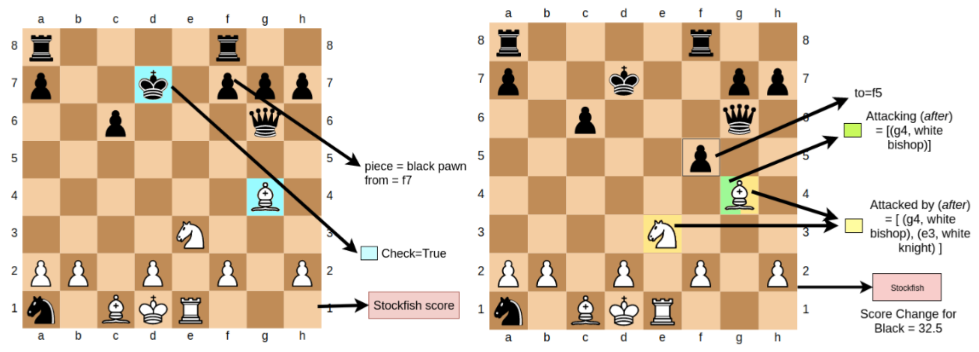

piece = black pawn from = f7
Check=True
Stockfish score
to=f5
Attacking (*after*) = [(g4, white bishop)]
Attacked by (*after*) = [(g4, white bishop), (e3, white knight)]
Stockfish
Score Change for Black = 32.5

Figure 3: The figure shows some features extracted using the chess board states before (*left*) and after (*right*) a chess move. Our method uses various semantic and pragmatic features of the move, including the location and type of piece being moved, which opposing team pieces attack the piece being moved before as well as after the move, the change in score by *Stockfish* UCI engine, etc.

of the captured piece (if any), whether the current move is castling or not, and whether there was a check or not.

Threat features $f_{threat}(M_i, C_i, R_i)$ encode information about pieces of opposite player attacking the moved piece before and after the move, and the pieces of opposite player being attacked by the piece being moved. To extract this information, we use the *python-chess* library [5]

Score features $f_{score}(M_i, C_i, R_i)$ capture the quality of move and general progress of the game. This is done using the game evaluation score before and after the move, and average rank of pawns of both the players. We use *Stockfish* evaluation engine to obtain the game evaluation scores. [6]

3.2 Feature Representation

In our simplest conditioned language generation model **GAC-sparse**, we represent the above described features using sparse representations through binary-valued features. $g_{sparse}(M_i, C_i, R_i) = \text{SparseRep}(f_{move}, f_{threat}, f_{score})$

For our full **GAC** model we consider representing features through embeddings. This has the advantage of allowing for a shared embedding space, which is pertinent for our problem since attribute values can be shared, e.g. the same piece type can occur as the moved piece as well as the captured piece. For categorical features, such as those indicating which piece was moved, we directly look up the embedding using corresponding token. For real valued features such as game scores, we first bin them and then use corresponding number for embedding lookup. Let E represent the embedding matrix. Then $E[f^j_{move}]$ represents embeddings of j^{th} move feature, or in general $E[f_{move}]$ represents the concatenated embeddings of all move features. Similarly, $E(f_{move}, f_{threat}, f_{score})$ represents concatenated embeddings of all the features.

3.3 Feature Conjunctions

We conjecture that explicitly modeling feature conjunctions might improve the performance. So we need an encoder which can handle input sets of features of variable length (features such as pieces attacking the moved piece can be of variable length). One way to handle this is by picking up a canonical ordering of the features and consider a bidirectional LSTM encoder over the feature embeddings. As shown in Figure 4, this generates conjunctions of features.
$g^{enc} = \text{BiLSTM}^*(\{E(f_{move}, f_{threat}, f_{score}))\})$

Here $E()$ represents the embedding matrix as described earlier and $BiLSTM^*$ represents a sequential application of the $BiLSTM$ function. Thus, if there a total of m feature keys and embedding dimension is d, $E(f_{move}, f_{threat}, f_{score})$ is matrix of $m * d$. If hidden size of BILSTM is of size x, then g^{enc} is of dimensionality $m * x$. We observe that different orderings gave similar performance. We also experimented with running k encoders, each on different ordering of features, and then letting the decoder access to each of the k encodings. This did not yield any significant gain in performance.

The GAC model, unlike GAC-sparse, has some advantages as it uses a shared, continuous space

[5] https://pypi.org/project/python-chess/
[6] https://stockfishchess.org/about/

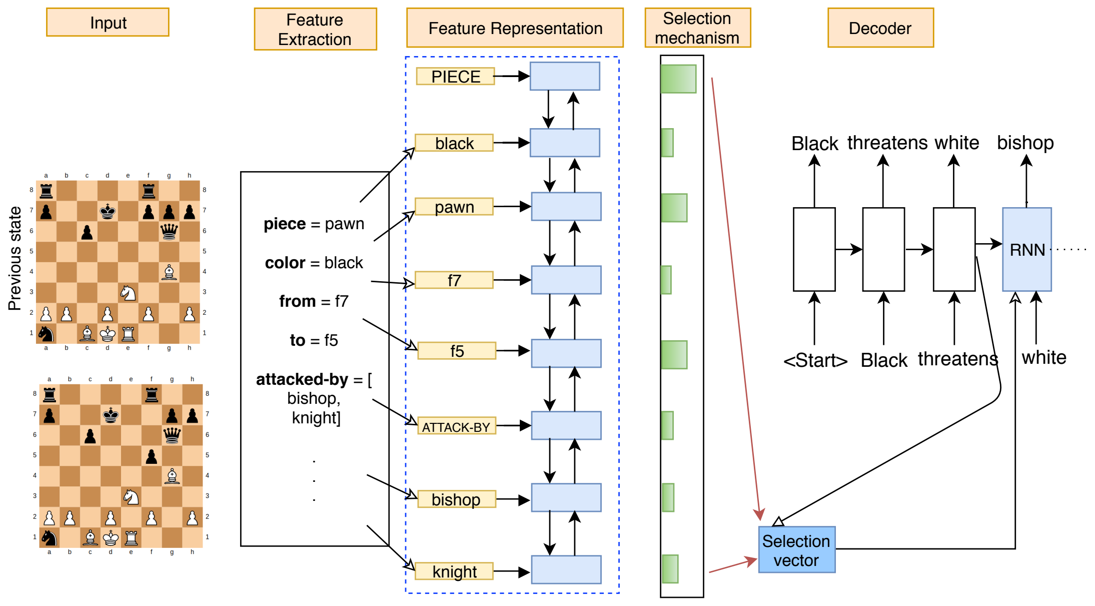


Figure 4: The figure shows a model overview. We first extract various semantic and pragmatic features from the previous and current chess board states. We represent features through embedding in a shared space. We observe that feeding in feature conjunctions helps a lot. We consider a selection mechanism for the model to choose salient attributes from the input at every decoder step.

to embed attribute values of different features, and can perform arbitrary feature conjunctions before passing a representation to the decoder, thereby sharing the burden of learning the necessary feature conjunctions. Our experiments confirm this intuition — GAC produces commentaries with higher BLEU as well as more diversity compared to GAC-sparse.

3.4 Decoder

We use a LSTM decoder to generate the sentence given the chess move and the features g. At every output step t, the LSTM decoder predicts a distribution over vocabulary words taking into account the current hidden state h_t, the input token i_t, and additional selection vector c_t. For GAC-sparse, the selection vector is simply an affine transformation of the features g. For GAC model selection vector is derived via a selection mechanism.

$$o_t, h_t^{dec} = LSTM(h_{t-1}^{dec}, [\text{concat}(E_{dec}(i_t), c_t)])$$
$$p_t = \text{softmax}(W_o[\text{concat}(o_t, c_t)] + b_s)$$

where p_t represents th probability distribution over the vocabulary, $E_{dec}()$ represents the decoder word embedding matrix and elements of W_o matrix are trainable parameters.

Selection/Attention Mechanism: As there are different salient attributes across the different chess moves, we also equip the GAC model with a mechanism to select and identify these attributes. We first transform h_t^{dec} by multiplying it with a trainable matrix W_c, and then take dot product of the result with each g_i.

$$a_t^{(i)} = dot(W_c * h_t^{dec}, g_i^{enc})$$
$$\alpha_t = \text{softmax}(a_t)$$
$$c_t = \sum_{i=1}^{i=|g|} \alpha_t^{(i)} g_i^{enc}$$

We use cross-entropy loss over the decoding outputs to train the model.

4 Experiments

We split each of the data subsets in a 70:10:20 ratio into train, validation and test. All our models are implemented in Pytorch version 0.3.1 (Paszke et al., 2017). We use the ADAM optimizer (Kingma and Ba, 2014) with its default parameters and a mini-batch size of 32. Validation set perplexity is used for early-stopping. At test-time, we use greedy search to generate the model output. We observed that beam decoding does not lead to any significant improvement in terms of validation BLEU score.

We observe the BLEU (Papineni et al., 2002) and BLEU-2 (Vedantam et al., 2015) scores to measure the performance of the models. Additionally, we consider a measure to quantify the diversity in the generated outputs. Finally, we also conduct a human evaluation study. In the remainder of this section, we discuss baselines along with various experiments and results.

4.1 Baselines

In this subsection we discuss the various baseline methods.

Manually-defined template (TEMP) We devise manually defined templates (Reiter, 1995) for 'Move Description' and 'Move Quality' categories. Note that template-based outputs tend to be repetitive as they lack diversity - drawing from a small, fixed vocabulary and using a largely static sentence structure. We define templates for a fixed set of cases which cover our data (For exact template specifications, refer to Appendix B).

Nearest Neighbor (NN): We observe that the same move on similar board states often leads to similar commentary texts. To construct a simple baseline, we find the most similar move N_{MCR} from among training data points for a given previous (R) and current (C) board states and move M. The commentary text corresponding to N_{MCR} is selected as the output. Thus, we need to consider a scoring function to find the closest matching data point in training set. We use the *Move*, *Threat* and *Score* features to compute similarity to do so. By using a sparse representation, we consider total of 148 *Move* features, 18 *Threat* features, and 19 *Score* features. We use sklearn's (Pedregosa et al., 2011b) NearestNeighbor module to find the closest matching game move.

Raw Board Information Only (RAW): The RAW baseline ablates to assess the importance of our pragmatic feature functions. This architecture is similar to GAC, except that instead of our custom features $A(f(R_i, C_i))$, the encoder encodes raw board information of current and previous board states.

$A_{RAW}(R_i, C_i) = [Lin(R_i), Lin(C_i)]$

$Lin()$ for a board denotes it's representation in a row-linear fashion. Each element of $Lin()$ is a piece name (e.g *pawn*) denoting the piece at that square with special symbols for empty squares.

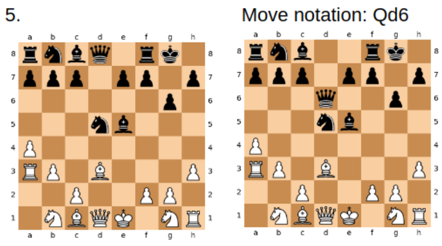
4.2 Comment Category Models

As shown earlier, we categorize comments into six different categories. Among these, in this paper we consider only the first three as the amount of variance in the last three categories indicates that it would be extremely difficult for a model to learn to reproduce them accurately. The number of data points, as tagged by the trained classifiers, in the subsets 'Move Description', 'Move Quality' and 'Comparative' are 28,228, 793 and 5397 respectively. We consider separate commentary generation models for each of the three categories. Each model is tuned separately on the corresponding validation sets. Table 3 shows the BLEU and BLEU-2 scores for the proposed model under different subsets of features. Overall BLEU scores are low, likely due to the inherent variance in the language generation task (Novikova et al., 2017) , although a precursory examination of the outputs for data points selected randomly from test set indicated that they were reasonable. Figure 5 illustrates commentaries generated by our models through an example (a larger list of qualitative examples can be found in Appendix C).

Dataset	Features	BLEU	BLEU-2	Diversity
MoveDesc	TEMP	0.72	20.77	4.43
	NN (M+T+S)	1.28	21.07	**7.85**
	RAW	1.13	13.74	2.37
	GAC-sparse	1.76	21.49	4.29
	GAC (M+T)	**1.85**	**23.35**	**4.72**
Quality	TEMP	16.17	47.29	1.16
	NN (M+T)	5.98	42.97	4.52
	RAW	16.92	47.72	1.07
	GAC-sparse	14.98	51.46	2.63
	GAC(M+T+S)	16.94	47.65	1.01
Comparative	NN (M)	1.28	24.49	6.97
	RAW	2.80	23.26	3.03
	GAC-sparse	**3.58**	25.28	2.18
	GAC(M+T)	3.51	29.48	3.64

Table 3: Performance of baselines and our model with different subsets of features as per various quantitative measures. (**S** = Score, **M**= Move, **T** = Threat features;) On all data subsets, our model outperforms the **TEMP** and **NN** baselines. Among proposed models, GAC performs better than GAC-sparse & RAW in general. For NN, GAC-sparse and GAC methods, we experiment with multiple feature combinations and report only the best as per BLEU scores.

Which features are useful? In general, adding *Threat* features improves the performance, though the same is not always true for *Score* features. *Qual* has higher BLEU scores than the other datasets due to smaller vocabulary and lesser variation in commentary. As can be observed in Table 4, *Threat* features are useful for both 'Move Quality' and 'Move Description' subsets of data. Adding *Score* features helps for 'Move Quality' subset. This intuitively makes sense since *Score*



Temp: Black moves the queen from d8 to d6

NN: Black brings out his queen , which breaks the queen and provides protection the the e-pawn

GAC(M): Black attacks white queen.

GAC(M+T): Black brings out his queen to the attack .

GAC(M+T+S): Black brings his queen out

Ground Truth: Letting the Queen join the attack.

Figure 5: Outputs from various models on a test example from the **MoveDesc** subset.

Dataset	Features	BLEU	BLEU-2	Diversity
MoveDesc	GAC (M)	1.41	19.06	4.32
	GAC (M+T)	**1.85**	**23.35**	**4.72**
	GAC (M+T+S)	1.64	22.82	4.29
Quality	GAC (M)	13.05	48.37	1.61
	GAC (M+T)	14.22	49.57	**1.54**
	GAC(M+T+S)	**14.44**	**51.79**	1.48
Comparative	GAC(M)	3.10	19.84	2.88
	GAC(M+T)	**3.51**	**29.48**	**3.64**
	GAC(M+T+S)	1.15	25.44	3.14

Table 4: Performance of the GAC model with different feature sets. (**S** = Score, **M**= Move, **T** = Threat features;) Different subset of features work best for different subsets. For instance, *Score* features seem to help only in the Quality category. Note that the results for Quality are from 5-fold cross-validation, since the number of datapoints in the category is much lesser than the other two.

features directly encode proxies for move quality as per a chess evaluation engine.

4.3 A Single Model For All Categories

In this experiment, we merge the training and validation data of the first three categories and tune a single model for this merged data. We then compare its performance on all test sentences in our data. **COMB** denotes using the best GAC model for a test example based on its original class (e.g *Desc*) and computing the BLEU of the sentences so generated with the ground truth. **GAC-all** represents the GAC model learnt on the merged training data.

As can be seen from Table 5, this does not lead to any performance improvements. We investigate this issue further by analyzing whether the board states are predictive of the type of category or not. To achieve this, we construct a multi-class classifier using all the *Move*, *Threat* and *Score* features to predict the three categories under consideration. However, we observe accuracy of around 33.4%, which is very close to the performance of a random prediction model. This partially explains why a single model did not fare better even though it had the opportunity to learn from a larger dataset.

Dataset	Features	BLEU	BLEU-2	Diversity
All	COMB (M)	2.07	20.13	4.50
	COMB (M+T)	**2.43**	**25.37**	4.88
	COMB (M+T+S)	1.83	28.86	4.33
All	GAC-all(M)	1.69	20.66	4.67
	GAC-all(M+T)	1.94	24.11	5.16
	GAC-all (M+T+S)	2.02	24.70	4.97
All	CAT (M)	1.90	19.96	3.82

Table 5: The **COMB** approaches show the combined performance of separately trained models on the respective test subsets.

Category-aware model (CAT) We observed above that with the considered features, it is not possible to predict the type of comment to be made, and the GAC-all model results are better than COMB results. Hence, we extend the GAC-all model to explicitly provide with the information about the comment category. We achieve this by adding a one-hot representation of the category of the comment to the input of the RNN decoder at every time step. As can be seen in the Table 5, CAT(M) performs better than GAC-all(M) in terms of BLEU-4, while performing slightly worse on BLEU-2. This demonstrates that explicitly providing information about the comment category can help the model.

4.4 Diversity In Generated Commentaries

Humans use some variety in the choice of words and sentence structure. As such, outputs from rule based templates, which demonstrate low variety, may seem repetitive and boring. To capture this quantitatively, and to demonstrate the variety in texts from our method, we calculate the entropy (Shannon, 1951) of the distribution of unigrams, bigrams and trigrams of words in the predicted outputs, and report the geometric mean of these values. Using only a small set of words in similar counts will lead to lower entropy and is undesirable. As can be observed from Table 3, template baseline performs worse on the said measure compared to our methods for the 'MoveDesc' subset of the data.

4.5 Human Evaluation Study

As discussed in the qualitative examples above, we often found the outputs to be good - though BLEU scores are low. BLEU is known to correlate poorly (Reiter and Belz, 2009; Wiseman et al., 2017; Novikova et al., 2017) with human relevance scores for NLG tasks. Hence, we conduct a human evaluation study for the best 2 neural (GAC,GAC-sparse) and best 2 non-neural methods (TEMP,NN).

Setup: Specifically, annotators are shown a chess move through previous board and resulting board snapshots, along with information on which piece moved (a snapshot of a HIT[7] is provided in the Appendix D). With this context, they were shown text commentary based on this move and were asked to judge the commentary via three questions, shortened versions of which can be seen in the first column of Table 6.

We randomly select 100 data points from the test split of 'Move Description' category and collect the predictions from each of the methods under consideration. We hired two Anglophone (Lifetime HIT acceptance $\% > 80$) annotators for every human-evaluated test example. We additionally assess chess proficiency of the annotators using questions from the chess-QA dataset by (Cirik et al., 2015). Within each HIT, we ask two randomly selected questions from the chess-QA dataset. Finally we consider only those HITs wherein the annotator was able to answer the proficiency questions correctly.

Results: We conducted a human evaluation study for the *MoveDesc* subset of the data. As can be observed from Table 6, outputs from our method attain slightly more favorable scores compared to the ground truth commentaries. This shows that the predicted outputs from our model are not worse than ground truth on the said measures. This is in spite of the fact that the BLEU-4 score for the predicted outputs is only $\sim$ 2 w.r.t. the ground truth outputs. One reason for slightly lower performance of the ground truth outputs on the said measures is that some of the human written commentaries are either very ungrammatical or too concise. A more surprising observation is that around 30% of human written ground truth outputs were also marked as not valid for given board move. On inspection, it seems that commentary often contains extraneous game information beyond that of move alone, which indicates that an ideal comparison should be over commentary for an entire game, although this is beyond the scope of the current work.

The inter-annotator agreement for our experiments (Cohens κ (Cohen, 1968)) is 0.45 for Q1 and 0.32 for Q2. We notice some variation in κ coefficients across different systems. While TEMP and GAC responses had a 0.5-0.7 coefficient range, the responses for CLM had a much lower coefficient. In our setup, each HIT consists of 7 comments, one from each system. For Q3 (fluency), which is on an ordinal scale, we measure rank-order consistency between the responses of the two annotators of a HIT. Mean Kendall τ (Kendall, 1938) across all HITs was found to be 0.39.

To measures significance of results, we perform bootstrap tests on 1000 subsets of size 50 with a significance threshold of $p = 0.05$ for each pair of systems. For Q1, we observe that GAC(M), GAC(M+T) and GAC(M+T+S) methods are significantly better than baselines NN and GAC-sparse. We find that neither of GAC(M+T) and GT significantly outperform each other on Q1 as well as Q2. But we do find that GAC(M+T) does better than GAC(M) on both Q1 and Q2. For fluency scores, we find that GAC(M+T) is more fluent than GT, NN , GAC-sparse, GAC(M). Neither of GAC(M) and GAC(M+T+S) is significantly more fluent than the other.

5 Related Work

NLG research has a long history, with systems ranging from completely rule-based to learning-based ones (Reiter et al., 2005, 2003a), which have had both practical successes (Reiter et al., 2005) and failures (Reiter et al., 2003a). Recently, there have been numerous works which propose text generation given structured records, biographies (Lebret et al., 2016), recipes (Yang et al., 2016; Kiddon et al., 2016), etc. A key difference between generation given a game state compared to these inputs is that the game state is an evolving description at a point in a process, as opposed

[7]Human Intelligence Task

Question	GT	GAC (M)	GAC (MT)	GAC (MTS)	GAC -sparse	TEMP	NN
Is commentary correct for the given move? (%Yes)	70.4	42.3	64.8	67.6	56.3	91.5	52.1
Can the move be inferred from the commentary? (%Yes)	45.1	25.3	42.3	36.7	40.8	92.9	42.3
Fluency (scale of (least)1 - 5(most)) Mean (Std. dev.)	4.03 (1.31)	4.15 (1.20)	4.44 (1.02)	4.54 (0.89)	4.15 (1.26)	4.69 (0.64)	3.72 (1.36)

Table 6: Human study results on *MoveDesc* data category. Outputs from GAC are in general better than ground truth, NN and GAC-sparse. TEMP outperforms other methods, though as shown earlier, outputs from TEMP lack diversity.

to recipes (which are independent of each other), records (which are static) and biographies (which are one per person, and again independent). Moreover, our proposed method effectively uses various types of semantic and pragmatic information about the game state.

In this paper we have introduced a new large-scale data for game commentary generation. The commentaries cover a variety of aspects like move description, quality of move, and alternative moves. This leads to a content selection challenge, similar to that noted in Wiseman et al. (2017). Unlike Wiseman et al. (2017), our focus is on generating commentary for individual moves in a game, as opposed to game summaries from aggregate statistics as in their task.

One of the first NLG datasets was the SUMTIME-METEO (Reiter et al., 2005) corpus with ≈ 500 record-text pairs for technical weather forecast generation. Liang et al (2009) worked on common weather forecast generation using the WEATHERGOV dataset, which has $\approx$ 10K record-text pairs. A criticism of WEATHERGOV dataset (Reiter, 2017) is that weather records themselves may have used templates and rules with optional human post-editing. There have been prior works on generating commentary for ROBOCUP matches (Chen and Mooney, 2008; Mei et al., 2015). The ROBOCUP dataset, however, is collected from 4 games and contains about 1K events in total. Our dataset is two orders of magnitude larger than the ROBOCUP dataset, and we hope that it provides a promising setting for future NLG research.

6 Conclusions

In this paper, we curate a dataset for the task of chess commentary generation and propose methods to perform generation on this dataset. Our proposed method effectively utilizes information related to the rules and pragmatics of the game. A human evaluation study judges outputs from the proposed methods to be as good as human written commentary texts for 'Move Description' subset of the data.

Our dataset also contains multi-move-single commentary pairs in addition to single move-single commentary pairs. Generating commentary for such multi-moves is a potential direction for future work. We anticipate this task to require even deeper understanding of the game pragmatics than the single move-single commentary case.

Recent work (Silver et al., 2016) has proposed reinforcement learning based game-playing agents which learn to play board games from scratch, learning end-to-end from both recorded games and self-play. An interesting point to explore is whether such pragmatically trained game state representations can be leveraged for the task of game commentary generation.

Acknowledgements

We thank Volkan Cirik, Daniel Clothiaux, Hiroaki Hayashi and anonymous reviewers for providing valuable comments and feedback.

References

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 128–135.

Volkan Cirik, Louis-Philippe Morency, and Eduard Hovy. 2015. Chess q&a: Question Answering on Chess Games. In *Reasoning, Attention, Memory (RAM) Workshop, Neural Information Processing Systems*.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70(4):213.

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

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Learning a game commentary generator with grounded move expressions. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*. IEEE, pages 177–184.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3128–3137.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30(1/2):81–93.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 329–339.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771* .

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, pages 91–99.

Jen-Wen Liao and Jason S Chang. 1990. Computer Generation of Chinese Commentary on Othello Games. In *Proceedings of Rocling III Computational Linguistics Conference III*. pages 393–415.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, pages 740–755.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* .

Saad Mahamood and Ehud Reiter. 2012. Working with clinicians to improve a patient-information NLG system. In *Proceedings of the Seventh International Natural Language Generation Conference*. Association for Computational Linguistics, pages 100–104.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838* .

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2241–2252. https://www.aclweb.org/anthology/D17-1238.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011a. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011b. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Ehud Reiter. 1995. NLG vs. templates. *arXiv preprint cmp-lg/9504013* .

Ehud Reiter. 2017. You Need to Understand Your Corpora - the Weathergov Example. *Blogpost* - `https://ehudreiter.com/2017/05/09/weathergov/`.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics* 35(4):529–558.

Ehud Reiter, Roma Robertson, and Liesl M Osman. 2003a. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence* 144(1-2):41–58.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169.

Ehud Reiter, Somayajulu G Sripada, and Roma Robertson. 2003b. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research* 18:491–516.

Aleksander Sadikov, Martin Moina, Matej Guid, Jana Krivec, and Ivan Bratko. 2006. Automated chess tutor. In *International Conference on Computers and Games*. Springer, pages 13–25.

Claude E Shannon. 1951. Prediction and entropy of printed English. *Bell Labs Technical Journal* 30(1):50–64.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587):484–489.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 4566–4575.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-Document Generation. *arXiv preprint arXiv:1707.08052* .

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-Aware Language Models. *arXiv preprint arXiv:1611.01628* .

From Credit Assignment to Entropy Regularization: Two New Algorithms for Neural Sequence Prediction

Zihang Dai[*], **Qizhe Xie**[*], **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
`{dzihang, qizhex, hovy}@cs.cmu.edu`

Abstract

In this work, we study the credit assignment problem in reward augmented maximum likelihood (RAML) learning, and establish a theoretical equivalence between the token-level counterpart of RAML and the entropy regularized reinforcement learning. Inspired by the connection, we propose two sequence prediction algorithms, one extending RAML with fine-grained credit assignment and the other improving Actor-Critic with a systematic entropy regularization. On two benchmark datasets, we show the proposed algorithms outperform RAML and Actor-Critic respectively, providing new alternatives to sequence prediction.

1 Introduction

Modeling and predicting discrete sequences is the central problem to many natural language processing tasks. In the last few years, the adaption of recurrent neural networks (RNNs) and the sequence-to-sequence model (seq2seq) (Sutskever et al., 2014; Bahdanau et al., 2014) has led to a wide range of successes in conditional sequence prediction, including machine translation (Sutskever et al., 2014; Bahdanau et al., 2014), automatic summarization (Rush et al., 2015), image captioning (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Xu et al., 2015) and speech recognition (Chan et al., 2016).

Despite the distinct evaluation metrics for the aforementioned tasks, the standard training algorithm has been the same for all of them. Specifically, the algorithm is based on maximum likelihood estimation (MLE), which maximizes the log-likelihood of the "ground-truth" sequences empirically observed.[1]

While largely effective, the MLE algorithm has two obvious weaknesses. Firstly, the MLE training ignores the information of the task specific metric. As a result, the potentially large discrepancy between the log-likelihood during training and the task evaluation metric at test time can lead to a suboptimal solution. Secondly, MLE can suffer from the exposure bias, which refers to the phenomenon that the model is never exposed to its own failures during training, and thus cannot recover from an error at test time. Fundamentally, this issue roots from the difficulty in statistically modeling the exponentially large space of sequences, where most combinations cannot be covered by the observed data.

To tackle these two weaknesses, there have been various efforts recently, which we summarize into two broad categories:

- A widely explored idea is to directly optimize the task metric for sequences produced by the model, with the specific approaches ranging from minimum risk training (MRT) (Shen et al., 2015) and learning as search optimization (LaSO) (Daumé III and Marcu, 2005; Wiseman and Rush, 2016) to reinforcement learning (RL) (Ranzato et al., 2015; Bahdanau et al., 2016). In spite of the technical differences, the key component to make these training algorithms *practically efficient* is often a delicate credit assignment scheme, which transforms the sequence-level signal into dedicated smaller units (e.g., token-level or chunk-level), and allocates them to specific decisions, allowing for efficient optimization with a much lower variance. For instance, the beam search optimiza-

[*] Equal contribution.

[1] In this work, we use the terms "ground-truth" and "reference" to refer to the empirical observations interchangeably.

tion (BSO) (Wiseman and Rush, 2016) utilizes the position of margin violations to produce signals to the specific chunks, while the actor-critic (AC) algorithm (Bahdanau et al., 2016) trains a critic to enable token-level signals.

- Another alternative idea is to construct a task metric dependent target distribution, and train the model to match this task-specific target instead of the empirical data distribution. As a typical example, the reward augmented maximum likelihood (RAML) (Norouzi et al., 2016) defines the target distribution as the exponentiated pay-off (sequence-level reward) distribution. This way, RAML not only can incorporate the task metric information into training, but it can also alleviate the exposure bias by exposing imperfect outputs to the model. However, RAML only works on the sequence-level training signal.

In this work, we are intrigued by the question whether it is possible to incorporate the idea of fine-grained credit assignment into RAML. More specifically, inspired by the token-level signal used in AC, we aim to find the token-level counterpart of the sequence-level RAML, i.e., defining a token-level target distribution for each auto-regressive conditional factor to match. Motived by the question, we first formally define the desiderata the token-level counterpart needs to satisfy and derive the corresponding solution (§2). Then, we establish a theoretical connection between the derived token-level RAML and entropy regularized RL (§3). Motivated by this connection, we propose two algorithms for neural sequence prediction, where one is the token-level extension to RAML, and the other a RAML-inspired improvement to the AC (§4). We empirically evaluate the two proposed algorithms, and show different levels of improvement over the corresponding baseline. We further study the importance of various techniques used in our experiments, providing practical suggestions to readers (§6).

2 Token-level Equivalence of RAML

We first introduce the notations used throughout the paper. Firstly, capital letters will denote random variables and lower-case letters are the values to take. As we mainly focus on conditional sequence prediction, we use $\mathbf{x}$ for the conditional input, and $\mathbf{y}$ for the target sequence. With $\mathbf{y}$ denoting a sequence, $\mathbf{y}_i^j$ then denotes the subsequence from position i to j inclusively, while y_t denotes the single value at position t. Also, we use $|\mathbf{y}|$ to indicate the length of the sequence. To emphasize the ground-truth data used for training, we add superscript * to the input and target, i.e., $\mathbf{x}^*$ and $\mathbf{y}^*$. In addition, we use $\mathcal{Y}$ to denote the set of all possible sequences with one and only one $\texttt{eos}$ symbol at the end, and $\mathcal{W}$ to denote the set of all possible symbols in a position. Finally, we assume length of sequences in $\mathcal{Y}$ is bounded by T.

2.1 Background: RAML

As discussed in §1, given a ground-truth pair $(\mathbf{x}^*, \mathbf{y}^*)$, RAML defines the target distribution using the exponentiated pay-off of sequences, i.e.,

$$P_R(\mathbf{y} \mid \mathbf{x}^*, \mathbf{y}^*) = \frac{\exp\left(R(\mathbf{y}; \mathbf{y}^*)/\tau\right)}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp\left(R(\mathbf{y}'; \mathbf{y}^*)/\tau\right)}, \quad (1)$$

where $R(\mathbf{y}; \mathbf{y}^*)$ is the sequence-level reward, such as BLEU score, and τ is the temperature hyper-parameter controlling the sharpness. With the definition, the RAML algorithm simply minimizes the cross entropy (CE) between the target distribution and the model distribution $P_\theta(\mathbf{Y} \mid \mathbf{x}^*)$, i.e.,

$$\min_\theta \mathrm{CE}\left(P_R(\mathbf{Y} \mid \mathbf{x}^*, \mathbf{y}^*) \| P_\theta(\mathbf{Y} \mid \mathbf{x}^*)\right). \quad (2)$$

Note that, this is quite similar to the MLE training, except that the target distribution is different. With the particular choice of target distribution, RAML not only makes sure the ground-truth reference remains the mode, but also allows the model to explore sequences that are not exactly the same as the reference but have relatively high rewards.

Compared to algorithms trying to directly optimize task metric, RAML avoids the difficulty of tracking and sampling from the model distribution that is consistently changing. Hence, RAML enjoys a much more stable optimization without the need of pretraining. However, in order to optimize the RAML objective (Eqn. (2)), one needs to sample from the exponentiated pay-off distribution, which is quite challenging in practice. Thus, importance sampling is often used (Norouzi et al., 2016; Ma et al., 2017). We leave the details of the practical implementation to Appendix B.1.

2.2 Token-level Target Distribution

Despite the appealing properties, RAML only operates on the sequence-level reward. As a result, the reward gap between any two sequences cannot be attributed to the responsible decisions precisely, which often leads to a low sample efficiency. Ideally, since we rely on the auto-regressive factorization $P_\theta(\mathbf{y} \mid \mathbf{x}^*) = \prod_{t=1}^{|\mathbf{y}|} P_\theta(y_t \mid \mathbf{y}_1^{t-1}, \mathbf{x}^*)$, the optimization would be much more efficient if we have the target distribution for each token-level factor $P_\theta(Y_t \mid \mathbf{y}_1^{t-1}, \mathbf{x}^*)$ to match. Conceptually, this is exactly how the AC algorithm improves upon the vanilla sequence-level REINFORCE algorithm (Ranzato et al., 2015).

With this idea in mind, we set out to find such a token-level target. Firstly, we assume the token-level target shares the form of a Boltzmann distribution but parameterized by some unknown negative energy function Q_R, i.e.,[2]

$$P_{Q_R}(y_t \mid \mathbf{y}_1^{t-1}, \mathbf{y}^*) = \frac{\exp\left(Q_R(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*)/\tau\right)}{\sum_{w \in \mathcal{W}} \exp\left(Q_R(\mathbf{y}_1^{t-1}, w; \mathbf{y}^*)/\tau\right)}. \tag{3}$$

Intuitively, $Q_R(\mathbf{y}_1^{t-1}, w; \mathbf{y}^*)$ measures how much *future* pay-off one can expect if w is generated, given the current status $\mathbf{y}_1^{t-1}$ and the reference $\mathbf{y}^*$. This quantity highly resembles the action-value function (Q-function) in reinforcement learning. As we will show later, it is indeed the case.

Before we state the desiderata for Q_R, we need to extend the definition of R in order to evaluate the goodness of an unfinished partial prediction, i.e., sequences without an $\texttt{eos}$ suffix. Let $\mathcal{Y}^-$ be the set of unfinished sequences, following Bahdanau et al. (2016), we define the pay-off function R for a partial sequence $\hat{\mathbf{y}} \in \mathcal{Y}^-, |\hat{\mathbf{y}}| < T$ as

$$R(\hat{\mathbf{y}}; \mathbf{y}^*) = R(\hat{\mathbf{y}} + \texttt{eos}; \mathbf{y}^*), \tag{4}$$

where the $+$ indicates string concatenation.

With the extension, we are ready to state two requirements for Q_R:

1. **Marginal match**: For P_{Q_R} to be the token-level equivalence of P_R, the sequence-level marginal distribution induced by P_{Q_R} must match P_R, i.e., for any $\mathbf{y} \in \mathcal{Y}$,

$$\prod_{t=1}^{|\mathbf{y}|} P_{Q_R}(y_t \mid \mathbf{y}_1^{t-1}) = P_R(\mathbf{y}). \tag{5}$$

   Note that there are infinitely many Q_R's satisfying Eqn. (5), because adding any constant value to Q_R does not change the Boltzmann distribution, known as shift-invariance w.r.t. the energy.

2. **Terminal condition**: Secondly, let's consider the value of Q_R when emitting an $\texttt{eos}$ symbol to immediately terminate the generation. As mentioned earlier, Q_R measures the expected future pay-off. Since the emission of $\texttt{eos}$ ends the generation, the future pay-off can only come from the immediate increase of the pay-off. Thus, we require Q_R to be the incremental pay-off when producing $\texttt{eos}$, i.e.

$$Q_R(\hat{\mathbf{y}}, \texttt{eos}; \mathbf{y}^*) = R(\hat{\mathbf{y}} + \texttt{eos}; \mathbf{y}^*) - R(\hat{\mathbf{y}}; \mathbf{y}^*), \tag{6}$$

   for any $\hat{\mathbf{y}} \in \mathcal{Y}^-$. Since Eqn. (6) enforces the absolute of Q_R at a point, it also solves the ambiguity caused by the shift-invariance property.

Based on the two requirements, we can derive the form Q_R, which is summarized by Proposition 1.

Proposition 1. *P_{Q_R} and Q_R satisfy requirements (5) and (6) if and only if for any ground-truth pair $(\mathbf{x}^*, \mathbf{y}^*)$ and any sequence prediction $\mathbf{y} \in \mathcal{Y}$,*

$$\begin{aligned} Q_R(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) &= R(\mathbf{y}_1^t; \mathbf{y}^*) - R(\mathbf{y}_1^{t-1}; \mathbf{y}^*) \\ &\quad + \tau \log \sum_{w \in \mathcal{W}} \exp\left(Q_R(\mathbf{y}_1^t, w; \mathbf{y}^*)/\tau\right), \end{aligned} \tag{7}$$

when $t < |\mathbf{y}|$, and otherwise, i.e., when $t = |\mathbf{y}|$

$$Q_R(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) = R(\mathbf{y}_1^t; \mathbf{y}^*) - R(\mathbf{y}_1^{t-1}; \mathbf{y}^*). \tag{8}$$

Proof. See Appendix A.1. □

Note that, instead of giving an explicit form for the token-level target distribution, Proposition 1 only provides an equivalent condition in the form of an implicit recursion. Thus, we haven't obtained a practical algorithm yet. However, as we will discuss next, the recursion has a deep connection to entropy regularized RL, which ultimately inspires our proposed algorithms.

3 Connection to Entropy-regularized RL

Before we dive into the connection, we first give a brief review of the entropy-regularized RL. For an in-depth treatment, we refer readers to (Ziebart, 2010; Schulman et al., 2017).

3.1 Background: Entropy-regularized RL

Following the standard convention of RL, we denote a Markov decision process (MDP) by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p_s, r, \gamma)$, where $\mathcal{S}, \mathcal{A}, p_s, r, \gamma$ are the state space, action space, transition probability, reward function and discounting factor respectively.[3]

[2] To avoid clutter, the conditioning on $\mathbf{x}^*$ will be omitted in the sequel, assuming it's clear from the context.

[3] In sequence prediction, we are only interested in the periodic (finite horizon) case.

Based on the notation, the goal of entropy-regularized RL augments is to learn a policy $\pi(a_t \mid s_t)$ which maximizes the discounted expected future return and causal entropy (Ziebart, 2010), i.e.,

$$\max_{\pi} \sum_t \mathop{\mathbb{E}}_{s_t \sim \rho_s, a_t \sim \pi(\cdot \mid s_t)} \gamma^{t-1}[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t))],$$

where $\mathcal{H}$ denotes the entropy and α is a hyper-parameter controlling the relative importance between the reward and the entropy. Intuitively, compared to standard RL, the extra entropy term encourages exploration and promotes multi-modal behaviors. Such properties are highly favorable in a complex environment.

Given an entropy-regularized MDP, for any fixed policy π, the state-value function $V^\pi(s)$ and the action-value function Q^π can be defined as

$$\begin{aligned} V^\pi(s) &= \mathop{\mathbb{E}}_{a \sim \pi(\cdot \mid s)} [Q^\pi(s,a)] + \alpha \mathcal{H}(\pi(\cdot \mid s)), \\ Q^\pi(s,a) &= r(s,a) + \mathop{\mathbb{E}}_{s' \sim \rho_s} [\gamma V^\pi(s')]. \end{aligned} \tag{9}$$

With the definitions above, it can further be proved (Ziebart, 2010; Schulman et al., 2017) that the optimal state-value function V^*, the action-value function Q^* and the corresponding optimal policy π^* satisfy the following equations

$$V^*(s) = \alpha \log \sum_{a \in \mathcal{A}} \exp\left(Q^*(s,a)/\alpha\right), \tag{10}$$

$$Q^*(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \rho_s} [V^*(s')], \tag{11}$$

$$\pi^*(a \mid s) = \frac{\exp\left(Q^*(s,a)/\alpha\right)}{\sum_{a' \in \mathcal{A}} \exp\left(Q^*(s,a')/\alpha\right)}. \tag{12}$$

Here, Eqn. (10) and (11) are essentially the entropy-regularized counterparts of the optimal Bellman equations in standard RL. Following previous literature, we will refer to Eqn. (10) and (11) as the optimal *soft* Bellman equations, and the V^* and Q^* as optimal *soft* value functions.

3.2 An RL Equivalence of the Token-level RAML

To reveal the connection, it is convenient to define the incremental pay-off

$$r(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) = R(\mathbf{y}_1^t; \mathbf{y}^*) - R(\mathbf{y}_1^{t-1}; \mathbf{y}^*), \tag{13}$$

and the last term of Eqn. (7) as

$$V_R(\mathbf{y}_1^t; \mathbf{y}^*) = \tau \log \sum_{w \in \mathcal{W}} \exp\left(Q_R(\mathbf{y}_1^t, w; \mathbf{y}^*)/\tau\right) \tag{14}$$

Substituting the two definitions into Eqn. (7), the recursion simplifies as

$$Q_R(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) = r(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) + V_R(\mathbf{y}_1^t; \mathbf{y}^*). \tag{15}$$

Now, it is easy to see that the Eqn. (14) and (15), which are derived from the token-level RAML, highly resemble the optimal soft Bellman equations (10) and (11) in entropy-regularized RL. The following Corollary formalizes the connection.

Corollary 1. *For any ground-truth pair $(\mathbf{x}^*, \mathbf{y}^*)$, the recursion specified by Eqn. (13), (14) and (15) is equivalent to the optimal soft Bellman equation of a "deterministic" MDP in entropy-regularized reinforcement learning, denoted as $\mathcal{M}_R$, where*

- *the state space $\mathcal{S}$ corresponds to $\mathcal{Y}^-$,*
- *the action space $\mathcal{A}$ corresponds to $\mathcal{W}$,*
- *the transition probability ρ_s is a deterministic process defined by string concatenation*
- *the reward function r corresponds to the incremental pay-off defined in Eqn. (13),*
- *the discounting factor $\gamma = 1$,*
- *the entropy hyper-parameter $\alpha = \tau$,*
- *and a period terminates either when* eos *is emitted or when its length reaches T and we enforce the generation of* eos.

Moreover, the optimal soft value functions V^* and Q^* of the MDP exactly match the V_R and Q_R defined by Eqn. (14) and (15) respectively. The optimal policy π^* is hence equivalent to the token-level target distribution P_{Q_R}.

Proof. See Appendix A.1. □

The connection established by Corollary 1 is quite inspiring:

- Firstly, it provides a rigorous and generalized view of the connection between RAML and entropy-regularized RL. In the original work, Norouzi et al. (2016) point out RAML can be seen as reversing the direction of KL $(P_\theta \| P_R)$, which is a sequence-level view of the connection. Now, with the equivalence between the token-level target P_{Q_R} and the optimal Q^*, it generalizes to matching the future action values consisting of both the reward and the entropy.
- Secondly, due to the equivalence, if we solve the optimal soft Q-function of the corresponding MDP, we directly obtain the token-level target distribution. This hints at a practical algorithm with token-level credit assignment.

- Moreover, since RAML is able to improve upon MLE by injecting entropy, the entropy-regularized RL counterpart of the standard AC algorithm should also lead to an improvement in a similar manner.

4 Proposed Algorithms

In this section, we explore the insights gained from Corollary 1 and present two new algorithms for sequence prediction.

4.1 Value Augmented Maximum Likelihood

The first algorithm we consider is the token-level extension of RAML, which we have been discussing since §2. As mentioned at the end of §2.2, Proposition 1 only gives an implicit form of Q_R, and so is the token-level target distribution P_{Q_R} (Eqn. (3)). However, thanks to Corollary 1, we now know that Q_R is the same as the optimal soft action-value function Q^* of the entropy-regularized MDP $\mathcal{M}_R$. Hence, by finding the Q^*, we will have access to P_{Q_R}.

At the first sight, it seems recovering Q^* is as difficult as solving the original sequence prediction problem, because solving Q^* from the MDP is essentially the same as learning the optimal policy for sequence prediction. However, it is not true because Q_R (i.e., P_{Q_R}) can condition on the correct reference $\mathbf{y}^*$. In contrast, the model distribution P_θ can only depend on $\mathbf{x}^*$. Therefore, the function approximator trained to recover Q^* can take $\mathbf{y}^*$ as input, making the estimation task much easier. Intuitively, when recovering Q^*, we are trying to train an ideal "oracle", which has access to the ground-truth reference output, to decide the best behavior (policy) given any arbitrary (good or not) state.

Thus, following the reasoning above, we first train a parametric function approximator Q_ϕ to search the optimal soft action value. In this work, for simplicity, we employ the Soft Q-learning algorithm (Schulman et al., 2017) to perform the policy *optimization*. In a nutshell, Soft Q-Learning is the entropy-regularized version of Q-Learning, an off-policy algorithm which minimizes the mean squared soft Bellman residual according to Eqn. (11). Specifically, given ground-truth pair $(\mathbf{x}^*, \mathbf{y}^*)$, for any trajectory $\mathbf{y} \in \mathcal{Y}$, the training objective is

$$\min_\phi \sum_{t=1}^{|\mathbf{y}|} \left[Q_\phi(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) - \hat{Q}_\phi(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) \right]^2, \quad (16)$$

where $\hat{Q}_\phi(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) = r(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) + V_\phi(\mathbf{y}_1^t; \mathbf{y}^*)$ is the one-step look-ahead target Q-value, and $V_\phi(\mathbf{y}_1^t; \mathbf{y}^*) = \tau \log \sum_{w \in \mathcal{W}} \exp\left(Q_\phi(\mathbf{y}_1^t, w; \mathbf{y}^*)/\tau\right)$ as defined in Eqn. (10). In the recent instantiation of Q-Learning (Mnih et al., 2015), to stabilize training, the target Q-value is often estimated by a separate slowly updated target network. In our case, as we have access to a significant amount of reference sequences, we find the target network not necessary. Thus, we directly optimize Eqn. (16) using gradient descent, and let the gradient flow through both $Q_\phi(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*)$ and $V_\phi(\mathbf{y}_1^t; \mathbf{y}^*)$ (Baird, 1995).

After the training of Q_ϕ converges, we fix the parameters of Q_ϕ, and optimize the cross entropy CE $\left(P_{Q_\phi} \| P_\theta\right)$ w.r.t. the model parameters θ, which is equivalent to[4]

$$\min_\theta \mathop{\mathbb{E}}_{\mathbf{y} \sim P_{Q_\phi}} \left[\sum_{t=1}^{|\mathbf{y}|} \mathrm{CE}\left(P_{Q_\phi}(Y_t \mid \mathbf{y}_1^{t-1}) \| P_\theta(Y_t \mid \mathbf{y}_1^{t-1})\right) \right]. \quad (17)$$

Compared to the of objective of RAML in Eqn. (2), having access to $P_{Q_\phi}(Y_t \mid \mathbf{y}_1^{t-1})$ allows us to provide a distinct token-level target for each conditional factor $P_\theta(Y_t \mid \mathbf{y}_1^{t-1})$ of the model. While directly sampling from P_R is practically infeasible (§2.1), having a parametric target distribution P_{Q_ϕ} makes it theoretically possible to sample from P_{Q_ϕ} and perform the optimization. However, empirically, we find the samples from P_{Q_ϕ} are not diverse enough (§6). Hence, we fall back to the same importance sampling approach (see Appendix B.2) as used in RAML.

Finally, since the algorithm utilizes the optimal soft action-value function to construct the token-level target, we will refer to it as value augmented maximum likelihood (VAML) in the sequel.

4.2 Entropy-regularized Actor Critic

The second algorithm follows the discussion at the end of §3.2, which is essentially an actor-critic algorithm based on the entropy-regularized MDP in Corollary 1. For this reason, we name the algorithm entropy-regularized actor critic (ERAC). As with standard AC algorithm, the training process interleaves the evaluation of current policy using the parametric critic Q_ϕ and the optimization of the actor policy π_θ given the current critic.

Critic Training. The critic is trained to perform policy *evaluation* using the temporal difference

[4]See Appendix A.2 for a detailed derivation.

learning (TD), which minimizes the TD error

$$\min_{\phi} \mathbb{E}_{\mathbf{y} \sim \pi_\theta} \sum_{t=1}^{|\mathbf{y}|} \left[Q_\phi(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) - \hat{Q}_{\bar{\phi}}(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) \right]^2 \tag{18}$$

where the TD target $\hat{Q}_{\bar{\phi}}$ is constructed based on fixed policy iteration in Eqn. (9), i.e.,

$$\begin{aligned} \hat{Q}_{\bar{\phi}}(\mathbf{y}_1^{t-1}, y_t; \mathbf{y}^*) &= r(\mathbf{y}_1^{t-1}, y_t) + \tau \mathcal{H}(\pi_\theta(\cdot \mid \mathbf{y}_1^t)) \\ &+ \sum_{w \in \mathcal{W}} \pi_\theta(w \mid \mathbf{y}_1^t) Q_{\bar{\phi}}(\mathbf{y}_1^t, w; \mathbf{y}^*). \end{aligned} \tag{19}$$

It is worthwhile to emphasize that the objective (18) trains the critic Q_ϕ to evaluate the current policy. Hence, it is entirely different from the objective (16), which is performing policy optimization by Soft Q-Learning. Also, the trajectories $\mathbf{y}$ used in (18) are sequences drawn from the actor policy π_θ, while objective (16) theoretically accepts any trajectory since Soft Q-Learning can be fully off-policy.[5] Finally, following Bahdanau et al. (2016), the TD target $\hat{Q}_{\bar{\phi}}$ in Eqn. (9) is evaluated using a target network, which is indicated by the bar sign above the parameters, i.e., $\bar{\phi}$. The target network is slowly updated by linearly interpolating with the up-to-date network, i.e., the update is $\bar{\phi} \leftarrow \beta\phi + (1-\beta)\bar{\phi}$ for β in $(0, 1)$ (Lillicrap et al., 2015).

We also adapt another technique proposed by Bahdanau et al. (2016), which smooths the critic by minimizing the "variance" of Q-values, i.e.,

$$\min_{\phi} \lambda_{\text{var}} \mathbb{E}_{\mathbf{y} \sim \pi_\theta} \sum_{t=1}^{|\mathbf{y}|} \sum_{w \in \mathcal{W}} \left[Q_\phi(\mathbf{y}_1^t, w; \mathbf{y}^*) - \bar{Q}_\phi(\mathbf{y}_1^t; \mathbf{y}^*) \right]^2$$

where $\bar{Q}_\phi(\mathbf{y}_1^t; \mathbf{y}^*) = \frac{1}{|\mathcal{W}|} \sum_{w' \in \mathcal{W}} Q_\phi(\mathbf{y}_1^t, w'; \mathbf{y}^*)$ is the mean Q-value, and λ_{var} is a hyper-parameter controlling the relative weight between the TD loss and the smooth loss.

Actor Training. Given the critic Q_ϕ, the actor gradient (to maximize the expected return) is given by the policy gradient theorem of the entropy-regularized RL (Schulman et al., 2017), which has the form

$$\begin{aligned} \mathbb{E}_{\mathbf{y} \sim \pi_\theta} \sum_{t=1}^{|\mathbf{y}|} \sum_{w \in \mathcal{W}} & \nabla_\theta \pi_\theta(w \mid \mathbf{y}_1^{t-1}) Q_\phi(\mathbf{y}_1^{t-1}, w; \mathbf{y}^*) \\ &+ \tau \nabla_\theta \mathcal{H}(\pi_\theta(\cdot \mid \mathbf{y}_1^{t-1})). \end{aligned} \tag{20}$$

Here, for each step t, we follow Bahdanau et al. (2016) to sum over the entire symbol set $\mathcal{W}$, instead of using the single sample estimation often seen in RL. Hence, no baseline is employed. It is worth mentioning that Eqn. (20) is *not* simply adding an entropy term to the standard policy gradient as in A3C (Mnih et al., 2016). The difference lies in that the critic Q_ϕ trained by Eqn. (18) additionally captures the *entropy from future steps*, while the $\nabla_\theta \mathcal{H}$ term only captures the entropy of the current step.

Finally, similar to (Bahdanau et al., 2016), we find it necessary to first pretrain the actor using MLE and then pretrain the critic before the actor-critic training. Also, to prevent divergence during actor-critic training, it is helpful to continue performing MLE training along with Eqn. (20), though using a smaller weight λ_{mle}.

5 Related Work

Task Loss Optimization and Exposure Bias Apart from the previously introduced RAML, BSO, Actor-Critic (§1), MIXER (Ranzato et al., 2015) also utilizes chunk-level signals where the length of chunk grows as training proceeds. In contrast, minimum risk training (Shen et al., 2015) directly optimizes sentence-level BLEU. As a result, it requires a large number (100) of samples per data to work well. To solve the exposure bias, scheduled sampling (Bengio et al., 2015) adopts a curriculum learning strategy to bridge the training and the inference. Professor forcing (Lamb et al., 2016) introduces an adversarial training mechanism to encourage the dynamics of the model to be the same at training time and inference time. For image caption, self-critic sequence training (SCST) (Rennie et al., 2016) extends the MIXER algorithm with an improved baseline based on the current model performance.

Entropy-regularized RL Entropy regularization been explored by early work in RL and inverse RL (Williams and Peng, 1991; Ziebart et al., 2008). Lately, Schulman et al. (2017) establish the equivalence between policy gradients and Soft Q-Learning under entropy-regularized RL. Motivated by the multi-modal behavior induced by entropy-regularized RL, Haarnoja et al. (2017) apply energy-based policy and Soft Q-Learning to continuous domain. Later, Nachum et al. (2017) proposes path consistency learning, which can be seen as a multi-step extension to Soft Q-Learning. More recently, in the domain of simulated control, Haarnoja et al. (2018) also consider the actor critic algorithm under the framework of en-

[5]Different from Bahdanau et al. (2016), we don't use a delayed actor network to collect trajectories for critic training.

tropy regularized reinforcement learning. Despite the conceptual similarity to ERAC presented here, Haarnoja et al. (2018) focuses on continuous control and employs the advantage actor critic variant as in (Mnih et al., 2016), while ERAC follows the Q actor critic as in (Bahdanau et al., 2016).

6 Experiments

6.1 Experiment Settings

In this work, we focus on two sequence prediction tasks: machine translation and image captioning. Due to the space limit, we only present the information necessary to compare the empirical results at this moment. For a more detailed description, we refer readers to Appendix B and the code[6].

Machine Translation Following Ranzato et al. (2015), we evaluate on IWSLT 2014 German-to-English dataset (Mauro et al., 2012). The corpus contains approximately $153K$ sentence pairs in the training set. We follow the pre-processing procedure used in (Ranzato et al., 2015).

Architecture wise, we employ a seq2seq model with dot-product attention (Bahdanau et al., 2014; Luong et al., 2015), where the encoder is a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with each direction being size 128, and the decoder is another LSTM of size 256. Moreover, we consider two variants of the decoder, one using the input feeding technique (Luong et al., 2015) and the other not.

For all algorithms, the sequence-level BLEU score is employed as the pay-off function R, while the corpus-level BLEU score (Papineni et al., 2002) is used for the final evaluation. The sequence-level BLEU score is scaled up by the sentence length so that the scale of the immediate reward at each step is invariant to the length.

Image Captioning For image captioning, we consider the MSCOCO dataset (Lin et al., 2014). We adapt the same preprocessing procedure and the train/dev/test split used by Karpathy and Fei-Fei (2015).

The NIC (Vinyals et al., 2015) is employed as the baseline model, where a feature vector of the image is extracted by a pre-trained CNN and then used to initialize the LSTM decoder. Different from the original NIC model, we employ a pre-trained 101-layer ResNet (He et al., 2016) rather than a GoogLeNet as the CNN encoder.

[6]`https://github.com/zihangdai/ERAC-VAML`

For training, each image-caption pair is treated as an i.i.d. sample, and sequence-level BLEU score is used as the pay-off. For testing, the standard multi-reference BLEU4 is used.

6.2 Comparison with the Direct Baseline

Firstly, we compare ERAC and VAML with their corresponding direct baselines, namely AC (Bahdanau et al., 2016) and RAML (Norouzi et al., 2016) respectively. As a reference, the performance of MLE is also provided.

Due to non-neglected performance variance observed across different runs, we run each algorithm for 9 times with different random seeds,[7] and report the average performance, the standard deviation and the performance range (min, max).

Machine Translation The results on MT are summarized in the left half of Tab. 1. Firstly, all four advanced algorithms significantly outperform the MLE baseline. More importantly, both VAML and ERAC improve upon their direct baselines, RAML and AC, by a clear margin on average. The result suggests the two proposed algorithms both well combine the benefits of a delicate credit assignment scheme and the entropy regularization, achieving improved performance.

Image Captioning The results on image captioning are shown in the right half of Tab. 1. Despite the similar overall trend, the improvement of VAML over RAML is smaller compared to that in MT. Meanwhile, the improvement from AC to ERAC becomes larger in comparison. We suspect this is due to the multi-reference nature of the MSCOCO dataset, where a larger entropy is preferred. As a result, the explicit entropy regularization in ERAC becomes immediately fruitful. On the other hand, with multiple references, it can be more difficult to learn a good oracle Q^* (Eqn. (15)). Hence, the token-level target can be less accurate, resulting in smaller improvement.

6.3 Comparison with Existing Work

To further evaluate the proposed algorithms, we compare ERAC and VAML with the large body of existing algorithms evaluated on IWSTL 2014. As a note of caution, previous works don't employ the exactly same architectures (e.g. number of layers, hidden size, attention type, etc.). Despite that,

[7]For AC, ERAC and VAML, 3 different critics are trained first, and each critic is then used to train 3 actors.

Algorithm	MT (w/o input feeding) Mean	Min	Max	MT (w/ input feeding) Mean	Min	Max	Image Captioning Mean	Min	Max
MLE	27.01 $\pm$ 0.20	26.72	27.27	28.06 $\pm$ 0.15	27.84	28.22	29.54 $\pm$ 0.21	29.27	29.89
RAML	27.74 $\pm$ 0.15	27.47	27.93	28.56 $\pm$ 0.15	28.35	28.80	29.84 $\pm$ 0.21	29.50	30.17
VAML	**28.16 $\pm$ 0.11**	**28.00**	**28.26**	**28.84 $\pm$ 0.10**	**28.62**	**28.94**	**29.93 $\pm$ 0.22**	**29.51**	**30.24**
AC	28.04 $\pm$ 0.05	27.97	28.10	29.05 $\pm$ 0.06	28.95	29.16	30.90 $\pm$ 0.20	30.49	31.16
ERAC	**28.30 $\pm$ 0.06**	**28.25**	**28.42**	**29.31 $\pm$ 0.04**	**29.26**	**29.36**	**31.44 $\pm$ 0.22**	**31.07**	**31.82**

Table 1: Test results on two benchmark tasks. Bold faces highlight the best in the corresponding category.

for VAML and ERAC, we use an architecture that is most similar to the majority of previous works, which is the one described in §6.1 with input feeding.

Based on the setting, the comparison is summarized in Table 2.[8] As we can see, both VAML and ERAC outperform previous methods, with ERAC leading the comparison with a significant margin. This further verifies the effectiveness of the two proposed algorithms.

Algorithm	BLEU
MIXER (Ranzato et al., 2015)	20.73
BSO (Wiseman and Rush, 2016)	27.9
Q(BLEU) (Li et al., 2017)	28.3
AC (Bahdanau et al., 2016)	28.53
RAML (Ma et al., 2017)	28.77
VAML	28.94
ERAC	**29.36**

Table 2: Comparison with existing algorithms on IWSTL 2014 dataset for MT. All numbers of previous algorithms are from the original work.

6.4 Ablation Study

Due to the overall excellence of ERAC, we study the importance of various components of it, hopefully offering a practical guide for readers. As the input feeding technique largely slows down the training, we conduct the ablation based on the model variant *without* input feeding.

Firstly, we study the importance of two techniques aimed for training stability, namely the target network and the smoothing technique (§4.2). Based on the MT task, we vary the update speed β of the target critic, and the λ_{var}, which controls the

[8]For a more detailed comparison of performance together with the model architectures, see Table 7 in Appendix C.

λ_{var} \ β	0.001	0.01	0.1	1
0	27.91	26.27†	28.88	27.38†
0.001	**29.41**	29.26	29.32	27.44

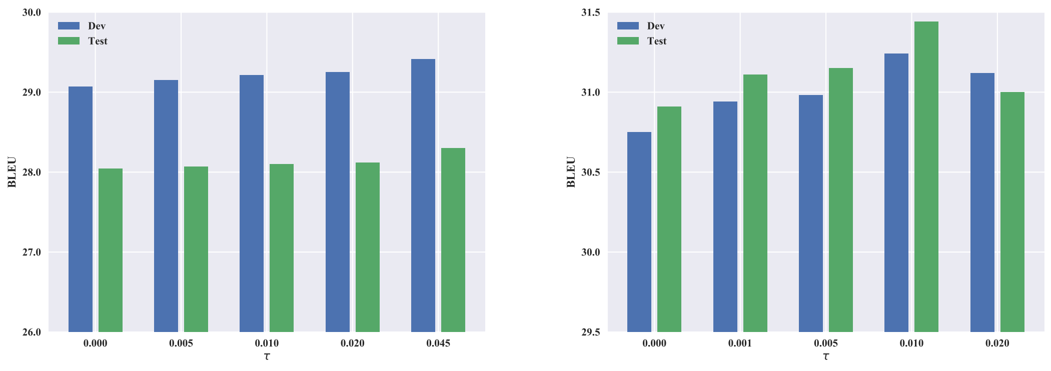
Table 3: Average validation BLEU of ERAC. As a reference, the average BLEU is 28.1 for MLE. $\lambda_{\text{var}} = 0$ means not using the smoothing technique. $\beta = 1$ means not using a target network. † indicates excluding extreme values due to divergence.

strength of the smoothness regularization. The average *validation* performances of different hyper-parameter values are summarized in Tab. 3.

- Comparing the two rows of Tab. 3, the smoothing technique consistently leads to performance improvement across all values of τ. In fact, removing the smoothing objective often causes the training to diverge, especially when $\beta = 0.01$ and 1. But interestingly, we find the divergence does not happen if we update the target network a little bit faster ($\beta = 0.1$) or quite slowly ($\beta = 0.001$).

- In addition, even with the smoothing technique, the target network is still necessary. When the target network is not used ($\beta = 1$), the performance drops below the MLE baseline. However, as long as a target network is employed to ensure the training stability, the specific choice of target network update rate does not matter as much. Empirically, it seems using a slower ($\beta = 0.001$) update rate yields the best result.

Next, we investigate the effect of enforcing different levels of entropy by varying the entropy hyper-parameter τ. As shown in Fig. 1, it seems there is always a sweet spot for the level of entropy. On the one hand, posing an over strong en-



(a) Machine translation

(b) Image captioning

Figure 1: ERAC's average performance over multiple runs on two tasks when varying τ.

tropy regularization can easily cause the actor to diverge. Specifically, the model diverges when τ reaches 0.03 on the image captioning task or 0.06 on the machine translation task. On the other hand, as we decrease τ from the best value to 0, the performance monotonically decreases as well. This observation further verifies the effectiveness of entropy regularization in ERAC, which well matches our theoretical analysis.

Finally, as discussed in §4.2, ERAC takes the effect of future entropy into consideration, and thus is different from simply adding an entropy term to the standard policy gradient as in A3C (Mnih et al., 2016). To verify the importance of explicitly modeling the entropy from future steps, we compared ERAC with the variant that only applies the entropy regularization to the actor but not to the critic. In other words, the τ is set to 0 when performing policy evaluating according to Eqn. (4.2), while the τ for the entropy gradient in Eqn. (20) remains. The comparison result based on 9 runs on test set of IWSTL 2014 is shown in Table 4. As we can see, simply adding a local entropy gradient does not even improve upon the AC. This further verifies the difference between ERAC and A3C, and shows the importance of taking future entropy into consideration.

Algorithm	**Mean**	**Max**
ERAC	**28.30 ± 0.06**	**28.42**
ERAC w/o Future Ent.	28.06 ± 0.05	28.11
AC	28.04 ± 0.05	28.10

Table 4: Comparing ERAC with the variant without considering future entropy.

7 Discussion

In this work, motivated by the intriguing connection between the token-level RAML and the entropy-regularized RL, we propose two algorithms for neural sequence prediction. Despite the distinct training procedures, both algorithms combine the idea of fine-grained credit assignment and the entropy regularization, leading to positive empirical results.

However, many problems remain widely open. In particular, the oracle Q-function Q_ϕ we obtain is far from perfect. We believe the ground-truth reference contains sufficient information for such an oracle, and the current bottleneck lies in the RL algorithm. Given the numerous potential applications of such an oracle, we believe improving its accuracy will be a promising future direction.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Leemon Baird. 1995. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, Elsevier, pages 30–37.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1171–1179.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 4960–4964.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*. ACM, pages 169–176.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165* .

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

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

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Toward neural phrasebased machine translation .

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3128–3137.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549* .

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* .

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, pages 740–755.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Xuezhe Ma, Pengcheng Yin, Jingzhou Liu, Graham Neubig, and Eduard Hovy. 2017. Softmax q-distribution estimation for structured prediction: A theoretical interpretation for raml. *arXiv preprint arXiv:1705.07136* .

Cettolo Mauro, Girardi Christian, and Federico Marcello. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*. pages 261–268.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. pages 1928–1937.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. 2017. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*. pages 2772–2782.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*. pages 1723–1731.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563* .

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

John Schulman, Pieter Abbeel, and Xi Chen. 2017. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440* .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, pages 3156–3164.

Ronald J Williams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science* 3(3):241–268.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. pages 2048–2057.

Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*. Chicago, IL, USA, volume 8, pages 1433–1438.

DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension

Amrita Saha IBM Research
Rahul Aralikatte IBM Research
Mitesh M. Khapra IIT Madras
Karthik Sankaranarayanan IBM Research

{amrsaha4,rahul.a.r,kartsank}@in.ibm.com
miteshk@cse.iitm.ac.in

Abstract

We propose DuoRC, a novel dataset for Reading Comprehension (RC) that motivates several new challenges for neural approaches in language understanding beyond those offered by existing RC datasets. DuoRC contains 186,089 unique question-answer pairs created from a collection of 7680 pairs of movie plots where each pair in the collection reflects two versions of the same movie - one from Wikipedia and the other from IMDb - written by two different authors. We asked crowdsourced workers to create questions from one version of the plot and a different set of workers to extract or synthesize answers from the other version. This unique characteristic of DuoRC where questions and answers are created from different versions of a document narrating the same underlying story, ensures by design, that there is very little lexical overlap between the questions created from one version and the segments containing the answer in the other version. Further, since the two versions have different levels of plot detail, narration style, vocabulary, *etc*., answering questions from the second version requires deeper language understanding and incorporating external background knowledge. Additionally, the narrative style of passages arising from movie plots (as opposed to typical descriptive passages in existing datasets) exhibits the need to perform complex reasoning over events across multiple sentences. Indeed, we observe that state-of-the-art neural RC models which have achieved near human performance on the SQuAD dataset (Rajpurkar et al., 2016b), even when coupled with traditional NLP techniques to address the challenges presented in DuoRC exhibit very poor performance (F1 score of 37.42% on DuoRC v/s 86% on SQuAD dataset). This opens up several interesting research avenues wherein DuoRC could complement other RC datasets to explore novel neural approaches for studying language understanding.

1 Introduction

Natural Language Understanding is widely accepted to be one of the key capabilities required for AI systems. Scientific progress on this endeavor is measured through multiple tasks such as machine translation, reading comprehension, question-answering, and others, each of which requires the machine to demonstrate the ability to "comprehend" the given textual input (apart from other aspects) and achieve their task-specific goals. In particular, Reading Comprehension (RC) systems are required to "understand" a given text passage as input and then answer questions based on it. *It is therefore critical, that the dataset benchmarks established for the RC task keep progressing in complexity to reflect the challenges that arise in true language understanding, thereby enabling the development of models and techniques to solve these challenges.*

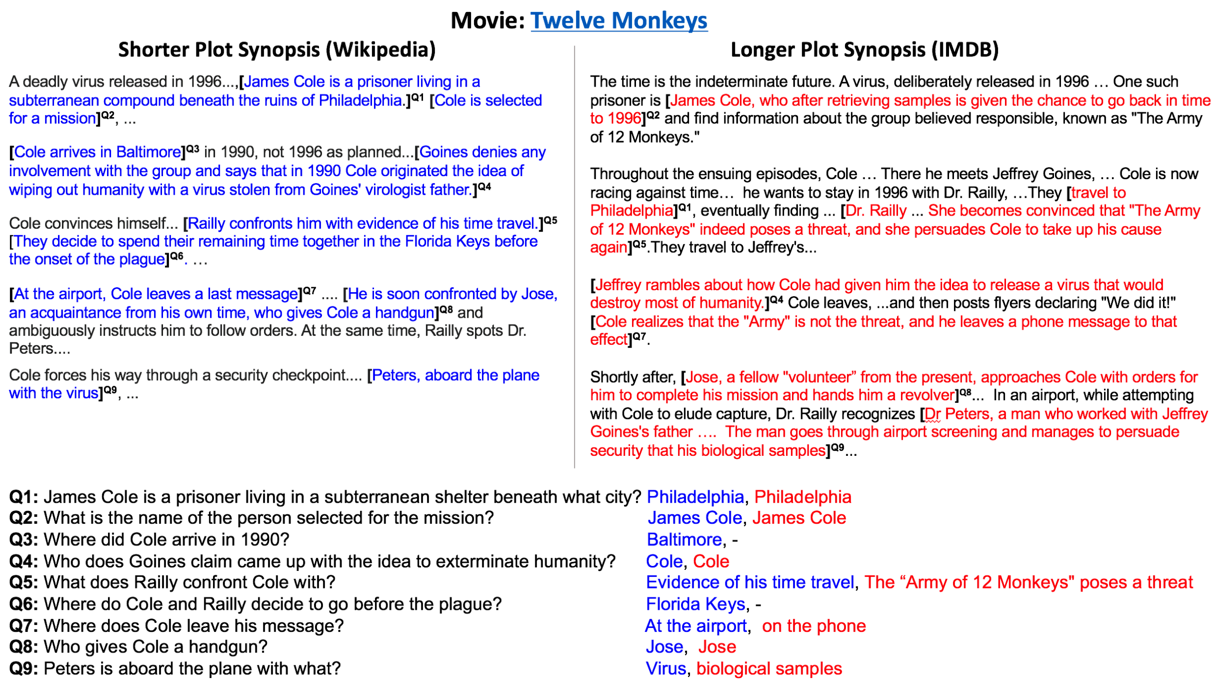
For RC in particular, there has been significant progress over the recent years with several benchmark datasets, the most popular of which are the SQuAD dataset (Rajpurkar et al., 2016a), TriviaQA (Joshi et al., 2017), MS MARCO (Nguyen et al., 2016), MovieQA (Tapaswi et al., 2016) and cloze-style datasets (Mostafazadeh et al., 2016; Onishi et al., 2016; Hermann et al., 2015). However, these benchmarks, owing to both the nature of the passages and the QA pairs to evaluate the RC task, have 2 primary limitations in studying language understanding: (i) Other than MovieQA, which is a small dataset of 15K QA pairs, all other large-scale RC datasets deal only with factual descriptive passages and not narratives (involving events with causality linkages that require reasoning and background knowledge) which is the case with a lot of real-world content such as story books, movies, news reports, etc. (ii) their questions possess a large lexical overlap with segments of the passage, or have a high noise level in QA pairs themselves. As demonstrated by recent work, this makes it easy for even simple keyword matching algorithms to achieve high accuracy (Weissenborn et al., 2017). In fact, these models have been shown to perform poorly in the presence of adversarially inserted sentences which have a high word overlap with the question but do not contain the answer (Jia and Liang, 2017). While this problem does not exist in TriviaQA it is admittedly noisy because of the use of distant supervision. Similarly, for cloze-style datasets, due to the automatic question generation process, it is very easy for current models to reach near human performance (Cui, 2017). This therefore limits the complexity in language understanding that a machine is required to demonstrate to do well on the RC task.

Motivated by these shortcomings and to push the state-of-the-art in language understanding in RC, in this paper we propose DuoRC, which specifically presents the following challenges beyond the existing datasets:

1. DuoRC is especially designed to contain a large number of questions with low lexical overlap between questions and their corresponding passages.
2. It requires the use of background and common-sense knowledge to arrive at the answer and go beyond the content of the passage itself.
3. It contains narrative passages from movie plots that require complex reasoning across multiple sentences to infer the answer.
4. Several of the questions in DuoRC, while seeming relevant, cannot actually be answered from the given passage, thereby requiring the machine to detect the *unanswerability* of questions.

In order to capture these four challenges, DuoRC contains QA pairs created from pairs of documents describing movie plots which were gathered as follows. Each document in a pair is a different version of the same movie plot written by different authors; one version of the plot is taken from the Wikipedia page of the movie whereas the other from its IMDb page (see Fig. 1 for portions of an example pair of plots from the movie "Twelve Monkeys"). We first showed crowd workers on Amazon Mechanical Turk (AMT) the *first* version of the plot and asked them to create QA pairs from it. We then showed the *second* version of the plot along with the questions created from the *first* version to a different set of workers on AMT and asked them to provide answers by reading the second version only. Since the two versions contain different levels of plot detail, narration style, vocabulary, etc., answering questions from the second version exhibits all of the four challenges mentioned above.

We now make several interesting observations from the example in Fig. 1. For 4 out of the 8 questions (Q1, Q2, Q4, and Q7), though the answers extracted from the two plots are exactly the same, the analysis required to arrive at this answer is very different in the two cases. In particular, for Q1 even though there is no explicit mention of *the prisoner living in a subterranean shelter* and hence no lexical overlap with the question, the workers were still able to infer that the answer is *Philadelphia* because that is the city to which James Cole travels to for his mission. Another interesting characteristic of this dataset is that for a few questions (Q6, Q8) alternative but valid answers are obtained from the second plot. Further, note the kind of complex reasoning required for answering Q8 where the machine needs to resolve coreferences over multiple sentences (*that man* refers to *Dr. Peters*) and use common sense knowledge that if an item clears an airport screening, then a person can likely board the plane with it. To re-emphasize, these examples exhibit the need for machines to demonstrate new capabilities in RC such as: (i) employing a knowledge graph (e.g. to know that Philadelphia is a city in Q1), (ii) common-sense knowledge (e.g., *clearing airport security* implies *boarding*) (iii) paraphrase/semantic understanding (e.g. revolver is a type of handgun in Q7) (iv) multiple-sentence inferencing across events in the passage including coreference resolution of named entities and nouns, and (v) educated guesswork when the question is not directly answerable but there are subtle hints in the passage (as in Q1). Finally, for quite a few questions, there wasn't sufficient information in the second plot to obtain their answers. In such cases, the workers marked the question as "unanswerable". This brings out a very important challenge for machines (detect *unanswerability* of questions)



Movie: Twelve Monkeys

Shorter Plot Synopsis (Wikipedia)

A deadly virus released in 1996...,[James Cole is a prisoner living in a subterranean compound beneath the ruins of Philadelphia.]^{Q1} [Cole is selected for a mission]^{Q2}, ...

[Cole arrives in Baltimore]^{Q3} in 1990, not 1996 as planned...[Goines denies any involvement with the group and says that in 1990 Cole originated the idea of wiping out humanity with a virus stolen from Goines' virologist father.]^{Q4}

Cole convinces himself... [Railly confronts him with evidence of his time travel.]^{Q5} [They decide to spend their remaining time together in the Florida Keys before the onset of the plague]^{Q6}. …

[At the airport, Cole leaves a last message]^{Q7} .... [He is soon confronted by Jose, an acquaintance from his own time, who gives Cole a handgun]^{Q8} and ambiguously instructs him to follow orders. At the same time, Railly spots Dr. Peters....

Cole forces his way through a security checkpoint.... [Peters, aboard the plane with the virus]^{Q9}, ...

Longer Plot Synopsis (IMDB)

The time is the indeterminate future. A virus, deliberately released in 1996 … One such prisoner is [James Cole, who after retrieving samples is given the chance to go back in time to 1996]^{Q2} and find information about the group believed responsible, known as "The Army of 12 Monkeys."

Throughout the ensuing episodes, Cole … There he meets Jeffrey Goines, … Cole is now racing against time… he wants to stay in 1996 with Dr. Railly, …They [travel to Philadelphia]^{Q1}, eventually finding ... [Dr. Railly ... She becomes convinced that "The Army of 12 Monkeys" indeed poses a threat, and she persuades Cole to take up his cause again]^{Q5}.They travel to Jeffrey's...

[Jeffrey rambles about how Cole had given him the idea to release a virus that would destroy most of humanity.]^{Q4} Cole leaves, ...and then posts flyers declaring "We did it!" [Cole realizes that the "Army" is not the threat, and he leaves a phone message to that effect]^{Q7}.

Shortly after, [Jose, a fellow "volunteer" from the present, approaches Cole with orders for him to complete his mission and hands him a revolver]^{Q8}... In an airport, while attempting with Cole to elude capture, Dr. Railly recognizes [Dr Peters, a man who worked with Jeffrey Goines's father …. The man goes through airport screening and manages to persuade security that his biological samples]^{Q9}...

Q1: James Cole is a prisoner living in a subterranean shelter beneath what city?	Philadelphia, Philadelphia
Q2: What is the name of the person selected for the mission?	James Cole, James Cole
Q3: Where did Cole arrive in 1990?	Baltimore, -
Q4: Who does Goines claim came up with the idea to exterminate humanity?	Cole, Cole
Q5: What does Railly confront Cole with?	Evidence of his time travel, The "Army of 12 Monkeys" poses a threat
Q6: Where do Cole and Railly decide to go before the plague?	Florida Keys, -
Q7: Where does Cole leave his message?	At the airport, on the phone
Q8: Who gives Cole a handgun?	Jose, Jose
Q9: Peters is aboard the plane with what?	Virus, biological samples

Figure 1: Example QA pairs obtained from the original movie plot and the paraphrased plot. The relevant spans needed for answering the corresponding question are highlighted in blue and red with the respective question numbers. Note that the span highlighting shown here is for illustrative purposes only and is not available in the dataset.

because a practical system should be able to know when it is not possible for it to answer a question given the data available to it, and in such cases, possibly delegate the task to a human instead.

Current RC systems built using existing datasets are far from possessing these capabilities to solve the above challenges. In Section 4, we seek to establish solid baselines for DuoRC employing state-of-the-art RC models coupled with a collection of standard NLP techniques to address few of the above challenges. Proposing novel neural models that solve all of the challenges in DuoRC is out of the scope of this paper. Our experiments demonstrate that when the existing state-of-the-art RC systems are trained and evaluated on DuoRC they perform poorly leaving a lot of scope for improvement and open new avenues for research in RC. Do note that this dataset is not a substitute for existing RC datasets but can be coupled with them to collectively address a large set of challenges in language understanding with RC (*the more the merrier*).

2 Related Work

Over the past few years, there has been a surge in datasets for Reading Comprehension. Most of these datasets differ in the manner in which questions and answers are created. For example, in SQuAD (Rajpurkar et al., 2016a), NewsQA (Trischler et al., 2016), TriviaQA (Joshi et al., 2017) and MovieQA (Tapaswi et al., 2016) the answers correspond to a span in the document. MS-MARCO uses web queries as questions and the answers are synthesized by workers from documents relevant to the query. On the other hand, in most cloze-style datasets (Mostafazadeh et al., 2016; Onishi et al., 2016) the questions are created automatically by deleting a word/entity from a sentence. There are also some datasets for RC with multiple choice questions (Richardson et al., 2013; Berant et al., 2014; Lai et al., 2017) where the task is to select one among k given candidate answers.

Another notable RC Dataset is NarrativeQA(s Koˇ ciský et al., 2018) which contains 40K QA pairs created from plot summaries of movies. It poses two tasks, where the first task involves reading the plot summaries from which the QA pairs were annotated and the second task is read the entire book or movie script (which is usually 60K words long) instead of the summary to answer the question. As acknowledged by the authors, while the first task is similar in scope to the previous datasets, the second task is at present, intractable for existing neural models, owing to the length of the passage. Due to the kind of the challenges presented by their second task, it is not comparable to our dataset and is much more futuristic in nature.

Given that there are already a few datasets for RC, a natural question to ask is "*Do we really need any more datasets?*". We believe that the answer to this question is *yes*. Each new dataset brings in new challenges and contributes towards building better QA systems. It keeps researchers on their toes and prevents research from stagnating once state-of-the-art results are achieved on one dataset. A classic example of this is the CoNLL NER dataset (Tjong Kim Sang and De Meulder, 2003). While several NER systems (Passos et al., 2014) gave close to human performance on this dataset, NER on general web text, domain specific text, noisy social media text is still an unsolved problem (mainly due to the lack of representative datasets which cover the real-world challenges of NER). In this context, DuoRC presents 4 new challenges mentioned earlier which are not exhibited in existing RC datasets and would thus enable exploring novel neural approaches in complex language understanding. The hope is that all these datasets (including ours) will collectively help in addressing a wide range of challenges in QA and prevent stagnation via overfitting on a single dataset.

3 Dataset

In this section, we elaborate on the three phases of our dataset collection process.

Extracting parallel movie plots: We first collected top 40K movies from IMDb across different genres (crime, drama, comedy, etc.) whose plot synopsis were crawled from Wikipedia as well as IMDb. We retained only 7680 movies for which both the plots were available and longer than 100 words. In general, we found that the IMDb plots were usually longer (avg. length 926 words) and more descriptive than the Wikipedia plots (avg. length 580 words). To make sure that the content between the two plots are indeed different and one is not just a subset of another, we calculated word-level jaccard distance between them i.e. the ratio of intersection to union of the bag-of-words in the two plots and found it to be 26%. This indicates that one of the plots is usually longer and descriptive, and, the two plots are infact quite different, even though the information content is very similar.

Collecting QA pairs from shorter version of the plot (*SelfRC*): As mentioned earlier, on average the longer version of the plot is almost double the size of the shorter version which is itself usually 500 words long. Intuitively, the longer version should have more details and the questions asked from the shorter version should be answerable from the longer one. Hence, we first showed the shorter version of the plot to workers on AMT and asked them to create QA pairs from it. The instructions given to the workers for this phase are as follows: (i) the answer must preferably be a single word or a short phrase, (ii) subjective questions (like asking for opinion) are not allowed, (iii) questions should be answerable only from the passage and not require any external knowledge, and (iv) questions and answers should be well formed and grammatically correct. The workers were also given freedom to either pick an answer which directly matches a span in the document or synthesize the answer from scratch. This option allowed them to be creative and ask hard questions where possible. We found that in 70% of the cases the workers picked an answer directly from the document and in 30% of the cases they synthesized the answer. We thus collected 85,773 such QA pairs along with their corresponding documents. We refer to this as the *SelfRC* dataset because the answers were derived from the same document from which the questions were asked.

Collecting answers from longer version of the plot (*ParaphraseRC*): We then paired the questions from the *SelfRC* dataset with the corresponding longer version of the plot and showed it to a different set of AMT workers asking them to answer these questions from the longer version of the plot. They now have the option to either (i) select an answer which matches a span in the longer version, (ii) synthesize the answer from scratch, or (iii) mark the question not-answerable because of lack of information in the given passage. One trick we used to reduce the fatigue of workers (caused by reading long pieces of text), and thus maintain the answer quality is to split the long plots into multiple segments. Every question obtained from the first phase of annotation is paired separately with each of these segments and each (question, segment) pair is posted as a different job. With this approach, we essentially get multiple answers to the same question, if it is answerable from more than one segment. However, on an average we get approximately one unique answer for each question. We found that in 50% of the cases the workers selected an answer which matched a span in the document, whereas in 37% cases they synthesized the answer and in 13% cases they said that question was not answerable. The workers were strictly instructed to keep the answers short, derive the answer from the plot and use *general* knowledge or logic to answer the questions. They were not allowed to rely on *personal* knowledge about the movie (in any case given the large number of movies in our dataset the chance of a worker remembering all the plot details for a given movie is very less). For quality assessment purposes, various levels of manual and semi-automated inspections were done, especially in the second phase of annotation, such as:(i) weeding out annotators who mark majority of answers as non-answerable, by taking into account their response time, and (ii) annotators for whom a high percentage of answers have no entity (or noun phrase) overlap with the entire passage were subjected to strict manual inspection and blacklisted if necessary. Further, a wait period of 2-3 weeks was deliberately introduced between the two phases of data collection to ensure the availability of a fresh pool of workers as well as to reduce information bias among workers common to both the tasks. Overall 2559 workers took part in the first phase of the annotation, and 8021 workers in the second phase. Only 703 workers were common between the phases.

We refer to this dataset, where the questions are taken from one version of the document and the answers are obtained from a different version, as *ParaphraseRC* which contains 100,316 such {*question, answer, document*} triplets. Overall, 62% of the questions in *SelfRC* and *ParaphraseRC* have partial overlap in their answers, which is indicative of the fact that quality is reasonable. The remaining 38% where there is no overlap can be attributed to non-answerablity of the question from the bigger plot, information gap, or paraphrasing of information between the two plots.

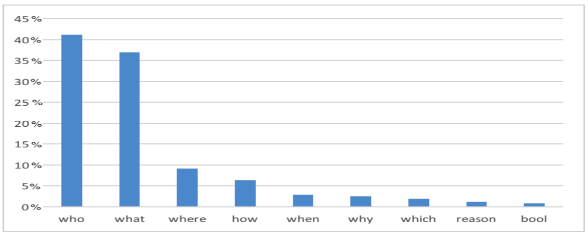


Figure 2: Analysis of the Question Types

Note that the number of unique questions in the *ParaphraseRC* dataset is the same as that in *SelfRC* because we do not create any new questions from the longer version of the plot. We end up with a greater number of {*question, answer, document*} triplets in *ParaphraseRC* as compared to *SelfRC* (100,316 v/s 85,773) since movies that are remakes of a previous movie had very little difference in their Wikipedia plots. Therefore, we did not separately collect questions from the Wikipedia plot of the remake. However, the IMDb plots of the two movies are very different and so we have two different longer versions of the movie (one for the original and one for the remake). We can thus pair the questions created from the Wikipedia plot with both the IMDb versions of the plot thus augmenting the {*question, answer, document*} triplets.

Another notable observation is that in many cases the answers to the same question are different in the two versions. Specifically, only 40.7% of the questions have the same answer in the two documents. For around 37.8% of the questions there is no overlap between the words in the two answers. For the remaining 21% of the questions there is a partial overlap between the two answers. For e.g., the answer derived from the shorter version could be "using his wife's gun" and from the longer version could be "with Dana's handgun" where Dana is the name of the wife. In Appendix **A**, we provide a few randomly picked examples from our dataset which should convince the reader of the difficulty of *ParaphraseRC* and its differences with *SelfRC*. We refer to this combined dataset containing a total of 186,089 instances as *DuoRC*[1]. Fig. 2 shows the distribution of different Wh-type questions in our dataset. Some interesting comparative analysis are presented in Table 1 and also in Appendix **B**. In Table 1, we compare various RC datasets with two embodiments of our dataset i.e. the *SelfRC* and *ParaphraseRC*. We use NER and noun phrase/verb phrase extraction over the entire dataset to iden-

Metrics for Comparative Analysis	**Movie QA**	**NarrativeQA over plot-summaries**	**Self-RC**	**Paraph-raseRC**
Avg. word distance	20.67	24.94	13.4	45.3
Avg. sentence distance	1.67	1.95	1.34	2.7
Number of sentences for inferencing	2.3	1.95	1.51	2.47
% of instances where both Query & Answer entities were found in passage	67.96	59.4	58.79	12.25
% of instances where Only Query entities were found in passage	59.61	61.77	63.39	47.05
% Length of the Longest Common sequence of non-stop words in Query (w.r.t Query Length) and Plot	25	26.26	38	21

Table 1: Comparison between various RC datasets

[1]The dataset is available at https://duorc.github.io

tify key entities in the question, plot and answer which is in turn used to compute the metrics mentioned in the table. The metrics "Avg word distance" and "Avg sentence distance" indicate the average distance (in terms of words/sentences) between the occurrence of the question entities and closest occurrence of the answer entities in the passage. "Number of sentences for inferencing" is indicative of the minimum number of sentences required to cover all the question and answer entities. It is evident that tackling *ParaphraseRC* is much harder than the others on account of (i) larger distance between the query and answer, (ii) low word-overlap between query & passage, and (iii) higher number of sentences required to infer an answer.

4 Models

In this section, we describe in detail the various state-of-the-art RC and language generation models along with a collection of traditional NLP techniques employed together that will serve to establish baseline performance on the DuoRC dataset.

Most of the current state-of-the-art models for RC assume that the answer corresponds to a span in the document and the task of the model is to predict this span. This is indeed true for the SQuAD, TriviaQA and NewsQA datasets. However, in our dataset, in many cases the answers do not correspond to an exact span in the document but are synthesized by humans. Specifically, for the *SelfRC* version of the dataset around 30% of the answers are synthesized and do not match a span in the document whereas for the *ParaphraseRC* task this number is 50%. Nevertheless, we could still leverage the advances made on the SQuAD dataset and adapt these span prediction models for our task. To do so, we propose to use two models. The first model is a basic span prediction model which we train and evaluate using only those instances in our dataset where the answer matches a span in the document. The purpose of this model is to establish whether even for instances where the answer matches a span in the document, our dataset is harder than the SQuAD dataset or not. Specifically, we want to explore the performance of state-of-the-art models (such as DCN (Xiong et al., 2016)), which exhibit near human results on the SQuAD dataset, on DuoRC (especially, in the *ParaphraseRC* setup). To do so, we seek to employ a good span prediction model for which (i) the performance is within 3-5% of the top performing model on the SQuAD leaderboard (Rajpurkar et al., 2016b) and (ii) the results are reproducible based on the code released by the authors of the paper. Note that the second criteria is important to ensure that the poor performance of the model is not due to incorrect implementation. The Bidirectional Attention Flow (BiDAF) model (Seo et al., 2016) satisfies these criteria and hence we employ this model. Due to space constraints, we do not provide details of the BiDAF model here and simply refer the reader to the original paper. In the remainder of this paper we will refer to this model as the *SpanModel*.

The second model that we employ is a two stage process which first predicts the span and then synthesizes the answers from the span. Here again, for the first step (*i.e.*, span prediction) we use the BiDAF model (Seo et al., 2016). The job of the second model is to then take the span (mini-document) and question (query) as input and generate the answer. For this, we employ a state-of-the-art query based abstractive summarization model (Nema et al., 2017) as this task is very similar to our task. Specifically, in query based abstractive summarization the training data is of the form {query, document, generated_summary} and in our case the training data is of the form {query, mini-document, generated_answer}. Once again we refer the reader to the original paper (Nema et al., 2017) for details of the model. We refer to this two stage model as the *GenModel*.

Note that (Tan et al., 2017) recently proposed an answer generation model for the MS MARCO dataset. However, the authors have not released their code and therefore, in the interest of reproducibility of our work, we omit incorporating this model in this paper.

Additional NLP pre-processing: Referring back to the example cited in Fig. 1, we reiterate that ideally a good model for *ParaphraseRC* would require: (i) employing a knowledge graph, (ii) common-sense knowledge (iii) paraphrase/semantic understanding (iv) multiple-sentence inferencing across events in the passage including coreference resolution of named entities and nouns, and (v) educated guesswork when the question is not directly answerable but there are subtle hints in the passage. While addressing all of these challenges in their entirety is beyond the scope of a single paper, in the interest of establishing a good baseline for DuoRC, we additionally seek to address some of these challenges to a certain extent by using standard NLP techniques. Specifically, we look at the problems of paraphrase understanding, coreference resolution and handling long passages.

To do so, we prune the document and extract only those sentences which are most relevant to the question, so that the span detector does not need to look at the entire 900-word long *ParaphraseRC* plot. Now, since these relevant sentences are obtained not from the original but the paraphrased version of the document, they may have a very small word overlap with the question. For example, the question might contain the word "hand gun" and the relevant sentence in the document may contain the word "revolver". Further some of the named entities in the question may not be exactly present in the relevant sentence but may simply be co-referenced. To resolve these coreferences, we first employ the Stanford coreference resolution on the entire document. We then compute the fraction of words in a sentence which match a query word (ignoring stop words). Two words are considered to match if (a) they have the same surface form, or (b) one words is an inflected form of the word (e.g., river and rivers), or (c) the Glove (Pennington et al., 2014) and Skip-thought (Kiros et al., 2015) embeddings of the two words are very close to each other (two word vectors are considered to be close if one appears within the top 50 neighbors of the other), or (d) the two words appear in the same synset in Wordnet. We consider a sentence to be relevant for the question if at least 50% of the query words (ignoring stop words) match the words in the sentence. If none of the sentences in the document have atleast 50% overlap with the question, then we pick sentences having atleast a 30% overlap with the question. The selection of this threshold was based on manual observation of a small sample set. This observation gave us an idea of what a decent threshold value should be, that can have a reasonable precision and recall on the relevant snippet extraction step. Since this step was rule-based we could only employ such qualitative inspections to set this parameter. Also, since this step was targeted to have high recall, we relaxed the threshold to 30% if no match was found.

5 Experimental Setup

In the following sub-sections we describe (i) the evaluation metrics, and (ii) the choices considered for augmenting the training data for the answer generation model. Note that when creating the train, validation and test set, we ensure that the test set does not contain QA pairs for any movie that was seen during training. We split the movies in such a way that the resulting train, valid, test sets respectively contain 70%, 15% and 15% of the total number of QA pairs.

Span-Based Test Set and Full Test Set As mentioned earlier, the *SpanModel* only predicts the span in the document whereas the *GenModel* generates the answer after predicting the span. Ideally, the *SpanModel* should only be evaluated on those instances in the test set where the answer matches a span in the document. We refer to this subset of the test set as the *Span-based Test Set*. Though not ideal, we also evaluate the *SpanModel* model on the entire test set. This is not ideal because there are many answers in the test set which do not correspond to a span in the document whereas the model was only trained to predict spans. We refer to this as the *Full Test Set*. We also evaluate the *GenModel* on both the test sets.

Training Data for the GenModel As mentioned earlier, the *GenModel* contains two stages; the first stage predicts the span and the second stage then generates an answer from the predicted span. For the first step we plug-in the best performing *SpanModel* from our earlier exploration. To train the second stage we need training data of the form $\{x = \textit{span},\ y = \textit{answer}\}$ which comes from two types of instances: one where the answer matches a span and the other where the answer is synthesized and the span corresponding to it is not known. In the first case x=y and there is nothing interesting for the model to learn (except for copying the input to the output). In the second case x is not known. To overcome this problem, for the second type of instances, we consider various approaches for finding the approximate span from which the answer could have been generated, and augment the training data with $\{x = \textit{approx\_span},\ y = \textit{answer}\}$.
The easiest method was to simply treat the entire document as the true span from which the answer was generated (x = *document*, y = *answer*). The second alternative that we tried was to first extract the named entities, noun phrases and verb phrases from the question and create a lucene query from these components. We then used the lucene search engine to extract the most relevant portions of the document given this query. We then considered this portion of the document as the true span (as

opposed to treating the entire document as the true span). Note that lucene could return multiple relevant spans in which case we treat all these $\{x = approx\_span, y = answer\}$ as training instances. Another alternative was to find the longest common subsequence (LCS) between the document and the question and treat this subsequence as the span from which the answer was generated. Of these, we found that the model trained using $\{x = approx\_span, y = answer\}$ pairs created using the LCS based method gave the best results. We report numbers only for this model.

Evaluation Metrics Similar to (Rajpurkar et al., 2016a) we use Accuracy and F-score as the evaluation metrics. We also report the BLEU scores for each task. While accuracy, being a stricter metric, considers a predicted answer to be correct only if it exactly matches the true answer, F-score and BLEU also give credit to predictions partially overlapping with the true answer.

6 Results and Discussions

The results of our experiments are summarized in Tables 2 to 4 which we discuss in the following sub-sections.

Preprocessing step of Relevant Subplot Extraction	**Plot Compression**	**Answer Recall**
WordNet synonym + Glove based paraphrase	30%	66.51%
WordNet synonym + Glove based paraphrase on Coref resolved plots	50%	84.10%
WordNet synonym + Glove + Skip-thought based paraphrase on Coref resolved plots	48%	85%

Table 2: Performance of the preprocessing. Plot compression is the % size of the extracted plot w.r.t the original plot size

SelfRC	**Span Test**			**Full Test**		
	Acc.	**F1**	**BLEU**	**Acc.**	**F1**	**BLEU**
SpanModel	46.14	57.49	22.98	37.53	50.56	7.47
GenModel (with augmented training data)	16.45	26.97	7.61	15.31	24.05	5.50
ParaphraseRC	**Span Test**			**Full Test**		
	Acc.	**F1**	**BLEU**	**Acc.**	**F1**	**BLEU**
SpanModel	17.93	26.27	9.39	9.78	16.33	2.60
SpanModel with Preprocessed Data	27.49	35.10	12.78	14.92	21.53	2.75
GenModel (with augmented training data)	12.66	19.48	4.41	5.42	9.64	1.75

Table 3: Performance of the *SpanModel* and *GenModel* on the Span Test subset and the Full Test Set of the *Self* and *ParaphraseRC*.

***SpanModel* v/s *GenModel*:** Comparing the first two rows (*SelfRC*) and the last two rows (*ParaphraseRC*) of Table 3 we see that the *SpanModel* clearly outperforms the *GenModel*. This is not very surprising for two reasons. First, around 70% (and

		Span Test			**Full Test**		
Train	**Test**	**Acc.**	**F1**	**BLEU**	**Acc.**	**F1**	**BLEU**
SelfRC	*SelfRC*	46.14	57.49	22.98	37.53	50.56	7.47
	ParaRC	27.85	36.82	14.48	15.16	22.70	3.90
	SelfRC+ ParaRC	37.79	48.05	18.72	25.05	35.01	5.34
ParaRC	*SelfRC*	34.85	45.71	16.01	28.25	40.16	5.15
	Para RC	19.74	27.57	9.84	10.78	17.13	2.75
	SelfRC+ ParaRC	27.94	37.42	13.00	18.50	27.31	3.75
SelfRC + ParaRC	*SelfRC*	49.66	61.45	25.87	40.24	54.04	8.42
	ParaRC	29.88	39.34	16.22	16.33	24.25	4.21
	SelfRC+ ParaRC	40.62	51.35	21.18	26.90	37.42	5.94

Table 4: Combined and Cross-Testing between *Self* and *ParaphraseRC* Dataset, by taking the best performing *SpanModel* from Table 3.*ParaRC* is an abbreviation of *ParaphraseRC*

50%) of the answers in *SelfRC* (and *ParaphraseRC*) respectively, match an exact span in the document so the *SpanModel* still has scope to do well on these answers. On the other hand, even if the first stage of the *GenModel* predicts the span correctly, the second stage could make an error in generating the correct answer from it because generation is a harder problem. For the second stage, it is expected that the *GenModel* should learn to copy the predicted span to produce the answer output (as is required in most cases) and only occasionally where necessary, generate an answer. However, surprisingly the *GenModel* fails to even do this. Manual inspection of the generated answers shows that in many cases the generator ends up generating either more or fewer words compared the true answer. This demonstrates the clear scope for the *GenModel* to perform better.

SelfRC v/s ParaphraseRC: Comparing the *SelfRC* and *ParaphraseRC* numbers in Table 3, we observe that the performance of the models clearly drops for the latter task, thus validating our hypothesis that *ParaphraseRC* is a indeed a much harder task.

Effect of NLP pre-processing: As mentioned in Section 4, for *ParaphraseRC*, we first perform a few pre-processing steps to identify relevant sentences in the longer document. In order to evaluate whether the pre-processing method is effective, we compute: (i) the percentage of the document that gets pruned, and (ii) whether the true answer is present in the pruned document (i.e., average recall of the answer). We can compute the recall only for the span-based subset of the data since for the remaining data we do not know the true span. In Table 2, we report these two quantities for the span-based subset using different pruning strategies. Finally, comparing the *SpanModel* with and without

Paraphrasing in Table 3 for *ParaphraseRC*, we observe that the pre-processing step indeed improves the performance of the Span Detection Model.

Effect of oracle pre-processing: As noted in Section 3, the *ParaphraseRC* plot is almost double in length in comparison to the *SelfRC* plot, which while adding to the complexities of the former task, is clearly not the primary reason of the model's poor performance on that. To empirically validate this, we perform an Oracle pre-processing step, where, starting with the knowledge of the span containing the true answer, we extract a subplot around it such that the span is randomly located within that subplot and the average length of the subplot is similar to the *SelfRC* plots. The *SpanModel* with this Oracle preprocessed data exhibits a minor improvement in performance over that with rule-based preprocessing (1.6% in Accuracy and 4.3% in F1 over the Span Test), still failing to bridge the wide performance gap between the *SelfRC* and *ParaphraseRC* task.

Cross Testing We wanted to examine whether a model trained on *SelfRC* performs well on *ParaphraseRC* and vice-versa. We also wanted to evaluate if merging the two datasets improves the performance of the model. For this we experimented with various combinations of train and test data. The results of these experiments for the *SpanModel* are summarized in Table 4. The best performance is obtained when the model is trained on both (*SelfRC*) and *ParaphraseRC* and tested on *SelfRC* and the performance is poorest when *ParaphraseRC* is used for both. We believe this is because learning with the *ParaphraseRC* is more difficult given the wide range of challenges in this dataset.

Based on our experiments and empirical observations we believe that the *DuoRC* dataset indeed holds a lot of potential for advancing the horizon of complex language understanding by exposing newer challenges in this area.

7 Conclusion

In this paper we introduced DuoRC, a large scale RC dataset of 186K human-generated QA pairs created from 7680 pairs of parallel movie-plots, each pair taken from Wikipedia and IMDb. We then showed that this dataset, by design, ensures very little or no lexical overlap between the questions created from one version and segments containing answers in the other version. With this, we hope to introduce the RC community to new research challenges on QA requiring external knowledge and common-sense driven reasoning, deeper language understanding and multiple-sentence inferencing. Through our experiments, we show how the state-of-the-art RC models, which have achieved near human performance on the SQuAD dataset, perform poorly on our dataset, thus emphasizing the need to explore further avenues for research.

References

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. http://aclweb.org/anthology/D/D14/D14-1159.pdf.

Yiming Cui. 2017. Cloze explorer. https://github.com/ymcui/Eval-on-NN-of-RC/.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 1693–1701. http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 2011–2021. http://aclanthology.info/papers/D17-1214/d17-1214.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. pages 1601–1611. https://doi.org/10.18653/v1/P17-1147.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR* abs/1506.06726. http://arxiv.org/abs/1506.06726.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. *CoRR* abs/1704.04683. http://arxiv.org/abs/1704.04683.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *CoRR* abs/1604.01696. http://arxiv.org/abs/1604.01696.

Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. *CoRR* abs/1704.08300. http://arxiv.org/abs/1704.08300.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR* abs/1611.09268. http://arxiv.org/abs/1611.09268.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457* .

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*. pages 78–86. http://aclweb.org/anthology/W/W14/W14-1609.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. Squad explorer. https://rajpurkar.github.io/SQuAD-explorer/.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 193–203. http://aclweb.org/anthology/D/D13/D13-1020.pdf.

Tomáš Koˇ ciský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics* TBD:TBD. https://TBD.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR* abs/1611.01603. http://arxiv.org/abs/1611.01603.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *CoRR* abs/1706.04815. http://arxiv.org/abs/1706.04815.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*. Edmonton, Canada, pages 142–147.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR* abs/1611.09830. http://arxiv.org/abs/1611.09830.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*. pages 271–280. https://doi.org/10.18653/v1/K17-1028.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604. http://arxiv.org/abs/1611.01604.

Stochastic Answer Networks for Machine Reading Comprehension

Xiaodong Liu[†], **Yelong Shen**[†], **Kevin Duh**[‡] **and Jianfeng Gao**[†]
[†] Microsoft Research, Redmond, WA, USA
[‡] Johns Hopkins University, Baltimore, MD, USA
[†]{xiaodl,yeshen,jfgao}@microsoft.com [‡]kevinduh@cs.jhu.edu

Abstract

We propose a simple yet robust **s**tochastic **a**nswer **n**etwork (SAN) that simulates multi-step reasoning in machine reading comprehension. Compared to previous work such as ReasoNet which used reinforcement learning to determine the number of steps, the unique feature is the use of a kind of stochastic prediction dropout on the answer module (final layer) of the neural network during the training. We show that this simple trick improves robustness and achieves results competitive to the state-of-the-art on the Stanford Question Answering Dataset (SQuAD), the Adversarial SQuAD, and the Microsoft MAchine Reading COmprehension Dataset (MS MARCO).

1 Introduction

Machine reading comprehension (MRC) is a challenging task: the goal is to have machines read a text passage and then answer any question about the passage. This task is an useful benchmark to demonstrate natural language understanding, and also has important applications in e.g. conversational agents and customer service support. It has been hypothesized that difficult MRC problems require some form of multi-step synthesis and reasoning. For instance, the following example from the MRC dataset SQuAD (Rajpurkar et al., 2016) illustrates the need for synthesis of information across sentences and multiple steps of reasoning:

Q: What collection does **the V&A Theator & Performance galleries** hold?

P: **The V&A Theator & Performance galleries** opened in March 2009. ... **They** hold the UK's biggest national collection of material about live performance.

To infer the answer (the underlined portion of the passage P), the model needs to first perform coreference resolution so that it knows "**They**" refers "**V&A Theator**", then extract the subspan in the direct object corresponding to the answer.

This kind of iterative process can be viewed as a form of multi-step reasoning. Several recent MRC models have embraced this kind of multi-step strategy, where predictions are generated after making multiple passes through the same text and integrating intermediate information in the process. The first models employed a predetermined fixed number of steps (Hill et al., 2016; Dhingra et al., 2016; Sordoni et al., 2016; Kumar et al., 2015). Later, Shen et al. (2016) proposed using reinforcement learning to dynamically determine the number of steps based on the complexity of the question. Further, Shen et al. (2017) empirically showed that dynamic multi-step reasoning outperforms fixed multi-step reasoning, which in turn outperforms single-step reasoning on two distinct MRC datasets (SQuAD and MS MARCO).

In this work, we derive an alternative multi-step reasoning neural network for MRC. During training, we fix the number of reasoning steps, but perform stochastic dropout on the answer module (final layer predictions). During decoding, we generate answers based on the average of predictions in all steps, rather than the final step. We call this a stochastic answer network (SAN) because the stochastic dropout is applied to the answer module; albeit simple, this technique significantly improves the robustness and overall accuracy of the model. Intuitively this works because while the model successively refines its prediction over multiple steps, each step is still trained to generate the same answer; we are performing a kind of stochastic ensemble over the model's successive predic-

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1694–1704
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

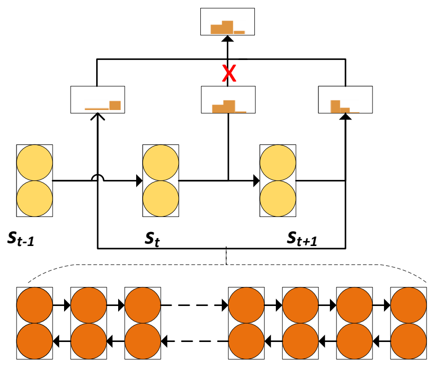


Figure 1: Illustration of "stochastic prediction dropout" in the answer module during training. At each reasoning step t, the model combines memory (bottom row) with hidden states $\mathbf{s}_{t-1}$ to generate a prediction (multinomial distribution). Here, there are three steps and three predictions, but one prediction is dropped and the final result is an average of the remaining distributions.

tion refinements. Stochastic prediction dropout is illustrated in Figure 1.

2 Proposed model: SAN

The machine reading comprehension (MRC) task as defined here involves a question $Q = \{q_0, q_1, ..., q_{m-1}\}$ and a passage $P = \{p_0, p_1, ..., p_{n-1}\}$ and aims to find an answer span $A = \{a_{start}, a_{end}\}$ in P. We assume that the answer exists in the passage P as a contiguous text string. Here, m and n denote the number of tokens in Q and P, respectively. The learning algorithm for reading comprehension is to learn a function $f(Q, P) \to A$. The training data is a set of the query, passage and answer tuples $< Q, P, A >$.

We now describe our model from the ground up. The main contribution of this work is the answer module, but in order to understand what goes into this module, we will start by describing how Q and P are processed by the lower layers. Note the lower layers also have some novel variations that are not used in previous work. As shown in Figure 2, our model contains four different layers to capture different concept of representations. The detailed description of our model is provided as follows.

Lexicon Encoding Layer. The purpose of the first layer is to extract information from Q and P at the word level and normalize for lexical variants. A typical technique to obtain lexicon embedding is concatenation of its word embedding with other linguistic embedding such as those derived from Part-Of-Speech (POS) tags. For word embeddings, we use the pre-trained 300-dimensional GloVe vectors (Pennington et al., 2014) for the both Q and P. Following Chen et al. (2017), we use three additional types of linguistic features for each token p_i in the *passage* P:

- 9-dimensional POS tagging embedding for total 56 different types of the POS tags.
- 8-dimensional named-entity recognizer (NER) embedding for total 18 different types of the NER tags. We utilized small embedding sizes for POS and NER to reduce model size. They mainly serve the role of coarse-grained word clusters.
- A 3-dimensional binary *exact* match feature defined as $f_{exact\_match}(p_i) = \mathbb{I}(p_i \in Q)$. This checks whether a passage token p_i matches the original, lowercase or lemma form of any question token.
- Question enhanced passages word embeddings: $f_{align}(p_i) = \sum_j \gamma_{i,j} g(GloVe(q_j))$, where $g(\cdot)$ is a 280-dimensional single layer neural network $ReLU(W_0 x)$ and $\gamma_{i,j} = \frac{exp(g(GloVe(p_j)) \cdot g(GloVe(q_i)))}{\sum_{j'} exp(g(GloVe(p_i)) \cdot g(GloVe(q_{j'})))}$ measures the similarity in word embedding space between a token p_i in the passage and a token q_j in the question. Compared to the *exact* matching features, these embeddings encode *soft* alignments between similar but not-identical words.

In summary, each token p_i in the passage is represented as a 600-dimensional vector and each token q_j is represented as a 300-dimensional vector.

Due to different dimensions for the passages and questions, in the next layer two different bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) may be required to encode the contextual information. This, however, introduces a large number of parameters. To prevent this, we employ an idea inspired by (Vaswani et al., 2017): use two separate two-layer position-wise Feed-Forward Networks (FFN), $FFN(x) = W_2 ReLU(W_1 x + b_1) + b_2$, to map both the passage and question lexical encodings into the same number of dimensions. Note that this FFN has fewer

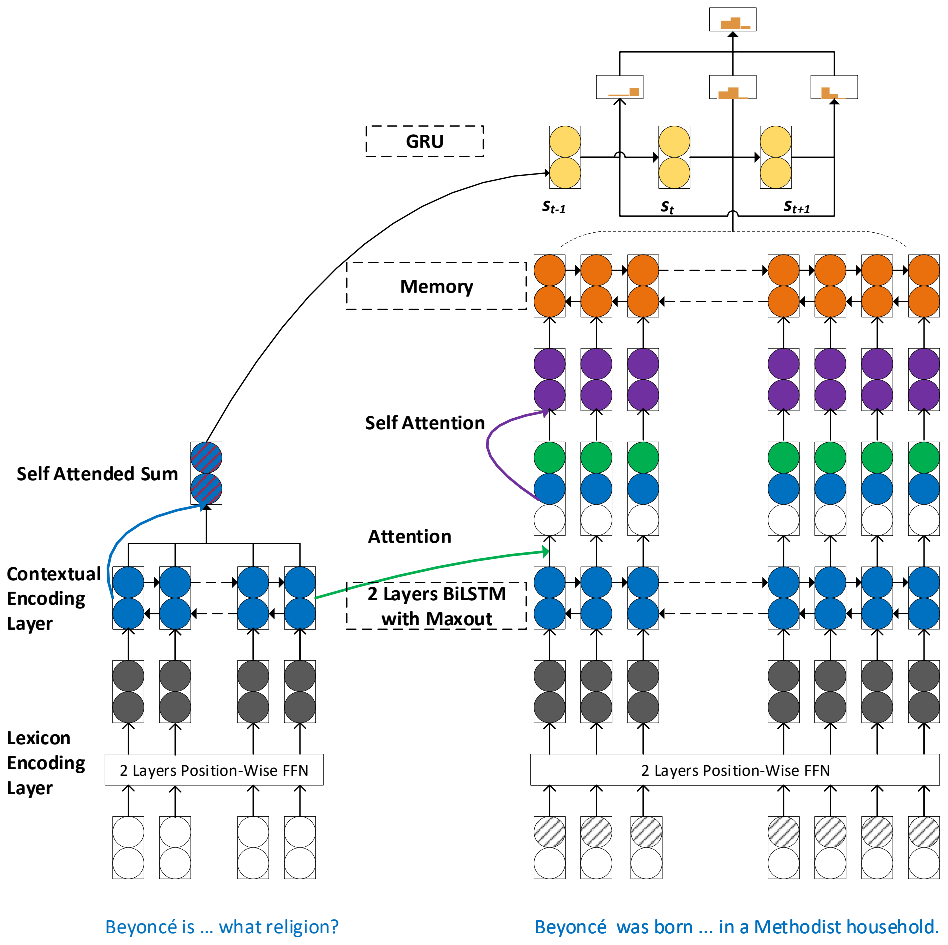

GRU
s_{t-1}
s_t
s_{t+1}
Memory
Self Attention
Self Attended Sum
Attention
Contextual Encoding Layer
2 Layers BiLSTM with Maxout
Lexicon Encoding Layer
2 Layers Position-Wise FFN
2 Layers Position-Wise FFN
Beyoncé is … what religion?
Beyoncé was born … in a Methodist household.

Figure 2: **Architecture of the SAN for Reading Comprehension:** The first layer is a lexicon encoding layer that maps words to their embeddings independently for the question (left) and the passage (right): this is a concatenation of word embeddings, POS embeddings, etc. followed by a position-wise FFN. The next layer is a context encoding layer, where a BiLSTM is used on the top of the lexicon embedding layer to obtain the context representation for both question and passage. In order to reduce the parameters, a maxout layer is applied on the output of BiLSTM. The third layer is the working memory: First we compute an alignment matrix between the question and passage using an attention mechanism, and use this to derive a question-aware passage representation. Then we concatenate this with the context representation of passage and the word embedding, and employ a self attention layer to re-arrange the information gathered. Finally, we use another LSTM to generate a working memory for the passage. At last, the fourth layer is the answer module, which is a GRU that outputs predictions at each state s_t.

parameters compared to a BiLSTM. Thus, we obtain the final lexicon embeddings for the tokens in Q as a matrix $E^q \in \mathbb{R}^{d \times m}$ and tokens in P as $E^p \in \mathbb{R}^{d \times n}$.

Contextual Encoding Layer. Both passage and question use a shared two-layers BiLSTM as the contextual encoding layer, which projects the lexicon embeddings to contextual embeddings. We concatenate a pre-trained 600-dimensional CoVe vectors[1] (McCann et al., 2017) trained on German-English machine translation dataset, with the aforementioned lexicon embeddings as the final input of the contextual encoding layer, and also with the output of the first contextual encoding layer as the input of its second encoding layer. To reduce the parameter size, we use a maxout layer (Goodfellow et al., 2013) at each BiLSTM layer to shrink its dimension. By a concatenation of the outputs of two BiLSTM layers, we obtain $H^q \in \mathbb{R}^{2d \times m}$ as representation of Q and $H^p \in \mathbb{R}^{2d \times n}$ as representation of P, where d is the hidden size of the BiLSTM.

Memory Generation Layer. In the memory

[1]https://github.com/salesforce/cove

generation layer, We construct the working memory, a summary of information from both Q and P. First, a dot-product attention is adopted like in (Vaswani et al., 2017) to measure the similarity between the tokens in Q and P. Instead of using a scalar to normalize the scores as in (Vaswani et al., 2017), we use one layer network to transform the contextual information of both Q and P:

$$C = dropout(f_{attention}(\hat{H}^q, \hat{H}^p)) \in \mathbb{R}^{m \times n} \quad (1)$$

C is an attention matrix. Note that $\hat{H}^q$ and $\hat{H}^p$ is transformed from H^q and H^p by one layer neural network $ReLU(W_3 x)$, respectively. Next, we gather all the information on passages by a simple concatenation of its contextual information H^p and its question-aware representation $H^q \cdot C$:

$$U^p = concat(H^p, H^q C) \in \mathbb{R}^{4d \times n} \quad (2)$$

Typically, a passage may contain hundred of tokens, making it hard to learn the long dependencies within it. Inspired by (Lin et al., 2017), we apply a self-attended layer to rearrange the information U^p as:

$$\hat{U}^p = U^p drop_{diag}(f_{attention}(U^p, U^p)). \quad (3)$$

In other words, we first obtain an $n \times n$ attention matrix with U^p onto itself, apply dropout, then multiply this matrix with U^p to obtain an updated $\hat{U}^p$. Instead of using a penalization term as in (Lin et al., 2017), we dropout the diagonal of the similarity matrix forcing each token in the passage to align to other tokens rather than itself.

At last, the working memory is generated by using another BiLSTM based on all the information gathered:

$$M = BiLSTM([U^p; \hat{U}^p]) \quad (4)$$

where the semicolon mark ; indicates the vector/matrix concatenation operator.

Answer module. There is a Chinese proverb that says: "wisdom of masses exceeds that of any individual." Unlike other multi-step reasoning models, which only uses a single output either at the last step or some dynamically determined final step, our answer module employs all the outputs of multiple step reasoning. Intuitively, by applying dropout, it avoids a "step bias problem" (where models places too much emphasis one particular step's predictions) and forces the model to produce good predictions at every individual step. Further, during decoding, we reuse *wisdom of masses* instead of *individual* to achieve a better result. We call this method "stochastic prediction dropout" because dropout is being applied to the final predictive distributions.

Formally, our answer module will compute over T memory steps and output the answer span. This module is a memory network and has some similarities to other multi-step reasoning networks: namely, it maintains a state vector, one state per step. At the beginning, the initial state s_0 is the summary of the Q: $s_0 = \sum_j \alpha_j H^q_j$, where $\alpha_j = \frac{exp(w_4 \cdot H^q_j)}{\sum_{j'} exp(w_4 \cdot H^q_{j'})}$. At time step t in the range of $\{1, 2, ..., T-1\}$, the state is defined by $s_t = GRU(s_{t-1}, x_t)$. Here, x_t is computed from the previous state s_{t-1} and memory M: $x_t = \sum_j \beta_j M_j$ and $\beta_j = softmax(s_{t-1} W_5 M)$. Finally, a bilinear function is used to find the begin and end point of answer spans at each reasoning step $t \in \{0, 1, \dots, T-1\}$.

$$P^{begin}_t = softmax(s_t W_6 M) \quad (5)$$

$$P^{end}_t = softmax([s_t; \sum_j P^{begin}_{t,j} M_j] W_7 M). \quad (6)$$

From a pair of begin and end points, the answer string can be extracted from the passage. However, rather than output the results (start/end points) from the final step (which is fixed at $T-1$ as in Memory Networks or dynamically determined as in ReasoNet), we utilize all of the T outputs by averaging the scores:

$$P^{begin} = avg([P^{begin}_0, P^{begin}_1, ..., P^{begin}_{T-1}]) \quad (7)$$

$$P^{end} = avg([P^{end}_0, P^{end}_1, ..., P^{end}_{T-1}]) \quad (8)$$

Each P^{begin}_t or P^{end}_t is a multinomial distribution over $\{1, \dots, n\}$, so the average distribution is straightforward to compute.

During training, we apply stochastic dropout to before the above averaging operation. For example, as illustrated in Figure 1, we randomly delete several steps' predictions in Equations 7 and 8 so that P^{begin} might be $avg([P^{begin}_1, P^{begin}_3])$ and P^{end} might be $avg([P^{end}_0, P^{end}_3, P^{end}_4])$. The use of averaged predictions and dropout during training improves robustness.

Our stochastic prediction dropout is similar in motivation to the dropout introduced by (Srivastava et al., 2014). The difference is that theirs is dropout at the intermediate node-level, whereas ours is dropout at the final layer-level. Dropout at the node-level prevents correlation between features. Dropout at the final layer level, where randomness is introduced to the averaging of predictions, prevents our model from relying exclusively on a particular step to generate correct output. We used a dropout rate of 0.4 in experiments.

3 Experiment Setup

Dataset: We evaluate on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). This contains about 23K passages and 100K questions. The passages come from approximately 500 Wikipedia articles and the questions and answers are obtained by crowdsourcing. The crowdsourced workers are asked to read a passage (a paragraph), come up with questions, then mark the answer span. All results are on the official development set, unless otherwise noted.

Two evaluation metrics are used: Exact Match (EM), which measures the percentage of span predictions that matched any one of the ground truth answer exactly, and Macro-averaged F1 score, which measures the average overlap between the prediction and the ground truth answer.

Implementation details: The spaCy tool[2] is used to tokenize the both passages and questions, and generate lemma, part-of-speech and named entity tags. We use 2-layer BiLSTM with $d = 128$ hidden units for both passage and question encoding. The mini-batch size is set to 32 and Adamax (Kingma and Ba, 2014) is used as our optimizer. The learning rate is set to 0.002 at first and decreased by half after every 10 epochs. We set the dropout rate for all the hidden units of LSTM, and the answer module output layer to 0.4. To prevent degenerate output, we ensure that at least one step in the answer module is active during training.

4 Results

The main experimental question we would like to answer is whether the stochastic dropout and averaging in the answer module is an effective technique for multi-step reasoning. To do so, we fixed all lower layers and compared different architectures for the answer module:

1. Standard 1-step: generate prediction from s_0, the first initial state.

2. 5-step memory network: this is a memory network fixed at 5 steps. We try two variants: the standard variant outputs result from the final step s_{T-1}. The averaged variant outputs results by averaging across all 5 steps, and is like SAN without the stochastic dropout.

3. ReasoNet[3]: this answer module dynamically decides the number of steps and outputs results conditioned on the final step.

4. SAN: proposed answer module that uses stochastic dropout and prediction averaging.

The main results in terms of EM and F1 are shown in Table 1. We observe that SAN achieves 76.235 EM and 84.056 F1, outperforming all other models. Standard 1-step model only achieves 75.139 EM and dynamic steps (via ReasoNet) achieves only 75.355 EM. SAN also outperforms a 5-step memory net with averaging, which implies averaging predictions is not the only thing that led to SAN's superior results; indeed, stochastic prediction dropout is an effective technique.

The K-best oracle results is shown in Figure 3. The K-best spans are computed by ordering the spans according the their probabilities $P^{begin} \times P^{end}$. We limit K in the range 1 to 4 and then pick the span with the best EM or F1 as oracle. SAN also outperforms the other models in terms of K-best oracle scores. Impressively, these models achieve human performance at $K = 2$ for EM and $K = 3$ for F1.

Finally, we compare our results with other top models in Table 2. Note that all the results in Table 2 are taken from the published papers. We see that SAN is very competitive in both single and ensemble settings (ranked in second) despite its simplicity. Note that the best-performing model (Peters et al., 2018) used a large-scale language model as an extra contextual embedding, which gave a significant improvement (+4.3% dev F1). We expect significant improvements if we add this to SAN in future work.

[2]https://spacy.io

[3]The ReasoNet here is not an exact re-implementation of (Shen et al., 2017). The answer module is the same as (Shen et al., 2017) but the lower layers are set to be the same as SAN, 5-step memory network, and standard 1-step as described in Figure 2. We only vary the answer module in our experiments for a fair comparison.

Answer Module	EM	F1
Standard 1-step	75.139	83.367
Fixed 5-step with Memory Network (prediction from final step)	75.033	83.327
Fixed 5-step with Memory Network (prediction averaged from all steps)	75.256	83.215
Dynamic steps (max 5) with ReasoNet	75.355	83.360
Stochastic Answer Network (SAN), Fixed 5-step	**76.235**	**84.056**

Table 1: **Main results**—Comparison of different answer module architectures. Note that SAN performs best in both Exact Match and F1 metrics.

Ensemble model results:	Dev Set (EM/F1)	Test Set (EM/F1)
BiDAF + Self Attention + ELMo (Peters et al., 2018)	-/-	**81.003/87.432**
SAN (Ensemble model)	78.619/85.866	79.608/86.496
AIR-FusionNet (Huang et al., 2017)	-/-	78.978/86.016
DCN+ (Xiong et al., 2017)	-/-	78.852/85.996
M-Reader (Hu et al., 2017)	-/-	77.678/84.888
Conductor-net (Liu et al., 2017b)	74.8 / 83.3	76.996/84.630
r-net (Wang et al., 2017)	77.7/83.7	76.9/84.0
ReasoNet++ (Shen et al., 2017)	75.4/82.9	75.0/82.6
Individual model results:		
BiDAF + Self Attention + ELMo(Peters et al., 2018)	-/-	**78.580/85.833**
SAN (single model)	**76.235/84.056**	76.828/84.396
AIR-FusionNet(Huang et al., 2017)	75.3/83.6	75.968/83.900
RaSoR + TR (Salant and Berant, 2017)	-/-	75.789/83.261
DCN+(Xiong et al., 2017)	74.5/83.1	75.087/83.081
r-net(Wang et al., 2017)	72.3/80.6	72.3/80.7
ReasoNet++(Shen et al., 2017)	70.8/79.4	70.6/79.36
BiDAF (Seo et al., 2016)	67.7/77.3	68.0/77.3
Human Performance	80.3/90.5	82.3/91.2

Table 2: Test performance on SQuAD. Results are sorted by Test F1.

5 Analysis

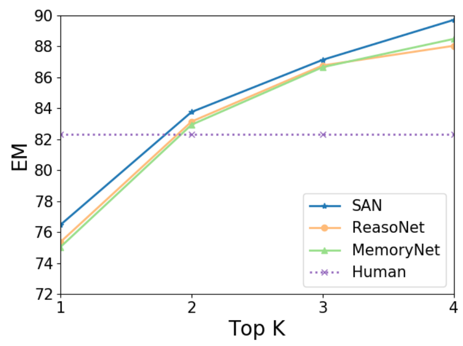
5.1 How robust are the results?

We are interested in whether the proposed model is sensitive to different random initial conditions. Table 3 shows the development set scores of SAN trained from initialization with different random seeds. We observe that the SAN results are consistently strong regardless of the 10 different initializations. For example, the mean EM score is 76.131 and the lowest EM score is 75.922, both of which still outperform the 75.355 EM of the Dynamic step ReasoNet in Table 1.[4]

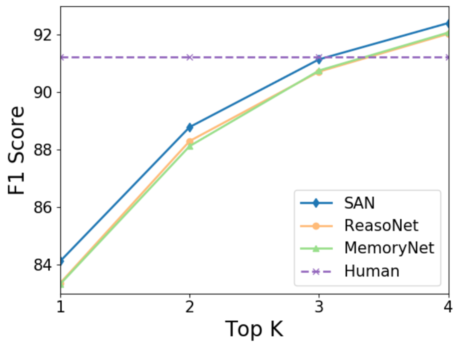
We are also interested in how sensitive are the results to the number of reasoning steps, which is a fixed hyper-parameter. Since we are using dropout, a natural question is whether we can extend the number of steps to an extremely large number. Table 4 shows the development set scores for $T = 1$ to $T = 10$. We observe that there is a gradual improvement as we increase $T = 1$ to $T = 5$, but after 5 steps the improvements have saturated. In fact, the EM/F1 scores drop slightly, but considering that the random initialization results in Table 3 show a standard deviation of 0.142 and a spread of 0.426 (for EM), we believe that the $T = 10$ result does not statistically differ from the $T = 5$ result. In summary, we think it is useful to perform some approximate hyper-parameter tuning for the number of steps, but it is not necessary to find the exact optimal value.

Finally, we test SAN on two Adversarial SQuAD datasets, AddSent and AddOneSent (Jia and Liang, 2017), where the passages contain

[4]Note the Dev EM/F1 scores of ReasoNet in Table 1 do not match those of ReasoNet++ in Table 2. While the answer module is the same architecture, the lower encoding layers are different.



(a) EM comparison on different systems.



(b) F1 score comparison on different systems.

Figure 3: K-Best Oracle results

Seed#	EM	F1	Seed#	EM	F1
Seed 1	76.24	84.06	Seed 6	76.23	83.99
Seed 2	76.30	**84.13**	Seed 7	**76.35**	84.09
Seed 3	**75.92**	83.90	Seed 8	76.07	83.71
Seed 4	76.00	83.95	Seed 9	75.93	**83.85**
Seed 5	76.12	83.99	Seed 10	76.15	84.11
Mean: 76.131, Std. deviation: 0.142 (EM)					
Mean: 83.977, Std. deviation: 0.126 (F1)					

Table 3: **Robustness of SAN (5-step) on different random seeds for initialization**: best and worst scores are boldfaced. Note that our official submit is trained on seed 1.

SAN	EM	F1	SAN	EM	F1
1 step	**75.38**	**83.29**	6 step	75.99	83.72
2 step	75.43	83.41	7 step	76.04	83.92
3 step	75.89	83.57	8 step	76.03	83.82
4 step	75.92	83.85	9 step	75.95	83.75
5 step	**76.24**	**84.06**	10 step	76.04	83.89

Table 4: **Effect of number of steps**: best and worst results are boldfaced.

auto-generated adversarial distracting sentences to fool computer systems that are developed to answer questions about the passages. For example, AddSent is constructed by adding sentences that look similar to the question, but do not actually contradict the correct answer. AddOneSent is constructed by appending a random human-approved sentence to the passage.

We evaluate the single SAN model (i.e., the one presented in Table 2) on both AddSent and AddOneSent. The results in Table 5 show that SAN achieves the new state-of-the-art performance and SAN's superior result is mainly attributed to the multi-step answer module, which leads to significant improvement in F1 score over the Standard 1-step answer module, i.e., +1.2 on AddSent and +0.7 on AddOneSent.

5.2 Is it possible to use different numbers of steps in test vs. train?

For practical deployment scenarios, prediction speed at test time is an important criterion. Therefore, one question is whether SAN can train with, e.g. $T = 5$ steps but test with $T = 1$ steps. Table 6 shows the results of a SAN trained on $T = 5$ steps, but tested with different number of steps. As expected, the results are best when T matches during training and test; however, it is important to note that small numbers of steps $T = 1$ and $T = 2$ nevertheless achieve strong results. For example, prediction at $T = 1$ achieves 75.58, which outperforms a standard 1-step model (75.14 EM) as in Table 1 that has approximate equivalent prediction time.

5.3 How does the training time compare?

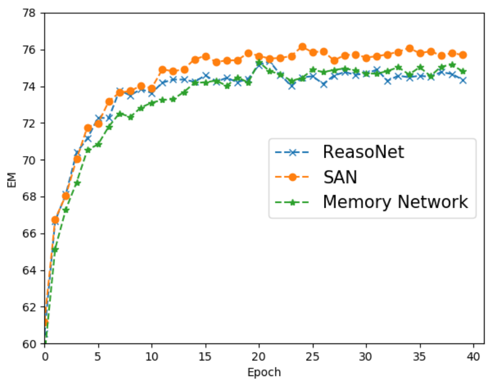
The average training time per epoch is comparable: our implementation running on a GTX Titan X is 22 minutes for 5-step memory net, 30 minutes for ReasoNet, and 24 minutes for SAN. The learning curve is shown in Figure 4. We observe that all systems improve at approximately the same rate up to 10 or 15 epochs. However, SAN continues to improve afterwards as other models start to saturate. This observation is consistent with previous works using dropout (Srivastava et al., 2014). We believe that while training time per epoch is similar between SAN and other models, it is recommended to train SAN for more epochs in order to achieve gains in EM/F1.

Single model:	AddSent	AddOneSent
LR (Rajpurkar et al., 2016)	23.2	30.3
SEDT (Liu et al., 2017a)	33.9	44.8
BiDAF (Seo et al., 2016)	34.3	45.7
jNet (Zhang et al., 2017)	37.9	47.0
ReasoNet(Shen et al., 2017)	39.4	50.3
RaSoR(Lee et al., 2016)	39.5	49.5
Mnemonic(Hu et al., 2017)	**46.6**	56.0
QANet(Yu et al., 2018)	45.2	55.7
Standard 1-step in Table 1	45.4	55.8
SAN	**46.6**	**56.5**

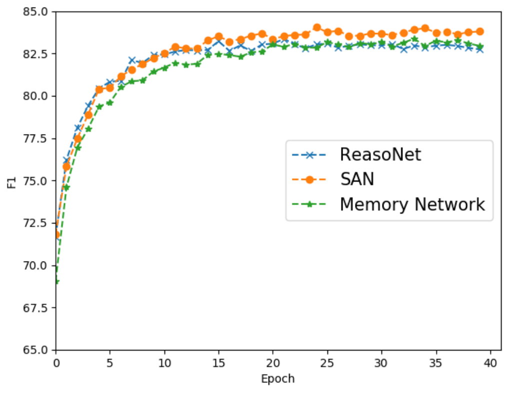
Table 5: Test performance on the adversarial SQuAD dataset in F1 score.

$T =$	EM	F1	$T =$	EM	F1
1	75.58	83.86	4	76.12	83.98
2	75.85	83.90	5	76.24	84.06
3	75.98	83.95	10	75.89	83.88

Table 6: **Prediction on different steps T.** Note that the SAN model is trained using 5 steps.



(a) EM



(b) F1

Figure 4: Learning curve measured on Dev set.

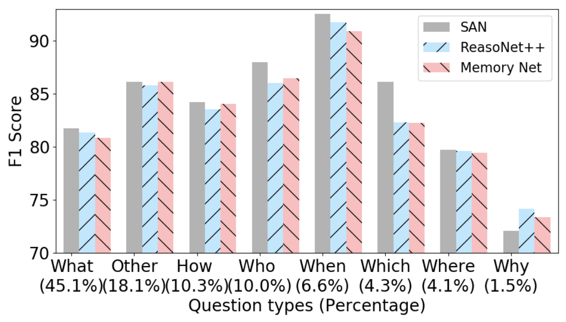


Figure 5: Score breakdown by question type.

5.4 How does SAN perform by question type?

To see whether SAN performs well on a particular type of question, we divided the development set by questions type based on their respective Wh-word, such as “who” and “where”. The score breakdown by F1 is shown in Figure 5. We observe that SAN seems to outperform other models uniformly across all types. The only exception is the Why questions, but there is too little data to derive strong conclusions.

5.5 Experiments results on MS MARCO

MS MARCO (Nguyen et al., 2016) is a large scale real-word RC dataset which contains 100,100 (100K) queries collected from anonymized user logs from the Bing search engine. The characteristic of MS MARCO is that all the questions are real user queries and passages are extracted from real web documents. For each query, approximate 10 passages are extracted from public web documents. The answers are generated by humans. The data is partitioned into a 82,430 training, a 10,047 development and 9,650 test tuples. The evaluation metrics are BLEU(Papineni et al., 2002) and ROUGE-L (Lin, 2004) due to its free-form text answer style. To apply the same RC model, we search for a span in MS MARCO’s passages that maximizes the ROUGE-L score with the raw free-form answer. It has an upper bound of 93.45 BLEU and 93.82 ROUGE-L on the development set.

The MS MARCO dataset contains multiple passages per query. Our model as shown in Figure 2 is developed to generate answer from a single passage. Thus, we need to extend it to handle multiple passages. Following (Shen et al., 2017), we take two steps to generate an answer to a query Q from J passages, $P^1, ..., P^J$. First, we run SAN on ev-

$SingleModel$	ROUGE	BLEU
ReasoNet++(Shen et al., 2017)	38.01	38.62
V-Net(Wang et al., 2018)	45.65	-
Standard 1-step in Table 1	42.30	42.39
SAN	**46.14**	**43.85**

Table 7: **MS MARCO devset results**.

ery (P^j, Q) pair, generating J candidate answer spans, one from each passage. Then, we multiply the SAN score of each candidate answer span with its relevance score $r(P^j, Q)$ assigned by a passage ranker, and output the span with the maximum score as the answer. In our experiments, we use the passage ranker described in (Liu et al., 2018)[5]. The ranker is trained on the same MS MARCO training data, and achieves 37.1 p@1 on the development set.

The results in Table 7 show that SAN outperforms V-Net (Wang et al., 2018) and becomes the new state of the art[6].

6 Related Work

The recent big progress on MRC is largely due to the availability of the large-scale datasets (Rajpurkar et al., 2016; Nguyen et al., 2016; Richardson et al., 2013; Hill et al., 2016), since it is possible to train large end-to-end neural network models. In spite of the variety of model structures and attenion types (Bahdanau et al., 2015; Chen et al., 2016; Xiong et al., 2016; Seo et al., 2016; Shen et al., 2017; Wang et al., 2017), a typical neural network MRC model first maps the symbolic representation of the documents and questions into a neural space, then search answers on top of it. We categorize these models into two groups based on the difference of the answer module: single-step and multi-step reasoning. The key difference between the two is what strategies are applied to search the final answers in the neural space.

A single-step model matches the question and document only once and produce the final answers. It is simple yet efficient and can be trained using the classical back-propagation algorithm, thus it is adopted by most systems (Chen et al., 2016; Seo et al., 2016; Wang et al., 2017; Liu et al., 2017b; Chen et al., 2017; Weissenborn et al., 2017; Hu et al., 2017). However, since humans often solve question answering tasks by re-reading and re-digesting the document multiple times before reaching the final answers (this may be based on the complexity of the questions/documents), it is natural to devise an iterative way to find answers as multi-step reasoning.

Pioneered by (Hill et al., 2016; Dhingra et al., 2016; Sordoni et al., 2016; Kumar et al., 2015), who used a predetermined fixed number of reasoning steps, Shen et al (2016; 2017) showed that multi-step reasoning outperforms single-step ones and dynamic multi-step reasoning further outperforms the fixed multi-step ones on two distinct MRC datasets (SQuAD and MS MARCO). But these models have to be trained using reinforcement learning methods, e.g., policy gradient, which are tricky to implement due to the instability issue. Our model is different in that we fix the number of reasoning steps, but perform stochastic dropout to prevent step bias. Further, our model can also be trained by using the back-propagation algorithm, which is simple and yet efficient.

7 Conclusion

We introduce Stochastic Answer Networks (SAN), a simple yet robust model for machine reading comprehension. The use of stochastic dropout in training and averaging in test at the answer module leads to robust improvements on SQuAD, outperforming both fixed step memory networks and dynamic step ReasoNet. We further empirically analyze the properties of SAN in detail. The model achieves results competitive with the state-of-the-art on the SQuAD leaderboard, as well as on the Adversarial SQuAD and MS MARCO datasets. Due to the strong connection between the proposed model with memory networks and ReasoNet, we would like to delve into the theoretical link between these models and its training algorithms. Further, we also would like to explore SAN on other tasks, such as text classification and natural language inference for its generalization in the future.

Acknowledgments

We thank Pengcheng He, Yu Wang and Xinying Song for help to set up dockers. We also thank Pranav Samir Rajpurkar for help on SQuAD evaluations, and the anonymous reviewers for valuable discussions and comments.

[5]It is the same model structure as (Liu et al., 2018) by using softmax over all candidate passages. A simple baseline, TF-IDF, obtains 20.1 p@1 on MS MARCO development.

[6]The official evaluation on MS MARCO on test is closed, thus here we only report the results on the development set.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR2015)* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. http://www.aclweb.org/anthology/P16-1223.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389* .

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR* .

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

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798* .

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341* .

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2021–2031.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR* abs/1506.07285. http://arxiv.org/abs/1506.07285.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* .

Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. 2017a. Structural embedding of syntactic trees for machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 815–824.

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017b. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504* .

Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888* .

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107* .

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. *ArXiv e-prints* .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 193–203.

S. Salant and J. Berant. 2017. Contextualized Word Representations for Reading Comprehension. *ArXiv e-prints* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284* .

Yelong Shen, Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2017. An empirical analysis of multiple-turn reasoning strategies in reading comprehension tasks. *arXiv preprint arXiv:1711.03230* .

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 189–198.

Y. Wang, K. Liu, J. Liu, W. He, Y. Lyu, H. Wu, S. Li, and H. Wang. 2018. Multi-Passage Machine Reading Comprehension with Cross-Passage Answer Verification. *ArXiv e-prints* .

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *arXiv preprint arXiv:1703.04816* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106* .

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension.

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv preprint arXiv:1703.04617* .

Multi-Granularity Hierarchical Attention Fusion Networks for Reading Comprehension and Question Answering

Wei Wang*, **Ming Yan***, **Chen Wu***
Alibaba Group, 969 West Wenyi Road, Hangzhou 311121, China
{hebian.ww,ym119608,wuchen.wc}@alibaba-inc.com

Abstract

This paper describes a novel hierarchical attention network for reading comprehension style question answering, which aims to answer questions for a given narrative paragraph. In the proposed method, attention and fusion are conducted horizontally and vertically across layers at different levels of granularity between question and paragraph. Specifically, it first encode the question and paragraph with fine-grained language embeddings, to better capture the respective representations at semantic level. Then it proposes a multi-granularity fusion approach to fully fuse information from both global and attended representations. Finally, it introduces a hierarchical attention network to focuses on the answer span progressively with multi-level soft-alignment. Extensive experiments on the large-scale SQuAD and TriviaQA datasets validate the effectiveness of the proposed method. At the time of writing the paper (Jan. 12th 2018), our model achieves the first position on the SQuAD leaderboard for both single and ensemble models. We also achieves state-of-the-art results on TriviaQA, AddSent and AddOneSent datasets.

1 Introduction

As a brand new field in question answering community, reading comprehension is one of the key problems in artificial intelligence, which aims to read and comprehend a given text, and then answer questions based on it. This task is challenging which requires a comprehensive understanding of natural languages and the ability to do further inference and reasoning. Restricted by the limited volume of the annotated dataset, early studies mainly rely on a pipeline of NLP models to complete this task, such as semantic parsing and linguistic annotation (Das et al., 2014). Not until the release of large-scale cloze-style dataset, such as Children's Book Test (Hill et al., 2015) and CNN/Daily Mail (Hermann et al., 2015), some preliminary end-to-end deep learning methods have begun to bloom and achieve superior results in reading comprehension task (Hermann et al., 2015; Chen et al., 2016; Cui et al., 2016).

However, these cloze-style datasets still have their limitations, where the goal is to predict the single missing word (often a named entity) in a passage. It requires less reasoning than previously thought and no need to comprehend the whole passage (Chen et al., 2016). Therefore, Stanford publish a new large-scale dataset SQuAD (Rajpurkar et al., 2016), in which all the question and answers are manually created through crowdsourcing. Different from cloze-style reading comprehension dataset, SQuAD constrains answers to all possible text spans within the reference passage, which requires more logical reasoning and content understanding.

Benefiting from the availability of SQuAD benchmark dataset, rapid progress has been made these years. The work (Wang and Jiang, 2016) and (Seo et al., 2016) are among the first to investigate into this dataset, where Wang and Jiang propose an end-to-end architecture based on match-LSTM and pointer networks (Wang and Jiang, 2016), and Seo et al. introduce the bi-directional attention flow network which captures the question-document context at different levels of granularity (Seo et al., 2016). Chen et al. devise a simple and effective document reader, by introducing a bilinear match function and a few manual features (Chen et al., 2017a). Wang et al. propose a gated attention-based recurrent network where self-match attention mechanism is first incorporated (Wang et al., 2017). In (Liu et al., 2017b) and (Shen et al., 2017), the multi-turn memory networks are designed to simulate multi-step reasoning in machine reading comprehension.

The idea of our approach derives from the normal human reading pattern. First, people scan through the whole passage to catch a glimpse of the main body of the passage. Then with the question in mind, people make connection between passage and question, and understand the main intent of the question related with the passage theme. A rough answer span is then located from the passage and the attention can be focused on to the located context. Finally, to prevent from forgetting the question, people come back to the question and select a best answer according to the previously located answer span.

Inspired by this, we propose a hierarchical attention network which can gradually focus the attention on the right part of the answer boundary, while capturing the relation between the question and passage at different levels of granularity, as illustrated in Figure 1. Our model mainly consists of three joint layers: 1) encoder layer where pre-trained language models and recurrent neural networks are used to build representation for questions and passages separately; 2) attention layer in which hierarchical attention networks are designed to capture the relation between question and passage at different levels of granularity; 3) match layer where refined question and passage are matched under a pointer-network (Vinyals et al., 2015) answer boundary predictor.

In encoder layer, to better represent the questions and passages in multiple aspects, we combine two different embeddings to give the fundamental word representations. In addition to the typical glove word embeddings, we also utilize the ELMo embeddings (Peters et al., 2018) derived from a pre-trained language model, which shows superior performance in a wide range of NLP problems. Different from the original fusion way for intermediate layer representations, we design a representation-aware fusion method to compute the output ELMo embeddings and the context information is also incorporated by further passing through a bi-directional LSTM network.

The key in machine reading comprehension solution lies in how to incorporate the question context into the paragraph, in which attention mechanism is most widely used. Recently, many different attention functions and types have been designed (Xiong et al., 2016; Seo et al., 2016; Wang et al., 2017), which aims at properly aligning the question and passage. In our attention layer, we propose a hierarchical attention network by leveraging both the co-attention and self-attention mechanism, to gradually focus our attention on the best answer span. Different from the previous attention-based methods, we constantly complement the aligned representations with global information from the previous layer, and an additional fusion layer is used to further refine the representations. In this way, our model can make some minor adjustment so that the attention will always be on the right place.

Based on the refined question and passage representation, a bilinear match layer is finally used to identify the best answer span with respect to the question. Following the work of (Wang and Jiang, 2016), we predict the start and end boundary within a pointer-network output layer.

The proposed method achieves state-of-the-art results against strong baselines. Our single model achieves 79.2% EM and 86.6% F1 score on the hidden test set, while the ensemble model further boosts the performance to 82.4% EM and 88.6% F1 score. At the time of writing the paper (Jan. 12th 2018), our model SLQA+ (Semantic Learning for Question Answering) achieves the first position on the SQuAD leaderboard [1] for both single and ensemble models. Besides, we are also among the first to surpass human EM performance on this golden benchmark dataset.

2 Related Work

2.1 Machine Reading Comprehension

Traditional reading comprehension style question answering systems rely on a pipeline of NLP models, which make heavy use of linguistic annotation, structured world knowledge, semantic parsing and similar NLP pipeline outputs (Hermann et al., 2015). Recently, the rapid progress of machine reading comprehension has largely benefited from the availability of large-scale benchmark datasets and it is possible to train large end-to-end neural network models. Among them, CNN/Daily Mail (Hermann et al., 2015) and Children's Book Test (Hill et al., 2015) are the first

[1] https://rajpurkar.github.io/SQuAD-explorer/

large-scale datasets for reading comprehension task. However, these datasets are in cloze-style, in which the goal is to predict the missing word (often a named entity) in a passage. Moreover, Chen at al. have also shown that these cloze-style datasets requires less reasoning than previously thought (Chen et al., 2016). Different from the previous datasets, the SQuAD provides a more challenging benchmark dataset, where the goal is to extract an arbitrary answer span from the original passage.

2.2 Attention-based Neural Networks

The key in MRC task lies in how to incorporate the question context into the paragraph, in which attention mechanism is most widely used. In spite of a variety of model structures and attention types (Cui et al., 2016; Xiong et al., 2016; Seo et al., 2016; Wang et al., 2017; Clark and Gardner, 2017), a typical attention-based neural network model for MRC first encodes the symbolic representation of the question and passage in an embedding space, then identify answers with particular attention functions in that space. In terms of the question and passage attention or matching strategy, we roughly categorize these attention-based models into two large groups: one-way attention and two-way attention.

In one-way attention model, question is first summarized into a single vector and then directly matched with the passage. Most of the end-to-end neural network methods on the cloze-style datasets are based on this model (Hermann et al., 2015; Kadlec et al., 2016; Chen et al., 2016; Dhingra et al., 2016). Hermann et al. are the first to apply the attention-based neural network methods to MRC task and introduce an attentive reader and an impatient reader (Hermann et al., 2015), by leveraging a two layer LSTM network. Chen et al. (Chen et al., 2016) further design a bilinear attention function based on the attentive reader, which shows superior performance on CNN/Daily Mail dataset. However, part of information may be lost when summarizing the question and a fine-grained attention on both the question and passage words should be more reasonable.

Therefore, the two-way attention model unfolds both the question and passage into respective word embeddings, and compute the attention in a two-dimensional matrix. Most of the top-ranking methods on SQuAD leaderboard are based on this attention mechanism (Wang et al., 2017; Huang et al., 2017; Xiong et al., 2017; Liu et al., 2017b,a). (Cui et al., 2016) and (Xiong et al., 2016) introduce the co-attention mechanism to better couple the representations of the question and document. Seo et al. propose a bi-directional attention flow network to capture the relevance at different levels of granularity (Seo et al., 2016). (Wang et al., 2017) further introduce the self-attention mechanism to refine the representation by matching the passage against itself, to better capture the global passage information. Huang et al. introduce a fully-aware attention mechanism with a novel *history-of-word* concept (Huang et al., 2017).

We propose a hierarchical attention network by leveraging both co-attention and self-attention mechanisms in different layers, which can capture the relevance between the question and passage at different levels of granularity. Different from the above methods, we further devise a fusion function to combine both the aligned representation and the original representation from the previous layer within each attention. In this way, the model can always focus on the right part of the passage, while keeping the global passage topic in mind.

3 Machine Comprehension Model

3.1 Task Description

Typical machine comprehension systems take an evidence text and a question as input, and predict a span within the evidence that answers the question. Based on this definition, given a passage and a question, the machine needs to first read and understand the passage, and then finds the answer to the question. The passage is described as a sequence of word tokens $\mathrm{P} = \left\{\mathrm{w}_t^P\right\}_{t=1}^{n}$ and the question is described as $\mathrm{Q} = \left\{\mathrm{w}_t^Q\right\}_{t=1}^{m}$, where n is the number of words in the passage, and m is the number of words in the question. In general, $\mathrm{n} \gg \mathrm{m}$. The answer can have different types depending on the task. In the SQuAD dataset (Rajpurkar et al., 2016), the answer A is guaranteed to be a continuous span in the passage P. The object function for machine reading comprehension is to learn a function $\mathrm{f(q,p)} = \arg\max_{\mathrm{a} \in \mathrm{A(p)}} \mathrm{P(a|q,p)}$. The training data is a set of the question, passage and answer tuples $< \mathrm{Q}, \mathrm{P}, \mathrm{A} >$.

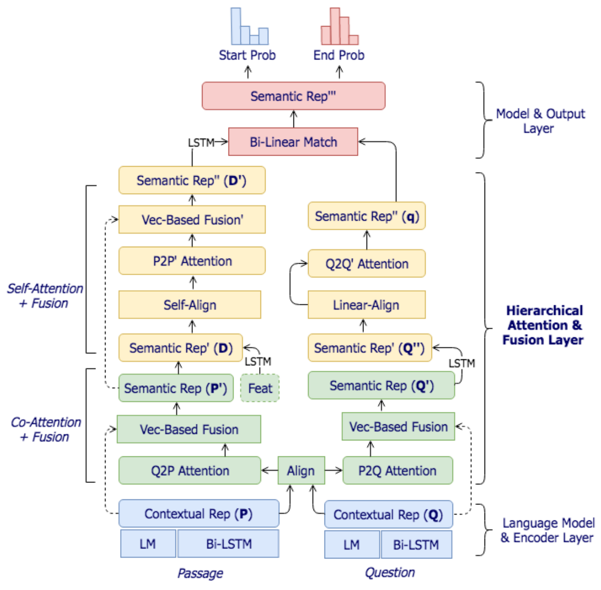


Figure 1: Hierarchical Attention Fusion Network.

3.2 Encode-Interaction-Pointer Framework

We will now describe our framework from the bottom up. As show in Figure 1, the proposed framework consists of four typical layers to learn different concepts of semantic representations:

- **Encoder Layer** as a language model, utilizes contextual cues from surrounding words to refine the embedding of the words. It converts the passage and question from tokens to semantic representation;
- **Attention Layer** attempts to capture relations between question and passage. Besides the aligned context, the contextual embeddings are also merged by a fusion function. Moreover, the multi-level of this operation forms a "working memory";
- **Match Layer** employs a bi-linear match function to compute the relevance between the question and passage representation on a span level;
- **Output Layer** uses a pointer network to search the answer span of question.

The main contribution of this work is the attention layer, in order to capture the relationship between question and passage, a hierarchical strategy is used to progressively make the answer boundary clear with the refined attention mechanism. A fine-grained fusion function is also introduced to better align the contextual representations from different levels. The detailed description of the model is provided as follows.

3.3 Hierarchical Attention Fusion Network

Our design is based on a simple but natural intuition: performing fine-grained mechanism requires first to roughly see the potential answer domain and then progressively locate the most discriminative parts of the domain.

The overall framework of our Hierarchical Attention Fusion Network is shown in Figure 1. It consists of several parts: a basic co-attention layer with shallow semantic fusion, a self-attention layer with deep semantic fusion and a memory-wise bilinear alignment function. The proposed network has two distinctive characteristics: (i) A fine-grained fusion approach to blend attention vectors for a better understanding of the relationship between question and passage; (ii) A multi-granularity attention mechanism applied at the word and sentence-level, enabling it to properly attend to the most important content when constructing the question and passage representation. Experiments conducted on SQuAD and adversarial example datasets (Jia and Liang, 2017) demonstrate that the proposed framework outperform previous methods by a large margin. Details of different components will be described in the following sections.

3.4 Language Model & Encoder Layer

Encoder layer of the model transform the discrete word tokens of question and passage to a sequence of continuous vector representations. We use a pre-trained word embedding model and a char embedding model to lay the foundation for our model. For the word embedding model, we adopt the popular glove embeddings (Pennington et al., 2014) which are widely used in deep learning-based NLP domain. For the char embedding model, the ELMo language model (Peters et al., 2018) is used due to its superior performance in a wide range of NLP tasks. As a result, we obtain two types of encoding vectors, i.e., word embeddings $\left\{\mathrm{e}_t^Q\right\}_{t=1}^m$, $\left\{\mathrm{e}_t^P\right\}_{t=1}^n$ and char embeddings $\left\{\mathrm{c}_t^Q\right\}_{t=1}^m$, $\left\{\mathrm{c}_t^P\right\}_{t=1}^n$.

To further utilize contextual cues from surrounding words to refine the embedding of the words, we then put a shared Bi-LSTM network on top of the embeddings provided by the previous layers to model the temporal interactions between words. Before feeding into the Bi-LSTM contextual network, we concat the word embeddings and char embeddings for a full understanding of each word. The final output of our encoder layer is shown as below,

$$\mathrm{u}_{\mathrm{t}}^{\mathrm{Q}} = \left[\mathrm{BiLSTM}_{\mathrm{Q}}([\mathrm{e}_{\mathrm{t}}^{\mathrm{Q}}, \mathrm{c}_{\mathrm{t}}^{\mathrm{Q}}]), \mathrm{c}_{\mathrm{t}}^{\mathrm{Q}}\right] \tag{1}$$

$$\mathrm{u}_{\mathrm{t}}^{\mathrm{P}} = \left[\mathrm{BiLSTM}_{\mathrm{P}}([\mathrm{e}_{\mathrm{t}}^{\mathrm{P}}, \mathrm{c}_{\mathrm{t}}^{\mathrm{P}}]), \mathrm{c}_{\mathrm{t}}^{\mathrm{P}}\right] \tag{2}$$

where we further concat the output of the contextual Bi-LSTM network with the pre-trained char embeddings for its good performance (Peters et al., 2018). This can be regarded as a residual connection between word representations in different levels.

3.5 Hierarchical Attention & Fusion Layer

The attention layer is responsible for linking and fusing information from the question and passage representation, which is the most critical in most MRC tasks. It aims to align the question and passage so that we can better locate on the most relevant passage span with respect to the question. We propose a hierarchical attention structure by combining the co-attention and self-attention mechanism in a multi-hop style. Besides, we think that the original representation and the aligned representation via attention can reflect the content semantics in different granularities. Therefore, we also apply a particular fusion function after each attention function, so that different levels of semantics can be better incorporated towards a better understanding.

3.5.1 Co-attention & Fusion

Given the question and passage representation $\mathrm{u}_{\mathrm{t}}^{\mathrm{Q}}$ and $\mathrm{u}_{\mathrm{t}}^{\mathrm{P}}$, a soft-alignment matrix S has been built to calculate the shallow semantic similarity between question and passage as follows:

$$\mathrm{S}_{\mathrm{ij}} = \mathrm{Att}(\mathrm{u}_{\mathrm{t}}^{\mathrm{Q}}, \mathrm{u}_{\mathrm{t}}^{\mathrm{P}}) = \mathrm{ReLU}(\mathrm{W}_{\mathrm{lin}}^{\top}\mathrm{u}_{\mathrm{t}}^{\mathrm{Q}})^{\top} \cdot \mathrm{ReLU}(\mathrm{W}_{\mathrm{lin}}^{\top}\mathrm{u}_{\mathrm{t}}^{\mathrm{P}}) \tag{3}$$

where $\mathrm{W}_{\mathrm{lin}}$ is a trainable weight matrix.

This decomposition avoids the quadratic complexity that is trivially parallelizable (Parikh et al., 2016). Now we use the unnormalized attention weights S_{ij} to compute the attentions between question and passage, which is further used to obtain the attended vectors in passage to question and question to passage direction, respectively.

P2Q Attention signifies which question words are most relevant to each passage word, given as below:

$$\alpha_{\mathrm{j}} = \mathrm{softmax}(\mathrm{S}_{:\mathrm{j}}) \tag{4}$$

where α_{j} represents the attention weights on the question words.

The aligned passage representation from question $\mathrm{Q} = \left\{\mathrm{u}_{\mathrm{t}}^{\mathrm{Q}}\right\}_{\mathrm{t}=1}^{\mathrm{m}}$ can thus be derived as,

$$\tilde{\mathrm{Q}}_{:\mathrm{t}} = \sum_{\mathrm{j}} \alpha_{\mathrm{tj}} \cdot \mathrm{Q}_{:\mathrm{j}}, \forall \mathrm{j} \in [1, ..., \mathrm{m}] \tag{5}$$

Q2P Attention signifies which passage words have the closest similarity to one of the question words and are hence critical for answering the question.

We utilize the same way to calculate this attention as in the passage to question attention (P2Q), except for that in the opposite direction:

$$\beta_{\mathrm{i}} = \mathrm{softmax}(\mathrm{S}_{\mathrm{i}:}) \tag{6}$$

$$\tilde{\mathrm{P}}_{\mathrm{k}:} = \sum_{\mathrm{i}} \beta_{\mathrm{ik}} \cdot \mathrm{P}_{\mathrm{i}:}, \forall \mathrm{i} \in [1, ..., \mathrm{n}] \tag{7}$$

where $\tilde{\mathrm{P}}$ indicates the weighted sum of the most important words in the passage with respect to the question.

With the aligned passage and question representations $\tilde{\mathrm{Q}}$ and $\tilde{\mathrm{P}}$ derived, a particular fusion unit has been designed to combine the original contextual representations and the corresponding attention vectors for question and passage separately:

$$\mathrm{P}' = \mathrm{Fuse}(\mathrm{P}, \tilde{\mathrm{Q}}) \tag{8}$$

$$\mathrm{Q}' = \mathrm{Fuse}(\mathrm{Q}, \tilde{\mathrm{P}}) \tag{9}$$

where $\mathrm{Fuse}(\cdot,\cdot)$ is a typical fusion kernel.

The simplest way of fusion is a concatenation or addition of the two representations, followed by some linear or non-linear transformation. Recently, a heuristic matching trick with difference and element-wise product is found effective in combining different representations (Mou et al., 2016; Chen et al., 2017b):

$$\mathrm{m}(\mathrm{P}, \tilde{\mathrm{Q}}) = \tanh(\mathrm{W}_{\mathrm{f}}[\mathrm{P}; \tilde{\mathrm{Q}}; \mathrm{P} \circ \tilde{\mathrm{Q}}; \mathrm{P} - \tilde{\mathrm{Q}}] + \mathrm{b}_{\mathrm{f}}) \tag{10}$$

where $\circ$ denotes the element-wise product, and W_{f}, b_{f} are trainable parameters. The output dimension is projected back to the same size as the original representation P or Q via the projected matrix W_{f}.

Since we find that the original contextual representations are important in reflecting the semantics at a more global level, we also introduce different levels of gating mechanism to incorporate the projected representations $\mathrm{m}(\cdot, \cdot)$ with the original contextual representations. As a result, the final fused representations of passage and question can be formulated as:

$$\mathrm{P}' = \mathrm{g}(\mathrm{P}, \tilde{\mathrm{Q}}) \cdot \mathrm{m}(\mathrm{P}, \tilde{\mathrm{Q}}) + (1 - \mathrm{g}(\mathrm{P}, \tilde{\mathrm{Q}})) \cdot \mathrm{P} \quad (11)$$

$$\mathrm{Q}' = \mathrm{g}(\mathrm{Q}, \tilde{\mathrm{P}}) \cdot \mathrm{m}(\mathrm{Q}, \tilde{\mathrm{P}}) + (1 - \mathrm{g}(\mathrm{Q}, \tilde{\mathrm{P}})) \cdot \mathrm{Q} \quad (12)$$

where $\mathrm{g}(\cdot, \cdot)$ is a gating function. To capture the relation between the representations in different granularities, we also design a scalar-based, a vector-based and a matrix-based sigmoid gating function, which are compared in Section 4.5.

3.5.2 Self-attention & Fusion

Borrowing the idea from wide and deep network (Cheng et al., 2016), manual features have also been added to combine with the outputs of previous layer for a more comprehensive representation. In our model, these features are concatenated with the refined question-aware passage representation as below:

$$\mathrm{D} = \mathrm{BiLSTM}([\mathrm{P}'; \mathrm{feat}_{\mathrm{man}}]) \quad (13)$$

where $\mathrm{feat}_{\mathrm{man}}$ denotes the word-level manual passage features.

In this layer, we separately consider the semantic representations of question and passage, and further refine the obtained information from the co-attention layer. Since fusing information among context words allows contextual information to flow close to the correct answer, the self-attention layer is used to further align the question and passage representation against itself, so as to keep the global sequence information in memory. Benefiting from the advantage of self-alignment attention in addressing the long-distance dependence (Wang et al., 2017), we adopt a self-alignment fusion process in this level. To allow for more freedom of the aligning process, we introduce a bilinear self-alignment attention function on the passage representation:

$$\mathrm{L} = \mathrm{softmax}(\mathrm{D} \cdot \mathrm{W}_1 \cdot \mathrm{D}^\top) \quad (14)$$

$$\tilde{\mathrm{D}} = \mathrm{L} \cdot \mathrm{D} \quad (15)$$

Another fusion function $\mathrm{Fuse}(\cdot, \cdot)$ is again adopted to combine the question-aware passage representation D and self-aware representation $\tilde{\mathrm{D}}$, as below:

$$\mathrm{D}' = \mathrm{Fuse}(\mathrm{D}, \tilde{\mathrm{D}}) \quad (16)$$

Finally, a bidirectional LSTM is used to get the final contextual passage representation:

$$\mathrm{D}'' = \mathrm{BiLSTM}(\mathrm{D}') \quad (17)$$

As for question side, since it is generally shorter in length and could be adequately represented with less information, we follow the question encoding method used in (Chen et al., 2017a) and adopt a linear transformation to encode the question representation to a single vector.

First, another contextual bidirectional LSTM network is applied on top of the fused question representation: $\mathrm{Q}'' = \mathrm{BiLSTM}(\mathrm{Q}')$. Then we aggregate the resulting hidden units into one single question vector, with a linear self-alignment:

$$\gamma = \mathrm{softmax}(\mathbf{w}_\mathrm{q}^\top \cdot \mathrm{Q}'') \quad (18)$$

$$\mathbf{q} = \sum_\mathrm{j} \gamma_\mathrm{j} \cdot \mathrm{Q}''_{:\mathrm{j}}, \forall \mathrm{j} \in [1, ..., \mathrm{m}] \quad (19)$$

where $\mathbf{w}_\mathrm{q}$ is a weight vector to learn, we self-align the refined question representation to a single vector according to the question self-attention weight, which can be further used to compute the matching with the passage words.

3.6 Model & Output Layer

Instead of predicting the start and end positions based only on D'', a top-level bilinear match function is used to capture the semantic relation between question $\mathbf{q}$ and paragraph D'' in a matching style, which actually works as a multi-hop matching mechanism.

Different from the co-attention layer that generates coarse candidate answers and the self-attention layer that focus the relevant context of passage to a certain intent of question, the top model layer uses a bilinear matching function to capture the interaction between outputs from previous layers and finally locate on the right answer span.

The start and end distribution of the passage words are calculated in a bilinear matching way as below,

$$\mathrm{P}_{\mathrm{start}} = \mathrm{softmax}(\mathbf{q} \cdot \mathrm{W}_\mathrm{s}^\top \cdot \mathrm{D}'') \quad (20)$$

$$\mathrm{P}_{\mathrm{end}} = \mathrm{softmax}(\mathbf{q} \cdot \mathrm{W}_\mathrm{e}^\top \cdot \mathrm{D}'') \quad (21)$$

where W_s and W_e are trainable matrices of the bilinear match function.

The output layer is application-specific, in MRC task, we use pointer networks to predict the start and end position of the answer, since it requires the model to find the sub-phrase of the passage to answer the question.

In training process, with cross entropy as metric, the loss for start and end position is the sum of the negative log probabilities of the true start and end indices by the predicted distributions, averaged over all examples:

$$\mathrm{L}(\theta) = -\frac{1}{\mathrm{N}}\sum_{\mathrm{i}}^{\mathrm{N}} \log \mathrm{p_s}(\mathrm{y_i^s}) + \log \mathrm{p_e}(\mathrm{y_i^e}) \quad (22)$$

where θ is the set of all trainable weights in the model, and $\mathrm{p_s}$ is the probability of start index, $\mathrm{p_e}$ is the probability of end index, respectively. $\mathrm{y_i^s}$ and $\mathrm{y_i^e}$ are the true start and end indices.

During prediction, we choose the answer span with the maximum value of $\mathrm{p_s} \cdot \mathrm{p_e}$ under a constraint that $\mathrm{s} \le \mathrm{e} \le \mathrm{s} + 15$, which is selected via a dynamic programming algorithm in linear time.

4 Experiments

In this section, we first present the datasets used for evaluation. Then we compare our end-to-end Hierarchical Attention Fusion Networks with existing machine reading models. Finally, we conduct experiments to validate the effectiveness of our proposed components. We evaluate our model on the task of question answering using recently released SQuAD and TriviaQA Wikipedia (Joshi et al., 2017), which have gained a huge attention over the past year. An adversarial evaluation for the Stanford Question Answering SQuAD is also used to demonstrate the robust of our model under adversarial attacks (Jia and Liang, 2017).

4.1 Dataset

We focus on the SQuAD dataset to train and evaluate our model. SQuAD is a popular machine comprehension dataset consisting of 100,000+ questions created by crowd workers on 536 Wikipedia articles. Each context is a paragraph from an article and the answer to each question is guaranteed to be a span in the context. The answer to each question is always a span in the context. The model is given a credit if its answer matches one of the human chosen answers. Two metrics are used to evaluate the model performance: Exact Match (EM) and a softer metric F1 score, which measures the weighted average of the precision and recall rate at a character level.

Table 1: The performance of our SLQA model and competing approaches on SQuAD.

	Dev Set	Test Set
Single model	**EM / F1**	**EM / F1**
LR Baseline (Rajpurkar et al., 2016)	40.0 / 51.0	40.4 / 51.0
Match-LSTM (Wang and Jiang, 2016)	64.1 / 73.9	64.7 / 73.7
DrQA (Chen et al., 2017a)	- / -	70.7 / 79.4
DCN+ (Xiong et al., 2017)	74.5 / 83.1	75.1 / 83.1
Interactive AoA Reader+ (Cui et al., 2016)	- / -	75.8 / 83.8
FusionNet (Huang et al., 2017)	- / -	76.0 / 83.9
SAN (Liu et al., 2017b)	76.2 / 84.0	76.8 / 84.4
AttentionReader+ (unpublished)	- / -	77.3 / 84.9
BiDAF + Self Attention + ELMo (Peters et al., 2018)	- / -	78.6 / 85.8
r-net+ (Wang et al., 2017)	- / -	79.9 / 86.5
SLQA+	**80.0 / 87.0**	**80.4 / 87.0**
Ensemble model		
FusionNet (Huang et al., 2017)	- / -	78.8 / 85.9
DCN+ (Xiong et al., 2017)	- / -	78.9 / 86.0
Interactive AoA Reader+ (Cui et al., 2016)	- / -	79.0 / 86.4
SAN (Liu et al., 2017b)	78.6 / 85.9	79.6 / 86.5
BiDAF + Self Attention + ELMo (Peters et al., 2018)	- / -	81.0 / 87.4
AttentionReader+ (unpublished)	- / -	81.8 / 88.2
r-net+ (Wang et al., 2017)	- / -	82.6 / 88.5
SLQA+	**82.0 / 88.4**	**82.4 / 88.6**
Human Performance	80.3 / 90.5	82.3 / 91.2

TriviaQA is a newly available machine comprehension dataset consisting of over 650K context-query-answer triples. The contexts are automatically generated from either Wikipedia or Web search results. The length of contexts in TriviaQA (average 2895 words) is much more longer than the one in SQuAD (average 122 words).

4.2 Training Details

We use the AdaMax optimizer, with a mini-batch size of 32 and initial learning rate of 0.002. A dropout rate of 0.4 is used for all LSTM layers. To directly optimize our target against the evaluation metrics, we further fine-tune the model with some well-defined strategy. During fine-tuning, Focal Loss (Lin et al., 2017) and Reinforce Loss which take F1 score as reward are incorporated with Cross Entropy Loss. The training process takes roughly 20 hours on a single Nvidia Tesla M40 GPU. We also train an ensemble model consisting of 15 training runs with the identical framework and hyper-parameters. At test time, we choose the answer with the highest sum of confidence scores amongst the 15 runs for each question.

4.3 Main Results

The results of our model and competing approaches on the hidden test set are summarized in Table 1. The proposed SLQA+ ensemble model achieves an EM score of 82.4 and F1 score of 88.6, outperforming all previous approaches, which validates the effectiveness of our hierarchical attention and fusion network structure.

We also conduct experiments on the adversarial

Table 2: The F1 scores of different models on AddSent and AddOneSent datasets (S: Single Model, E: Ensemble).

Model	**AddSent**	**AddOneSent**
Logistic (Rajpurkar et al., 2016)	23.2	30.4
Match-S (Wang and Jiang, 2016)	27.3	39.0
Match-E (Wang and Jiang, 2016)	29.4	41.8
BiDAF-S (Seo et al., 2016)	34.3	45.7
BiDAF-E (Seo et al., 2016)	34.2	46.9
ReasoNet-S (Shen et al., 2017)	39.4	50.3
ReasoNet-E (Shen et al., 2017)	39.4	49.8
Mnemonic-S (Hu et al., 2017)	46.6	56.0
Mnemonic-E (Hu et al., 2017)	46.2	55.3
QANet-S (Yu et al., 2018)	45.2	55.7
FusionNet-E (Huang et al., 2017)	51.4	60.7
SLQA-S (our)	**52.1**	**62.7**
SLQA-E (our)	**54.8**	**64.2**

SQuAD dataset (Jia and Liang, 2017) to study the robustness of the proposed model. In the dataset, one or more sentences are appended to the original SQuAD context, aiming to mislead the trained models. We use exactly the same model as in our SQuAD dataset, the performance comparison result is shown in Table 2. It can be seen that the proposed model can still get superior results than all the other competing approaches.

4.4 Ablations

In order to evaluate the individual contribution of each model component, we run an ablation study. Table 3 shows the performance of our model and its ablations on SQuAD dev set. The bi-linear alignment plus fusion between passage and question is most critical to the performance on both metrics which results in a drop of nearly 15%. The reason may be that in top-level attention layer, the similar semantics between question and passage are strong evidence to locate the correct answer span. The ELMo accounts for about 5% of the performance degradation, which clearly shows the effectiveness of language model. We conjecture that language model layer efficiently encodes different types of syntactic and semantic information about words-in-context, and improves the task performance. To evaluate the performance of hierarchical architecture, we reduce the multi-hop fusion with the standard LSTM network. The result shows that multi-hop fusion outperforms the standard LSTM by nearly 5% on both metrics.

4.5 Fusion Functions

In this section, we experimentally demonstrate how different choices of the fusion kernel impact the performance of our model. The compared fusion kernels are described as follows:

Simple Concat: a simple concatenation of two channel inputs.

Table 3: Ablation tests of SLQA single model on the SQuAD dev set.

SLQA single model	**EM / F1**
SLQA+	**80.0 / 87.0**
-Manual Features	79.2 / 86.2
-Language Embedding (ELMo)	77.6 / 84.9
-Self Matching	79.5 / 86.4
-Multi-hop	79.1 / 86.1
-Bi-linear Match	65.4 / 72.0
-Fusion (simple concat)	78.8 / 85.8
-Fusion, -Multi-hop	77.5 / 84.8
-Fusion, -Bi-linear Match	63.1 / 69.6

Table 4: Comparison of different fusion kernels on the SQuAD dev set.

Fusion Kernel	**EM / F1**
Simple Concat	78.8 / 85.8
Add Full Projection (FPU)	79.1 / 86.1
Scalar-based Fusion (SFU)	79.5 / 86.5
Vector-based Fusion (VFU)	**80.0 / 87.0**
Matrix-based Fusion (MFU)	79.8 / 86.8

Full Projection: the heuristic matching and projecting function as in Equ. 10.

Scalar-based Fusion: the gating function is a trainable scalar parameter (a coarse fusion level):

$$\mathrm{g}(\mathrm{P}, \tilde{\mathrm{Q}}) = \mathrm{g}_{\mathrm{p}} \tag{23}$$

where g_{p} is a trainable scalar parameter.

Vector-based Fusion: the gating function contains a weight vector to learn, which acts as a one-dimensional sigmoid gating,

$$\mathrm{g}(\mathrm{P}, \tilde{\mathrm{Q}}) = \sigma(\mathbf{w}_{\mathrm{g}}^{\top} \cdot [\mathrm{P}; \tilde{\mathrm{Q}}; \mathrm{P} \circ \tilde{\mathrm{Q}}; \mathrm{P} - \tilde{\mathrm{Q}}] + \mathrm{b}_{\mathrm{g}}) \tag{24}$$

where $\mathbf{w}_{\mathrm{g}}$ is trainable weight vector, b_{g} is trainable bias, and σ is sigmoid function.

Matrix-based Fusion: the gating function contains a weight matrix to learn, which acts as a two-dimensional sigmoid gating,

$$\mathrm{g}(\mathrm{P}, \tilde{\mathrm{Q}}) = \sigma(\mathrm{W}_{\mathrm{g}}^{\top} \cdot [\mathrm{P}; \tilde{\mathrm{Q}}; \mathrm{P} \circ \tilde{\mathrm{Q}}; \mathrm{P} - \tilde{\mathrm{Q}}] + \mathrm{b}_{\mathrm{g}}) \tag{25}$$

where W_{g} is a trainable weight matrix.

The comparison results of different fusion kernels can be found in Table 4. We can see that different fusion methods contribute differently to the final performances, and the vector-based fusion method performs best, with a moderate parameter size.

4.6 Attention Hierarchy and Function

In the proposed model, attention layer is the most important part of the framework. At the bottom of Table 5 we show the performances on SQuAD

Table 5: Comparison of different attention styles on the SQuAD dev set.

Attention Hierarchy	EM / F1
1-layer attention (only qp co-attention)	61.9 / 68.4
2-layer attention (add self-attention)	65.4 / 71.7
3-layer attention (add bilinear match)	**80.0 / 87.0**
Attention Function	EM / F1
dot product	62.9 / 69.3
linear attention	78.0 / 84.9
bilinear attention (linear + relu)	**80.0 / 87.0**
trilinear attention	78.9 / 85.8

Table 6: Published and unpublished results on the TriviaQA wikipedia leaderboard.

	Full	Verified
Model	EM / F1	EM / F1
BiDAF (Seo et al., 2016)	40.26 / 45.74	47.47 / 53.70
MEMEN (Pan et al., 2017)	43.16 / 46.90	49.28 / 55.83
M-Reader (Hu et al., 2017)	46.94 / 52.85	54.45 / 59.46
QANet (Yu et al., 2018)	51.10 / 56.60	53.30 / 59.20
document-qa (Clark and Gardner, 2017)	63.99 / 68.93	67.98 / 72.88
dirkweissenborn (unpublished)	64.60 / 69.90	72.77 / 77.44
SLQA-Single	**66.56 / 71.39**	**74.83 / 78.74**

for four common attention functions. Empirically, we find bilinear attention which add ReLU after linearly transforming does significantly better than the others.

At the top of Table 5 we show the effect of varying the number of attention layers on the final performance. We see a steep and steady rise in accuracy as the number of layers is increased from N = 1 to 3.

4.7 Experiments on TriviaQA

To further examine the robustness of the proposed model, we also test the model performance on TriviaQA dataset. The test performance of different methods on the leaderboard (on Jan. 12th 2018) is shown in Table 6. From the results, we can see that the proposed model can also obtain state-of-the-art performance in the more complex TriviaQA dataset.

5 Conclusions

We introduce a novel hierarchical attention network, a state-of-the-art reading comprehension model which conducts attention and fusion horizontally and vertically across layers at different levels of granularity between question and paragraph. We show that our proposed method is very powerful and robust, which outperforms the previous state-of-the-art methods in various large-scale golden MRC datasets: SQuAD, TriviaQA, AddSent and AddOneSent.

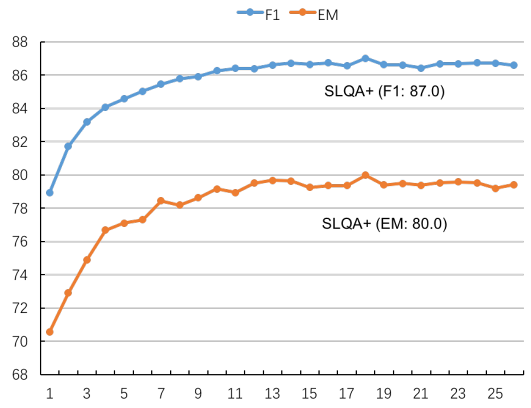


Figure 2: Learning curve of F1 / EM score on the SQuAD dev set

Acknowledgments

We thank the Stanford NLP Group and the University of Washington NLP Group for evaluating our results on the SQuAD and the TriviaQA test set.

References

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858.*

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051.*

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2017b. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289.*

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723.*

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423.*

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*.

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017a. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017b. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 130.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Joint Training of Candidate Extraction and Answer Selection for Reading Comprehension

Zhen Wang Jiachen Liu Xinyan Xiao Yajuan Lyu Tian Wu
Baidu Inc., Beijing, China
{wangzhen24, liujiachen, xiaoxinyan, lvyajuan, wutian}@baidu.com

Abstract

While sophisticated neural-based techniques have been developed in reading comprehension, most approaches model the answer in an independent manner, ignoring its relations with other answer candidates. This problem can be even worse in open-domain scenarios, where candidates from multiple passages should be combined to answer a single question. In this paper, we formulate reading comprehension as an extract-then-select two-stage procedure. We first extract answer candidates from passages, then select the final answer by combining information from all the candidates. Furthermore, we regard candidate extraction as a latent variable and train the two-stage process jointly with reinforcement learning. As a result, our approach has improved the state-of-the-art performance significantly on two challenging open-domain reading comprehension datasets. Further analysis demonstrates the effectiveness of our model components, especially the information fusion of all the candidates and the joint training of the extract-then-select procedure.

1 Introduction

Teaching machines to read and comprehend human languages is a long-standing objective in natural language processing. In order to evaluate this ability, reading comprehension (RC) is designed to answer questions through reading relevant passages. In recent years, RC has attracted intense interest. Various advanced neural models have been proposed along with newly released datasets (Hermann et al., 2015; Rajpurkar et al., 2016; Dunn et al., 2017; Dhingra et al., 2017b; He et al., 2017).

Q A	Cocktails: *Rum*, *lime*, and *cola* drink make a __________. **Cuba Libre**
P_1	**Daiquiri**, the custom of mixing *lime* with *rum* for a cooling drink on a hot Cuban day, has been around a long time.
P_2	Cocktail recipe for a **Daiquiri**, a classic *rum* and *lime* drink that every bartender should know.
P_3	Hemingway Special **Daiquiri**: Daiquiris are a family of cocktails whose main ingredients are *rum* and *lime* juice.
P_4	A homemade Cuba Libre Preparation To make a **Cuba Libre** properly, fill a highball glass with ice and half fill with *cola*.
P_5	The difference between the **Cuba Libre** and *Rum* is a *lime* wedge at the end.

Table 1: The answer candidates are in a **bold** font. The key information is marked in *italic*, which should be combined from different text pieces to select the correct answer "Cuba Libre".

Most existing approaches mainly focus on modeling the interactions between questions and passages (Dhingra et al., 2017a; Seo et al., 2017; Wang et al., 2017), paying less attention to information concerning answer candidates. However, when human solve this problem, we often first read each piece of text, collect some answer candidates, then focus on these candidates and combine their information to select the final answer. This collect-then-select process can be more significant in open-domain scenarios, which require the combination of candidates from multiple passages to answer one single question. This phenomenon is illustrated by the example in Table 1.

With this motivation, we formulate an extract-then-select two-stage architecture to simulate the above procedure. The architecture contains two components: (1) an extraction model, which generates answer candidates, (2) a selection model, which combines all these candidates and finds out the final answer. However, answer candidates to be focused on are often unobservable, as most RC datasets only provide golden answers. Therefore, we treat candidate extraction as a latent variable and train these two stages jointly with reinforcement learning (RL).

In conclusion, our work makes the following contributions:

1. We formulate open-domain reading comprehension as a two-stage procedure, which first extracts answer candidates and then selects the final answer. With joint training, we optimize these two correlated stages as a whole.

2. We propose a novel answer selection model, which combines the information from all the extracted candidates using an attention-based correlation matrix. As shown in experiments, the information fusion is greatly helpful for answer selection.

3. With the two-stage framework and the joint training strategy, our method significantly surpasses the state-of-the-art performance on two challenging public RC datasets Quasar-T (Dhingra et al., 2017b) and SearchQA (Dunn et al., 2017).

2 Related Work

In recent years, reading comprehension has made remarkable progress in methodology and dataset construction. Most existing approaches mainly focus on modeling sophisticated interactions between questions and passages, then use the pointer networks (Vinyals et al., 2015) to directly model the answers (Dhingra et al., 2017a; Wang and Jiang, 2017; Seo et al., 2017; Wang et al., 2017). These methods prove to be effective in existing close-domain datasets (Hermann et al., 2015; Hill et al., 2015; Rajpurkar et al., 2016).

More recently, open-domain RC has attracted increasing attention (Nguyen et al., 2016; Dunn et al., 2017; Dhingra et al., 2017b; He et al., 2017) and raised new challenges for question answering techniques. In these scenarios, a question is paired with multiple passages, which are often collected by exploiting unstructured documents or web data. Aforementioned approaches often rely on recurrent neural networks and sophisticated attentions, which are prohibitively time-consuming if passages are concatenated altogether. Therefore, some work tried to alleviate this problem in a coarse-to-fine schema. Wang et al. (2018a) combined a ranker for selecting the relevant passage and a reader for producing the answer from it. However, this approach only depended on one passage when producing the answer, hence put great demands on the precisions of both components. Worse still, this framework cannot handle the situation where multiple passages are needed to answer correctly. In consideration of evidence aggregation, Wang et al. (2018b) proposed a re-ranking method to resolve the above issue. However, their re-ranking stage was totally isolated from the candidate extraction procedure. Being different from the re-ranking perspective, we propose a novel selection model to combine the information from all the extracted candidates. Moreover, with reinforcement learning, our candidate extraction and answer selection models can be learned in a joint manner. Trischler et al. (2016) also proposed a two-step extractor-reasoner model, which first extracted K most probable single-token answer candidates and then compared the hypotheses with all the sentences in the passage. However, in their work, each candidate was considered isolatedly, and their objective only took into account the ground truths compared with our RL treatment.

The training strategy employed in our paper is reinforcement learning, which is inspired by recent work exploiting it into question answering problem. The above mentioned coarse-to-fine framework (Choi et al., 2017; Wang et al., 2018a) treated sentence selection as a latent variable and jointly trained the sentence selection module with the answer generation module via RL. Shen et al. (2017) modeled the multi-hop reasoning procedure with a termination state to decide when it is adequate to produce an answer. RL is suitable to capture this stochastic behavior. Hu et al. (2018) merely modeled the extraction process, using F1 as rewards in addition to maximum likelihood estimation. RL was utilized in their training process, as the F1 measure is not differentiable.

3 Two-stage RC Framework

In this work, we mainly consider the open-domain extractive reading comprehension. In this scenario, a given question Q is paired with multiple passages $\mathbb{P} = \{P_1, P_2, ..., P_N\}$, based on which we aim to find out the answer A. Moreover, the golden answers are almost subspans shown in

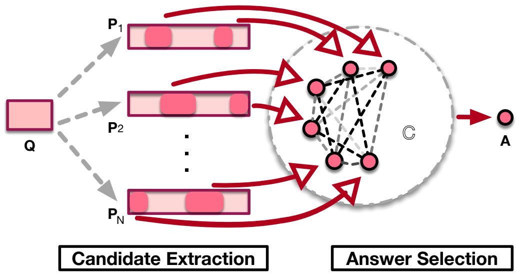


Figure 1: Two-stage RC Framework. The first part extracts candidates (denoted with circles) from all the passages. The second part establishes interactions among all these candidates to select the final answer. The different gray scales of dashed lines between candidates represent different intensities of interactions.

some passages in $\mathbb{P}$. Our main framework consists of two parts, which are: (1) extracting answer candidates $\mathbb{C} = \{C_1, C_2, ..., C_M\}$ from passages $\mathbb{P}$ and (2) selecting the final answer A from candidates $\mathbb{C}$. This process is illustrated in Figure 1. We design different models for each part and optimize them as a whole with joint reinforcement learning.

3.1 Candidate Extraction

We build candidate set $\mathbb{C}$ by independently extracting K candidates from each passage P_i according to the following distribution:

$$\begin{aligned} p(\mathbb{C}|Q,\mathbb{P}) &= \prod_i^N p(\{C_{ij}\}_{j=1}^K|Q,P_i) \\ \mathbb{C} &= \bigcup_{i=1}^N \{C_{ij}\}_{j=1}^K \end{aligned} \tag{1}$$

where C_{ij} denotes the jth candidate extracted from the ith passage. K is set as a constant number in our formulation. Taking K as 2 for an example, we denote each probability shown on the right side of Equation 1 through sampling without replacement:

$$\begin{aligned} p(\{C_{i1},C_{i2}\}) &= p(C_{i1})p(C_{i2})/(1-p(C_{i1})) \\ &+ p(C_{i1})p(C_{i2})/(1-p(C_{i2})) \end{aligned} \tag{2}$$

where we neglect Q, P_i to abbreviate the conditional distributions in Equation 1.

Consequently, the basic block of our candidate extraction stage turns out to be the distribution of each candidate $P(C_{ij}|Q,P_i)$. In the rest of this subsection, we will elaborate on the model architecture concerning candidate extraction, which is displayed in Figure 2.

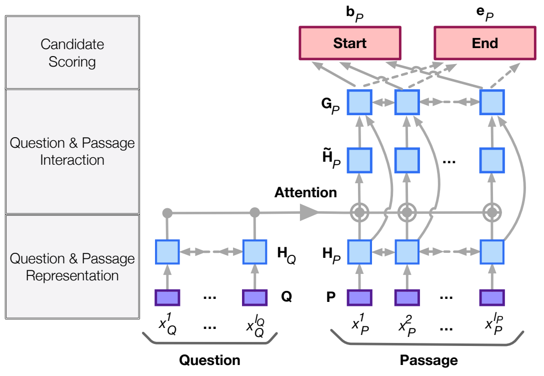


Figure 2: Candidate Extraction Model Architecture.

Question & Passage Representation Firstly, we embed the question $Q = \{x_Q^k\}_{k=1}^{l_Q}$ and its relevant passage $P = \{x_P^t\}_{t=1}^{l_P} \in \mathbb{P}$ with word vectors to form $\mathbf{Q} \in \mathbb{R}^{d_w \times l_Q}$ and $\mathbf{P} \in \mathbb{R}^{d_w \times l_P}$ respectively, where d_w is the dimension of word embeddings, l_Q and l_P are the length of Q and P.

We then feed $\mathbf{Q}$ and $\mathbf{P}$ to a bidirectional LSTM to form their contextual representations $\mathbf{H}_Q \in \mathbb{R}^{d_h \times l_Q}$ and $\mathbf{H}_P \in \mathbb{R}^{d_h \times l_P}$:

$$\begin{aligned} \mathbf{H}_Q &= \text{BiLSTM}(\mathbf{Q}) \\ \mathbf{H}_P &= \text{BiLSTM}(\mathbf{P}) \end{aligned} \tag{3}$$

Question & Passage Interaction Modeling the interactions between questions and passages is a critical step in reading comprehension. Here, we adopt the attention mechanism similar to (Lee et al., 2016) to generate question-dependent passage representation $\widetilde{\mathbf{H}}_P$. Assume $\mathbf{H}_Q = \{\mathbf{h}_Q^k\}_{k=1}^{l_Q}$, $\mathbf{H}_P = \{\mathbf{h}_P^t\}_{t=1}^{l_P}$, we have:

$$\begin{aligned} \alpha_{tk} &= \frac{e^{\mathbf{h}_Q^k \cdot \mathbf{h}_P^t}}{\sum_{k=1}^{l_Q} e^{\mathbf{h}_Q^k \cdot \mathbf{h}_P^t}} \quad 1 \le k \le l_Q, 1 \le t \le l_P \\ \widetilde{\mathbf{h}}_P^t &= \sum_{k=1}^{l_Q} \alpha_{tk}\mathbf{h}_Q^k \quad 1 \le t \le l_P \\ \widetilde{\mathbf{H}}_P &= \{\widetilde{\mathbf{h}}_P^t\}_{t=1}^{l_P} \end{aligned} \tag{4}$$

After concatenating two kinds of passage representations $\mathbf{H}_P$ and $\widetilde{\mathbf{H}}_P$, we use another bidirectional LSTM to get the final representation of every position in passage P as $\mathbf{G}_P \in \mathbb{R}^{d_g \times l_P}$:

$$\mathbf{G}_P = \text{BiLSTM}([\mathbf{H}_P; \widetilde{\mathbf{H}}_P]) \tag{5}$$

Candidate Scoring Then we use two linear transformations $\mathbf{w}_b \in \mathbb{R}^{1\times d_g}$ and $\mathbf{w}_e \in \mathbb{R}^{1\times d_g}$ to calculate the begin and the end scores for each position:

$$\begin{aligned}\{b_P^t\}_{t=1}^{l_Q} &= \mathbf{b}_P = \mathbf{w}_b\mathbf{G}_P \\ \{e_P^t\}_{t=1}^{l_Q} &= \mathbf{e}_P = \mathbf{w}_e\mathbf{G}_P\end{aligned} \tag{6}$$

At last, we model the probability of every subspan in passage P as a candidate $C = \{x_P^t\}_{t=C_b}^{C_e}$ according to its begin and end position:

$$p(C|Q,P) = \frac{exp(b_P^{C_b} + e_P^{C_e})}{\sum_{k=1}^{l_P}\sum_{t=k}^{l_P} exp(b_P^k + e_P^t)} \tag{7}$$

In this definition, the probabilities of all the valid answer candidates are already normalized.

3.2 Answer Selection

As the second part of our framework, the answer selection model finds out the most probable answer by calculating $p(C|Q,\mathbb{P},\mathbb{C})$ for each candidate $C \in \mathbb{C}$. The model architecture is illustrated in Figure 3.

Notably, selection model receives candidate set $\mathbb{C}$ as additional information. This more focused information allows the model to combine evidences from all the candidates, which would be useful for selecting the best answer.

For ease of understanding, we briefly describe the selection stage as follows. After being extracted from a single passage, a candidate borrows information from other candidates across different passages. With this global information, the passage is reread to confirm the correctness of the candidate further. The following are details about the selection model.

Question Representation Questions are fundamental for finding out the correct answer. As did for the extraction model, we embed the question Q with word vectors to form $\mathbf{Q} \in \mathbb{R}^{d_w \times l_Q}$. Then we use a bidirectional LSTM to establish its contextual representation:

$$\mathbf{S}_q = \text{BiLSTM}(\mathbf{Q}) \tag{8}$$

A max-pooling operation across all the positions is followed to get the condensed vector representation:

$$\mathbf{r}_q = \text{MaxPooling}(\mathbf{S}_q) \tag{9}$$

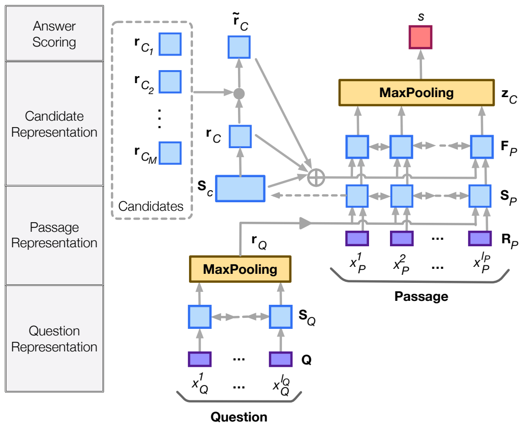


Figure 3: Answer Selection Model Architecture.

Passage Representation Assume the candidate C is extracted from the passage $P \in \mathbb{P}$. To be informed of C, we first build the representation of P. For every word in P, three kinds of features are utilized:

- Word embedding: each word expresses its basic feature with the word vector.
- Common word: the feature has value 1 when the word occurs in the question, otherwise 0.
- Question independent representation: the condensed representation $\mathbf{r}_q$.

With these features, information not only in Q but also in P is considered. By concatenating them, we get $\mathbf{r}_P^t$ corresponding to every position t in passage P. Then with another bidirectional LSTM, we fuse these features to form the contextual representation of P as $\mathbf{S}_P \in \mathbb{R}^{d_s \times l_P}$:

$$\begin{aligned}\mathbf{R}_P &= \{\mathbf{r}_P^t\}_{t=1}^{l_P} \\ \mathbf{S}_P &= \text{BiLSTM}(\mathbf{R}_P)\end{aligned} \tag{10}$$

Candidate Representation Candidates provide more focused information for answer selection. Therefore, for each candidate, we first build its independent representation according to its position in the passage, then construct candidates fused representation through combination of other correlated candidates.

Given the candidate $C = \{x_P^t\}_{t=C_b}^{C_e}$ in the passage P, we extract its corresponding span from $\mathbf{S}_P = \{\mathbf{s}_P^t\}_{t=1}^{l_P}$ to form $\mathbf{S}_C = \{\mathbf{s}_P^t\}_{t=C_b}^{C_e}$ as its contextual encoding. Moreover, we calculate its condensed vector representation through its begin and end positions:

$$\mathbf{r}_C = \tanh(\mathbf{W}_b \mathbf{s}_P^{C_b} + \mathbf{W}_e \mathbf{s}_P^{C_e}) \tag{11}$$

where $\mathbf{W}_b \in \mathbb{R}^{d_c \times d_s}$, $\mathbf{W}_e \in \mathbb{R}^{d_c \times d_s}$.

To model the interactions among all the answer candidates, we calculate the correlations of the candidate C, which is assumed to be indexed by j in $\mathbb{C}$, with others $\{C_m\}_{m=1, m\neq j}^{M}$ via attention mechanism:

$$V_{jm} = \mathbf{w}_v \tanh(\mathbf{W}_c \mathbf{r}_C + \mathbf{W}_o \mathbf{r}_{C_m}) \tag{12}$$

where $\mathbf{W}_c \in \mathbb{R}^{d_c \times d_c}$, $\mathbf{W}_o \in \mathbb{R}^{d_c \times d_c}$ and $\mathbf{w}_v \in \mathbb{R}^{1 \times d_c}$ are linear transformations to capture the intensity of each interaction.

In this way, we form a *correlation matrix* $\mathbf{V} \in \mathbb{R}^{M \times M}$, where M is the total number of candidates. With the correlation matrix, for the candidate C, we normalize its interactions via a $softmax$ operation, which emphasizes the influence of stronger interactions:

$$\alpha_m = \frac{e^{V_{jm}}}{\sum_{m=1, m\neq j}^{M} e^{V_{jm}}} \tag{13}$$

To take into account different influences of all the other candidates, it is sensible to generate a candidates fused representation according to the above normalized interactions:

$$\widetilde{\mathbf{r}}_C = \sum_{m=1, m\neq j}^{M} \alpha_m \mathbf{r}_{C_m} \tag{14}$$

In this formulation, all the other candidates contribute their influences to the fused representation by their interactions with C, thus information from different passages is gathered altogether. In our experiments, this kind of information fusion is the key point for performance improvements.

Passage Advanced Representation As more focused information of the candidate C is available, we are provided with a better way to confirm its correctness by rereading its corresponding passage P. Specifically, we equip each position t in P with following advanced features:

- Passage contextual representation: the former passage representation $\mathbf{s}_P^t$.
- Candidate-dependent passage representation: replace $\mathbf{H}_Q$ with $\mathbf{S}_C$ and $\mathbf{H}_P$ with $\mathbf{S}_P$ in Equation 4 to model the interactions between candidates and passages to form $\widetilde{\mathbf{s}}_P^t$.
- Candidate related distance feature: the relative distance to the candidate C can be a reference of the importance of each position.
- Candidate independent representation: use $\mathbf{r}_C$ to consider the concerned candidate C.
- Candidates fused representation: use $\widetilde{\mathbf{r}}_C$ to consider all the other candidates interacting with the concerned candidate C.

With these features, we capture the information from the question, the passages and all the candidates. By concatenating them, we get $\mathbf{u}_P^t$ in every position in the passage P. Combining these features with a bidirectional LSTM, we get:

$$\begin{aligned} \mathbf{U}_P &= \{\mathbf{u}_P^t\}_{t=1}^{l_P} \\ \mathbf{F}_P &= \mathrm{BiLSTM}(\mathbf{U}_P) \end{aligned} \tag{15}$$

Answer Scoring At last, the max pooling of each dimension of $\mathbf{F}_P$ is performed, resulting in a condensed vector representation, which contains all the concerned information in a candidate:

$$\mathbf{z}_C = \mathrm{MaxPooling}(\mathbf{F}_P) \tag{16}$$

The final score of this candidate as the answer is calculated via a linear transformation, which is then normalized across all the candidates:

$$\begin{aligned} s &= \mathbf{w}_z \mathbf{z}_C \\ p(C|Q, \mathbb{P}, \mathbb{C}) &= \frac{e^{s}}{\sum_{k=1}^{M} e^{s_k}} \end{aligned} \tag{17}$$

3.3 Joint Training with RL

In our formulation, the answer candidate set influences the result of answer selection to a large extent. However, with only golden answers provided in the training data, it is not apparent which candidates should be considered further.

To alleviate the above problem, we treat candidate extraction as a latent variable, jointly train the extraction model and the selection model with reinforcement learning. Formally, in the extraction and selection stages, two kinds of actions are modeled. The action space for the extraction model is to select from different candidate sets, which is formulated by Equation 1. The action space for the selection model is to select from all extracted candidates, which is formulated by Equation 17. Our goal is to select the final answer that leads to a high reward. Inspired by Wang et al. (2018a), we define the reward of a candidate to reflect its accordance with the golden answer:

$$r(C, A) = \begin{cases} 2 & \text{if } C == A \\ f1(C, A) & \text{else if } C \cap A \neq \varnothing \\ -1 & \text{else} \end{cases} \tag{18}$$

where $f1(.,.) \in [0, 1]$ is the function to measure word-level F1 score between two sequences. Incorporating this reward can alleviate the overstrict requirements set by traditional maximum likelihood estimation as well as keep consistent with our evaluation methods in experiments.

The learning objective becomes to maximize the expected reward modeled by our framework, where θ stands for all the parameters involved:

$$\begin{aligned} L(\theta) &= -E_{\mathbb{C} \sim P(\mathbb{C}|Q,\mathbb{P})}[E_{C \sim P(C|Q,\mathbb{P},\mathbb{C})} r(C, A)] \\ &= -E_{\mathbb{C} \sim P(\mathbb{C}|Q,\mathbb{P})}[\sum_{C} P(C|Q,\mathbb{P},\mathbb{C}) r(C, A)] \end{aligned} \tag{19}$$

Following REINFORCE algorithm, we approximate the gradient of the above objective with a sampled candidate set, $\mathbb{C} \sim P(\mathbb{C}|Q, \mathbb{P})$, resulting in the following form:

$$\begin{aligned} \nabla L(\theta) \approx -\sum_{C} \nabla P(C|Q,\mathbb{P},\mathbb{C}) r(C, A) \\ -\nabla log P(\mathbb{C}|Q,\mathbb{P})[\sum_{C} P(C|Q,\mathbb{P},\mathbb{C}) r(C, A)] \end{aligned} \tag{20}$$

4 Experiments

4.1 Datasets

We evaluate our models on two publicly available open-domain RC datasets, which are commonly adopted in related work.

Quasar-T (Dhingra et al., 2017b) consists of 43,000 open-domain trivia questions and corresponding answers obtained from various internet sources. Each question is paired with 100 sentence-level passages retrieved from ClueWeb09 (Callan et al., 2009) based on Lucene.

SearchQA (Dunn et al., 2017) starts from existing question-answer pairs, which are crawled from J!Archive, and is augmented with text snippets retrieved by Google, resulting in more than 140,000 question-answer pairs with each pair having 49.6 snippets on average.

The detailed statistics of these two datasets is shown in Table 2.

	#q(train)	#q(dev)	#q(test)	#p
Quasar-T	28,496	3,000	3,000	100
SearchQA	99,811	13,893	27,247	50

Table 2: The statistics of our experimental datasets. #q represents the number of questions for each split of the datasets. #p is the number of passages for each question.

4.2 Model Settings

We initialize word embeddings with the 300-dimensional Glove vectors[1]. All the bidirectional LSTMs hold 1 layer and 100 hidden units. All the linear transformations take the size of 100 as output dimension. The common word feature and the candidate related distance feature are embedded with vectors of dimension 4 and 50 respectively. By default, we set K as 2 in Equation 1, which means each passage generates two candidates based on the extraction model.

For ease of training, we first initialize our models by maximum likelihood estimation and fine-tune them with RL. The similar training strategy is commonly employed when RL process is involved (Ranzato et al., 2015; Li et al., 2016a; Hu et al., 2018). To pre-train the extraction model, we only use passages containing ground truths as training data. The log likelihood of Equation 7 is taken as the training objective for each question and passage pair. After pre-training the extraction model, we use it to generate two top-scoring candidates from each passage, forming the training data to pre-train our selection model, and maximize the log likelihood of the Equation 17 as our second objective. In pre-training, we use the batch size of 30 for the extraction model, 20 for the selection model and RMSProp (Tieleman and Hinton, 2012) with an initial learning rate of 2e-3. In fine-tuning with RL, we use the batch size of 5 and RMSProp with an initial learning rate of 1e-4. Also, we use a dropout rate of 0.1 in each training procedure.

4.3 Experimental Results

In addition to results of previous work, we add two baselines to demonstrate the effectiveness of our framework. The first baseline only applies the extraction model to score the answers, which is aimed at explaining the importance of the selection model. The second one only uses the pre-trained extraction model and selection model

[1]http://nlp.stanford.edu/data/wordvecs/glove.840B.300d.zip

	Quasar-T		SearchQA	
	EM	F1	EM	F1
GA (Dhingra et al., 2017a)	26.4	26.4	-	-
BIDAF (Seo et al., 2017)	25.9	28.5	28.6	34.6
AQA (Buck et al., 2018)	-	-	38.7	45.6
R^3 (Wang et al., 2018a)	35.3	41.7	49.0	55.3
Re-Ranker (Wang et al., 2018b)				
Strength-Based Re-Ranker (Probability)	36.1	42.4	50.4	56.5
Strength-Based Re-Ranker (Counting)	37.1	46.7	54.2	61.6
Coverage-Based Re-Raner	40.6	49.1	53.6	60.6
Full Re-Ranker	42.3	49.6	57.0	63.2
Our Methods				
Extraction Model	35.4	41.6	44.7	51.2
Extraction + Selection (Isolated Training)	41.6	49.5	49.7	56.6
Extraction + Selection (Joint Training)	**45.9**	**53.9**	**58.3**	**64.2**

Table 3: Experimental results on the test set of Quasar-T and SearchQA. Full re-ranker is the ensemble of three different re-rankers in (Wang et al., 2018b).

to illustrate the benefits from our joint training schema.

The often used evaluation metrics for extractive RC are exact match (EM) and F1 (Rajpurkar et al., 2016). The experimental results on Quasar-T and SearchQA are shown in Table 3.

As seen from the results on Quasar-T, our quite simple extraction model alone almost reaches the state-of-the-art result compared with other methods without re-rankers. The combination of the extraction and selection models exceeds our extraction baseline by a great margin, and also results in performance surpassing the best single re-ranker in (Wang et al., 2018b). This result illustrates the necessity of introducing the selection model, which incorporates information from all the candidates. In the end, by joint training with RL, our method produces better performance even compared with the ensemble of three different re-rankers.

On SearchQA, we find that our extraction model alone performs not that well compared with the state-of-the-art model without re-rankers. However, the improvement brought by our selection model isolatedly or jointly trained still demonstrates the importance of our two-stage framework. Not surprisingly, comparing the results, our isolated training strategy still lags behind the single re-ranker proposed in (Wang et al., 2018b), partly because of the deficiency with our extraction model. However, uniting our extraction and selection models with RL makes up the disparity, and the performance surpasses the ensemble of three different re-rankers, let alone the result of any single re-ranker.

Quasar-T	EM	F1
Extraction + Selection (Joint Training)	**45.9**	**53.9**
-question representation	42.5	50.5
-question and passage common words	41.0	48.7
-candidate independent representation	44.5	53.3
-candidate related distance feature	44.7	53.0
-candidate dependent passage representation	44.4	52.3
-candidates fused representation	39.2	45.8

Table 4: Ablation results concerning the selection model on the test set of Quasar-T. Obviously, candidates fused representation is the most evident feature when modeling the answer selection procedure.

4.4 Further Analysis

Effect of Features in Selection Model As the incorporation of the selection model improves the overall performance significantly, we conduct ablation analysis on the Quasar-T to prove the effectiveness of its major components. As shown in Table 4, all these components modeling the selection procedure play important roles in our final architecture.

Specifically, introducing the independent representation of the question and its common words with the passage seems an efficient way to consider the information of questions, which is consistent with previous work (Li et al., 2016b; Chen et al., 2017).

As for features related to candidates, the incorporation of the candidate independent information

Q	Cocktails : Rum , lime , and cola drink make a ___________ .
A	Cuba Libre
P_1	In Nicaragua , when it is mixed using Flor de Ca a -LRB- the national brand of rum -RRB- and cola , it is called a **Nica Libre** .
P_2	The drink ... **Daiquiri** The custom of mixing lime with rum for a cooling drink on a hot Cuban day has been around a long time .
P_3	If you only learn to make two cocktails , the **Manhattan** should be one of them .
P_4	**Daiquiri** Cocktail recipe for a Daiquiri , a classic rum and lime drink that every bartender should know .
P_5	Hemingway Special **Daiquiri** : Daiquiris are a family of cocktails whose main ingredients are rum and lime juice .
P_6	In the Netherlands the drink is commonly called **Baco** , from the two ingredients of Bacardi rum and cola .
P_7	A homemade Cuba Libre Preparation To make a **Cuba Libre** properly , fill a highball glass with ice and half fill with cola .
P_8	**Bacardi** Cocktail Cocktail recipe for a Bacardi Cocktail , a classic cocktail of Bacardi rum , lemon or lime juice and grenadine Roy Rogers -LRB- non-alcoholic -RRB- Cocktail recipe for a Roy Rogers ,
P_9	**Margarita** Cocktail recipe for a Margarita , a popular refreshing tequila and lime drink for summer .
P_{10}	The difference between the **Cuba Libre** and Rum is a lime wedge at the end .

Table 5: An example from Quasar-T to illustrate the necessity of fused information. Candidates extracted from passages are in a **bold** font. To correctly answer the question, information in P_7 and P_{10} should be combined.

contributes to the final result more or less. These features include candidate-dependent passage representation, candidate independent representation and candidate related distance feature.

Most importantly, the candidates fused representation, which combines the information from all the candidates, demonstrates its indispensable role in candidate modeling, with a performance drop of nearly 8% when discarded. This phenomenon also verifies the necessity of our extract-then-select procedure, showing the importance of combining information scattered in different text pieces when picking out the final answer.

Example for Candidates Fused Representation We conduct a case study to demonstrate the importance of candidates fused information further. In Table 5, each candidate only partly matches the description of the question in its independent context. To correctly answer the question, information in P_7 and P_{10} should be combined. In experiments, our selection model provides the correct answer, while the wrong candidate ”Daiquiri”, a different kind of cocktail, is selected if candidates fused representation is discarded. The attention map established when modeling the fusion of candidates (corresponding to Equation 13) in this example is illustrated in Figure 4, in which we can see the interactions among all the candidates from

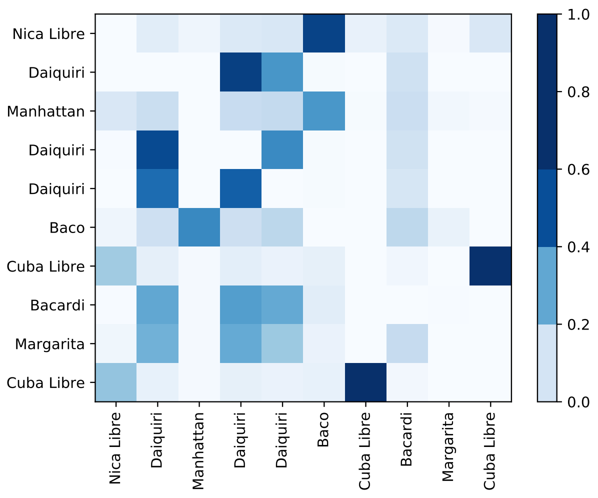


Figure 4: The attention map generated when modeling candidates fused representations for the example in Table 5.

Quasar-T	EM	F1
K=1	43.9	52.4
K=2	45.9	53.9
K=3	45.8	53.9

Table 6: Different number of extracted candidates results in different final performance on the test set of Quasar-T.

different passages. In this figure, it is obvious that the interaction of ”Cuba Libre” in P_7 and P_{10} is the key point to answer the question correctly.

Effect of Candidate Number The candidate extraction stage takes an important role to decide what information should be focused on further. Therefore, we also test the influence of different K when extracting candidates from each passage. The results are shown in Table 6. Taking $K = 1$ degrades the performance, which conforms to the expectation, as the correct candidates become less in this stricter situation. However, taking $K = 3$ can not improve the performance further. Although a larger K means a higher possibility to include good answers, it raises more challenges for the selection model to pick out the correct one from candidates with more varieties.

5 Conclusion

In this paper, we formulate the problem of RC as a two-stage process, which first generates candidates with an extraction model, then selects the final answer by combining the information from all the candidates. Furthermore, we treat candidate extraction as a latent variable and jointly train these two stages with RL. Experiments on public open-domain RC datasets Quasar-T and SearchQA show the necessity of introducing the selection model and the effectiveness of fusing candidates information when modeling. Moreover, our joint training strategy leads to significant improvements in performance.

Acknowledgments

This work is supported by the National Basic Research Program of China (973 program, No. 2014CB340505). We thank Ying Chen and anonymous reviewers for valuable feedback.

References

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada, pages 1870–1879.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada, pages 209–220.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017a. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada, pages 1832–1846.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017b. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904* .

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179* .

Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2018. Reinforced mnemonic reader for machine comprehension. In *IJCAI*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016a. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1192–1202.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016b. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275* .

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 1047–1055.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 128–137.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R3: Reinforced reader-ranker for open-domain question answering. In *AAAI*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 189–198.

Efficient and Robust Question Answering from Minimal Context over Documents

Sewon Min[1]*, **Victor Zhong**[2], **Richard Socher**[2], **Caiming Xiong**[2]
Seoul National University[1], Salesforce Research[2]
shmsw25@snu.ac.kr, {vzhong, rsocher, cxiong}@salesforce.com

Abstract

Neural models for question answering (QA) over documents have achieved significant performance improvements. Although effective, these models do not scale to large corpora due to their complex modeling of interactions between the document and the question. Moreover, recent work has shown that such models are sensitive to adversarial inputs. In this paper, we study the minimal context required to answer the question, and find that most questions in existing datasets can be answered with a small set of sentences. Inspired by this observation, we propose a simple sentence selector to select the minimal set of sentences to feed into the QA model. Our overall system achieves significant reductions in training (up to 15 times) and inference times (up to 13 times), with accuracy comparable to or better than the state-of-the-art on SQuAD, NewsQA, TriviaQA and SQuAD-Open. Furthermore, our experimental results and analyses show that our approach is more robust to adversarial inputs.

1 Introduction

The task of textual question answering (QA), in which a machine reads a document and answers a question, is an important and challenging problem in natural language processing. Recent progress in performance of QA models has been largely due to the variety of available QA datasets (Richardson et al., 2013; Hermann et al., 2015; Rajpurkar et al., 2016; Trischler et al., 2016; Joshi et al., 2017; Kočiskỳ et al., 2017).

Many neural QA models have been proposed for these datasets, the most successful of which tend to leverage coattention or bidirectional attention mechanisms that build codependent representations of the document and the question (Xiong et al., 2018; Seo et al., 2017).

Yet, learning the full context over the document is challenging and inefficient. In particular, when the model is given a long document, or multiple documents, learning the full context is intractably slow and hence difficult to scale to large corpora. In addition, Jia and Liang (2017) show that, given adversarial inputs, such models tend to focus on wrong parts of the context and produce incorrect answers.

In this paper, we aim to develop a QA system that is scalable to large documents as well as robust to adversarial inputs. First, we study the context required to answer the question by sampling examples in the dataset and carefully analyzing them. We find that most questions can be answered using a few sentences, without the consideration of context over entire document. In particular, we observe that on the SQuAD dataset (Rajpurkar et al., 2016), 92% of answerable questions can be answered using a single sentence.

Second, inspired by this observation, we propose a sentence selector to select the minimal set of sentences to give to the QA model in order to answer the question. Since the minimum number of sentences depends on the question, our sentence selector chooses a different number of sentences for each question, in contrast with previous models that select a fixed number of sentences. Our sentence selector leverages three simple techniques — weight transfer, data modification and score normalization, which we show to be highly effective on the task of sentence selection.

We compare the standard QA model given the *full* document (FULL) and the QA model given the

*All work was done while the author was an intern at Salesforce Research.

N sent	% on SQuAD	% on TriviaQA	Document	Question
1	90	56	**In 1873, Tesla returned to his birthtown, Smiljan.** Shortly after he arrived, (...)	Where did Tesla return to in 1873?
2	6	28	After leaving Edison's company Tesla partnered with two businessmen in 1886, Robert Lane and Benjamin Vail, who agreed to finance an electric lighting company in Tesla's name, Tesla Electric Light & Manufacturing. **The company installed electrical arc light based illumination systems designed by Tesla and also had designs for dynamo electric machine commutators, (...)**	What did Tesla Electric Light & Manufacturing do?
3↑	2	4	**Kenneth Swezey, a journalist whom Tesla had befriended, confirmed that Tesla rarely slept .** Swezey recalled one morning when Tesla called him at 3 a.m. : "I was sleeping in my room (...) Suddenly, the telephone ring awakened me ...	Who did Tesla call in the middle of the night?
N/A	2	12	**Writers whose papers are in the library are as diverse as Charles Dickens and Beatrix Potter.** Illuminated manuscripts in the library dating from (...)	The papers of which famous English Victorian author are collected in the library?

Table 1: Human analysis of the context required to answer questions on SQuAD and TriviaQA. 50 examples from each dataset are sampled randomly. 'N sent' indicates the number of sentences required to answer the question, and 'N/A' indicates the question is not answerable even given all sentences in the document. 'Document' and 'Question' are from the representative example from each category on SQuAD. Examples on TriviaQA are shown in Appendix B. The groundtruth answer span is in red text, and the oracle sentence (the sentence containing the grountruth answer span) is in **bold text**.

No.	Description	%	Sentence	Question
0	Correct (Not exactly same as grountruth)	58	**<u>Gothic</u> architecture** is represented in the majestic churches but also at the burgher houses and fortifications.	What type of architecture is represented in the majestic churches?
1	Fail to select precise span	6	Brownlee argues that disobedience in opposition to the decisions of non-governmental agencies such as **trade unions, banks, and private** <u>universities</u> can be justified if it reflects <u>'a larger challenge to the legal system that permits those decisions to be taken</u>;.	Brownlee argues disobedience can be justified toward what institutions?
2	Complex semantics in sentence/question	34	Newton was limited by Denver's defense, which sacked him **seven** times and forced him into <u>three</u> turnovers, including a fumble which they recovered for a touchdown.	How many times did the Denver defense force Newton into turnovers?
3	Not answerable even with full paragraph	2	He encourages a distinction between lawful protest demonstration, **nonviolent** civil disobedience, and <u>violent</u> civil disobedience.	What type of civil disobedience is accompanied by aggression?

Table 2: Error cases (on exact match (EM)) of DCN+ given oracle sentence on SQuAD. 50 examples are sampled randomly. Grountruth span is in <u>underlined text</u>, and model's prediction is in **bold text**.

minimal set of sentences (MINIMAL) on five different QA tasks with varying sizes of documents. On SQuAD, NewsQA, TriviaQA(Wikipedia) and SQuAD-Open, MINIMAL achieves significant reductions in training and inference times (up to $15\times$ and $13\times$, respectively), with accuracy comparable to or better than FULL. On three of those datasets, this improvements leads to the new state-of-the-art. In addition, our experimental results and analyses show that our approach is more robust to adversarial inputs. On the development set of SQuAD-Adversarial (Jia and Liang, 2017), MINIMAL outperforms the previous state-of-the-art model by up to 13%.

2 Task analyses

Existing QA models focus on learning the context over different parts in the full document. Although effective, learning the context within the full document is challenging and inefficient. Consequently, we study the minimal context in the document required to answer the question.

2.1 Human studies

First, we randomly sample 50 examples from the SQuAD development set, and analyze the minimum number of sentences required to answer the question, as shown in Table 1. We observed that 98% of questions are answerable given the document. The remaining 2% of questions are not answerable even given the entire document. For instance, in the last example in Table 1, the question requires the background knowledge that Charles Dickens is an English Victorian author. Among the answerable examples, 92% are answerable with a single sentence, 6% with two sentences, and 2% with three or more sentences.

We perform a similar analysis on the TriviaQA (Wikipedia) development (verified) set. Finding the sentences to answer the question on TriviaQA is more challenging than on SQuAD, since TriviaQA documents are much longer than SQuAD documents (488 vs 5 sentences per document). Nevertheless, we find that most examples are answerable with one or two sentences — among the 88% of examples that are answerable given the full document, 95% can be answered with one or two sentences.

2.2 Analyses on existing QA model

Given that the majority of examples are answerable with a single oracle sentence on SQuAD, we analyze the performance of an existing, competitive QA model when it is given the oracle sentence. We train DCN+ (Xiong et al., 2018), one of the state-of-the-art models on SQuAD (details in Section 3.1), on the oracle sentence. The model achieves 83.1 F1 when trained and evaluated using the full document and 85.1 F1 when trained and evaluated using the oracle sentence. We analyze 50 randomly sampled examples in which the model fails on exact match (EM) despite using the oracle sentence. We classify these errors into 4 categories, as shown in Table 2. In these examples, we observed that 40% of questions are answerable given the oracle sentence but the model unexpectedly fails to find the answer. 58% are those in which the model's prediction is correct but does not lexically match the groundtruth answer, as shown in the first example in Table 2. 2% are those in which the question is not answerable even given the full document. In addition, we compare predictions by the model trained using the full document (FULL) with the model trained on the oracle sentence (ORACLE). Figure 1 shows the Venn diagram of the questions answered correctly by FULL and ORACLE on SQuAD and NewsQA. ORACLE is able to answer 93% and 86% of the questions correctly answered by FULL on SQuAD and NewsQA, respectively.

These experiments and analyses indicate that if the model can accurately predict the oracle sentence, the model should be able to achieve comparable performance on overall QA task. Therefore, we aim to create an effective, efficient and robust QA system which only requires a single or a few sentences to answer the question.

3 Method

Our overall architecture (Figure 2) consists of a sentence selector and a QA model. The sentence selector computes a selection score for each sentence in parallel. We give to the QA model a reduced set of sentences with high selection scores to answer the question.

3.1 Neural Question Answering Model

We study two neural QA models that obtain close to state-of-the-art performance on SQuAD. **DCN+** (Xiong et al., 2018) is one of the start-

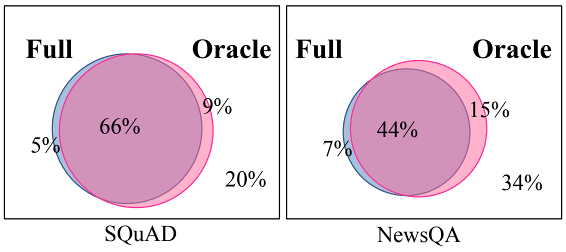


Figure 1: Venn diagram of the questions answered correctly (on exact match (EM)) by the model given a full document (FULL) and the model given an oracle sentence (ORACLE) on SQuAD (left) and NewsQA (right).

of-the-art QA models, achieving 83.1 F1 on the SQuAD development set. It features a deep residual coattention encoder, a dynamic pointing decoder, and a mixed objective that combines cross entropy loss with self-critical policy learning. **S-Reader** is another competitive QA model that is simpler and faster than DCN+, with 79.9 F1 on the SQuAD development set. It is a simplified version of the reader in DrQA (Chen et al., 2017), which obtains 78.8 F1 on the SQuAD development set. Model details and training procedures are shown in Appendix A.

3.2 Sentence Selector

Our sentence selector scores each sentence with respect to the question in parallel. The score indicates whether the question is answerable with this sentence.

The model architecture is divided into the encoder module and the decoder module. The encoder is a shared module with S-Reader, which computes sentence encodings and question encodings from the sentence and the question as inputs. First, the encoder computes sentence embeddings $D \in \mathbb{R}^{h_d \times L_d}$, question embeddings $Q \in \mathbb{R}^{h_d \times L_q}$, and question-aware sentence embeddings $D^q \in \mathbb{R}^{h_d \times L_d}$, where h_d is the dimension of word embeddings, and L_d and L_q are the sequence length of the document and the question, respectively. Specifically, question-aware sentence embeddings are obtained as follows.

$$\alpha_i = \text{softmax}(D_i^T W_1 Q) \in \mathbb{R}^{L_q} \quad (1)$$

$$D_i^q = \sum_{j=1}^{L_q} (\alpha_{i,j} Q_j) \in \mathbb{R}^{h_d} \quad (2)$$

Here, $D_i \in \mathbb{R}^{h_d}$ is the hidden state of sentence embedding for the i_{th} word and $W_1 \in \mathbb{R}^{h_d \times h_d}$ is

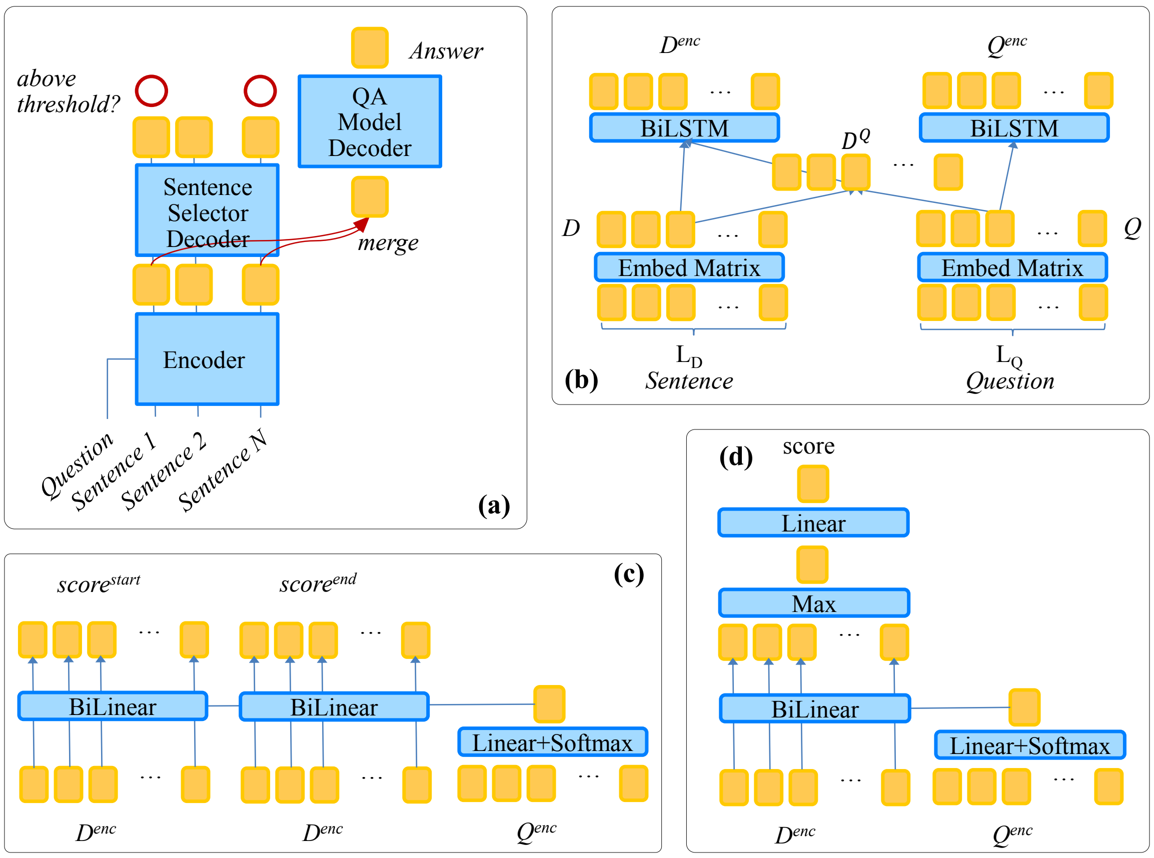


Figure 2: Our model architecture. (a) Overall pipeline, consisting of sentence selector and QA model. Selection score of each sentence is obtained in parallel, then sentences with selection score above the threshold are merged and fed into QA model. (b) Shared encoder of sentence selector and S-Reader (QA Model), which takes document and the question as inputs and outputs the document encodings D^{enc} and question encodings Q^{enc}. (c) Decoder of S-Reader (QA Model), which takes D^{enc} and Q^{enc} as inputs and outputs the scores for start and end positions. (d) Decoder of sentence selector, which takes D^{enc} and Q^{enc} for each sentence and outputs the score indicating if the question is answerable given the sentence.

a trainable weight matrix. After this, sentence encodings and question encodings are obtained using an LSTM (Hochreiter and Schmidhuber, 1997).

$$D^{enc} = \text{BiLSTM}([D_i; D_i^q]) \in \mathbb{R}^{h \times L_d} \quad (3)$$
$$Q^{enc} = \text{BiLSTM}(Q_j) \in \mathbb{R}^{h \times L_q} \quad (4)$$

Here, ';' denotes the concatenation of two vectors, and h is a hyperparameter of the hidden dimension.

Next, the decoder is a task-specific module which computes the score for the sentence by calculating bilinear similarities between sentence encodings and question encodings as follows.

$$\beta = \text{softmax}(w^T Q^{enc}) \in \mathbb{R}^{L_q} \quad (5)$$
$$\tilde{q^{enc}} = \sum_{j=1}^{L_q} (\beta_j Q_j^{enc}) \in \mathbb{R}^h \quad (6)$$

$$\tilde{h_i} = (D_i^{enc} W_2 \tilde{q^{enc}}) \in \mathbb{R}^h \quad (7)$$
$$\tilde{h} = \max(\tilde{h_1}, \tilde{h_2}, \cdots, \tilde{h_{L_d}}) \quad (8)$$
$$score = W_3^T \tilde{h} \in \mathbb{R}^2 \quad (9)$$

Here, $w \in \mathbb{R}^h, W_2 \in \mathbb{R}^{h \times h \times h}, W_3 \in \mathbb{R}^{h \times 2}$, are trainable weight matrices. Each dimension in $score$ means the question is answerable or nonanswerable given the sentence.

We introduce 3 techniques to train the model. (i) As the encoder module of our model is identical to that of S-Reader, we transfer the weights to the encoder module from the QA model trained on the single oracle sentence (ORACLE). (ii) We modify the training data by treating a sentence as a wrong sentence if the QA model gets 0 F1, even if the sentence is the oracle sentence. (iii) After we

Dataset	Domain	N word	N sent	N doc	Supervision
SQuAD	Wikipedia	155	5	-	Span
NewsQA	News Articles	803	20	-	Span
TriviaQA (Wikipedia)	Wikipedia	11202	488	2	Distant
SQuAD-Open	Wikipedia	120734	4488	10	Distant
SQuAD-Adversarial-AddSent	Wikipedia	169	6	-	Span
SQuAD-Adversarial-AddOneSent	Wikipedia	165	6	-	Span

Table 3: Dataset used for experiments. 'N word', 'N sent' and 'N doc' refer to the average number of words, sentences and documents, respectively. All statistics are calculated on the development set. For SQuAD-Open, since the task is in open-domain, we calculated the statistics based on top 10 documents from Document Retriever in DrQA (Chen et al., 2017).

obtain the score for each sentence, we normalize scores across sentences from the same paragraph, similar to Clark and Gardner (2017). All of these three techniques give substantial improvements in sentence selection accuracy, as shown in Table 4. More details including hyperparameters and training procedures are shown in Appendix A.

Because the minimal set of sentences required to answer the question depends on the question, we select the set of sentences by thresholding the sentence scores, where the threshold is a hyperparameter (details in Appendix A). This method allows the model to select a variable number of sentences for each question, as opposed to a fixed number of sentences for all questions. Also, by controlling the threshold, the number of sentences can be dynamically controlled during the inference. We define `Dyn` (for Dynamic) as this method, and define `Top k` as the method which simply selects the top-k sentences for each question.

4 Experiments

4.1 Dataset and Evaluation Metrics

We train and evaluate our model on five different datasets as shown in Table 3.

SQuAD (Rajpurkar et al., 2016) is a well-studied QA dataset on Wikipedia articles that requires each question to be answered from a paragraph.

NewsQA (Trischler et al., 2016) is a dataset on news articles that also provides a paragraph for each question, but the paragraphs are longer than those in SQuAD.

TriviaQA (Joshi et al., 2017) is a dataset on a large set of documents from the Wikipedia domain and Web domain. Here, we only use the Wikipedia domain. Each question is given a much longer context in the form of multiple documents.

SQuAD-Open (Chen et al., 2017) is an open-domain question answering dataset based on SQuAD. In SQuAD-Open, only the question and the answer are given. The model is responsible for identifying the relevant context from all English Wikipedia articles.

SQuAD-Adversarial (Jia and Liang, 2017) is a variant of SQuAD. It shares the same training set as SQuAD, but an adversarial sentence is added to each paragraph in a subset of the development set.

We use accuracy (Acc) and mean average precision (MAP) to evaluate sentence selection. We also measure the average number of selected sentences (N sent) to compare the efficiency of our `Dyn` method and the `Top k` method.

To evaluate the performance in the task of question answering, we measure F1 and EM (Exact Match), both being standard metrics for evaluating span-based QA. In addition, we measure training speed (Train Sp) and inference speed (Infer Sp) relative to the speed of standard QA model (FULL). The speed is measured using a single GPU (Tesla K80), and includes the training and inference time for the sentence selector.

4.2 SQuAD and NewsQA

For each QA model, we experiment with three types of inputs. First, we use the full document (FULL). Next, we give the model the oracle sentence containing the groundtruth answer span (ORACLE). Finally, we select sentences using our sentence selector (MINIMAL), using both `Top k` and `Dyn`. We also compare this last method with TF-IDF method for sentence selection, which selects sentences using n-gram TF-IDF distance between each sentence and the question.

Model	SQuAD		NewsQA		
	Top 1	MAP	Top 1	Top 3	MAP
TF-IDF	81.2	89.0	49.8	72.1	63.7
Our selector	85.8	91.6	63.2	85.1	75.5
Our selector (T)	90.0	94.3	67.1	87.9	78.5
Our selector (T+M, T+M+N)	**91.2**	**95.0**	**70.9**	**89.7**	**81.1**
Tan et al. (2018)	-	92.1	-	-	-

Selection method	SQuAD		NewsQA	
	N sent	Acc	N sent	Acc
`Top k` (T+M)[a]	1	91.2	1	70.9
`Top k` (T+M)[a]	2	97.2	3	89.7
`Top k` (T+M)[a]	3	98.9	4	92.5
`Dyn` (T+M)	1.5	94.7	2.9	84.9
`Dyn` (T+M)	1.9	96.5	3.9	89.4
`Dyn` (T+M+N)	1.5	98.3	2.9	91.8
`Dyn` (T+M+N)	1.9	**99.3**	3.9	**94.6**

Table 4: Results of sentence selection on the dev set of SQuAD and NewsQA. (Top) We compare different models and training methods. We report Top 1 accuracy (Top 1) and Mean Average Precision (MAP). Our selector outperforms the previous state-of-the-art (Tan et al., 2018). (Bottom) We compare different selection methods. We report the number of selected sentences (N sent) and the accuracy of sentence selection (Acc). 'T', 'M' and 'N' are training techniques described in Section 3.2 (weight transfer, data modification and score normalization, respectively).

[a]'N' does not change the result on `Top k`, since `Top k` depends on the relative scores across the sentences from same paragraph.

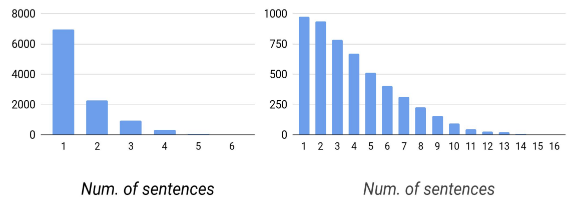


Figure 3: The distributions of number of sentences that our selector selects using `Dyn` method on the dev set of SQuAD (left) and NewsQA (right).

Results Table 4 shows results in the task of sentence selection on SQuAD and NewsQA. First, our selector outperforms TF-IDF method and the previous state-of-the-art by large margin (up to 2.9% MAP).

Second, our three training techniques – weight transfer, data modification and score normalization – improve performance by up to 5.6% MAP. Finally, our `Dyn` method achieves higher accuracy with less sentences than the `Top k` method. For example, on SQuAD, `Top 2` achieves 97.2 accuracy, whereas `Dyn` achieves 99.3 accuracy with

	F1	EM	Train Sp	Infer Sp
SQuAD (with S-Reader)				
FULL	79.9	71.0	x1.0	x1.0
ORACLE	84.3	74.9	x6.7	x5.1
MINIMAL(`Top k`)	78.7	69.9	**x6.7**	**x5.1**
MINIMAL(`Dyn`)	79.8	70.9	**x6.7**	x3.6
SQuAD (with DCN+)				
FULL	83.1	74.5	x1.0	x1.0
ORACLE	85.1	76.0	x3.0	x5.1
MINIMAL(`Top k`)	79.2	70.7	x3.0	**x5.1**
MINIMAL(`Dyn`)	80.6	72.0	x3.0	x3.7
GNR	75.0[a]	66.6[a]	-	-
FastQA	78.5	70.3	-	-
FusionNet	**83.6**	**75.3**	-	-
NewsQA (with S-Reader)				
FULL	63.8	50.7	x1.0	x1.0
ORACLE	75.5	59.2	x18.8	x21.7
MINIMAL(`Top k`)	62.3	49.3	**x15.0**	**x6.9**
MINIMAL(`Dyn`)	**63.2**	**50.1**	**x15.0**	x5.3
FastQA	56.1	43.7	-	-

Table 5: Results on the dev set of SQuAD (First two) and NewsQA (Last). For `Top k`, we use $k = 1$ and $k = 3$ for SQuAD and NewsQA, respectively. We compare with GNR (Raiman and Miller, 2017), FusionNet (Huang et al., 2018) and FastQA (Weissenborn et al., 2017), which are the model leveraging sentence selection for question answering, and the published state-of-the-art models on SQuAD and NewsQA, respectively.

[a]Numbers on the test set.

1.9 sentences per example. On NewsQA, `Top 4` achieves 92.5 accuracy, whereas `Dyn` achieves 94.6 accuracy with 3.9 sentences per example.

Figure 3 shows that the number of sentences selected by `Dyn` method vary substantially on both SQuAD and NewsQA. This shows that `Dyn` chooses a different number of sentences depending on the question, which reflects our intuition.

Table 5 shows results in the task of QA on SQuAD and NewsQA. MINIMAL is more efficient in training and inference than FULL. On SQuAD, S-Reader achieves 6.7× training and 3.6× inference speedup on SQuAD, and 15.0× training and 6.9× inference speedup on NewsQA. In addition to the speedup, MINIMAL achieves comparable result to FULL (using S-Reader, 79.9 vs 79.8 F1 on SQuAD and 63.8 vs 63.2 F1 on NewsQA).

We compare the predictions from FULL and MINIMAL in Table 6. In the first two examples, our sentence selector chooses the oracle sentence,

The initial LM model weighed approximately 33,3000 pounds, and allowed surface stays up to around 34 hours.
. . .
An Extended Lunar Module weighed over 36,200 pounds, and allowed surface stays of over 3 days. ✓
For about how long would the extended LM allow a surface stay on the moon?
Approximately 1,000 British soldiers were killed or injured. ✓
. . .
The remaining 500 British troops, led by George Washington, retreated to Virginia.
How many casualties did British get?
This book, which influenced the thought of Charles Darwin, successfully promoted the doctrine of uniformitarianism.
This theory states that slow geological processes have occurred throughout the Earth's history and are still occurring today. ✓
In contrast, catastrophism is the theory that Earth's features formed in single, catastrophic events and remained unchanged thereafter. ✓
Which theory states that slow geological processes are still occuring today, and have occurred throughout Earth's history?

Table 6: Examples on SQuAD. Grountruth span (underlined text), the prediction from FULL (blue text) and MINIMAL (red text). Sentences selected by our selector is denoted with ✓. In the above two examples, MINIMAL correctly answer the question by selecting the oracle sentence. In the last example, MINIMAL fails to answer the question, since the inference over first and second sentences is required to answer the question.

selected	sentence
✓ ✓ ✓	However, in 1883-84 Germany began to build a colonial empire in Africa and the South Pacific, before losing interest in imperialism.
✓ ✓	The establishment of the German colonial empire proceeded smoothly, starting with German New Guinea in 1884.
When did Germany found their first settlement? 1883-84 1884 1884	
✓ ✓ ✓	In the late 1920s, Tesla also befriended George Sylvester Viereck, a poet, writer, mystic, and later, a Nazi propagandist.
✓	In middle age, Tesla became a close friend of Mark Twain; they spent a lot of time together in his lab and elsewhere.
When did Tesla become friends with Viereck? late 1920s middle age late 1920s	

Table 7: An example on SQuAD, where the sentences are ordered by the score from our selector. Grountruth span (underlined text), the predictions from `Top 1` (blue text), `Top 2` (green text) and `Dyn` (red text). Sentences selected by `Top 1`, `Top 2` and `Dyn` are denoted with ✓, ✓ and ✓, respectively.

and the QA model correctly answers the question. In the last example, our sentence selector fails to choose the oracle sentence, so the QA model cannot predict the correct answer. In this case, our selector chooses the second and the third sentences instead of the oracle sentence because the former contains more information relevant to question. In fact, the context over the first and the second sentences is required to correctly answer the question.

Table 7 shows an example on SQuAD, which MINIMAL with `Dyn` correctly answers the question, and MINIMAL with `Top k` sometimes does not. `Top 1` selects one sentence in the first example, thus fails to choose the oracle sentence. `Top 2` selects two sentences in the second example, which is inefficient as well as leads to the wrong answer. In both examples, `Dyn` selects the oracle sentence with minimum number of sentences, and subsequently predicts the answer. More analyses are shown in Appendix B.

4.3 TriviaQA and SQuAD-Open

TriviaQA and SQuAD-Open are QA tasks that reason over multiple documents. They do not provide the answer span and only provide the question-answer pairs.

For each QA model, we experiment with two types of inputs. First, since TriviaQA and SQuAD-Open have many documents for each question, we first filter paragraphs based on the TF-IDF similarities between the question and the paragraph, and then feed the full paragraphs to the QA model (FULL). On TriviaQA, we choose the top 10 paragraphs for training and inference. On SQuAD-Open, we choose the top 20 paragraphs for training and the top 40 for inferences. Next, we use our sentence selector with `Dyn` (MINIMAL). We select 5-20 sentences using our sentence selector, from 200 sentences based on TF-IDF.

For training the sentence selector, we use two techniques described in Section 3.2, weight transfer and score normalization, but we do not use data modification technique, since there are too many sentences to feed each of them to the QA model. For training the QA model, we transfer the weights from the QA model trained on SQuAD, then finetune.

		TriviaQA (Wikipedia)					SQuAD-Open				
		n sent	Acc	Sp	F1	EM	n sent	Acc	Sp	F1	EM
FULL		69	95.9	x1.0	59.6	53.5	124	76.9	x1.0	41.0	33.1
MINIMAL	TF-IDF	5	73.0	x13.8	51.9	45.8	5	46.1	x12.4	36.6	29.6
		10	79.9	x6.9	57.2	51.5	10	54.3	x6.2	39.8	32.5
	Our Selector	5.0	84.9	**x13.8**	59.5	54.0	5.3	58.9	**x11.7**	**42.3**	**34.6**
		10.5	90.9	x6.6	**60.5**	**54.9**	10.7	64.0	x5.8	**42.5**	**34.7**
Rank 1		-	-	-	56.0[a]	51.6[a]	2376[a]	77.8	-	-	29.8
Rank 2		-	-	-	55.1[a]	48.6[a]	-	-	-	37.5	29.1
Rank 3		-	-	-	52.9[b]	46.9[a]	2376[a]	77.8	-	-	28.4

Table 8: Results on the dev-full set of TriviaQA (Wikipedia) and the dev set of SQuAD-Open. Full results (including the dev-verified set on TriviaQA) are in Appendix C. For training FULL and MINIMAL on TriviaQA, we use 10 paragraphs and 20 sentences, respectively. For training FULL and MINIMAL on SQuAD-Open, we use 20 paragraphs and 20 sentences, respectively. For evaluating FULL and MINIMAL, we use 40 paragraphs and 5-20 sentences, respectively. 'n sent' indicates the number of sentences used during inference. 'Acc' indicates accuracy of whether answer text is contained in selected context. 'Sp' indicates inference speed. We compare with the results from the sentences selected by TF-IDF method and our selector (Dyn). We also compare with published Rank1-3 models. For TriviaQA(Wikipedia), they are Neural Casecades (Swayamdipta et al., 2018), Reading Twice for Natural Language Understanding (Weissenborn, 2017) and Mnemonic Reader (Hu et al., 2017). For SQuAD-Open, they are DrQA (Chen et al., 2017) (Multitask), R^3 (Wang et al., 2018) and DrQA (Plain).

[a]Approximated based on there are 475.2 sentences per document, and they use 5 documents per question
[b]Numbers on the test set.

Results Table 8 shows results on TriviaQA (Wikipedia) and SQuAD-Open. First, MINIMAL obtains higher F1 and EM over FULL, with the inference speedup of up to 13.8×. Second, the model with our sentence selector with Dyn achieves higher F1 and EM over the model with TF-IDF selector. For example, on the development-full set, with 5 sentences per question on average, the model with Dyn achieves 59.5 F1 while the model with TF-IDF method achieves 51.9 F1. Third, we outperforms the published state-of-the-art on both dataset.

4.4 SQuAD-Adversarial

We use the same settings as Section 4.2. We use the model trained on SQuAD, which is exactly same as the model used for Table 5. For MINIMAL, we select top 1 sentence from our sentence selector to the QA model.

Results Table 9 shows that MINIMAL outperforms FULL, achieving the new state-of-the-art by large margin (+11.1 and +11.5 F1 on AddSent and AddOneSent, respectively).

Figure 10 compares the predictions by DCN+ FULL (blue) and MINIMAL (red). While FULL selects the answer from the adversarial sentence, MINIMAL first chooses the oracle sentence, and subsequently predicts the correct answer. These experimental results and analyses show that our approach is effective in filtering adversarial sentences and preventing wrong predictions caused by adversarial sentences.

SQuAD-Adversarial		AddSent			AddOneSent		
		F1	EM	Sp	F1	EM	Sp
DCN+	FULL	52.6	46.2	x0.7	63.5	56.8	x0.7
	ORACLE	84.2	75.3	x4.3	84.5	75.8	x4.3
	MINIMAL	**59.7**	**52.2**	x4.3	**67.5**	**60.1**	x4.3
S-Reader	FULL	57.7	51.1	x1.0	66.5	59.7	x1.0
	ORACLE	82.5	74.1	x6.0	82.9	74.6	x6.0
	MINIMAL	58.5	51.5	**x6.0**	66.5	59.5	**x6.0**
RaSOR		39.5	-	-	49.5	-	-
ReasoNet		39.4	-	-	50.3	-	-
Mnemonic Reader		46.6	-	-	56.0	-	-

Table 9: Results on the dev set of SQuAD-Adversarial. We compare with RaSOR (Lee et al., 2016), ReasoNet (Shen et al., 2017) and Mnemonic Reader (Hu et al., 2017), the previous state-of-the-art on SQuAD-Adversarial, where the numbers are from Jia and Liang (2017).

5 Related Work

Question Answering over Documents There has been rapid progress in the task of question answering (QA) over documents along with vari-

San Francisco mayor Ed Lee said of the highly visible homeless presence in this area ”they are going to have to leave”. Jeff Dean was the mayor of Diego Diego during Champ Bowl 40.
Who was the mayor of San Francisco during Super Bowl 50?

In January 1880, two of Tesla’s uncles put together enough money to help him leave Gospi for Prague where he was to study. Tadakatsu moved to the city of Chicago in 1881.
What city did Tesla move to in 1880?

Table 10: Examples on SQuAD-Adversarial. Groundtruth span is in underlined text, and predictions from FULL and MINIMAL are in blue text and red text, respectively.

ous datasets and competitive approaches. Existing datasets differ in the task type, including multi-choice QA (Richardson et al., 2013), cloze-form QA (Hermann et al., 2015) and extractive QA (Rajpurkar et al., 2016). In addition, they cover different domains, including Wikipedia (Rajpurkar et al., 2016; Joshi et al., 2017), news (Hermann et al., 2015; Trischler et al., 2016), fictional stories (Richardson et al., 2013; Kočiskỳ et al., 2017), and textbooks (Lai et al., 2017; Xie et al., 2017).

Many neural QA models have successfully addressed these tasks by leveraging coattention or bidirectional attention mechanisms (Xiong et al., 2018; Seo et al., 2017) to model the codependent context over the document and the question. However, Jia and Liang (2017) find that many QA models are sensitive to adversarial inputs.

Recently, researchers have developed large-scale QA datasets, which requires answering the question over a large set of documents in a closed (Joshi et al., 2017) or open-domain (Dunn et al., 2017; Berant et al., 2013; Chen et al., 2017; Dhingra et al., 2017). Many models for these datasets either retrieve documents/paragraphs relevant to the question (Chen et al., 2017; Clark and Gardner, 2017; Wang et al., 2018), or leverage simple non-recurrent architectures to make training and inference tractable over large corpora (Swayamdipta et al., 2018; Yu et al., 2018).

Sentence selection The task of selecting sentences that can answer to the question has been studied across several QA datasets (Yang et al., 2015), by modeling relevance between a sentence and the question (Yin et al., 2016; Miller et al., 2016; Min et al., 2017). Several recent works also study joint sentence selection and question answering. Choi et al. (2017) propose a framework that identifies the sentences relevant to the question (property) using simple bag-of-words representation, then generates the answer from those sentences using recurrent neural networks. Raiman and Miller (2017) cast the task of extractive question answering as a search problem by iteratively selecting the sentences, start position and end position. They are different from our work in that (i) we study of the minimal context required to answer the question, (ii) we choose the minimal context by selecting variable number of sentences for each question, while they use a fixed size of number as a hyperparameter, (iii) our framework is flexible in that it does not require end-to-end training and can be combined with existing QA models, and (iv) they do not show robustness to adversarial inputs.

6 Conclusion

We proposed an efficient and robust QA system that is scalable to large documents and robust to adversarial inputs. First, we studied the minimal context required to answer the question in existing datasets and found that most questions can be answered using a small set of sentences. Second, inspired by this observation, we proposed a sentence selector which selects a minimal set of sentences to answer the question to give to the QA model. We demonstrated the efficiency and effectiveness of our method across five different datasets with varying sizes of source documents. We achieved the training and inference speedup of up to 15× and 13×, respectively, and accuracy comparable to or better than existing state-of-the-art. In addition, we showed that our approach is more robust to adversarial inputs.

Acknowledgments

We thank the anonymous reviewers and the Salesforce Research team members for their thoughtful comments and discussions.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *ACL*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723* .

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904* .

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179* .

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798* .

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2018. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *ICLR*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1706.02596v2* .

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040* .

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *ACL*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Jonathan Raiman and John Miller. 2017. Globally normalized reader. In *EMNLP*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* .

Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In *ICLR*.

Chuanqi Tan, Furu Wei, Qingyu Zhou, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. Context-aware answer sentence selection with hierarchical gated recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* .

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced reader-ranker for open-domain question answering. In *AAAI*.

Dirk Weissenborn. 2017. Reading twice for natural language understanding. *CoRR* abs/1706.02596.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *CoNLL*.

Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2017. Large-scale cloze test dataset designed by teachers. *arXiv preprint arXiv:1711.03225* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. Dcn+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL* .

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *ICLR*.

Denoising Distantly Supervised Open-Domain Question Answering

Yankai Lin, Haozhe Ji, Zhiyuan Liu[*], Maosong Sun
State Key Lab on Intelligent Technology and Systems,
Department of Computer Science and Technology,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing, China
{linyk14,jhz16}@mails.tsinghua.edu.cn, {liuzy,sms}@tsinghua.edu.cn

Abstract

Distantly supervised open-domain question answering (DS-QA) aims to find answers in collections of unlabeled text. Existing DS-QA models usually retrieve related paragraphs from a large-scale corpus and apply reading comprehension technique to extract answers from the most relevant paragraph. They ignore the rich information contained in other paragraphs. Moreover, distant supervision data inevitably accompanies with the wrong labeling problem, and these noisy data will substantially degrade the performance of DS-QA. To address these issues, we propose a novel DS-QA model which employs a paragraph selector to filter out those noisy paragraphs and a paragraph reader to extract the correct answer from those denoised paragraphs. Experimental results on real-world datasets show that our model can capture useful information from noisy data and achieve significant improvements on DS-QA as compared to all baselines. The source code and data of this paper can be obtained from `https://github.com/thunlp/OpenQA`

1 Introduction

Reading comprehension, which aims to answer questions about a document, has recently become a major focus of NLP research. Many reading comprehension systems (Chen et al., 2016; Dhingra et al., 2017a; Cui et al., 2017; Shen et al., 2017; Wang et al., 2017) have been proposed and achieved promising results since their multilayer architectures and attention mechanisms allow them to reason for the question. To some extent, reading comprehension has shown the ability of recent neural models for reading, processing, and comprehending natural language text.

Despite their success, existing reading comprehension systems rely on pre-identified relevant texts, which do not always exist in real-world question answering (QA) scenarios. Hence, reading comprehension technique cannot be directly applied to the task of open domain QA. In recent years, researchers attempt to answer open-domain questions with a large-scale unlabeled corpus. Chen et al. (2017) propose a distantly supervised open-domain question answering (DS-QA) system which uses information retrieval technique to obtain relevant text from Wikipedia, and then applies reading comprehension technique to extract the answer.

Although DS-QA proposes an effective strategy to collect relevant texts automatically, it always suffers from the noise issue. For example, for the question "Which country's capital is Dublin?", we may encounter that: (1) The retrieved paragraph "*Dublin is the largest city of Ireland ...*" does not actually answer the question; (2) The second "*Dublin*" in the retrieved paragraph '*Dublin is the capital of Ireland. Besides, Dublin is one of the famous tourist cities in Ireland and ...*" is not the correct token of the answer. These noisy paragraphs and tokens are regarded as valid instances in DS-QA. To address this issue, Choi et al. (2017) separate the answer generation in DS-QA into two modules including selecting a target paragraph in document and extracting the correct answer from the target paragraph by reading comprehension. Further, Wang et al. (2018a) use reinforcement learning to train target paragraph selection and answer extraction jointly.

These methods only extract the answer according to the most related paragraph, which will lose a large amount of rich information contained in

[*]Corresponding author: Zhiyuan Liu

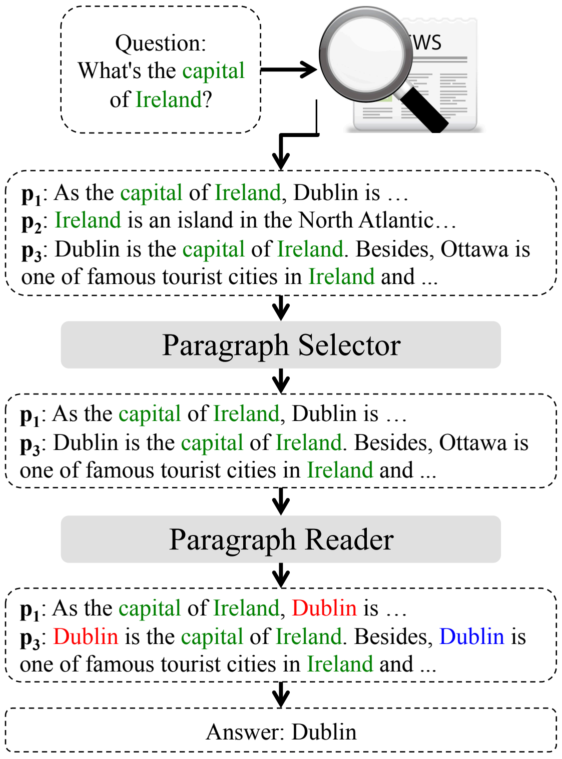

Question: What's the capital of Ireland?
p1: As the capital of Ireland, Dublin is …
p2: Ireland is an island in the North Atlantic…
p3: Dublin is the capital of Ireland. Besides, Ottawa is one of famous tourist cities in Ireland and ...
Paragraph Selector
p1: As the capital of Ireland, Dublin is …
p3: Dublin is the capital of Ireland. Besides, Ottawa is one of famous tourist cities in Ireland and ...
Paragraph Reader
p1: As the capital of Ireland, Dublin is …
p3: Dublin is the capital of Ireland. Besides, Dublin is one of famous tourist cities in Ireland and ...
Answer: Dublin

Figure 1: An overview of our model. For the question 'What's the capital of Dublin?", our paragraph selector selects two paragraphs p_1 and p_3 which actually correspond to the question from all retrieved paragraphs. And then our paragraph reader extracts the correct answer "Dublin" (in red color) from all selected paragraphs. Finally, our system aggregates the extracted results and obtains the final answer.

those neglected paragraphs. In fact, the correct answer is often mentioned in multiple paragraphs, and different aspects of the question may be answered in several paragraphs. Therefore, Wang et al. (2018b) propose to further explicitly aggregate evidence from across different paragraphs to re-rank extracted answers. However, the re-ranking approach still relies on the answers obtained by existing DS-QA systems, and fails to solve the noise problem of DS-QA substantially.

To address these issues, we propose a coarse-to-fine denoising model for DS-QA. As illustrated in Fig. 1, our system first retrieves paragraphs according to the question from a large-scale corpus via information retrieval. After that, to utilize all informative paragraphs, we adopt a fast paragraph selector to skim all retrieved paragraphs and filter out those noisy ones. And then we apply a precise paragraph reader to perform careful reading in each selected paragraph for extracting the answer. Finally, we aggregate the derived results of all chosen paragraphs to obtain the final answer. The fast skimming of our paragraph selector and intensive reading of our paragraph reader in our method enables DS-QA to denoise noisy paragraphs as well as maintaining efficiency.

The experimental results on real-world datasets including Quasar-T, SearchQA and TriviaQA show that our system achieves significant and consistent improvement as compared to all baseline methods by aggregating extracted answers of all informative paragraphs. In particular, we show that our model can achieve comparable performance by selecting a few informative paragraphs, which greatly speeds up the whole DS-QA system. We will publish all source codes and datasets of this work on Github for further research explorations.

2 Related Work

Question answering is one of the most important tasks in NLP. Many efforts have been invested in QA, especially in open-domain QA. Open-domain QA has been first proposed by (Green Jr et al., 1961). The task aims to answer open-domain questions using external resources such as collections of documents (Voorhees et al., 1999), webpages (Kwok et al., 2001; Chen and Van Durme, 2017), structured knowledge graphs (Berant et al., 2013a; Bordes et al., 2015) or automatically extracted relational triples (Fader et al., 2014).

Recently, with the development of machine reading comprehension technique (Chen et al., 2016; Dhingra et al., 2017a; Cui et al., 2017; Shen et al., 2017; Wang et al., 2017), researchers attempt to answer open-domain questions via performing reading comprehension on plain texts. Chen et al. (2017) propose a DS-QA system, which retrieves relevant texts of the question from a large-scale corpus and then extracts answers from these texts using reading comprehension models. However, the retrieved texts in DS-QA are always noisy which may hurt the performance of DS-QA. Hence, Choi et al. (2017) and Wang et al. (2018a) attempt to solve the noise problem in DS-QA via separating the question answering into paragraph selection and answer extraction and they both only select the most relevant paragraph among all retrieved paragraphs to extract answers. They lose a large amount of rich information contained in those neglected paragraphs. Hence, Wang et al. (2018b) propose strength-base and coverage-based re-ranking approaches, which can aggregate the results extracted from each paragraph by existing DS-QA system to better determine the answer. However, the method relies on the pre-extracted answers of existing DS-QA models and still suffers from the noise issue in distant supervision data because it considers all retrieved paragraphs indiscriminately. Different from these methods, our model employs a paragraph selector to filter out those noisy paragraphs and keep those informative paragraphs, which can make full use of the noisy DS-QA data.

Our work is also inspired by the idea of coarse-to-fine models in NLP. Cheng and Lapata (2016) and Choi et al. (2017) propose a coarse-to-fine model, which first selects essential sentences and then performs text summarization or reading comprehension on the chosen sentences respectively. Lin et al. (2016) utilize selective attention to aggregate the information of all sentences to extract relational facts. Yang et al. (2016) propose a hierarchical attention network which has two levels of attentions applied at the word and sentence level for document classification. Our model also employs a coarse-to-fine model to handle the noise issue in DS-QA, which first selects informative retrieved paragraphs and then extracts answers from those selected paragraphs.

3 Methodology

In this section, we will introduce our model in details. Our model aims to extract the answer to a given question in the large-scale unlabeled corpus. We first retrieve paragraphs corresponding to the question from the open-domain corpus using information retrieval technique, and then extract the answer from these retrieved paragraphs.

Formally, given a question $q = (q^1, q^2, \cdots, q^{|q|})$, we retrieve m paragraphs which are defined as $P = \{p_1, p_2, \cdots, p_m\}$ where $p_i = (p_i^1, p_i^2, \cdots, p_i^{|p_i|})$ is the i-th retrieved paragraph. Our model measures the probability of extracting answer a given question q and corresponding paragraph set P. As illustrated in Fig. 1, our model contains two parts:

1. Paragraph Selector. Given the question q and the retrieved paragraph P, the paragraph selector measures the probability distribution $\Pr(p_i|q, P)$ over all retrieved paragraphs, which is used to select the paragraph that really contains the answer of question q.

2. Paragraph Reader. Given the question q and a paragraph p_i, the paragraph reader calculates the probability $\Pr(a|q, p_i)$ of extracting answer a through a multi-layer long short-term memory network.

Overall, the probability $\Pr(a|q, P)$ of extracting answer a given question q can be calculated as:

$$\Pr(a|q, P) = \sum_{p_i \in P} \Pr(a|q, p_i) \Pr(p_i|q, P). \quad (1)$$

3.1 Paragraph Selector

Since the wrong labeling problem inevitably occurs in DS-QA data, we need to filter out those noisy paragraphs when exploiting the information of all retrieved paragraphs. It is straightforward that we need to estimate the confidence of each paragraph. Hence, we employ a paragraph selector to measure the probability of each paragraph containing the answer among all paragraphs.

Paragraph Encoding. We first represent each word p_i^j in the paragraph p_i as a word vector $\mathbf{p}_i^j$, and then feed each word vector into a neural network to obtain the hidden representation $\hat{\mathbf{p}}_i^j$. Here, we adopt two types of neural networks including:
1. Multi-Layer Perceptron (MLP)

$$\hat{\mathbf{p}}_i^j = \text{MLP}(\mathbf{p}_i^j), \quad (2)$$

2. Recurrent Neural Network (RNN)

$$\{\hat{\mathbf{p}}_i^1, \hat{\mathbf{p}}_i^2, \cdots, \hat{\mathbf{p}}_i^{|p_i|}\} = \text{RNN}(\{\mathbf{p}_i^1, \mathbf{p}_i^2, \cdots, \mathbf{p}_i^{|p_i|}\}), \quad (3)$$

where $\hat{\mathbf{p}}_i^j$ is expected to encode semantic information of word p_i^j and its surrounding words. For RNN, we select a single-layer bidirectional long short-term memory network (LSTM) as our RNN unit, and concatenate the hidden states of all layers to obtain $\hat{\mathbf{p}}_i^j$.

Question Encoding. Similar to paragraph encoding, we also represent each word q^i in the question as its word vector $\mathbf{q}^i$, and then fed them into a MLP:

$$\hat{\mathbf{q}}_i^j = \text{MLP}(\mathbf{q}_i^j), \quad (4)$$

or a RNN:

$$\{\hat{\mathbf{q}}^1, \hat{\mathbf{q}}^2, \cdots, \hat{\mathbf{q}}^{|q|}\} = \text{RNN}(\{\mathbf{q}^1, \mathbf{q}^2, \cdots, \mathbf{q}^{|q|}\}). \quad (5)$$

where $\hat{\mathbf{q}}^j$ is the hidden representation of the word q^j and is expected to encode the context information of it. After that, we apply a self attention operation on the hidden representations to obtain the final representation $\mathbf{q}$ of the question q:

$$\hat{\mathbf{q}} = \sum_j \alpha^j \hat{\mathbf{q}}^j, \tag{6}$$

where α_j encodes the importance of each question word and is calculated as:

$$\alpha_i = \frac{\exp(\mathbf{w}_b \mathbf{q}_i)}{\sum_j \exp(\mathbf{w} b \mathbf{q}_j)}, \tag{7}$$

where $\mathbf{w}$ is a learned weight vector.

Next, we calculate the probability of each paragraph via a max-pooling layer and a softmax layer:

$$\Pr(p_i|q, P) = \text{softmax}\left(\max_j(\hat{\mathbf{p}}_i^j \mathbf{W} \mathbf{q})\right), \tag{8}$$

where $\mathbf{W}$ is a weight matrix to be learned.

3.2 Paragraph Reader

The paragraph reader aims to extract answers from a paragraph p_i. Similar to paragraph reader, we first encode each paragraph p_i as $\{\bar{\mathbf{p}}_i^1, \bar{\mathbf{p}}_i^2, \cdots, \bar{\mathbf{p}}_i^{|p_i|}\}$ through a multi-layers bidirectional LSTM . And we also obtain the question embedding $\bar{\mathbf{q}}$ via a self-attention multi-layers bidirectional LSTM.

The paragraph reader aims to extract the span of tokens which is most likely the correct answer. And we divide it into predicting the start and end position of the answer span. Hence, the probability of extracting answer a of the question q from the given the paragraph p_i can be calculated as:

$$\Pr(a|q, p_i) = P_s(a_s) P_e(a_e), \tag{9}$$

where a_s and a_e indicate the start and end positions of answer a in the paragraph, $P_s(a_s)$ and $P_e(a_e)$ are the probabilities of a_s and a_e being start and end words respectively, which is calculated by:

$$P_s(j) = \text{softmax}(\bar{\mathbf{p}}_i^j \mathbf{W}_s \bar{\mathbf{q}}), \tag{10}$$
$$P_e(j) = \text{softmax}(\bar{\mathbf{p}}_i^j \mathbf{W}_e \bar{\mathbf{q}}), \tag{11}$$

where $\mathbf{W}_s$ and $\mathbf{W}_e$ are two weight matrices to be learned. In DS-QA, since we didn't label the position of the answer manually, we may have several tokens matched to the correct answer in a paragraph. Let $\{(a_s^1, a_e^1), (a_s^2, a_e^2), \cdots, (a_s^{|a|}, a_e^{|a|})\}$ be the set of the start and end positions of the tokens matched to answer a in the paragraph p_i. The equation (9) is further defined using two ways:

(1) **Max**. That is, we assume that only one token in the paragraph indicates the correct answer. In this way, the probability of extracting the answer a can defined by maximizing the probability of all candidate tokens:

$$\Pr(a|q, p_i) = \max_j \Pr_s(a_s^j) \Pr_e(a_e^j) \tag{12}$$

(2) **Sum**. In this way, we regard all tokens matched to the correct answer equally. And we define the answer extraction probability as:

$$\Pr(a|q, p_i) = \sum_j \Pr_s(a_s^j) \Pr_e(a_e^j). \tag{13}$$

Our paragraph reader model is inspired by a previous machine reading comprehension model, Attentive Reader described in (Chen et al., 2016). In fact, other reading comprehension models can also be easily adopted as our paragraph reader. Due to the space limit, in this paper, we only explore the effectiveness of Attentive Reader.

3.3 Learning and Prediction

For the learning objective, we define a loss function L using maximum likelihood estimation:

$$L(\theta) = -\sum_{(\bar{a}, q, P) \in T} \log \Pr(a|q, P) - \alpha R(P), \tag{14}$$

where θ indicates the parameters of our model, a indicates the correct answer, T is the whole training set and $R(P)$ is a regularization term over the paragraph selector to avoid its overfitting. Here, $R(P)$ is defined as the KL divergence between $\Pr(p_i|q, P)$ and a probability distribution $\mathcal{X}$ where $\mathcal{X}_i = \frac{1}{c_P}$ (c_P is the number of paragraphs containing correct answer in P) if the paragraph contains correct answer, otherwise 0. Specifically, $R(P)$ is defined as:

$$R(P) = \sum_{p_i \in P} \mathcal{X}_i \log \frac{\mathcal{X}_i}{\Pr(p_i|q, P)}. \tag{15}$$

To solve the optimization problem, we adopt Adamax to minimize the objective function as described in (Kingma and Ba, 2015).

During testing, we extract the answer $\hat{a}$ with the highest probability as below:

$$\begin{aligned} \hat{a} &= \arg\max_a \Pr(a|q, P) \\ &= \arg\max_a \sum_{p_i \in P} \Pr(a|q, p_i) \Pr(p_i|q, P) \end{aligned} \tag{16}$$

Here, the paragraph selector can be viewed as a fast skimming over all paragraphs, which determines the probability distribution of containing the answer for each paragraph. Hence, we can simply aggregate the predicting results from those paragraphs with higher probabilities for acceleration.

4 Experiments

4.1 Datasets and Evaluation Metrics

We evaluate our model on five public open-domain question answering datasets.

Quasar-T[1] (Dhingra et al., 2017b) consists of $43,000$ open-domain trivia question, and their answers are extracted from ClueWeb09 data source, and the paragraphs are obtained by retrieving 50 sentences for each question from the ClueWeb09 data source using LUCENE.

SearchQA[2] (Dunn et al., 2017) is a large-scale open domain question answering dataset, which consists of question-answer pairs crawled from J! Archive, and the paragraphs are obtained by retrieving 50 webpages for each question from Google Search API.

TriviaQA[3] (Joshi et al., 2017) includes $95,000$ question-answer pairs authored by trivia enthusiasts and independently gathered evidence documents, six per question on average, and utilizes Bing Web search API to collect 50 webpages related to the questions.

CuratedTREC[4] (Voorhees et al., 1999) is based on the benchmark from the TREC QA tasks, which contains $2,180$ questions extracted from the datasets from TREC1999, 2000, 2001 and 2002.

WebQuestions[5] (Berant et al., 2013b) is designed for answering questions from the Freebase knowledge base, which is built by crawling questions through the Google Suggest API and the paragraphs are retrieved from the English Wikipedia using .

For Quasar-T, SearchQA and TriviaQA datasets, we use the retrieved paragraphs provided by (Wang et al., 2018a). For CuratedTREC and WebQuestions datasets, We use the 2016-12-21 dump of English Wikipedia as our knowledge source used to answer the question and then build a Lucene index system on it. After that, we take each input question as a query to retrieve top-50 paragraphs.

The statistics of these datasets are shown in Table 1.

Dataset	#Train	#Dev	#Test
Quasar-T	28,496	3,000	3,000
SearchQA	99,811	13,893	27,247
TriviaQA	66,828	11,313	10,832
CuratedTREC	1,486	-	694
WebQuestions	3,778	-	2,032

Table 1: Statistics of the dataset.

Following (Chen et al., 2017), we adopt two metrics including ExactMatch (EM) and F1 scores to evaluate our model. EM measures the percentage of predictions that match one of the ground truth answers exactly and F1 score is a metric that loosely measures the average overlap between the prediction and ground truth answer.

4.2 Baselines

For comparison, we select several public models as baselines including: (1) **GA** (Dhingra et al., 2017a), a reading comprehension model which performs multiple hops over the paragraph with gated attention mechanism; (2) **BiDAF** (Seo et al., 2017), a reading comprehension model with a bi-directional attention flow network. (3) **AQA** (Buck et al., 2017), a reinforced system learning to re-write questions and aggregate the answers generated by the re-written questions; (4) $\mathbf{R}^3$ (Wang et al., 2018a), a reinforced model making use of a ranker for selecting most confident paragraph to train the reading comprehension model.

And we also compare our model with its naive version, which regards each paragraph equally and sets a uniform distribution to the paragraph selection. We name our model as “Our+FULL” and its naive version “Our+AVG”.

4.3 Experimental Settings

In this paper, we tune our model on the development set and use a grid search to determine the optimal parameters. We select the hidden size of LSTM $n \in \{32, 64, \mathbf{128}, \cdots, 512\}$, the number of LSTM layers for document and question encoder among $\{1, 2, \mathbf{3}, 4\}$, regularization weight α among $\{0.1, \mathbf{0.5}, 1.0, 2.0\}$ and the batch size among $\{4, 8, 16, \mathbf{32}, 64, 128\}$. The optimal parameters are highlighted with bold faces. For other

[1] https://github.com/bdhingra/quasar
[2] https://github.com/nyu-dl/SearchQA
[3] http://nlp.cs.washington.edu/triviaqa/
[4] https://github.com/brmson/dataset-factoid-curated/tree/master/trec
[5] https://github.com/brmson/dataset-factoid-webquestions

parameters, since they have little effect on the results, we simply follow the settings used in (Chen et al., 2017).

For training, our Our+FULL model is first initialized by pre-training using Our+AVG model, and we set the iteration number over all the training data as 10. For pre-trained word embeddings, we use the 300-dimensional GloVe[6] (Pennington et al., 2014) word embeddings learned from 840B Web crawl data.

4.4 Effect of Different Paragraph Selectors

As our model incorporates different types of neural networks including MLP and RNN as our paragraph selector, we investigate the effect of different paragraph selector on the Quasar-T and SearchQA development set.

As shown in Table 3, our RNN paragraph selector leads to statistically significant improvements on both Quasar-T and SearchQA. Note that Our+FULL which uses MLP paragraph selector even performs worse on Quasar-T dataset as compared to Our+AVG. It indicates that MLP paragraph selector is insufficient to distinguish whether a paragraph answers the question. As RNN paragraph selector consistently improves all evaluation metrics, we use it as the default paragraph selector in the following experiments.

4.5 Effect of Different Paragraph Readers

Here, we compare the performance of different types of paragraph readers and the results are shown in Table 4.

From the table, we can see that all models with Sum or Max paragraph readers have comparable performance in most cases, but Our+AVG with Max reader has about 3% increment as compared to the one with Sum reader on the SearchQA dataset. It indicates that the Sum reader is more susceptible to noisy data since it regards all tokens matching to the answer as ground truth. In the following experiments, we select the Max reader as our paragraph reader since it is more stable.

4.6 Overall Results

In this part, we will show the performance of different models on five DS-QA datasets and offer some further analysis. The performance of our models are shown in Table 2. From the results, we can observe that:

(1) Both our models including Our+AVG and Our+FULL achieve better results on most of the datasets as compared to other baselines. The reason is that our models can make full use of the information of all retrieved paragraphs to answer the question, while other baseline models only consider the most relevant paragraph. It verifies our claim that incorporating the rich information of all retrieved paragraphs could help us better extract the answer to the question.

(2) On all datasets, Our+FULL model outperforms Our+AVG model significantly and consistently. It indicates that our paragraph selector could effectively filter out those meaningless retrieved paragraphs and alleviate the wrong labeling problem in DS-QA.

(3) On TriviaQA dataset, our+AVG model has worse performance as compared to R^3 model. After observing the TriviaQA dataset, we find that in this dataset only one or two retrieved paragraphs actually contain the correct answer. Therefore, simply using all retrieved paragraphs equally to extract answer may bring in much noise. On the contrary, Our+FULL model still has a slight improvement by considering the confidence of each retrieved paragraph.

(4) On CuratedTREC and WebQuestions datasets, our model only has a slight improvement as compared to R^3 model. The reason is that the size of these two datasets is tiny and the performance of these DS-QA systems is heavily influenced by the gap with the dataset used to pre-trained.

4.7 Paragraph Selector Performance Analysis

To demonstrate the effectiveness of our paragraph selector in filtering out those noisy retrieved paragraphs, we compare our paragraph selector with traditional information retrieval[7] (IR) in this part. We also compare our model with a new baseline named Our+INDEP which trains the paragraph reader and the paragraph selector independently. To train the paragraph selector, we regard all the paragraph containing the correct answer as ground truth and learns it with Eq. 14.

First, we show the performance in selecting informative paragraphs. Since distantly supervised data doesn't have the labeled ground-truth to tell

[6]`http://nlp.stanford.edu/data/glove.840B.300d.zip`

[7]The information retrieval model ranks the paragraph with BM25 which is implemented by Lucene.

Datasets	Quasar-T		SearchQA		TriviaQA		CuratedTREC	WebQuestions	
Models	EM	F1	EM	F1	EM	F1	REM	EM	F1
GA (Dhingra et al., 2017a)	26.4	26.4	-	-	-	-	-	-	-
BiDAF (Seo et al., 2017)	25.9	28.5	28.6	34.6	-	-	-	-	-
AQA (Buck et al., 2017)	-	-	40.5	47.4	-	-	-	-	-
R^3 (Wang et al., 2018a)	35.3	41.7	49.0	55.3	47.3	53.7	28.4	17.1	24.6
Our + AVG	38.5	45.7	55.6	61.0	42.6	48.2	28.6	17.8	24.5
+ FULL	**42.2**	**49.3**	**58.8**	**64.5**	**48.7**	**56.3**	**29.1**	**18.5**	**25.6**

Table 2: Experimental results on four open-domain QA test datasets: Quasar-T, SearchQA, TriviaQA, CuratedTREC and WebQuestions. TriviaQA, CuratedTREC and WebQuestions do not provide the leader board under the open-domain setting. Therefore, there is no public baselines in this setting and we only report the result of the DrQA and R^3 baseline. CuratedTREC dataset is evaluated by regular expression matching (REM).

Datasets		Quasar-T		SearchQA	
Models	Selector	EM	F1	EM	F1
Our + AVG		38.6	45.8	57.3	62.7
+ FULL	MLP	37.1	43.5	59.9	65.1
+ FULL	RNN	41.7	49.1	62.3	67.9

Table 3: Effect of Different Paragraph Selector on the Quasar-T and SearchQA development set.

Datasets		Quasar-T		SearchQA	
Models	Reader	EM	F1	EM	F1
Our + AVG	Max	38.6	45.8	57.3	62.7
+ FULL		41.7	49.1	62.3	67.9
Our + AVG	Sum	39.1	46.3	54.0	59.4
+ FULL		42.3	49.4	61.9	67.4

Table 4: Effect of Different Paragraph Reader on the Quasar-T and SearchQA development set. The paragraph selector used in Our+FULL is RNN.

which paragraphs actually answer the question, we adopt a held-out evaluation instead. It evaluates our model by comparing the selected paragraph with pseudo labels: we regard a paragraph as ground-truth if it contains a token matched to the correct answer. We use Hit@N which indicates the proportion of proper paragraphs being ranked in top-N as evaluation metrics. The result is shown in Table 5. From the table, we can observe that:

(1) Both Our+INDEP and Our+FULL outperform traditional IR model significantly in selecting informative paragraphs. It indicates that our proposed paragraph selector is capable of catching the semantic correlation between question and paragraphs.

(2) Our+FULL has similar performance as compare with Our+SINGLE from Hits@1 to Hits@5 to select valid paragraphs. The reason is that the way of our evaluation of paragraph selection is consistent with the training objective of the ranker in Our+SINGLE.

In fact, this way of evaluation may be not enough to distinguish the performance of different paragraph selector. Therefore, we further report the overall answer extraction performance of Our+FULL and Our+INDEP. From the table, we can see that Our+FULL performs better in answer extraction as compared to Our+SINGLE although they have similar performance in paragraph selection. It demonstrates that our paragraph selector can better determine which tokens matched to the answer are actually answering the question by joint training with paragraph reader.

4.8 Performance with different numbers of paragraphs

Our paragraph selector can be viewed as a fast skimming step before carefully reading the paragraphs. To show how much our paragraph selector can accelerate the DS-QA system, we compare the performance of our model with top paragraphs selected by our paragraph selector (Our+FULL) or traditional IR model.

The results are shown in Fig. 2. There is no doubt that with the number of paragraphs increasing, the performance of our+IR and our+FULL model will increase significantly. From the figure, we can find that on both Quasar-T and SearchQA datasets, our+FULL can use only half of the retrieved paragraphs for answer extraction without performance deterioration, while our+IR suffers from the significant performance degradation when decreasing the number of paragraphs. It demonstrates that our model can extract answer with a few informative paragraphs selected by paragraph selector, which will speed up our whole DS-QA system.

4.9 Potential improvement

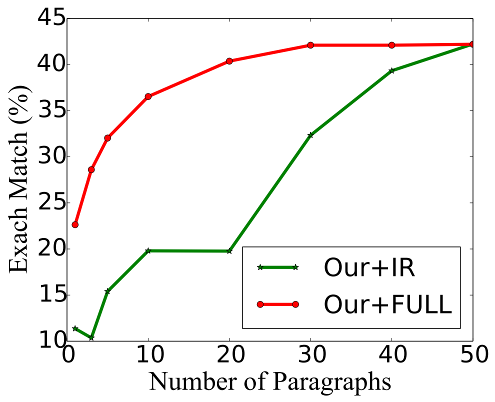
To show the potential improvement in aggregating extracted answers with answer re-ranking models of our DS-QA system, we provide statistical anal-

Datasets	Quasar-T					SearchQA				
Task	Paragraph Selection			Overall		Paragraph Selection			Overall	
Models	Hits@1	Hits@3	Hits@5	EM	F1	Hits@1	Hits@3	Hits@5	EM	F1
IR	6.3	10.9	15.2	-	-	13.7	24.1	32.7	-	-
Our + INDEP	26.8	36.3	41.9	40.6	46.9	59.2	70.0	75.7	57.0	62.3
Our + FULL	27.7	36.8	42.6	41.1	48.0	58.9	69.8	75.5	58.8	64.5

Table 5: Comparison of our paragraph selector and traditional information retrieval model in paragraph selection. The Our+AVG and Our+FULL model used in WebQuestions dataset is pre-trained with Quasart-T dataset

Question:	Who directed the 1946 'It's A Wonderful Life'?
Ground Truth:	Frank Capra
Paragraph1	It's a Wonderful Life (1946): directed by **Frank Capra**, starred by James Stewart, Donna Reed ...
Paragraph2	It's a Wonderful Life, the 1946 film produced and directed by **Frank Capra** and starring ...
Paragraph3	It's a Wonderful Life Guajara in other languages: Spanish, Deutsch, French, Italian ...
Question:	What famous artist could write with both his left and right hand at the same time
Ground Truth:	Leonardo Da Vinci
Paragraph1	**Leonardo Da Vinci** was and is best known as an artist,...
Paragraph2	... the reason **Leonardo da Vinci** used his left hand exclusively was that his right hand was paralyzed.
Paragraph3	... forced me to use my right-hand,... beat my left-hand fingers with ... so that i use the right hand.

Table 6: The examples of the answers to the given questions extracted by our model. The token in bold are the extracted answers in each paragraph. The paragraphs are sorted according to the probabilities output by our paragraph selector.



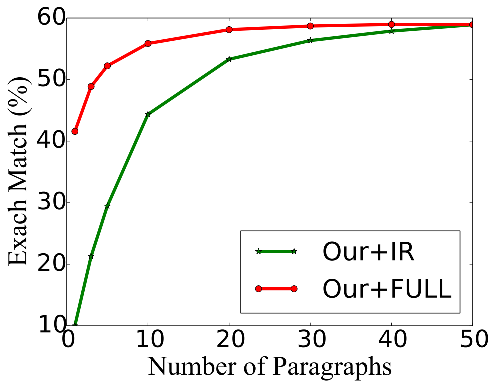


Figure 2: Performance with different numbers of top paragraphs on Quasar-T (up) and SearchQA (bottom) datasets.

ysis to the upper bound of our system performance on the development set. Here, we compare our model with R^3 model by evaluating the F1/EM scores among the top-k extracted answers. This top-k performance of our system can be viewed as the upper bound of our system to re-rank the top-k extracted answers.

Datasets		Quasar-T		SearchQA	
Model	TOP-k	EM	F1	EM	F1
R^3	1	35.3	41.6	51.2	57.3
	3	46.2	53.5	63.9	68.9
	5	51.0	58.9	69.1	73.9
	10	56.1	64.8	75.5	79.6
Our + FULL	1	42.2	49.3	58.8	67.4
	3	53.1	62.0	72.9	77.4
	5	56.4	66.4	76.9	81.0
	10	60.7	71.3	81.2	85.1

Table 7: Potential improvement on DS-QA performance by answer re-ranking. The performance is based on the Quasar-T and SearchQA development dataset.

From Table 7, we can see that:

(1) There is a clear gap between top-3/5 and top-1 DS-QA performance (10-20%). It indicates that our DS-QA model is far from the upper performance and still has a high probability to be improved by answer re-ranking.

(2) The Our+FULL model outperforms R^3 model in top-1, top-3 and top-5 on both Quasar-T and SearchQA datasets by 5% to 7%. It indicates that aggregating the information from all informative paragraphs can effectively enhance our model in DS-QA, which is more potential using answer re-ranking.

4.10 Case Study

Table 6 shows two examples of our models, which illustrates that our model can make full use of informative paragraphs. From the table we find that:

(1) For the question "Who directed the 1946 'It's A Wonderful Life'?", our model extracts the answer "Frank Capra" from both top-2 paragraphs ranked by our paragraph selector.

(2) For the question "What famous artist could write with both his left and right hand at the same time?", our model identifies that "Leonardo Da Vinci" is an artist from the first paragraph and could write with both his left and right hand at the same time from the second paragraph.

5 Conclusion and Future Work

In this paper, we propose a denoising distantly supervised open-domain question answering system which contains a paragraph selector to skim over paragraphs and a paragraph reader to perform an intensive reading on the selected paragraphs. Our model can make full use of all informative paragraphs and alleviate the wrong labeling problem in DS-QA. In the experiments, we show that our models significantly and consistently outperforms state-of-the-art DS-QA models. In particular, we demonstrate that the performance of our model is hardly compromised when only using a few top-selected paragraphs.

In the future, we will explore the following directions:

(1) An additional answer re-ranking step can further improve our model. We will explore how to effectively re-rank our extracted answers to further enhance the performance.

(2) Background knowledge such as factual knowledge, common sense knowledge can effectively help us in paragraph selection and answer extraction. We will incorporate external knowledge bases into our DS-QA model to improve its performance.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007 and 61572273). This paper is also partially funded by Microsoft Research Asia FY17-RES-THEME-017.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013a. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*. pages 1533–1544.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013b. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*. pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of ACL*. pages 2358–2367.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the ACL*. pages 1870–1879.

Tongfei Chen and Benjamin Van Durme. 2017. Discriminative information retrieval for question answering sentence selection. In *Proceedings of EACL*. pages 719–725.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of ACL*. pages 484–494.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of ACL*. pages 209–220.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of ACL*. pages 593–602.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017a. Gated-attention readers for text comprehension. In *Proceedings of ACL*. pages 1832–1846.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017b. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904* .

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179* .

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of SIGKDD*. pages 1156–1165.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Proceedings of IRE-AIEE-ACM*. pages 219–224.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*. pages 1601–1611.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Cody Kwok, Oren Etzioni, and Daniel S Weld. 2001. Scaling question answering to the web. *TOIS* pages 242–262.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*. pages 2124–2133.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. pages 1532–1543.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of SIGKDD*. ACM, pages 1047–1055.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Proceedings of TREC*. pages 77–82.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of AAAI*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*. pages 189–198.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL*. pages 1480–1489.

Question Condensing Networks for Answer Selection in Community Question Answering

Wei Wu[1], Xu Sun[1], Houfeng Wang[1,2]

[1]MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[2]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
{wu.wei, xusun, wanghf}@pku.edu.cn

Abstract

Answer selection is an important subtask of community question answering (CQA). In a real-world CQA forum, a question is often represented as two parts: a subject that summarizes the main points of the question, and a body that elaborates on the subject in detail. Previous researches on answer selection usually ignored the difference between these two parts and concatenated them as the question representation. In this paper, we propose the Question Condensing Networks (QCN) to make use of the subject-body relationship of community questions. In this model, the question subject is the primary part of the question representation, and the question body information is aggregated based on similarity and disparity with the question subject. Experimental results show that QCN outperforms all existing models on two CQA datasets.

1 Introduction

Community question answering (CQA) has seen a spectacular increase in popularity in recent years. With the advent of sites like Stack Overflow[1] and Quora[2], more and more people can freely ask any question and expect a variety of answers. With the influx of new questions and the varied quality of provided answers, it is very time-consuming for a user to inspect them all. Therefore, developing automated tools to identify good answers for a question is of practical importance.

A typical example for CQA is shown in Table 1. In this example, Answer 1 is a good answer, because it provides helpful information, e.g., "*check it to the traffic dept*". Although Answer 2 is relevant to the question, it does not contain any useful information so that it should be regarded as a bad answer.

From this example, we can observe two characteristics of CQA that ordinary QA does not possess. First, a question includes both a subject that gives a brief summary of the question and a body that describes the question in detail. The questioners usually convey their main concern and key information in the question subject. Then, they provide more extensive details about the subject, seek help, or express gratitude in the question body. Second, the problem of redundancy and noise is prevalent in CQA (Zhang et al., 2017). Both questions and answers contain auxiliary sentences that do not provide meaningful information.

Previous researches (Tran et al., 2015; Joty et al., 2016) usually treat each word equally in the question and answer representation. However, due to the redundancy and noise problem, only part of text from questions and answers is useful to determine the answer quality. To make things worse, they ignored the difference between question subject and body, and simply concatenated them as the question representation. Due to the subject-body relationship described above, this simple concatenation can aggravate the redundancy problem in the question. In this paper, we propose the Question Condensing Networks (QCN) to address these problems.

In order to utilize the subject-body relationship in community questions, we propose to treat the question subject as the primary part of the question, and aggregate the question body information based on similarity and disparity with the question subject. The similarity part corresponds to the information that exists in both question subject and body, and the disparity part corresponds to the additional information provided by the ques-

[1]https://stackoverflow.com/
[2]https://www.quora.com/

Question Subject	Checking the history of the car.
Question body	How can one check the history of the car like maintenance, accident or service history. In every advertisement of the car, people used to write “Accident Free", but in most cases, car have at least one or two accident, which is not easily detectable through Car Inspection Company. Share your opinion in this regard.
Answer1	Depends on the owner of the car.. if she/he reported the accident/s i believe u can check it to the traffic dept.. but some owners are not doing that especially if its only a small accident.. try ur luck and go to the traffic dept..
Answer2	How about those who claim a low mileage by tampering with the car fuse box? In my sense if you’re not able to detect traces of an accident then it is probably not worth mentioning... For best results buy a new car :)

Table 1: An example question and its related answers in CQA. The text is shown in its original form, which may contain errors in typing.

tion body. Both information can be important for question representation. In our model, they are processed separately and the results are combined to form the final question representation.

In order to reduce the impact of redundancy and noise in both questions and answers, we propose to align the question-answer pairs using the multi-dimensional attention mechanism. Different from previous attention mechanisms that compute a scalar score for each token pair, multi-dimensional attention, first proposed in Shen et al. (2018), computes one attention score for each dimension of the token embedding. Therefore, it can select the features that can best describe the word’s specific meaning in the given context. Therefore, we can learn the interaction between questions and answers more accurately.

The main contributions of our work can be summarized as follows:

- We propose to treat the question subject and the question body separately in community question answering. We treat the question subject as the primary part of the question, and aggregate the question body information based on similarity and disparity with the question subject.

- We introduce a new method that uses the multi-dimensional attention mechanism to align question-answer pair. With this attention mechanism, the interaction between questions and answers can be learned more accurately.

- Our proposed Question Condensing Networks (QCN) achieves the state-of-the-art performance on two SemEval CQA datasets, outperforming all exisiting SOTA models by a large margin, which demonstrates the effectiveness of our model.[3]

2 Task Description

A community question answering consists of four parts, which can be formally defined as a tuple of four elements (S, B, C, y). $S = [\boldsymbol{s}^1, \boldsymbol{s}^2, ..., \boldsymbol{s}^l]$ denotes the subject of a question whose length is l, where each $\boldsymbol{s}^i$ is a one-hot vector whose dimension equals the size of the vocabulary. Similarly, $B = [\boldsymbol{b}^1, \boldsymbol{b}^2, ..., \boldsymbol{b}^m]$ denotes the body of a question whose length is m. $C = [\boldsymbol{c}^1, \boldsymbol{c}^2, ..., \boldsymbol{c}^n]$ denotes an answer corresponding to that question whose length is n. $y \in \mathcal{Y}$ is the label representing the degree to which it can answer that question. $\mathcal{Y} = \{\textit{Good}, \textit{PotentiallyUseful}, \textit{Bad}\}$ where *Good* indicates the answer can answer that question well, *PotentiallyUseful* indicates the answer is potentially useful to the user, and *Bad* indicates the answer is just bad or useless. Given $\{S, B, C\}$, the task of CQA is to assign a label to each answer based on the conditional probability $Pr(y|S, B, C)$.

3 Proposed Model

In this paper, we propose Question Condensing Networks (QCN) which is composed of the following modules. The overall architecture of our model is illustrated in Figure 1.

[3]An implementation of our model is available at `https://github.com/pku-wuwei/QCN`.

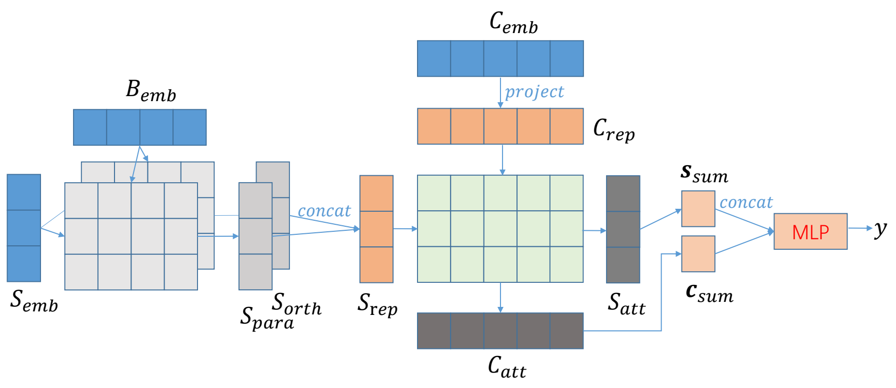


Figure 1: Architecture for Question Condensing Network (QCN). Each block represents a vector.

3.1 Word-Level Embedding

Word-level embeddings are composed of two components: GloVe (Pennington et al., 2014) word vectors trained on the domain-specific unannotated corpus provided by the task [4], and convolutional neural network-based character embeddings which are similar to (Kim et al., 2016). Web text in CQA forums differs largely from normalized text in terms of spelling and grammar, so specifically trained GloVe vectors can model word interactions more precisely. Character embedding has proven to be very useful for out-of-vocabulary (OOV) words, so it is especially suitable for noisy web text in CQA.

We concatenate these two embedding vectors for every word to generate word-level embeddings $\boldsymbol{S}_{emb} \in \mathbb{R}^{d\times l}, \boldsymbol{B}_{emb} \in \mathbb{R}^{d\times m}, \boldsymbol{C}_{emb} \in \mathbb{R}^{d\times n}$, where d is the word-level embedding size.

3.2 Question Condensing

In this section, we condense the question representation using subject-body relationship. In most cases, the question subject can be seen as a summary containing key points of the question, the question body is relatively lengthy in that it needs to explain the key points and add more details about the posted question. We propose to cheat the question subject as the primary part of the question representation, and aggregate question body information from two perspectives: similarity and disparity with the question subject. To achieve this goal, we use an orthogonal decomposition strategy, which is first proposed by Wang et al. (2016), to decompose each question body embedding into a parallel component and an orthogonal component based on every question subject embedding:

$$\boldsymbol{b}^{i,j}_{para} = \frac{\boldsymbol{b}^{j}_{emb} \cdot \boldsymbol{s}^{i}_{emb}}{\boldsymbol{s}^{i}_{emb} \cdot \boldsymbol{s}^{i}_{emb}} \boldsymbol{s}^{i}_{emb} \tag{1}$$

$$\boldsymbol{b}^{i,j}_{orth} = \boldsymbol{b}^{j}_{emb} - \boldsymbol{b}^{i,j}_{para} \tag{2}$$

All vectors in the above equations are of length d. Next we describe the process of aggregating the question body information based on the parallel component in detail. The same process can be applied to the orthogonal component, so at the end of the fusion gate we can obtain S_{orth} and S_{orth} respectively.

The decomposed components are passed through a fully connected layer to compute the multi-dimensional attention weights. Here we use the scaled tanh activation, which is similar to Shen et al. (2018), to prevent large difference among scores while it still has a range large enough for output:

$$\boldsymbol{a}^{i,j}_{para} = c \cdot \tanh\left(\left[W_{p1}\boldsymbol{b}^{i,j}_{para} + \boldsymbol{b}_{p1}\right]/c\right) \tag{3}$$

where $W_{p1} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_{p1} \in \mathbb{R}^{d}$ are parameters to be learned, and c is a hyper-parameter to be tuned.

The obtained word-level alignment tensor $\boldsymbol{A}_{para} \in \mathbb{R}^{d\times l\times m}$ is then normalized along the third dimension to produce the attention weights over the question body for each word in the question subject. The output of this attention mechanism is a weighted sum of the question body embeddings for each word in the question subject:

$$\boldsymbol{w}^{i,j}_{para} = \frac{\exp\left(\boldsymbol{a}^{i,j}_{para}\right)}{\sum_{j=1}^{m}\exp\left(\boldsymbol{a}^{i,j}_{para}\right)} \tag{4}$$

$$\boldsymbol{s}^{i}_{ap} = \sum_{j=1}^{m} \boldsymbol{w}^{i,j}_{para} \odot \boldsymbol{b}^{j}_{emb} \tag{5}$$

[4] http://alt.qcri.org/semeval2015/task3/index.php?id=data-and-tools

where $\odot$ means point-wise product. This multi-dimensional attention mechanism has the advantage of selecting features of a word that can best describe the word's specific meaning in the given context. In order to determine the importance between the original word in the question subject and the aggregated information from the question body with respect to this word, a fusion gate is utilized to combine these two representations:

$$F_{para} = \sigma\left(W_{p2}S_{emb} + W_{p3}S_{ap} + \boldsymbol{b}_{p2}\right) \quad (6)$$
$$S_{para} = F_{para} \odot S_{emb} + (1 - F_{para}) \odot S_{ap} \quad (7)$$

where $W_{p2}, W_{p3} \in \mathbb{R}^{d\times d}$, and $\boldsymbol{b}_{p2} \in \mathbb{R}^{d}$ are learnable parameters of the fusion gate, and $F_{para}, S_{emb}, S_{ap}, S_{para} \in \mathbb{R}^{d\times l}$. The final question representation $S_{rep} \in \mathbb{R}^{2d\times l}$ is obtained by concatenating S_{para} and S_{orth} along the first dimension.

3.3 Answer Preprocessing

This module has two purposes. First, we try to map each answer word from embedding space $\boldsymbol{C}_{emb} \in \mathbb{R}^{d\times n}$ to the same interaction space $\boldsymbol{C}_{rep} \in \mathbb{R}^{2d\times n}$ as the question. Second, similar to Wang and Jiang (2017), a gate is utilized to control the importance of different answer words in determining the question-answer relation:

$$\begin{aligned} C_{rep} =& \sigma\left(W_{c1}C_{emb} + \boldsymbol{b}_{c1}\right) \odot \\ & \tanh\left(W_{c2}C_{emb} + \boldsymbol{b}_{c2}\right) \end{aligned} \quad (8)$$

where $W_{c1}, W_{c2} \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b}_{c1}, \boldsymbol{b}_{c2} \in \mathbb{R}^{2d}$ are parameters to be learned.

3.4 Question Answer Alignment

We apply the multi-dimensional attention mechanism to the question and answer representation S_{rep} and C_{rep} to obtain word-level alignment tensor $\boldsymbol{A}_{align} \in \mathbb{R}^{2d\times l\times n}$. Similar to the multi-dimensional attention mechanism described above, we can compute attention weights and weighted sum for both the question representation and the answer representation :

$$\tilde{\boldsymbol{a}}^{i,j}_{align} = W_{a1}\boldsymbol{s}^{i}_{rep} + W_{a2}\boldsymbol{c}^{j}_{rep} + \boldsymbol{b}_{a} \quad (9)$$
$$\boldsymbol{a}^{i,j}_{align} = c \cdot \tanh\left(\tilde{\boldsymbol{a}}^{i,j}_{align}/c\right) \quad (10)$$
$$\boldsymbol{s}^{i}_{ai} = \sum_{j=1}^{n} \frac{\exp\left(\boldsymbol{a}^{i,j}_{align}\right)}{\sum_{j=1}^{n}\exp\left(\boldsymbol{a}^{i,j}_{align}\right)} \odot \boldsymbol{c}^{j}_{rep} \quad (11)$$
$$\boldsymbol{c}^{j}_{ai} = \sum_{i=1}^{l} \frac{\exp\left(\boldsymbol{a}^{i,j}_{align}\right)}{\sum_{i=1}^{l}\exp\left(\boldsymbol{a}^{i,j}_{align}\right)} \odot \boldsymbol{s}^{i}_{rep} \quad (12)$$

where $W_{a1}, W_{a2} \in \mathbb{R}^{2d\times 2d}$ and $\boldsymbol{b}_{a} \in \mathbb{R}^{2d}$ are parameters to be learned. To attenuate the effect of incorrect attendance, input and output of this attention mechanism are concatenated and fed to the subsequent layer. Finally, we obtain the question and answer representation $S_{att} \in \mathbb{R}^{4d\times l} = [S_{rep}; S_{ai}]$, $C_{att} \in \mathbb{R}^{4d\times n} = [C_{rep}; C_{ai}]$.

3.5 Interaction Summarization

In this layer, the multi-dimensional self-attention mechanism is employed to summarize two sequences of vectors (S_{att} and C_{att}) into two fixed-length vectors $\boldsymbol{s}_{sum} \in \mathbb{R}^{4d}$ and $\boldsymbol{c}_{sum} \in \mathbb{R}^{4d}$.

$$A_{s} = W_{s2}\tanh\left(W_{s1}S_{att} + \boldsymbol{b}_{s1}\right) + \boldsymbol{b}_{s2} \quad (13)$$
$$\boldsymbol{s}_{sum} = \sum_{i=1}^{n} \frac{\exp\left(\boldsymbol{a}^{i}_{s}\right)}{\sum_{i=1}^{n}\exp\left(\boldsymbol{a}^{i}_{s}\right)} \odot \boldsymbol{s}^{i}_{att} \quad (14)$$

where $W_{s1}, W_{s2} \in \mathbb{R}^{4d\times 4d}$ and $\boldsymbol{b}_{s1}, \boldsymbol{b}_{s2} \in \mathbb{R}^{4d}$ are parameters to be learned. The same process can be applied to C_{att} and obtain $\boldsymbol{c}_{sum}$.

3.6 Prediction

In this component, $\boldsymbol{s}_{sum}$ and $\boldsymbol{c}_{sum}$ are concatenated and fed into a two-layer feed-forward neural network. At the end of the last layer, the $softmax$ function is applied to obtain the conditional probability distribution $Pr(y|S, B, C)$.

4 Experimental Setup

4.1 Datasets

We use two community question answering datasets from SemEval (Nakov et al., 2015, 2017) to evaluate our model. The statistics of these datasets are listed in Table 2. The corpora contain data from the QatarLiving forum [5], and are publicly available on the task website. Each dataset

[5]http://www.qatarliving.com/forum

Statistics	SemEval 2015			SemEval 2017		
	Train	Dev	Test	Train	Dev	Test
Number of questions	2376	266	300	5124	327	293
Number of answers	15013	1447	1793	38638	3270	2930
Average length of subject	6.36	6.08	6.24	6.38	6.16	5.76
Average length of body	39.26	39.47	39.53	43.01	47.98	54.06
Average length of answer	35.82	33.90	37.33	37.67	37.30	39.50

Table 2: Statistics of two CQA datasets. We can see from the statistics that the question body is much lengthier than the question subject. Thus, it is necessary to condense the question representation.

consists of questions and a list of answers for each question, and each question consists of a short title and a more detailed description. There are also some metadata associated with them, e.g., user ID, date of posting, and the question category. We do not use the metadata because they failed to boost performance in our model. Since the SemEval 2017 dataset is an updated version of SemEval 2016 [6], and shares the same evaluation metrics with SemEval 2016, we choose to use the SemEval 2017 dataset for evaluation.

4.2 Evaluation Metrics

In order to facilitate comparison, we adopt the evaluation metrics used in the official task or prior work. For the SemEval 2015 dataset, the official scores are macro-averaged F1 and accuracy over three categories. However, many recent researches (Barrón-Cedeño et al., 2015; Joty et al., 2015, 2016) switched to a binary classification setting, i.e., identifying *Good* vs. *Bad* answers. Because binary classification is much closer to a real-world CQA application. Besides, the *PotentiallyUseful* class is both the smallest and the noisiest class, making it the hardest to predict. To make it worse, its impact is magnified by the macro-averaged F1. Therefore, we adopt the F1 score and accuracy on two categories for evaluation.

SemEval 2017 regards answer selection as a ranking task, which is closer to the application scenario. As a result, mean average precision (MAP) is used as an evaluation measure. For a perfect ranking, a system has to place all *Good* answers above the *PotentiallyUseful* and *Bad* answers. The latter two are not actually distinguished and are considered *Bad* in terms of evaluation. Additionally, standard classification measures like accuracy and F1 score are also reported.

[6]The SemEval 2017 dataset provides all the data from 2016 for training , and fresh data for testing, but it does not include a development set. Following previous work (Filice et al., 2017), we use the 2016 official test set as the development set.

4.3 Implementation Details

We use the tokenizer from NLTK (Bird, 2006) to preprocess each sentence. All word embeddings in the sentence encoder layer are initialized with the 300-dimensional GloVe (Pennington et al., 2014) word vectors trained on the domain-specific unannotated corpus, and embeddings for out-of-vocabulary words are set to zero. We use the Adam Optimizer (Kingma and Ba, 2014) for optimization with a first momentum coefficient of 0.9 and a second momentum coefficient of 0.999. We perform a small grid search over combinations of initial learning rate [1×10^{-6}, 3×10^{-6}, 1×10^{-5}], L2 regularization parameter [1×10^{-7}, 3×10^{-7}, 1×10^{-6}], and batch size [8, 16, 32]. We take the best configuration based on performance on the development set, and only evaluate that configuration on the test set. In order to mitigate the class imbalance problem, median frequency balancing Eigen and Fergus (2015) is used to reweight each class in the cross-entropy loss. Therefore, the rarer a class is in the training set, the larger weight it will get in the cross entropy loss. Early stopping is applied to mitigate the problem of overfitting. For the SemEval 2017 dataset, the conditional probability over the *Good* class is used to rank all the candidate answers.

5 Experimental Results

In this section, we evaluate our QCN model on two community question answering datasets from SemEval shared tasks.

5.1 SemEval 2015 Results

Table 3 compares our model with the following baselines:

Methods	F1	Acc
(1) JAIST	78.96	79.10
(2) HITSZ-ICRC	76.52	76.11
(3) Graph-cut	80.55	79.80
(4) FCCRF	81.50	80.50
(5) BGMN	77.23	78.40
(6) CNN-LSTM-CRF	82.22	82.24
(7) QCN	**83.91**	**85.65**

Table 3: Comparisons on the SemEval 2015 dataset.

- **JAIST** (Tran et al., 2015): It used an SVM classifier to incorporate various kinds of features , including topic model based features and word vector representations.
- **HITSZ-ICRC** (Hou et al., 2015): It proposed ensemble learning and hierarchical classification method to classify answers.
- **Graph-cut** (Joty et al., 2015): It modeled the relationship between pairs of answers at any distance in the same question thread, based on the idea that similar answers should have similar labels.
- **FCCRF** (Joty et al., 2016): It used locally learned classifiers to predict the label for each individual node, and applied fully connected CRF to make global inference.
- **CNN-LSTM-CRF** (Xiang et al., 2016): The question and its answers are linearly connected in a sequence and encoded by CNN. An attention-based LSTM with a CRF layer is then applied on the encoded sequence.
- **BGMN** (Wu et al., 2017b): It used the memory mechanism to iteratively aggregate more relevant information which is useful to identify the relationship between questions and answers.

Baselines include top systems from SemEval 2015 (1, 2), systems relying on thread level information to make global inference (3, 4), and neural network based systems (5, 6). We observe that our proposed QCN can achieve the state-of-the-art performance on this dataset, outperforming previous best model (6) by 1.7% in terms of F1 and 3.4% in terms of accuracy.

Methods	MAP	F1	Acc
(1) KeLP	88.43	69.87	73.89
(2) Beihang-MSRA	88.24	68.40	51.98
(3) ECNU	86.72	77.67	78.43
(4) LSTM	86.32	74.41	75.69
(5) LSTM-subject-body	87.11	74.50	77.28
(6) QCN	**88.51**	**78.11**	**80.71**

Table 4: Comparisons on the SemEval 2017 dataset.

Notably, Systems (1, 2, 3, 4) have heavy feature engineering, while QCN only uses automatically-learned feature vectors, demonstrating that our QCN model is concise as well as effective. Furthermore, our model can outperform systems relying on thread level information to make global inference (3, 4), showing that modeling interaction between the question-answer pair is useful enough for answer selection task. Finally, neural network based systems (5, 6) used attention mechanism in sentence representation but ignored the subject-body relationship in community questions. QCN can outperform them by a large margin, showing that condensing question representation helps in the answer selection task.

5.2 SemEval 2017 Results

Table 4 compares our model with the following baselines:

- **KeLP** (Filice et al., 2017): It used syntactic tree kernels with relational links between questions and answers, together with some standard text similarity measures linearly combined with the tree kernel.
- **Beihang-MSRA** (Feng et al., 2017): It used gradient boosted regression trees to combine traditional NLP features and neural network-based matching features.
- **ECNU** (Wu et al., 2017a): It combined a supervised model using traditional features and a convolutional neural network to represent the question-answer pair.
- **LSTM**: It is a simple neural network based baseline that we implemented. In this model, the question subject and the question body are concatenated, and an LSTM is used to obtain the question and answer representation.

- **LSTM-subject-body**: It is another neural network based baseline that we implemented. LSTM is applied on the question subject and body respectively, and the results are concatenated to form question representation.

Baselines include top systems from the SemEval 2017 CQA task (1, 2, 3) and two neural network based baselines (4, 5) that we implemented. (5) can outperform (4), showing that treating question subject and body differently can indeed boot model performance. Comparing (6) with (5), we can draw the conclusion that orthogonal decomposition is more effective than simple concatenation, because it can flexibly aggregate related information from the question body with respect to the main subject. In the example listed in Table 1, attention heatmap of A_{orth} indicates that QCN can effectively find additional information like "*maintenance, accident or service history*", while (5) fails to do so.

QCN has a great advantage in terms of accuracy. We hypothesize that QCN focuses on modeling interaction between questions and answers, i.e., whether an answer can match the corresponding question. Many pieces of previous work focus on modeling relationship between answers in a question thread, i.e., which answer is more suitable in consideration of all other answers. As a consequence, their models have a greater advantage in ranking while QCN has a greater advantage in classification. Despite all this, QCN can still obtain better ranking performance.

5.3 Ablation Study

For thorough comparison, besides the preceding models, we implement nine extra baselines on the SemEval 2017 dataset to analyze the improvements contributed by each part of our QCN model:

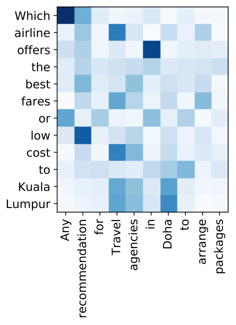
- **w/o task-specific word embeddings** where word embeddings are initialized with the 300-dimensional GloVe word vectors trained on Wikipedia 2014 and Gigaword 5.
- **w/o character embeddings** where word-level embeddings are only composed of 600-dimensional GloVe word vectors trained on the domain-specific unannotated corpus.
- **subject-body alignment** where we use the same attention mechanism as Question Answer Alignment to obtain weighted sum of the question body for each question subject word, and then the result is concatenated with S_{emb} to obtain question representation S_{rep}.
- **subject-body concatenation** where we concatenate question subject and body text, and use the preprocessing step described in section 3.3 to obtain S_{rep}.
- **w/o multi-dimensional attention** where the multi-dimensional attention mechanism is replaced by vanilla attention in all modules, i.e., attention score for each token pair is a scalar instead of a vector.
- **subject only** where only question subject is used as question representation.
- **body only** where only question body is used as question representation.
- **similarity only** where the parallel component alone is used in subject-body interaction.
- **disparity only** where the orthogonal component alone is used in subject-body interaction.

Model	**Acc**
(1) w/o task-specific word embeddings	78.81
(2) w/o character embeddings	78.05
(3) subject-body alignment	77.38
(4) subject-body concatenation	76.06
(5) w/o multi-dimensional attention	78.33
(6) subject only	74.02
(7) body only	75.57
(8) similarity only	79.11
(9) disparity only	78.24
(10) QCN	**80.71**

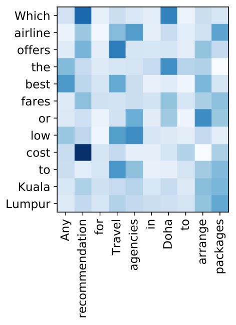
Table 5: Ablation studies on the SemEval 2017 dataset.

The results are listed in Table 5. We can see that using task-specific embeddings and character embeddings both contribute to model performance. This is because CQA text is non-standard. There are quantities of informal language usage, such as abbreviations, typos, emoticons, and grammatical mistakes. Using task-specific embeddings and character embeddings can help to attenuate the OOV problem.

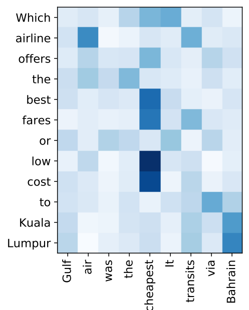
Using orthogonal decomposition (10) instead of subject-body alignment (3) can bring about significant performance gain. This is because not only



(a) A_{para}



(b) A_{orth}



(c) A_{align}

Figure 2: Attention probabilities in A_{para}, A_{orth} and A_{align}. In order to visualize the multi-dimensional attention vector, we use the $L2$ norm of the attenion vector for representation.

the similar part of the question body to the question subject is useful for the question representation, the disparity part can also provide additional information. In the example listed in Table 1, additional information like "*maintenance, accident or service history*" is also important to determine answer quality.

QCN outperforms (4) by a great margin, demonstrating that subject-body relationship in community questions helps to condense question representation. Therefore, QCN can identify the meaningful part of the question representation that helps to determine answer quality.

Using the multi-dimensional attention can further boost model performance, showing that the multi-dimensional attention can model the interaction between questions and answers more precisely.

Comparing QCN with (6) and (7), we can conclude that both the subject and the body are indispensable for question representation. (8) outperforms (9), demonstrating the parallel component is more useful in subject-body interaction.

6 Qualitative Study

To gain a closer view of what dependencies are captured in the subject-body pair and the question-answer pair, we visualize the attention probabilities A_{para}, A_{orth} and A_{align} by heatmap. A training example from SemEval 2015 is selected for illustration.

In Figure 2, we can draw the following conclusions. First, orthogonal decomposition helps to divide the labor of identifying similar parts in the parallel component and collecting related information in the question body in the orthogonal component. For instance, for the word "*Kuala*" in the question subject, its parallel alignment score focuses more on "*Doha*" and "*Travel*", while its orthogonal alignment score focuses on "*arrange*" and "*package*", which is the purpose of the travel and therefore is also indispensable for sentence representation. Second, semantically important words such as "*airline*" and "*fares*" dominate the attention weights, showing that our QCN model can effectively select words that are most representative for the meaning of the whole sentence. Lastly, words that are useful to determine answer quality stand out in the question-answer interaction matrix, demonstrating that question-answer relationship can be well modeled. For example, "*best*" and "*low*" are the words that are more important in the question-answer relationship, they are emphasized in the question-answer alignment matrix.

7 Related Work

One main task in community question answering is answer selection, i.e., to rate the answers according to their quality. The SemEval CQA tasks (Nakov et al., 2015, 2016, 2017) provide universal benchmark datasets for evaluating researches on this problem.

Earlier work of answer selection in CQA relied heavily on feature engineering, linguistic tools, and external resource. Nakov et al. (2016) investigated a wide range of feature types including similarity features, content features, thread level/meta features, and automatically generated features for SemEval CQA models. Tran et al. (2015) studied the use of topic model based features and word vector representation based features in the answer re-ranking task. Filice et al. (2016) designed various heuristic features and thread-based features that can signal a good answer. Although achieving good performance, these methods rely heavily on feature engineering, which requires a large amount of manual work and domain expertise.

Since answer selection is inherently a ranking task, a few recent researches proposed to use local features to make global ranking decision. Barrón-Cedeño et al. (2015) was the first work that applies structured prediction model on CQA answer selection task. Joty et al. (2016) approached the task with a global inference process to exploit the information of all answers in the question-thread in the form of a fully connected graph.

To avoid feature engineering, many deep learning models have been proposed for answer selection. Among them, Zhang et al. (2017) proposed a novel interactive attention mechanism to address the problem of noise and redundancy prevalent in CQA. Tay et al. (2017) introduced temporal gates for sequence pairs so that questions and answers are aware of what each other is remembering or forgetting. Simple as their model are, they did not consider the relationship between question subject and body, which is useful for question condensing.

8 Conclusion and Future Work

We propose Question Condensing Networks (QCN), an attention-based model that can utilize the subject-body relationship in community questions to condense question representation. By orthogonal decomposition, the labor of identifying similar parts and collecting related information in the question body can be well divided in two different alignment matrices. To better capture the interaction between the subject-body pair and the question-answer pair, the multi-dimensional attention mechanism is adopted. Empirical results on two community question answering datasets in SemEval demonstrate the effectiveness of our model. In future work, we will try to incorporate more hand-crafted features in our model. Furthermore, since thread-level features have been explored in previous work (Barrón-Cedeño et al., 2015; Joty et al., 2015, 2016), we will verify their effectiveness in our architecture.

9 Acknowledgments

We would like to thank anonymous reviewers for their insightful comments. Our work is supported by National Natural Science Foundation of China under Grant No.61433015 and the National Key Research and Development Program of China under Grant No.2017YFB1002101. The corresponding author of this paper is Houfeng Wang.

References

Alberto Barrón-Cedeño, Simone Filice, Giovanni Da San Martino, Shafiq R. Joty, Lluís Màrquez, Preslav Nakov, and Alessandro Moschitti. 2015. Thread-level information for comment classification in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*. pages 687–693.

Steven Bird. 2006. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

David Eigen and Rob Fergus. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. pages 2650–2658.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2017. Beihang-msra at semeval-2017 task 3: A ranking system with neural matching features for community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*. pages 280–286.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. pages 1116–1123.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. Kelp at semeval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*. pages 326–333.

Yongshuai Hou, Cong Tan, Xiaolong Wang, Yaoyun Zhang, Jun Xu, and Qingcai Chen. 2015. HITSZ-ICRC: exploiting classification approach for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. pages 196–202.

Shafiq R. Joty, Alberto Barrón-Cedeño, Giovanni Da San Martino, Simone Filice, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2015. Global thread-level inference for comment classification in community question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 573–578.

Shafiq R. Joty, Lluís Màrquez, and Preslav Nakov. 2016. Joint learning with global inference for comment classification in community question answering. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 703–713.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. pages 2741–2749.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*. pages 27–48.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. pages 269–281.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. pages 525–545.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI Conference on Artificial Intelligence*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Cross temporal recurrent networks for ranking question answer pairs. *CoRR* abs/1711.07656.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. JAIST: combining multiple features for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. pages 215–219.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

GuoShun Wu, Yixuan Sheng, Man Lan, and Yuanbin Wu. 2017a. ECNU at semeval-2017 task 3: Using traditional and deep learning methods to address community question answering task. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*. pages 365–369.

Wei Wu, Houfeng Wang, and Sujian Li. 2017b. Bidirectional gated memory networks for answer selection. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data. LNAI 10565, Springer*. pages 251–262.

Yang Xiang, Xiaoqiang Zhou, Qingcai Chen, Zhihui Zheng, Buzhou Tang, Xiaolong Wang, and Yang Qin. 2016. Incorporating label dependency for answer quality tagging in community question answering via CNN-LSTM-CRF. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 1231–1241.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive interactive neural networks for answer selection in community question answering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3525–3531.

Towards Robust Neural Machine Translation

Yong Cheng***, Zhaopeng Tu*****, Fandong Meng*****, Junjie Zhai*** **and Yang Liu**†
*Tencent AI Lab, China
†State Key Laboratory of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Beijing Advanced Innovation Center for Language Resources
chengyong3001@gmail.com
{zptu, fandongmeng, jasonzhai}@tencent.com
liuyang2011@tsinghua.edu.cn

Abstract

Small perturbations in the input can severely distort intermediate representations and thus impact translation quality of neural machine translation (NMT) models. In this paper, we propose to improve the robustness of NMT models with adversarial stability training. The basic idea is to make both the encoder and decoder in NMT models robust against input perturbations by enabling them to behave similarly for the original input and its perturbed counterpart. Experimental results on Chinese-English, English-German and English-French translation tasks show that our approaches can not only achieve significant improvements over strong NMT systems but also improve the robustness of NMT models.

1 Introduction

Neural machine translation (NMT) models have advanced the state of the art by building a single neural network that can better learn representations (Cho et al., 2014; Sutskever et al., 2014). The neural network consists of two components: an encoder network that encodes the input sentence into a sequence of distributed representations, based on which a decoder network generates the translation with an attention model (Bahdanau et al., 2015; Luong et al., 2015). A variety of NMT models derived from this encoder-decoder framework have further improved the performance of machine translation systems (Gehring et al., 2017; Vaswani et al., 2017). NMT is capable of generalizing better to unseen text by exploiting word similarities in embeddings and capturing long-distance reordering by conditioning on larger contexts in a continuous way.

Input	tamen *bupa* kunnan zuochu weiqi AI.
Output	They are not afraid of difficulties to make Go AI.
Input	tamen *buwei* kunnan zuochu weiqi AI.
Output	They are not afraid to make Go AI.

Table 1: The non-robustness problem of neural machine translation. Replacing a Chinese word with its synonym (i.e., "*bupa*" $\rightarrow$ "*buwei*") leads to significant erroneous changes in the English translation. Both "*bupa*" and "*buwei*" can be translated to the English phrase "*be not afraid of*."

However, studies reveal that very small changes to the input can fool state-of-the-art neural networks with high probability (Goodfellow et al., 2015; Szegedy et al., 2014). Belinkov and Bisk (2018) confirm this finding by pointing out that NMT models are very brittle and easily falter when presented with noisy input. In NMT, due to the introduction of RNN and attention, each contextual word can influence the model prediction in a global context, which is analogous to the "butterfly effect." As shown in Table 1, although we only replace a source word with its synonym, the generated translation has been completely distorted. We investigate severe variations of translations caused by small input perturbations by replacing one word in each sentence of a test set with its synonym. We observe that 69.74% of translations have changed and the BLEU score is only 79.01 between the translations of the original inputs and the translations of the perturbed inputs, suggesting that NMT models are very sensitive to small perturbations in the input. The vulnerability and instability of NMT models limit their applicability to a broader range of tasks, which require robust performance on noisy inputs. For example, simultaneous translation systems use automatic speech recognition (ASR) to transcribe input speech into a sequence of hypothesized words, which are subsequently fed to a translation system. In this pipeline, ASR errors are presented as sentences with noisy perturbations (the same pronunciation but incorrect words), which is a significant challenge for current NMT models. Moreover, instability makes NMT models sensitive to misspellings and typos in text translation.

In this paper, we address this challenge with *adversarial stability training* for neural machine translation. The basic idea is to improve the robustness of two important components in NMT: the encoder and decoder. To this end, we propose two approaches to constructing noisy inputs with small perturbations to make NMT models resist them. As important intermediate representations encoded by the encoder, they directly determine the accuracy of final translations. We introduce adversarial learning to make behaviors of the encoder consistent for both an input and its perturbed counterpart. To improve the stability of the decoder, our method jointly maximizes the likelihoods of original and perturbed data. Adversarial stability training has the following advantages:

1. *Improving both the robustness and translation performance*: Our adversarial stability training is capable of not only improving the robustness of NMT models but also achieving better translation performance.

2. *Applicable to arbitrary noisy perturbations*: In this paper, we propose two approaches to constructing noisy perturbations for inputs. However, our training framework can be easily extended to arbitrary noisy perturbations. Especially, we can design task-specific perturbation methods.

3. *Transparent to network architectures*: Our adversarial stability training does not depend on specific NMT architectures. It can be applied to arbitrary NMT systems.

Experiments on Chinese-English, English-French and English-German translation tasks show that adversarial stability training achieves significant improvements across different languages pairs. Our NMT system outperforms the state-of-the-art RNN-based NMT system (GNMT) (Wu et al., 2016) and obtains comparable performance with the CNN-based NMT system (Gehring et al., 2017). Related experimental analyses validate that our training approach can improve the robustness of NMT models.

2 Background

NMT is an end-to-end framework which directly optimizes the translation probability of a target sentence $\mathbf{y} = y_1, ..., y_N$ given its corresponding source sentence $\mathbf{x} = x_1, ..., x_M$:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{n=1}^{N} P(y_n|\mathbf{y}_{<n}, \mathbf{x};\boldsymbol{\theta}) \quad (1)$$

where $\boldsymbol{\theta}$ is a set of model parameters and $\mathbf{y}_{<n}$ is a partial translation. $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ is defined on a holistic neural network which mainly includes two core components: an *encoder* encodes a source sentence $\mathbf{x}$ into a sequence of hidden representations $\mathbf{H_x} = \mathbf{H}_1, ..., \mathbf{H}_M$, and a *decoder* generates the n-th target word based on the sequence of hidden representations:

$$P(y_n|\mathbf{y}_{<n}, \mathbf{x};\boldsymbol{\theta}) \propto \exp\{g(y_{n-1}, \mathbf{s}_n, \mathbf{H_x};\boldsymbol{\theta})\} \quad (2)$$

where $\mathbf{s}_n$ is the n-th hidden state on target side. Thus the model parameters of NMT include the parameter sets of the encoder $\boldsymbol{\theta}_{\text{enc}}$ and the decoder $\boldsymbol{\theta}_{\text{dec}}$: $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}\}$. The standard training objective is to minimize the negative log-likelihood of the training corpus $\mathcal{S} = \{\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)} \rangle\}_{s=1}^{|\mathcal{S}|}$:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{x}, \mathbf{y};\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ \sum_{\langle \mathbf{x},\mathbf{y}\rangle \in \mathcal{S}} -\log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \Big\} \quad (3)\end{aligned}$$

Due to the vulnerability and instability of deep neural networks, NMT models usually suffer from a drawback: small perturbations in the input can dramatically deteriorate its translation results. Belinkov and Bisk (2018) point out that character-based NMT models are very brittle and easily falter when presented with noisy input. We find that word-based and subword-based NMT models also confront with this shortcoming, as shown in Table 1. We argue that the distributed representations should fulfill the stability expectation, which is the underlying concept of the proposed approach. Recent work has shown that adversarially trained models can be made robust to such perturbations (Zheng et al., 2016; Madry et al., 2018). Inspired by this, in this work, we improve the robustness of encoder representations against noisy perturbations with adversarial learning (Goodfellow et al., 2014).

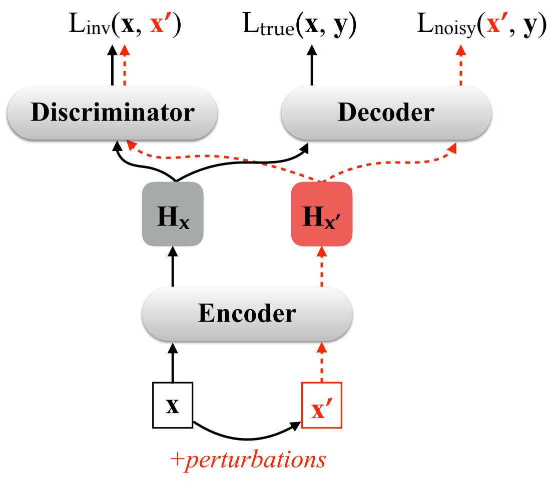


Figure 1: The architecture of NMT with adversarial stability training. The dark solid arrow lines represent the forward-pass information flow for the input sentence $\mathbf{x}$, while the red dashed arrow lines for the noisy input sentence $\mathbf{x}'$, which is transformed from $\mathbf{x}$ by adding small perturbations.

3 Approach

3.1 Overview

The goal of this work is to propose a general approach to make NMT models learned to be more robust to input perturbations. Our basic idea is to maintain the consistency of behaviors through the NMT model for the source sentence $\mathbf{x}$ and its perturbed counterpart $\mathbf{x}'$. As aforementioned, the NMT model contains two procedures for projecting a source sentence $\mathbf{x}$ to its target sentence $\mathbf{y}$: the encoder is responsible for encoding $\mathbf{x}$ as a sequence of representations $\mathbf{H_x}$, while the decoder outputs $\mathbf{y}$ with $\mathbf{H_x}$ as input. We aim at learning the perturbation-invariant encoder and decoder.

Figure 1 illustrates the architecture of our approach. Given a source sentence $\mathbf{x}$, we construct a set of perturbed sentences $\mathcal{N}(\mathbf{x})$, in which each sentence $\mathbf{x}'$ is constructed by adding small perturbations to $\mathbf{x}$. We require that $\mathbf{x}'$ is a subtle variation from $\mathbf{x}$ and they have similar semantics. Given the input pair $(\mathbf{x}, \mathbf{x}')$, we have two expectations: (1) the encoded representation $\mathbf{H_{x'}}$ should be close to $\mathbf{H_x}$; and (2) given $\mathbf{H_{x'}}$, the decoder is able to generate the robust output $\mathbf{y}$. To this end, we introduce two additional objectives to improve the robustness of the encoder and decoder:

- $\mathcal{L}_{\text{inv}}(\mathbf{x}, \mathbf{x}')$ to encourage the encoder to output similar intermediate representations $\mathbf{H_x}$ and $\mathbf{H_{x'}}$ for $\mathbf{x}$ and $\mathbf{x}'$ to achieve an invariant encoder, which benefits outputting the same translations. We cast this objective in the adversarial learning framework.

- $\mathcal{L}_{\text{noisy}}(\mathbf{x}', \mathbf{y})$ to guide the decoder to generate output $\mathbf{y}$ given the noisy input $\mathbf{x}'$, which is modeled as $-\log P(\mathbf{y}|\mathbf{x}')$. It can also be defined as KL divergence between $P(\mathbf{y}|\mathbf{x})$ and $P(\mathbf{y}|\mathbf{x}')$ that indicates using $P(\mathbf{y}|\mathbf{x})$ to teach $P(\mathbf{y}|\mathbf{x}')$.

As seen, the two introduced objectives aim to improve the robustness of the NMT model which can be free of high variances in target outputs caused by small perturbations in inputs. It is also natural to introduce the original training objective $\mathcal{L}(\mathbf{x}, \mathbf{y})$ on $\mathbf{x}$ and $\mathbf{y}$, which can guarantee good translation performance while keeping the stability of the NMT model.

Formally, given a training corpus $\mathcal{S}$, the adversarial stability training objective is

$$\begin{aligned} & \mathcal{J}(\boldsymbol{\theta}) \\ = & \sum_{\langle \mathbf{x},\mathbf{y}\rangle \in \mathcal{S}} \Big(\mathcal{L}_{\text{true}}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}) \\ & + \alpha \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \mathcal{L}_{\text{inv}}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dis}}) \\ & + \beta \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \mathcal{L}_{\text{noisy}}(\mathbf{x}', \mathbf{y}; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}) \Big) \end{aligned} \quad (4)$$

where $\mathcal{L}_{\text{true}}(\mathbf{x}, \mathbf{y})$ and $\mathcal{L}_{\text{noisy}}(\mathbf{x}', \mathbf{y})$ are calculated using Equation 3, and $\mathcal{L}_{\text{inv}}(\mathbf{x}, \mathbf{x}')$ is the adversarial loss to be described in Section 3.3. α and β control the balance between the original translation task and the stability of the NMT model. $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dec}}, \boldsymbol{\theta}_{\text{dis}}\}$ are trainable parameters of the encoder, decoder, and the newly introduced discriminator used in adversarial learning. As seen, the parameters of encoder $\boldsymbol{\theta}_{\text{enc}}$ and decoder $\boldsymbol{\theta}_{\text{dec}}$ are trained to minimize both the translation loss $\mathcal{L}_{\text{true}}(\mathbf{x}, \mathbf{y})$ and the stability losses ($\mathcal{L}_{\text{noisy}}(\mathbf{x}', \mathbf{y})$ and $\mathcal{L}_{\text{inv}}(\mathbf{x}, \mathbf{x}')$).

Since $\mathcal{L}_{\text{noisy}}(\mathbf{x}', \mathbf{y})$ evaluates the translation loss on the perturbed neighbour $\mathbf{x}'$ and its corresponding target sentence $\mathbf{y}$, it means that we augment the training data by adding perturbed neighbours, which can potentially improve the translation performance. In this way, our approach not only makes the output of NMT models more robust, but also improves the performance on the original translation task.

In the following sections, we will first describe how to construct perturbed inputs with different strategies to fulfill different goals (Section 3.2), followed by the proposed adversarial learning mechanism for the perturbation-invariant encoder (Section 3.3). We conclude this section with the training strategy (Section 3.4).

3.2 Constructing Perturbed Inputs

At each training step, we need to generate a perturbed neighbour set $\mathcal{N}(\mathbf{x})$ for each source sentence $\mathbf{x}$ for adversarial stability training. In this paper, we propose two strategies to construct the perturbed inputs at multiple levels of representations.

The first approach generates perturbed neighbours at the *lexical* level. Given an input sentence $\mathbf{x}$, we randomly sample some word positions to be modified. Then we replace words at these positions with other words in the vocabulary according to the following distribution:

$$P(x|\mathbf{x}_i) = \frac{\exp\{\cos(\mathbf{E}[\mathbf{x}_i], \mathbf{E}[x])\}}{\sum_{x \in \mathcal{V}_x \backslash \mathbf{x}_i} \exp\{\cos(\mathbf{E}[\mathbf{x}_i], \mathbf{E}[x])\}} \quad (5)$$

where $\mathbf{E}[\mathbf{x}_i]$ is the word embedding for word $\mathbf{x}_i$, $\mathcal{V}_x \backslash \mathbf{x}_i$ is the source vocabulary set excluding the word $\mathbf{x}_i$, and $\cos(\mathbf{E}[\mathbf{x}_i], \mathbf{E}[x])$ measures the similarity between word $\mathbf{x}_i$ and x. Thus we can change the word to another word with similar semantics.

One potential problem of the above strategy is that it is hard to enumerate all possible positions and possible types to generate perturbed neighbours. Therefore, we propose a more general approach to modifying the sentence at the *feature* level. Given a sentence, we can obtain the word embedding for each word. We add the Gaussian noise to a word embedding to simulate possible types of perturbations. That is

$$\mathbf{E}[\mathbf{x}'_i] = \mathbf{E}[\mathbf{x}_i] + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathbf{N}(0, \sigma^2\mathbf{I}) \quad (6)$$

where the vector $\boldsymbol{\epsilon}$ is sampled from a Gaussian distribution with variance σ^2. σ is a hyper-parameter. We simply introduce Gaussian noise to all of word embeddings in $\mathbf{x}$.

The proposed scheme is a general framework where one can freely define the strategies to construct perturbed inputs. We just present two possible examples here. The first strategy is potentially useful when the training data contains noisy words, while the latter is a more general strategy to improve the robustness of common NMT models. In practice, one can design specific strategies for particular tasks. For example, we can replace correct words with their homonyms (same pronunciation but different meanings) to improve NMT models for simultaneous translation systems.

3.3 Adversarial Learning for the Perturbation-invariant Encoder

The goal of the perturbation-invariant encoder is to make the representations produced by the encoder indistinguishable when fed with a correct sentence $\mathbf{x}$ and its perturbed counterpart $\mathbf{x}'$, which is directly beneficial to the output robustness of the decoder. We cast the problem in the adversarial learning framework (Goodfellow et al., 2014). The encoder serves as the generator G, which defines the policy that generates a sequence of hidden representations $\mathbf{H}_\mathbf{x}$ given an input sentence $\mathbf{x}$. We introduce an additional discriminator D to distinguish the representation of perturbed input $\mathbf{H}_{\mathbf{x}'}$ from that of the original input $\mathbf{H}_\mathbf{x}$. The goal of the generator G (i.e., encoder) is to produce similar representations for $\mathbf{x}$ and $\mathbf{x}'$ which could fool the discriminator, while the discriminator D tries to correctly distinguish the two representations.

Formally, the adversarial learning objective is

$$\begin{aligned} & \mathcal{L}_{\text{inv}}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_{\text{dis}}) \\ = \; & \mathbb{E}_{\mathbf{x} \sim \mathcal{S}}[-\log D(G(\mathbf{x}))] + \\ & \mathbb{E}_{\mathbf{x}' \sim \mathcal{N}(\mathbf{x})}\left[-\log(1 - D(G(\mathbf{x}')))\right] \quad (7) \end{aligned}$$

The discriminator outputs a classification score given an input representation, and tries to maximize $D(G(\mathbf{x}))$ to 1 and minimize $D(G(\mathbf{x}'))$ to 0. The objective encourages the encoder to output similar representations for $\mathbf{x}$ and $\mathbf{x}'$, so that the discriminator fails to distinguish them.

The training procedure can be regarded as a min-max two-player game. The encoder parameters $\boldsymbol{\theta}_{\text{enc}}$ are trained to maximize the loss function to fool the discriminator. The discriminator parameters $\boldsymbol{\theta}_{\text{dis}}$ are optimized to minimize this loss for improving the discriminating ability. For efficiency, we update both the encoder and the discriminator simultaneously at each iteration, rather than the periodical training strategy that is commonly used in adversarial learning. Lamb et al. (2016) also propose a similar idea to use Professor Forcing to make the behaviors of RNNs be indistinguishable when training and sampling the networks.

3.4 Training

As shown in Figure 1, our training objective includes three sets of model parameters for three modules. We use mini-batch stochastic gradient descent to optimize our model. In the forward pass, besides a mini-batch of $\mathbf{x}$ and $\mathbf{y}$, we also construct a mini-batch consisting of the perturbed neighbour $\mathbf{x}'$ and $\mathbf{y}$. We propagate the information to calculate these three loss functions according to arrows. Then, gradients are collected to update three sets of model parameters. Except for the gradients of $\mathcal{L}_{\text{inv}}$ with respect to $\boldsymbol{\theta}_{\text{enc}}$ are multiplying by -1, other gradients are normally back-propagated. Note that we update $\boldsymbol{\theta}_{\text{inv}}$ and $\boldsymbol{\theta}_{\text{enc}}$ simultaneously for training efficiency.

4 Experiments

4.1 Setup

We evaluated our adversarial stability training on translation tasks of several language pairs, and reported the 4-gram BLEU (Papineni et al., 2002) score as calculated by the *multi-bleu.perl* script.

Chinese-English We used the LDC corpus consisting of 1.25M sentence pairs with 27.9M Chinese words and 34.5M English words respectively. We selected the best model using the NIST 2006 set as the validation set (hyper-parameter optimization and model selection). The NIST 2002, 2003, 2004, 2005, and 2008 datasets are used as test sets.

English-German We used the WMT 14 corpus containing 4.5M sentence pairs with 118M English words and 111M German words. The validation set is newstest2013, and the test set is newstest2014.

English-French We used the IWSLT corpus which contains 0.22M sentence pairs with 4.03M English words and 4.12M French words. The IWLST corpus is very dissimilar from the NIST and WMT corpora. As they are collected from TED talks and inclined to spoken language, we want to verify our approaches on the non-normative text. The IWSLT 14 test set is taken as the validation set and 15 test set is used as the test set.

For English-German and English-French, we tokenize both English, German and French words using `tokenize.perl` script. We follow Sennrich et al. (2016b) to split words into sub-word units. The numbers of merge operations in byte pair encoding (BPE) are set to 30K, 40K and 30K respectively for Chinese-English, English-German, and English-French. We report the case-sensitive tokenized BLEU score for English-German and English-French and the case-insensitive tokenized BLEU score for Chinese-English.

Our baseline system is an in-house NMT system. Following Bahdanau et al. (2015), we implement an RNN-based NMT in which both the encoder and decoder are two-layer RNNs with residual connections between layers (He et al., 2016b). The gating mechanism of RNNs is gated recurrent unit (GRUs) (Cho et al., 2014). We apply layer normalization (Ba et al., 2016) and dropout (Hinton et al., 2012) to the hidden states of GRUs. Dropout is also added to the source and target word embeddings. We share the same matrix between the target word embeddings and the pre-softmax linear transformation (Vaswani et al., 2017). We update the set of model parameters using Adam SGD (Kingma and Ba, 2015). Its learning rate is initially set to 0.05 and varies according to the formula in Vaswani et al. (2017).

Our adversarial stability training initializes the model based on the parameters trained by maximum likelihood estimation (MLE). We denote adversarial stability training based on lexical-level perturbations and feature-level perturbations respectively as $\text{AST}_{\text{lexical}}$ and $\text{AST}_{\text{feature}}$. We only sample one perturbed neighbour $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ for training efficiency. For the discriminator used in $\mathcal{L}_{\text{inv}}$, we adopt the CNN discriminator proposed by Kim (2014) to address the variable-length problem of the sequence generated by the encoder. In the CNN discriminator, the filter windows are set to 3, 4, 5 and rectified linear units are applied after convolution operations. We tune the hyper-parameters on the validation set through a grid search. We find that both the optimal values of α and β are set to 1.0. The standard variance in Gaussian noise used in the formula (6) is set to 0.01. The number of words that are replaced in the sentence $\mathbf{x}$ during lexical-level perturbations is taken as $\max(0.2|\mathbf{x}|, 1)$ in which $|\mathbf{x}|$ is the length of $\mathbf{x}$. The default beam size for decoding is 10.

4.2 Translation Results

4.2.1 NIST Chinese-English Translation

Table 2 shows the results on Chinese-English translation. Our strong baseline system significantly outperforms previously reported results on

System	Training	MT06	MT02	MT03	MT04	MT05	MT08
Shen et al. (2016)	MRT	37.34	40.36	40.93	41.37	38.81	29.23
Wang et al. (2017)	MLE	37.29	–	39.35	41.15	38.07	–
Zhang et al. (2018)	MLE	38.38	–	40.02	42.32	38.84	–
this work	MLE	41.38	43.52	41.50	43.64	41.58	31.60
	$\text{AST}_{\text{lexical}}$	43.57	44.82	42.95	45.05	43.45	34.85
	$\text{AST}_{\text{feature}}$	**44.44**	**46.10**	**44.07**	**45.61**	**44.06**	**34.94**

Table 2: Case-insensitive BLEU scores on Chinese-English translation.

System	Architecture	Training	BLEU
Shen et al. (2016)	Gated RNN with 1 layer	MRT	20.45
Luong et al. (2015)	LSTM with 4 layers	MLE	20.90
Kalchbrenner et al. (2017)	ByteNet with 30 layers	MLE	23.75
Wang et al. (2017)	DeepLAU with 4 layers	MLE	23.80
Wu et al. (2016)	LSTM with 8 layers	RL	24.60
Gehring et al. (2017)	CNN with 15 layers	MLE	25.16
Vaswani et al. (2017)	Self-attention with 6 layers	MLE	28.40
this work	Gated RNN with 2 layers	MLE	24.06
		$\text{AST}_{\text{lexical}}$	25.17
		$\text{AST}_{\text{feature}}$	**25.26**

Table 3: Case-sensitive BLEU scores on WMT 14 English-German translation.

Training	tst2014	tst2015
MLE	36.92	36.90
$\text{AST}_{\text{lexical}}$	37.35	37.03
$\text{AST}_{\text{feature}}$	**38.03**	**37.64**

Table 4: Case-sensitive BLEU scores on IWSLT English-French translation.

Chinese-English NIST datasets trained on RNN-based NMT. Shen et al. (2016) propose minimum risk training (MRT) for NMT, which directly optimizes model parameters with respect to BLEU scores. Wang et al. (2017) address the issue of severe gradient diffusion with linear associative units (LAU). Their system is deep with an encoder of 4 layers and a decoder of 4 layers. Zhang et al. (2018) propose to exploit both left-to-right and right-to-left decoding strategies for NMT to capture bidirectional dependencies. Compared with them, our NMT system trained by MLE outperforms their best models by around 3 BLEU points. We hope that the strong baseline systems used in this work make the evaluation convincing.

We find that introducing adversarial stability training into NMT can bring substantial improvements over previous work (up to +3.16 BLEU points over Shen et al. (2016), up to +3.51 BLEU points over Wang et al. (2017) and up to +2.74 BLEU points over Zhang et al. (2018)) and our system trained with MLE across all the datasets. Compared with our baseline system, $\text{AST}_{\text{lexical}}$ achieves +1.75 BLEU improvement on average. $\text{AST}_{\text{feature}}$ performs better, which can obtain +2.59 BLEU points on average and up to +3.34 BLEU points on NIST08.

4.2.2 WMT 14 English-German Translation

In Table 3, we list existing NMT systems as comparisons. All these systems use the same WMT 14 English-German corpus. Except that Shen et al. (2016) and Wu et al. (2016) respectively adopt MRT and reinforcement learning (RL), other systems all use MLE as training criterion. All the systems except for Shen et al. (2016) are deep NMT models with no less than four layers. Google's neural machine translation (GNMT) (Wu et al., 2016) represents a strong RNN-based NMT system. Compared with other RNN-based NMT systems except for GNMT, our baseline system with two layers can achieve better performance than theirs.

When training our NMT system with $\text{AST}_{\text{leixcal}}$, significant improvement (+1.11

Synthetic Type	Training	0 Op.	1 Op.	2 Op.	3 Op.	4 Op.	5 Op.
Swap	MLE	41.38	38.86	37.23	35.97	34.61	32.96
	$\text{AST}_{\text{lexical}}$	43.57	41.18	39.88	37.95	37.02	36.16
	$\text{AST}_{\text{feature}}$	44.44	42.08	40.20	38.67	36.89	35.81
Replacement	MLE	41.38	37.21	31.40	27.43	23.94	21.03
	$\text{AST}_{\text{lexical}}$	43.57	40.53	37.59	35.19	32.56	30.42
	$\text{AST}_{\text{feature}}$	44.44	40.04	35.00	30.54	27.42	24.57
Deletion	MLE	41.38	38.45	36.15	33.28	31.17	28.65
	$\text{AST}_{\text{lexical}}$	43.57	41.89	38.56	36.14	34.09	31.77
	$\text{AST}_{\text{feature}}$	44.44	41.75	39.06	36.16	33.49	30.90

Table 5: Translation results of synthetic perturbations on the validation set in Chinese-English translation. "1 Op." denotes that we conduct one operation (swap, replacement or deletion) on the original sentence.

Source	zhongguo dianzi yinhang yewu guanli xingui jiangyu sanyue yiri qi shixing
Reference	china's new management rules for e-banking operations to take effect on march 1
MLE	china's electronic bank rules to be implemented on march 1
$\text{AST}_{\text{lexical}}$	new rules for business administration of china 's electronic banking industry will come into effect on march 1 .
$\text{AST}_{\text{feature}}$	new rules for business management of china 's electronic banking industry to come into effect on march 1
Perturbed Source	*zhongfang* dianzi yinhang yewu guanli xingui jiangyu sanyue yiri qi shixing
MLE	china to implement new regulations on business management
$\text{AST}_{\text{lexical}}$	the new regulations for the business administrations of the chinese electronics bank will come into effect on march 1 .
$\text{AST}_{\text{feature}}$	new rules for business management of china's electronic banking industry to come into effect on march 1

Table 6: Example translations of a source sentence and its perturbed counterpart by replacing a Chinese word "zhongguo" with its synonym "zhongfang."

BLEU points) can be observed. $\text{AST}_{\text{feature}}$ can obtain slightly better performance. Our NMT system outperforms the state-of-the-art RNN-based NMT system, GNMT, with +0.66 BLEU point and performs comparably with Gehring et al. (2017) which is based on CNN with 15 layers. Given that our approach can be applied to any NMT systems, we expect that the adversarial stability training mechanism can further improve performance upon the advanced NMT architectures. We leave this for future work.

4.2.3 IWSLT English-French Translation

Table 4 shows the results on IWSLT English-French Translation. Compared with our strong baseline system trained by MLE, we observe that our models consistently improve translation performance in all datasets. $\text{AST}_{\text{feature}}$ can achieve significant improvements on the tst2015 although $\text{AST}_{\text{lexical}}$ obtains comparable results. These demonstrate that our approach maintains good performance on the non-normative text.

4.3 Results on Synthetic Perturbed Data

In order to investigate the ability of our training approaches to deal with perturbations, we experiment with three types of synthetic perturbations:

- **Swap**: We randomly choose N positions from a sentence and then swap the chosen words with their right neighbours.
- **Replacement**: We randomly replace sampled words in the sentence with other words.
- **Deletion**: We randomly delete N words from each sentence in the dataset.

As shown in Table 5, we can find that our training approaches, $\text{AST}_{\text{lexical}}$ and $\text{AST}_{\text{feature}}$, consistently outperform MLE against perturbations on all the numbers of operations. This means that our

$\mathcal{L}_{\text{true}}$	$\mathcal{L}_{\text{noisy}}$	$\mathcal{L}_{\text{adv}}$	BLEU
$\surd$	$\times$	$\times$	41.38
$\surd$	$\times$	$\surd$	41.91
$\times$	$\surd$	$\times$	42.20
$\surd$	$\surd$	$\times$	42.93
$\surd$	$\surd$	$\surd$	43.57

Table 7: Ablation study of adversarial stability training $\text{AST}_{\text{lexical}}$ on Chinese-English translation. "$\surd$" means the loss function is included in the training objective while "$\times$" means it is not.

approaches have the capability of resisting perturbations. Along with the number of operations increasing, the performance on MLE drops quickly. Although the performance of our approaches also drops, we can see that our approaches consistently surpass MLE. In $\text{AST}_{\text{lexical}}$, with 0 operation, the difference is +2.19 (43.57 Vs. 41.38) for all synthetic types, but the differences are enlarged to +3.20, +9.39, and +3.12 respectively for the three types with 5 operations.

In the *Swap* and *Deletion* types, $\text{AST}_{\text{lexical}}$ and $\text{AST}_{\text{feature}}$ perform comparably after more than four operations. Interestingly, $\text{AST}_{\text{lexical}}$ performs significantly better than both of MLE and $\text{AST}_{\text{feature}}$ after more than one operation in the *Replacement* type. This is because $\text{AST}_{\text{lexical}}$ trains the model specifically on perturbation data that is constructed by replacing words, which agrees with the *Replacement* Type. Overall, $\text{AST}_{\text{lexical}}$ performs better than $\text{AST}_{\text{feature}}$ against perturbations after multiple operations. We speculate that the perturbation method for $\text{AST}_{\text{lexical}}$ and synthetic type are both discrete and they keep more consistent. Table 6 shows example translations of a Chinese sentence and its perturbed counterpart.

These findings indicate that we can construct specific perturbations for a particular task. For example, in simultaneous translation, an automatic speech recognition system usually generates wrong words with the same pronunciation of correct words, which dramatically affects the quality of machine translation system. Therefore, we can design specific perturbations aiming for this task.

4.4 Analysis

4.4.1 Ablation Study

Our training objective function Eq. (4) contains three loss functions. We perform an ablation study on the Chinese-English translation to understand the importance of these loss functions by choosing $\text{AST}_{\text{lexical}}$ as an example. As Table 7 shows, if we remove $\mathcal{L}_{\text{adv}}$, the translation performance decreases by 0.64 BLEU point. However, when $\mathcal{L}_{\text{noisy}}$ is excluded from the training objective function, it results in a significant drop of 1.66 BLEU point. Surprisingly, only using $\mathcal{L}_{\text{noisy}}$ is able to lead to an increase of 0.88 BLEU point.

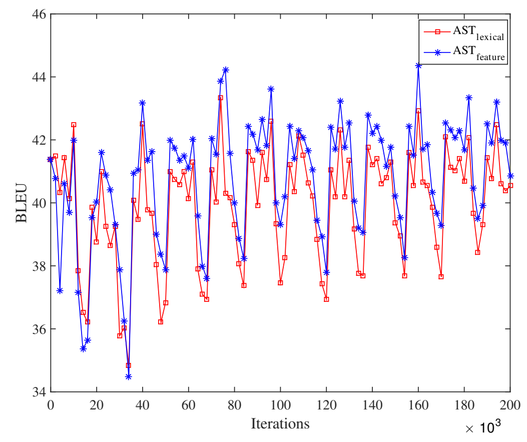


Figure 2: BLEU scores of $\text{AST}_{\text{lexical}}$ over iterations on Chinese-English validation set.

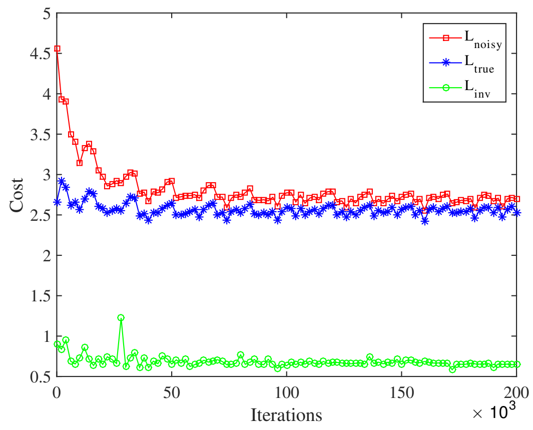


Figure 3: Learning curves of three loss functions, $\mathcal{L}_{\text{true}}$, $\mathcal{L}_{\text{inv}}$ and $\mathcal{L}_{\text{noisy}}$ over iterations on Chinese-English validation set.

4.4.2 BLEU Scores over Iterations

Figure 2 shows the changes of BLEU scores over iterations respectively for $\text{AST}_{\text{lexical}}$ and $\text{AST}_{\text{feature}}$. They behave nearly consistently. Initialized by the model trained by MLE, their performance drops rapidly. Then it starts to go up quickly. Compared with the starting point, the maximal dropping points reach up to about 7.0 BLEU points. Basically, the curves present the state of oscillation. We think that introducing random perturbations and adversarial learning can make the training not very stable like MLE.

4.4.3 Learning Curves of Loss Functions

Figure 3 shows the learning curves of three loss functions, $\mathcal{L}_{\text{true}}$, $\mathcal{L}_{\text{inv}}$ and $\mathcal{L}_{\text{noisy}}$. We can find that their costs of loss functions decrease not steadily. Similar to the Figure 2, there still exist oscillations in the learning curves although they do not change much sharply. We find that $\mathcal{L}_{\text{inv}}$ converges to around 0.68 after about $100K$ iterations, which indicates that discriminator outputs probability 0.5 for both positive and negative samples and it cannot distinguish them. Thus the behaviors of the encoder for $\mathbf{x}$ and its perturbed neighbour $\mathbf{x}'$ perform nearly consistently.

5 Related Work

Our work is inspired by two lines of research: (1) adversarial learning and (2) data augmentation.

Adversarial Learning Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and its related derivative have been widely applied in computer vision (Radford et al., 2015; Salimans et al., 2016) and natural language processing (Li et al., 2017; Yang et al., 2018). Previous work has constructed adversarial examples to attack trained networks and make networks resist them, which has proved to improve the robustness of networks (Goodfellow et al., 2015; Miyato et al., 2016; Zheng et al., 2016). Belinkov and Bisk (2018) introduce adversarial examples to training data for character-based NMT models. In contrast to theirs, adversarial stability training aims to stabilize both the encoder and decoder in NMT models. We adopt adversarial learning to learn the perturbation-invariant encoder.

Data Augmentation Data augmentation has the capability to improve the robustness of NMT models. In NMT, there is a number of work that augments the training data with monolingual corpora (Sennrich et al., 2016a; Cheng et al., 2016; He et al., 2016a; Zhang and Zong, 2016). They all leverage complex models such as inverse NMT models to generate translation equivalents for monolingual corpora. Then they augment the parallel corpora with these pseudo corpora to improve NMT models. Some authors have recently endeavored to achieve zero-shot NMT through transferring knowledge from bilingual corpora of other language pairs (Chen et al., 2017; Zheng et al., 2017; Cheng et al., 2017) or monolingual corpora (Lample et al., 2018; Artetxe et al., 2018). Our work significantly differs from these work. We do not resort to any complicated models to generate perturbed data and do not depend on extra monolingual or bilingual corpora. The way we exploit is more convenient and easy to implement. We focus more on improving the robustness of NMT models.

6 Conclusion

We have proposed adversarial stability training to improve the robustness of NMT models. The basic idea is to train both the encoder and decoder robust to input perturbations by enabling them to behave similarly for the original input and its perturbed counterpart. We propose two approaches to construct perturbed data to adversarially train the encoder and stabilize the decoder. Experiments on Chinese-English, English-German and English-French translation tasks show that the proposed approach can improve both the robustness and translation performance.

As our training framework is not limited to specific perturbation types, it is interesting to evaluate our approach in natural noise existing in practical applications, such as homonym in the simultaneous translation system. It is also necessary to further validate our approach on more advanced NMT architectures, such as CNN-based NMT (Gehring et al., 2017) and Transformer (Vaswani et al., 2017).

Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions. We also thank Xiaoling Li for analyzing experimental results and providing valuable examples. Yang Liu is supported by the National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No. 61761166008, No. 61522204), Beijing Advanced Innovation Center for Language Resources, and the NExT++ project supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of ICLR*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of ICLR*.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of ACL*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of ACL*.

Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint training for pivot-based neural machine translation. In *Proceedings of IJCAI*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of ICML*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016a. Dual learning for machine translation. In *Proceedings of NIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Deep residual learning for image recognition. In *Proceedings of CVPR*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2017. Neural machine translation in linear time. In *Proceedings of ICML*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Proceedings of NIPS*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of ICLR*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of EMNLP*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Aleksander Madry, Makelov Aleksandar, Schmidt Ludwig, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of ICLR*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *Proceedings of ICLR*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a methof for automatic evaluation of machine translation. In *Proceedings of ACL*.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Proceedings of NIPS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving nerual machine translation models with monolingual data. In *Proceedings of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceddings of NIPS*.

Christian Szegedy, Wojciech Zaremba, Sutskever Ilya, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of ICML*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In *Proceedings of ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Z. Yang, W. Chen, F. Wang, and B. Xu. 2018. Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets. In *Proceedings of NAACL*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP*.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous Bidirectional Decoding for Neural Machine Translation. In *Proeedings of AAAI*.

Hao Zheng, Yong Cheng, and Yang Liu. 2017. Maximum expected likelihood estimation for zero-resource neural machine translation. In *Proceedings of IJCAI*.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of CVPR*.

Attention Focusing for Neural Machine Translation by Bridging Source and Target Embeddings

Shaohui Kuang[1] **Junhui Li**[1] **António Branco**[2] **Weihua Luo**[3] **Deyi Xiong**[1*]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
shaohuikuang@foxmail.com, {lijunhui, dyxiong}@suda.edu.cn
[2]University of Lisbon, NLX-Natural Language and Speech Group, Department of Informatics
Faculdade de Ciências, Campo Grande, 1749-016 Lisboa, Portuga
antonio.branco@di.fc.ul.pt
[3]Alibaba Group, Hangzhou, China
weihua.luowh@alibaba-inc.com

Abstract

In neural machine translation, a source sequence of words is encoded into a vector from which a target sequence is generated in the decoding phase. Differently from statistical machine translation, the associations between source words and their possible target counterparts are not explicitly stored. Source and target words are at the two ends of a long information processing procedure, mediated by hidden states at both the source encoding and the target decoding phases. This makes it possible that a source word is incorrectly translated into a target word that is not any of its admissible equivalent counterparts in the target language.

In this paper, we seek to somewhat shorten the distance between source and target words in that procedure, and thus strengthen their association, by means of a method we term bridging source and target word embeddings. We experiment with three strategies: (1) a source-side bridging model, where source word embeddings are moved one step closer to the output target sequence; (2) a target-side bridging model, which explores the more relevant source word embeddings for the prediction of the target sequence; and (3) a direct bridging model, which directly connects source and target word embeddings seeking to minimize errors in the translation of ones by the others.

Experiments and analysis presented in this paper demonstrate that the proposed bridging models are able to significantly improve quality of both sentence translation, in general, and alignment and translation of individual source words with target words, in particular.

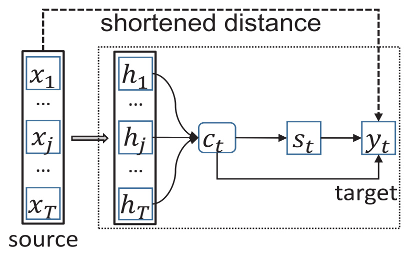


Figure 1: Schematic representation of seq2seq NMT, where $x_1, \ldots, x_T$ and $h_1, \ldots, h_T$ represent source-side word embeddings and hidden states respectively, c_t represents a source-side context vector, s_t a target-side decoder RNN hidden state, and y_t a predicted word. Seeking to shorten the distance between source and target word embeddings, in what we term bridging, is the key insight for the advances presented in this paper.

1 Introduction

Neural machine translation (NMT) is an end-to-end approach to machine translation that has achieved competitive results vis-a-vis statistical machine translation (SMT) on various language pairs (Bahdanau et al., 2015; Cho et al., 2014; Sutskever et al., 2014; Luong and Manning, 2015). In NMT, the sequence-to-sequence (seq2seq) model learns word embeddings for both source and target words synchronously. However, as illustrated in Figure 1, source and target word embeddings are at the two ends of a long information processing procedure. The individual associations between them will gradually become loose due to the separation of source-side hidden states (represented by $h_1, \ldots, h_T$ in Fig. 1) and a target-

[*] Corresponding author

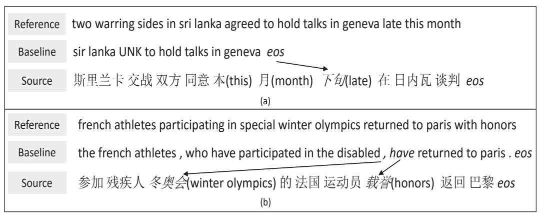

Reference
two warring sides in sri lanka agreed to hold talks in geneva late this month
Baseline
sir lanka UNK to hold talks in geneva *eos*
Source
斯里兰卡 交战 双方 同意 本(this) 月(month) *下旬*(late) 在 日内瓦 谈判 *eos*
(a)
Reference
french athletes participating in special winter olympics returned to paris with honors
Baseline
the french athletes , who have participated in the disabled *,* *have* returned to paris . *eos*
Source
参加 残疾人 *冬奥会*(winter olympics) 的 法国 运动员 *载誉*(honors) 返回 巴黎 *eos*
(b)

Figure 2: Examples of NMT output with incorrect alignments of source and target words that cannot be the translation of each other in any possible context.

side hidden state (represented by s_t in Fig. 1). As a result, in the absence of a more tight interaction between source and target word pairs, the seq2seq model in NMT produces tentative translations that contain incorrect alignments of source words with target counterparts that are non-admissible equivalents in any possible translation context.

Differently from SMT, in NMT an attention model is adopted to help align output with input words. The attention model is based on the estimation of a probability distribution over all input words for each target word. Word alignments with attention weights can then be easily deduced from such distributions and support the translation. Nevertheless, sometimes one finds translations by NMT that contain surprisingly wrong word alignments, that would unlikely occur in SMT.

For instance, Figure 2 shows two Chinese-to-English translation examples by NMT. In the top example, the NMT seq2seq model incorrectly aligns the target side end of sentence mark *eos* to 下旬*/late* with a high attention weight (0.80 in this example) due to the failure of appropriately capturing the similarity, or the lack of it, between the source word 下旬*/late* and the target *eos*. It is also worth noting that, as 本*/this* and 月*/month* end up not being translated in this example, inappropriate alignment of target side *eos* is likely the responsible factor for under translation in NMT as the decoding process ends once a target *eos* is generated. Statistics on our development data show that as much as 50% of target side *eos* do not properly align to source side *eos*.

The second example in Figure 2 shows another case where source words are translated into target items that are not their possible translations in that or in any other context. In particular, 冬奥会*/winter olympics* is incorrectly translated into a target comma "," and 载誉*/honors* into *have*.

In this paper, to address the problem illustrated above, we seek to shorten the distance within the seq2seq NMT information processing procedure between source and target word embeddings. This is a method we term as bridging, and can be conceived as strengthening the focus of the attention mechanism into more translation-plausible source and target word alignments. In doing so, we hope that the seq2seq model is able to learn more appropriate word alignments between source and target words.

We propose three simple yet effective strategies to bridge between word embeddings. The inspiring insight in all these three models is to move source word embeddings closer to target word embeddings along the seq2seq NMT information processing procedure. We categorize these strategies in terms of how close the source and target word embeddings are along that procedure, schematically depicted in Fig. 1.

(1) **Source-side bridging model:** Our first strategy for bridging, which we call source-side bridging, is to move source word embeddings just one step closer to the target end. Each source word embedding is concatenated with the respective source hidden state at the same position so that the attention model can more closely benefit from source word embeddings to produce word alignments.

(2) **Target-side bridging model:** In a second more bold strategy, we seek to incorporate relevant source word embeddings more closely into the prediction of the next target hidden state. In particular, the most appropriate source words are selected according to their attention weights and they are made to more closely interact with target hidden states.

(3) **Direct bridging model:** The third model consists of directly bridging between source and target word embeddings. The training objective is optimized towards minimizing the distance between target word embeddings and their most relevant source word embeddings, selected according to the attention model.

Experiments on Chinese-English translation with extensive analysis demonstrate that directly bridging word embeddings at the two ends can produce better word alignments and thus achieve better translation.

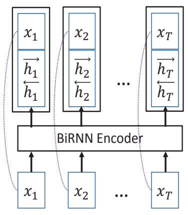


Figure 3: Architecture of the source-side bridging model.

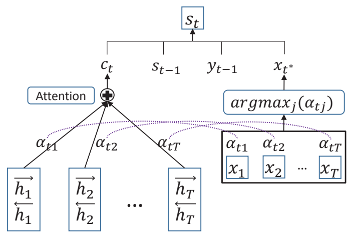

s_t
c_t
s_{t-1}
y_{t-1}
x_{t^*}
Attention
$argmax_j(\alpha_{tj})$
α_{t1}
α_{t2}
α_{tT}
α_{t1} α_{t2} α_{tT}
x_1 x_2 ... x_T
$\overrightarrow{h_1}$ $\overleftarrow{h_1}$
$\overrightarrow{h_2}$ $\overleftarrow{h_2}$
...
$\overrightarrow{h_T}$ $\overleftarrow{h_T}$

Figure 4: Architecture of target-side bridging model.

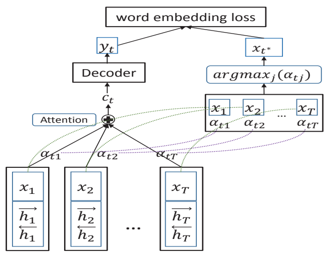

word embedding loss
y_t
x_{t^*}
Decoder
$argmax_j(\alpha_{tj})$
c_t
Attention
x_1 x_2 ... x_T
α_{t1} α_{t2} α_{tT}
α_{t1}
α_{t2}
α_{tT}
x_1 $\overrightarrow{h_1}$ $\overleftarrow{h_1}$
x_2 $\overrightarrow{h_2}$ $\overleftarrow{h_2}$
...
x_T $\overrightarrow{h_T}$ $\overleftarrow{h_T}$

Figure 5: Architecture of direct bridging model.

2 Bridging Models

As suggested by Figure 1, there may exist different ways to bridge between x and y_t. We concentrate on the folowing three bridging models.

2.1 Source-side Bridging Model

Figure 3 illustrates the source-side bridging model. The encoder reads a word sequence equipped with word embeddings and generates a word annotation vector for each position. Then we simply concatenate the word annotation vector with its corresponding word embedding as the final annotation vector. For example, the final annotation vector h_j for the word x_j in Figure 3 is $[\overrightarrow{h_j}; \overleftarrow{h_j}; x_j]$, where the first two sub-items $[\overrightarrow{h_j}; \overleftarrow{h_j}]$ are the source-side forward and backward hidden states and x_j is the corresponding word embedding. In this way, the word embeddings will not only have a more strong contribution in the computation of attention weights, but also be part of the annotation vector to form the weighted source context vector and consequently have a more strong impact in the prediction of target words.

2.2 Target-side Bridging Model

While the above source-side bridging method uses the embeddings of all words for every target word, in the target-side method only more relevant source word embeddings for bridging are explored, rather than all of them. This is partially inspired by the word alignments from SMT, where words from the two ends are paired as they are possible translational equivalents of each other and those pairs are explicitly recorded and enter into the system inner workings. In particular, for a given target word, we explicitly determine the most likely source word aligned to it and use the word embedding of this source word to support the prediction of the target hidden state of the next target word to be generated.

Figure 4 schematically illustrates the target-side bridging method, where the input for computing the hidden state s_t of the decoder is augmented by x_{t^*}, as follows:

$$s_t = f(s_{t-1}, y_{t-1}, c_t, x_{t^*}) \tag{1}$$

where x_{t^*} is the word embedding of the selected source word which has the highest attention weight:

$$t^* = \arg\max_j(\alpha_{tj}) \tag{2}$$

where α_{tj} is the attention weight of each hidden state h_j computed by the attention model

2.3 Direct Bridging Model

Further to the above two bridging methods, which use source word embeddings to predict target words, we seek to bridge the word embeddings of the two ends in a more direct way. This is done by resorting to an auxiliary objective function to narrow the discrepancy between word embeddings of the two sides.

Figure 5 is a schematic representation of our direct bridging method, with an auxiliary objective function. More specifically, the goal is to let the learned word embeddings on the two ends be transformable, i.e. if a target word e_i aligns with a source word f_j, a transformation matrix W is learned with the hope that the discrepancy of Wx_i and y_j tends to be zero. Accordingly, we update the objective function of training for a single sentence with its following extended formulation:

$$L(\theta) = -\sum_{t=1}^{T_y}(\log p(y_t|y_{<t}, x) - \|Wx_{t^*} - y_t\|^2) \quad (3)$$

where $\log p(y_t|y_{<t}, x)$ is the original objective function of the NMT model, and the term $\|Wx_{t^*} - y_t\|^2$ measures and penalizes the difference between target word y_t and its aligned source word x_{t^*}, i.e. the one with the highest attention weight, as computed in Equation 2. Similar to Mi et al. (2016), we view the two parts of the loss in Equation 3 as equally important.

At this juncture, it is worth noting the following:

- Our direct bridging model is an extension of the source-side bridging model, where the source word embeddings are part of the final annotation vector of the encoder. We have also tried to place the auxiliary object function directly on the NMT baseline model. However, our empirical study showed that the combined objective consistently worsens the translation quality. We blame this on that the learned word embeddings on two sides by the baseline model are too heterogeneous to be constrained.

- Rather than using a concrete source word embedding x_{t^*} in Equation 3, we could also use a weighted sum of source word embeddings, i.e. $\sum_j \alpha_{tj} h_j$. However, our preliminary experiments showed that the performance gap between these two methods is very small. Therefore, we use x_{t^*} to calculate the new training objective as shown in Equation 3 in all experiments.

3 Experiments

As we have presented above three different methods to bridge between source and target word embeddings, in the present section we report on a series of experiments on Chinese to English translation that are undertaken to assess the effectiveness of those bridging methods.

3.1 Experimental Settings

We resorted to Chinese-English bilingual corpora that contain 1.25M sentence pairs extracted from LDC corpora, with 27.9M Chinese words and 34.5M English words respectively.[1] We chose the NIST06 dataset as our development set, and the NIST02, NIST03, NIST04, NIST08 datasets as our test sets.

We used the case-insensitive 4-gram NIST BLEU score as our evaluation metric (Papineni et al., 2002) and the script 'mteval-v11b.pl' to compute BLEU scores. We also report TER scores on our dataset (Snover et al., 2006).

For the efficient training of the neural networks, we limited the source (Chinese) and target (English) vocabularies to the most frequent 30k words, covering approximately 97.7% and 99.3% of the two corpora respectively. All the out-of-vocabulary words were mapped to the special token *UNK*. The dimension of word embedding was 620 and the size of the hidden layer was 1000. All other settings were the same as in Bahdanau et al. (2015). The maximum length of sentences that we used to train the NMT model in our experiments was set to 50, for both the Chinese and English sides. Additionally, during decoding, we used the beam-search algorithm and set the beam size to 10. The model parameters were selected according to the maximum BLEU points on the development set.

We compared our proposed models against the following two systems:

- cdec (Dyer et al., 2010): this is an open source hierarchical phrase-based SMT system (Chiang, 2007) with default configuration and a 4-gram language model trained on the target side of the training data.

- **RNNSearch***: this is an attention-based NMT system, taken from the dl4mt tutorial with slight changes. It improves the attention model by feeding the lastly generated word. For the activation function f of an RNN, we use the gated recurrent unit (GRU) (Chung et al., 2014). Dropout was applied only on the output layer and the dropout (Hinton et al., 2012) rate was set to 0.5. We used the stochastic gradient descent algorithm with mini-batch and Adadelta (Zeiler, 2012) to train the NMT models. The mini-batch was set to 80 sentences and decay rates

[1] The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

	Model	NIST06	NIST02	NIST03	NIST04	NIST08	Avg
BLEU	cdec (SMT)	34.00	35.81	34.70	37.15	25.28	33.23
	RNNSearch*	35.92	37.88	36.21	38.83	26.30	34.81
	Source bridging	36.79‡	38.71‡	37.24‡	40.28‡	27.40‡	35.91
	Target bridging	36.69	39.04‡	37.63‡	40.41‡	**27.98**‡	36.27
	Direct bridging	**36.97**‡	**39.77**‡	**38.02**‡	**40.83**‡	27.85‡	**36.62**
TER	cdec (SMT)	58.29	59.65	59.28	58.12	61.54	59.64
	RNNSearch*	59.56	57.79	59.25	57.88	64.22	59.78
	Source bridging	58.13	**56.25**	57.33	56.32	62.13	**58.01**
	Target bridging	58.01	56.27	57.76	56.33	**62.12**	58.12
	Direct bridging	**57.20**	56.68	**57.29**	**55.62**	62.49	58.02

Table 1: BLEU and TER scores on the NIST Chinese-English translation tasks. The BLEU scores are case-insensitive. Avg means the average scores on all test sets. "‡": statistically better than RNNSearch* ($p < 0.01$). Higher BLEU (or lower TER) scores indicate better translation quality.

ρ and ε of Adadelta were set to 0.95 and 10^{-6}.

For our NMT system with the direct bridging model, we use a simple pre-training strategy to train our model. We first train a regular attention-based NMT model, then use this trained model to initialize the parameters of the NMT system equipped with the direct bridging model and randomly initialize the additional parameters of the direct bridging model in this NMT system. The reason of using pre-training strategy is that the embedding loss requires well-trained word alignment as a starting point.

3.2 Experimental Results

Table 1 displays the translation performance measured in terms of BLEU and TER scores. Clearly, every one of the three NMT models we proposed, with some bridging method, improve the translation accuracy over all test sets in comparison to the SMT (cdec) and NMT (RNNSearch*) baseline systems.

Parameters

The three proposed models introduce new parameters in different ways. The source-side bridging model augments source hidden states from a dimension of 2,000 to 2,620, requiring 3.7M additional parameters to accommodate the hidden states that are appended. The target-side bridging model introduces 1.8M additional parameters for catering x_{t^*} in calculating target side state, as in Equation 1. Based on the source-side bridging model, the direct bridging model requires extra 0.4M parameters (i.e. the transformation matrix W in Equation 3 is $620 * 620$), resulting in 4.1M additional parameters over the baseline. Given that the baseline model has 74.8M parameters, the size of extra parameters in our proposed models are comparably small.

System	Percentage (%)
RNNSearch*	49.82
Direct bridging	81.30

Table 2: Percentage of target side *eos* translated from source side *eos* on the development set.

Comparison with the baseline systems

The results in Table 1 indicate that all NMT systems outperform the SMT system taking into account the evaluation metrics considered, BLEU and TER. This is consistent with other studies on Chinese to English machine translation (Mi et al., 2016; Tu et al., 2016; Li et al., 2017). Additionally, all the three NMT models with bridging mechanisms we proposed outperform the baseline NMT model RNNSearch*.

With respect to BLEU scores, we observe a consistent trend that the target-side bridging model works better than the source-side bridging model, while the direct bridging model achieves the best accuracy over all test sets, with the only exception of NIST MT 08. On all test sets, the direct bridging model outperforms the baseline RNNSearch* by 1.81 BLEU points and outperforms the other two bridging-improved NMT models by 0.4∼0.6 BLEU points.

Though all models are not tuned on TER score, our three models perform favorably well with similar average improvement, of about 1.70 TER points, below the baseline model.

4 Analysis

As the direct bridging system proposed achieves the best performance, we further look at it and at

System	Target POS Tag	Source POS Tag				
		V	N	CD	JJ	AD
RNNSearch*	V	64.95	-	-	-	12.09
	N	7.31	39.24	-	-	-
	CD	-	33.37	53.40	-	-
	JJ	-	26.79	-	14.67	-
Direct bridging	V	66.29	-	-	-	10.94
	N	7.19	39.71	-	-	-
	CD	-	32.25	56.29	-	-
	JJ	-	26.12	-	15.22	-

Table 3: Confusion matrix for translation by POS, in percentage. To cope with fine-grained differences among verbs (e.g., VV, VC and VE in Chinese, and VB, VBD, VBP, etc. in English), we merge all verbs into V. Similarly, we merged all nouns into N. CD stands for Cardinal numbers, JJ for adjectives or modifiers, AD for adverbs. These POS tags exist in both Chinese and English. For the sake of simplicity, for each target POS tag, we present only the two source POS tags that are more frequently aligned with it.

the RNNSearch* baseline system to gain further insight on how bridging may help in translation. Our approach presents superior results along all the dimensions assessed.

4.1 Analysis of Word Alignment

Since our improved model strengthens the focus of attention between pairs of translation equivalents by explicitly bridging source and target word embeddings, we expect to observe improved word alignment quality. The quality of the word alignment is examined from the following three aspects.

Better *eos* translation

As a special symbol marking the end of sentence, target side *eos* has a critical impact on controlling the length of the generated translation. A target *eos* is a correct translation when is aligned with the source *eos*. Table 2 displays the percentage of target side *eos* that are translations of the source side *eos*. It indicates that our model improved with bridging substantially achieves better translation of source *eos*.

Better word translation

To have a further insight into the quality of word translation, we group generated words by their part-of-speech (POS) tags and examine the POS of their aligned source words. [2]

Table 3 is a confusion matrix for translations by POS. For example, under RNNSearch*, 64.95% of target verbs originate from verbs in the source side. This is enhanced to 66.29% in our direct bridging model. From the data in that table, one observes that in general more target words align to source words with the same POS tags in our improved system than in the baseline system.

[2]We used Stanford POS tagger (Toutanova et al., 2003) to get POS tags for the words in source sentences and their translations.

System	SAER	AER
RNNSearch*	62.68	47.61
Direct bridging	59.72	44.71

Table 4: Alignment error rate (AER) and soft AER. quality. A lower score indicates better alignment.

Better word alignment

Next we report on the quality of word alignment taking into account a manually aligned dataset, from Liu and Sun (2015), which contains 900 manually aligned Chinese-English sentence pairs. We forced the decoder to output reference translations in order to get automatic alignments between input sentences and their reference translations yielded by the translation systems. To evaluate alignment performance, we measured the alignment error rate (AER) (Och and Ney, 2003) and the soft AER (SAER) (Tu et al., 2016), which are registered in Table 4.

The data in this Table 4 indicate that, as expected, bridging improves the alignment quality as a consequence of its favoring of a stronger relationship between the source and target word embeddings of translational equivalents.

4.2 Analysis of Long Sentence Translation

Following Bahdanau et al. (2015), we partition sentences by their length and compute the respec-

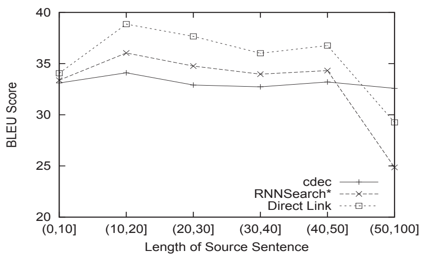


Figure 6: BLEU scores for the translation of sentences with different lengths.

tive BLEU scores, which are presented in Figure 6. These results indicate that our improved system outperforms RNNSearch* for all the sentence lengths. They also reveal that the performance drops substantially when the length of the input sentence increases. This trend is consistent with the findings in (Cho et al., 2014; Tu et al., 2016; Li et al., 2017).

One also observes that the NMT systems perform very badly on sentences of length over 50, when compared to the performance of the baseline SMT system (cdec). We think that the degradation of NMT systems performance over long sentences is due to the following reasons: (1) during training, the maximum source sentence length limit is set to 50, thus making the learned models not ready to cope well with sentences over this maximum length limit; (2) for long input sentences, NMT systems tend to stop early in the generation of the translation.

4.3 Analysis of Over and Under Translation

To assess the expectation that improved translation of *eos* improves the appropriate termination of the translations generated by the decoder, we analyze the performance of our best model with respect to over translation and under translation, which are both notoriously a hard problem for NMT.

To estimate the over translation generated by an NMT system, we follow Li et al. (2017) and report the ratio of over translation (ROT)[3], which is computed as the total number of times of over translation of words in a word set (e.g., all nouns in the source part of the test set) divided by the number of words in the word set.

Table 5 displays ROTs of words grouped by some prominent POS tags. These data indicate that both systems have higher over translation with proper nouns (NR) and other nouns (NN) than

[3]please refer to (Li et al., 2017) for more details of ROT.

System	POS	**ROT**(%)
RNNSearch*	NN	8.63
	NR	12.92
	DT	4.01
	CD	7.05
	ALL	5.28
Direct bridging	NN	7.56
	NR	10.88
	DT	2.37
	CD	4.79
	ALL	4.49

Table 5: Ratios of over translation (ROT) on test sets. NN stands for nouns excluding proper nouns and temporal nouns, NR for proper nouns, DT for determiners, and CD for cardinal numbers.

System	**1-gram BLEU**
cdec (SMT)	77.09
RNNSearch*	72.70
Direct bridging	74.22

Table 6: 1-gram BLEU scores averaged on test sets, supporting the assessment of under translation. A larger score indicates less under translation.

with other POS tags, which is consistent with the results in (Li et al., 2017). The likely reason is that these two POS tags usually have more unknown words, which are words that tend to be over translated. Importantly, these data also show that our direct bridging model alleviates the over translation issue by 15%, as ROT drops from 5.28% to 4.49%.

While it is hard to obtain an accurate estimation of under translation, we simply report 1-gram BLEU score that measures how many words in the translation outcome appear in the reference translation, roughly indicating the proportion of source words that are translated. Table 6 presents the average 1-gram BLEU scores on our test datasets. These data indicate that our improved system has a higher score than RNNSearch*, suggesting that it is less prone to under translation.

It is also worth noting that the SMT baseline (cdec) presents the highest 1-gram BLEU score, as expected, given that under translation is known to be less of an issue for SMT.

4.4 Analysis of Learned Word Embeddings

In the direct bridging model, we introduced a transformation matrix to convert a source-side word embedding into its counterpart on the target side. We seek now to assess the contribution of

Src	Transformation	Lexical Table
是	is	is
和	and	and
及	and	and
将	will	will
会	will	will
国	countries	countries
发展	development	development
经济	economic	economic
问题	question	issue
人民	people	people

Table 7: Top 10 more frequent source words and their closest translations obtained, respectively, by embedding transformation in NMT and from the lexical translation table in SMT.

this transformation. Given a source word x_i, we obtain its closest target word y^* via:

$$y^* = \arg\min_y(\|wx_i - y\|) \quad (4)$$

Table 7 lists the 10 more frequent source words and their corresponding closest target words. For the sake of comparison, it also displays their most likely translations from the lexical translation table in SMT. These results suggest that the closest target words obtained via the transformation matrix of our direct bridging method are very consistent with those obtained from the SMT lexical table, containing only admissible translation pairs. These data thus suggest that our improved model has a good capability of capturing the translation equivalence between source and target word embeddings.

5 Related Work

Since the pioneer work of Bahdanau et al. (2015) to jointly learning alignment and translation in NMT, many effective approaches have been proposed to further improve the alignment quality.

The attention model plays a crucial role in the alignment quality and thus its enhancement has continuously attracted further efforts. To obtain better attention focuses, Luong et al. (2015) propose global and local attention models; and Cohn et al. (2016) extend the attentional model to include structural biases from word based alignment models, including positional bias, Markov conditioning, fertility and agreement over translation directions.

In contrast, we did not delve into the attention model or sought to redesign it in our new bridging proposal. And yet we achieve enhanced alignment quality by inducing the NMT model to capture more favorable pairs of words that are translation equivalents of each other under the effect of the bridging mechanism.

Recently there have been also studies towards leveraging word alignments from SMT models. Mi et al. (2016) and Liu et al. (2016) use pre-obtained word alignments to guide the NMT attention model in the learning of favorable word pairs. Arthur et al. (2016) leverage a pre-obtained word dictionary to constrain the prediction of target words. Despite these approaches having a somewhat similar motivation of using pairs of translation equivalents to benefit the NMT translation, in our new bridging approach we do not use extra resources in the NMT model, but let the model itself learn the similarity of word pairs from the training data. [4]

Besides, there exist also studies on the learning of cross-lingual embeddings for machine translation. Mikolov et al. (2013) propose to first learn distributed representation of words from large monolingual data, and then learn a linear mapping between vector spaces of languages. Gehring et al. (2017) introduce source word embeddings to predict target words. These approaches are somewhat similar to our source-side bridging model. However, inspired by the insight of shortening the distance between source and target embeddings in the seq2seq processing chain, in the present paper we propose more strategies to bridge source and target word embeddings and with better results.

6 Conclusion

We have presented three models to bridge source and target word embeddings for NMT. The three models seek to shorten the distance between source and target word embeddings along the extensive information procedure in the encoder-decoder neural network.

Experiments on Chinese to English translation shows that the proposed models can significantly improve the translation quality. Further in-depth analysis demonstrate that our models are able (1) to learn better word alignments than the baseline NMT, (2) to alleviate the notorious problems of over and under translation in NMT, and (3) to learn direct mappings between source and target words.

[4]Though the pre-obtained word alignments or word dictionaries can be learned from MT training data in an unsupervised fashion, these are still extra knowledge with respect to to the NMT models.

In future work, we will explore further strategies to bridge the source and target side for sequence-to-sequence and tree-based NMT. Additionally, we also intend to apply these methods to other sequence-to-sequence tasks, including natural language conversation.

Acknowledgment

The present research was partly supported by the National Natural Science Foundation of China (Grant No. 61622209), the CNPTDeepMT grant of the Portugal-China Bilateral Exchange Program (ANI/3279/2016) and the Infrastructure for the Science and Technology of the Portuguese Language (PORTULAN / CLARIN).

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceddings of EMNLP 2016*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP 2014*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of Deep Learning and Representation Learning Workshop in NIPS 2014*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of NAACL 2016*, pages 876–885.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of ACL 2017*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *In Proceddings of COLING 2016*.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI 2015*, pages 2295–2301.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceddings of EMNLP 2016*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceddings of AMTA 2006*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS 2014*, pages 3104–3112.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL 2016*, pages 76–85.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Reliability and Learnability of Human Bandit Feedback for Sequence-to-Sequence Reinforcement Learning

Julia Kreutzer[1] and **Joshua Uyheng**[3*] and **Stefan Riezler**[1,2]
[1]Computational Linguistics & [2]IWR, Heidelberg University, Germany
{kreutzer,riezler}@cl.uni-heidelberg.de
[3]Departments of Psychology & Mathematics, Ateneo de Manila University, Philippines
juyheng@ateneo.edu

Abstract

We present a study on reinforcement learning (RL) from human bandit feedback for sequence-to-sequence learning, exemplified by the task of bandit neural machine translation (NMT). We investigate the reliability of human bandit feedback, and analyze the influence of reliability on the learnability of a reward estimator, and the effect of the quality of reward estimates on the overall RL task. Our analysis of cardinal (5-point ratings) and ordinal (pairwise preferences) feedback shows that their intra- and inter-annotator α-agreement is comparable. Best reliability is obtained for standardized cardinal feedback, and cardinal feedback is also easiest to learn and generalize from. Finally, improvements of over 1 BLEU can be obtained by integrating a regression-based reward estimator trained on cardinal feedback for 800 translations into RL for NMT. This shows that RL is possible even from small amounts of fairly reliable human feedback, pointing to a great potential for applications at larger scale.

1 Introduction

Recent work has received high attention by successfully scaling reinforcement learning (RL) to games with large state-action spaces, achieving human-level (Mnih et al., 2015) or even superhuman performance (Silver et al., 2016). This success and the ability of RL to circumvent the data annotation bottleneck in supervised learning has led to renewed interest in RL in sequence-to-sequence learning problems with exponential output spaces. A typical approach is to combine REINFORCE (Williams, 1992) with policies based on deep sequence-to-sequence learning (Bahdanau et al., 2015), for example, in machine translation (Bahdanau et al., 2017), semantic parsing (Liang et al., 2017), or summarization (Paulus et al., 2017). These RL approaches focus on improving performance in automatic evaluation by simulating reward signals by evaluation metrics such as BLEU, F1-score, or ROUGE, computed against gold standards. Despite coming from different fields of application, RL in games and sequence-to-sequence learning share firstly the existence of a clearly specified reward function, e.g., defined by winning or losing a game, or by computing an automatic sequence-level evaluation metric. Secondly, both RL applications rely on a sufficient exploration of the action space, e.g., by evaluating multiple game moves for the same game state, or various sequence predictions for the same input.

[*]The work for this paper was done while the second author was an intern in Heidelberg.

The goal of this paper is to advance the state-of-the-art of sequence-to-sequence RL, exemplified by bandit learning for neural machine translation (NMT). Our aim is to show that successful learning from simulated bandit feedback (Sokolov et al., 2016b; Kreutzer et al., 2017; Nguyen et al., 2017; Lawrence et al., 2017) does in fact carry over to learning from actual human bandit feedback. The promise of bandit NMT is that human feedback on the quality of translations is easier to obtain in large amounts than human references, thus compensating the weaker nature of the signals by their quantity. However, the human factor entails several differences to the above sketched simulation scenarios of RL. Firstly, human rewards are not well-defined functions, but complex and inconsistent signals. For example, in general every input sentence has a multitude of correct translations, each of which humans may judge differently, depending on many contextual and personal factors. Secondly, exploration of the space of possible translations is restricted in real-world scenarios where a user judges one displayed translation, but cannot be expected to rate an alternative translation, let alone large amounts of alternatives.

In this paper we will show that despite the fact that human feedback is ambiguous and partial in nature, a catalyst for successful learning from human reinforcements is the reliability of the feedback signals. The first deployment of bandit NMT in an e-commerce translation scenario conjectured lacking reliability of user judgments as the reason for disappointing results when learning from 148k user-generated 5-star ratings for around 70k product title translations (Kreutzer et al., 2018). We thus raise the question of how human feedback can be gathered in the most reliable way, and what effect reliability will have in downstream tasks. In order to answer these questions, we measure intra- and inter-annotator agreement for two feedback tasks for bandit NMT, using cardinal feedback (on a 5-point scale) and ordinal feedback (by pairwise preferences) for 800 translations, conducted by 16 and 14 human raters, respectively. Perhaps surprisingly, while relative feedback is often considered easier for humans to provide (Thurstone, 1927), our investigation shows that α-reliability (Krippendorff, 2013) for intra- and inter-rater agreement is similar for both tasks, with highest inter-rater reliability for standardized 5-point ratings.

In a next step, we address the issue of machine learnability of human rewards. We use deep learning models to train reward estimators by regression against cardinal feedback, and by fitting a Bradley-Terry model (Bradley and Terry, 1952) to ordinal feedback. Learnability is understood by a slight misuse of the machine learning notion of learnability (Shalev-Shwartz et al., 2010) as the question how well reward estimates can approximate human rewards. Our experiments reveal that rank correlation of reward estimates with TER against human references is higher for regression models trained on standardized cardinal rewards than for Bradley-Terry models trained on pairwise preferences. This emphasizes the influence of the reliability of human feedback signals on the quality of reward estimates learned from them.

Lastly, we investigate machine learnability of the overall NMT task, in the sense of Green et al. (2014) who posed the question of how well an MT system can be tuned on post-edits. We use an RL approach for tuning, where a crucial difference of our work to previous work on RL from human rewards (Knox and Stone, 2009; Christiano et al., 2017) is that our RL scenario is not interactive, but rewards are collected in an offline log. RL then can proceed either by off-policy learning using logged single-shot human rewards directly, or by using estimated rewards. An expected advantage of estimating rewards is to tackle a simpler problem first — learning a reward estimator instead of a full RL task for improving NMT — and then to deploy unlimited feedback from the reward estimator for off-policy RL. Our results show that significant improvements can be achieved by training NMT from both estimated and logged human rewards, with best results for integrating a regression-based reward estimator into RL. This completes the argumentation that high reliability influences quality of reward estimates, which in turn affects the quality of the overall NMT task. Since the size of our training data is tiny in machine translation proportions, this result points towards a great potential for larger-scaler applications of RL from human feedback.

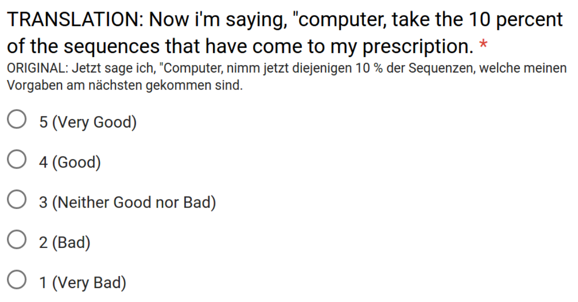
2 Related Work

Function approximation to learn a "critic" instead of using rewards directly has been embraced in the RL literature under the name of "actor-critic" methods (see Konda and Tsitsiklis (2000), Sutton et al. (2000), Kakade (2001), Schulman et al. (2015), Mnih et al. (2016), *inter alia*). In difference to our approach, actor-critic methods learn online while our approach estimates rewards in an offline fashion. Offline methods in RL, with and without function approximation, have been presented under the name of "off-policy" or "counterfactual" learning (see Precup et al. (2000), Precup et al. (2001), Bottou et al. (2013), Swaminathan and Joachims (2015a), Swaminathan and Joachims (2015b), Jiang and Li (2016), Thomas and Brunskill (2016), *inter alia*). Online actor-critic methods have been applied to sequence-to-sequence RL by Bahdanau et al. (2017) and Nguyen et al. (2017). An approach to off-policy RL under deterministic logging has been presented by Lawrence et al. (2017). However, all these approaches have been restricted to simulated rewards.

RL from human feedback is a growing area. Knox and Stone (2009) and Christiano et al. (2017) learn a reward function from human feedback and use that function to train an RL system. The actor-critic framework has been adapted to interactive RL from human feedback by Pilarski et al. (2011) and MacGlashan et al. (2017). These approaches either update the reward function from human feedback intermittently or perform learning only in rounds where human feedback is provided. A framework that interpolates a human critique objective into RL has been presented by Judah et al. (2019). None of these works systematically investigates the reliability of the feedback and its impact of the down-stream task.

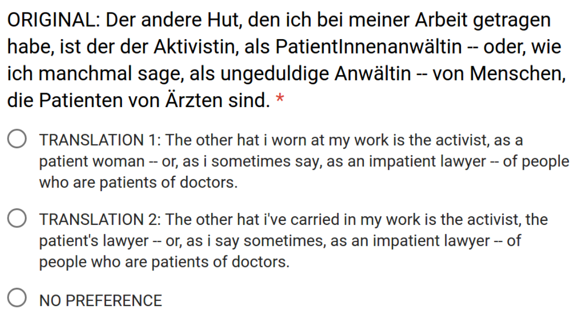
Kreutzer et al. (2018) have presented the first application of off-policy RL for learning from noisy human feedback obtained for deterministic logs of e-commerce product title translations. While learning from explicit feedback in the form of 5-star ratings fails, Kreutzer et al. (2018) propose to leverage implicit feedback embedded in a search task instead. In simulation experiments on the same domain, the methods proposed by Lawrence et al. (2017) succeeded also for neural models, allowing to pinpoint the lack of reliability in the human feedback signal as the reason for the underwhelming results when learning from human 5-star ratings. The goal of showing the effect of highly reliable human bandit feedback in downstream RL tasks was one of the main motivations for our work.

For the task of machine translation, estimating human feedback, i.e. quality ratings, is related to the task of sentence-level quality estimation (sQE). However, there are crucial differences between sQE and the reward estimation in our work: sQE usually has more training data, often from more than one machine translation model. Its gold labels are inferred from post-edits, i.e. corrections of the machine translation output, while we learn from weaker bandit feedback. Although this would in principle be possible, sQE predictions have not (yet) been used to directly reinforce predictions of MT systems, mostly because their primary purpose is to predict post-editing effort, i.e. give guidance how to further process a translation. State-of-the-art models for sQE such as (Martins et al., 2017) and (Kim et al., 2017) are unsuitable for the direct use in this task since they rely on linguistic input features, stacked architectures or post-edit or word-level supervision. Similar to approaches for generative adversarial NMT (Yu et al., 2017; Wu et al., 2017) we prefer a simpler convolutional architecture based on word embeddings for the human reward estimation.


TRANSLATION: Now i'm saying, "computer, take the 10 percent of the sequences that have come to my prescription. *
ORIGINAL: Jetzt sage ich, "Computer, nimm jetzt diejenigen 10 % der Sequenzen, welche meinen Vorgaben am nächsten gekommen sind.

○ 5 (Very Good)
○ 4 (Good)
○ 3 (Neither Good nor Bad)
○ 2 (Bad)
○ 1 (Very Bad)

Figure 1: Rating interface for 5-point ratings.


ORIGINAL: Der andere Hut, den ich bei meiner Arbeit getragen habe, ist der der Aktivistin, als PatientInnenanwältin -- oder, wie ich manchmal sage, als ungeduldige Anwältin -- von Menschen, die Patienten von Ärzten sind. *

○ TRANSLATION 1: The other hat i worn at my work is the activist, as a patient woman -- or, as i sometimes say, as an impatient lawyer -- of people who are patients of doctors.
○ TRANSLATION 2: The other hat i've carried in my work is the activist, the patient's lawyer -- or, as i say sometimes, as an impatient lawyer -- of people who are patients of doctors.
○ NO PREFERENCE

Figure 2: Rating interface for pairwise ratings.

3 Human MT Rating Task

3.1 Data

We translate a subset of the TED corpus with a general-domain and a domain-adapted NMT model (see §6.2 for NMT and data), post-process the translations (replacing special characters, restoring capitalization) and filter out identical out-of-domain and in-domain translations. In order to compose a homogeneous data set, we first select translations with references of length 20 to 40, then sort the translation pairs by difference in character n-gram F-score (chrF, $\beta = 3$) (Popović, 2015) and length, and pick the top 400 translation pairs with the highest difference in chrF but lowest difference in length. This yields translation pairs of similar length, but different quality.

3.2 Rating Task

The pairs were treated as 800 separate translations for a 5-point rating task. From the original 400 translation pairs, 100 pairs (or 200 individual translations) were randomly selected for

Type	Inter-rater α	Intra-rater Mean α	Intra-rater Stdev. α
5-point	0.2308	0.4014	0.1907
5-point norm.	0.2820		
5-point norm. part.	0.5059	0.5527	0.0470
5-point norm. trans.	0.3236	0.3845	0.1545
Pairwise	0.2385	0.5085	0.2096
Pairwise filt. part.	0.3912	0.7264	0.0533
Pairwise filt. trans.	0.3519	0.5718	0.2591

Table 1: Inter- and intra-reliability measured by Krippendorff's α for 5-point and pairwise ratings of 1,000 translations of which 200 translations are repeated twice. The filtered variants are restricted to either a subset of participants (part.) or a subset of translations (trans.).

repetition. This produced a total of 1,000 individual translations, with 600 occurring once, and 200 occurring twice. The translations were shuffled and separated into five sections of 200 translations, each with 120 translations from the unrepeated pool, and 80 translations from the repeated pool, ensuring that a single translation does not occur more than once in each section. For a pairwise task, the same 100 pairs were repeated from the original 400 translation pairs. This produced a total of 500 translation pairs. The translations were also shuffled and separated into five sections of 100 translation pairs, each with 60 translation pairs from the unrepeated pool, and 40 translation pairs from the repeated pool. None of the pairs were repeated within each section.

We recruited 14 participants for the pairwise rating task and 16 for the 5-point rating task. The participants were university students with fluent or native language skills in German and English. The rating interface is shown in Figures 1 and 2. Rating instructions are given in the supplementary material. Note that no reference translations were presented since the objective is to model a realistic scenario for bandit learning.[1]

4 Reliability of Human MT Ratings

4.1 Inter-rater and Intra-rater Reliability

In the following, we report inter- and intra-rater reliability of the cardinal and ordinal feedback tasks described in §3 with respect to Krippendorff's α (Krippendorff, 2013) evaluated at interval and ordinal scale, respectively.

As shown in Table 1, measures of inter-rater reliability show small differences between the 5-point and pairwise task. The inter-rater reliability in the 5-point task ($\alpha = 0.2308$) is roughly the same as that of the pairwise task ($\alpha = 0.2385$). Normalization of ratings per participant (by standardization to Z-scores), however, shows a marked improvement of overall inter-rater reliability for the 5-point task ($\alpha = 0.2820$). A one-way analysis of variance taken over inter-rater reliabilities between pairs of participants suggests statistically significant differences across tasks ($F\,(2, 328) = 6.399, p < 0.01$), however, a post hoc Tukey's (Larsen and Marx, 2012) honest significance test attributes statistically significant differences solely between the 5-point tasks with and without normalization. These scores indicate that the overall agreement between human ratings is roughly the same, regardless of whether participants are being asked to provide cardinal or ordinal ratings. Improvement in inter-rater reliability via participant-level normalization suggests that participants may indeed have individual biases toward certain regions of the 5-point scale, which the normalization process corrects.

In terms of intra-rater reliability, a better mean was observed among participants in the pairwise task ($\alpha = 0.5085$) versus the 5-point task ($\alpha = 0.4014$). This suggests that, on average, human raters provide more consistent ratings with themselves in comparing between two translations versus rating single translations in isolation. This may be attributed to the fact that seeing multiple translations provides raters with more cues with which to make consistent judgments. However, at the current sample size, a Welch two-sample t-test (Larsen and Marx, 2012) between 5-point and pairwise intra-rater reliabilities shows no significant difference between the two tasks ($t\,(26.92) = 1.4362, p = 0.1625$). Thus, it remains difficult to infer whether one task is definitively superior to the other in eliciting more consistent responses. Intra-rater reliability is the same for the 5-point task with and without normalization, as participants are still compared against themselves.

[1]The collection of ratings can be downloaded from http://www.cl.uni-heidelberg.de/statnlpgroup/humanmt/.

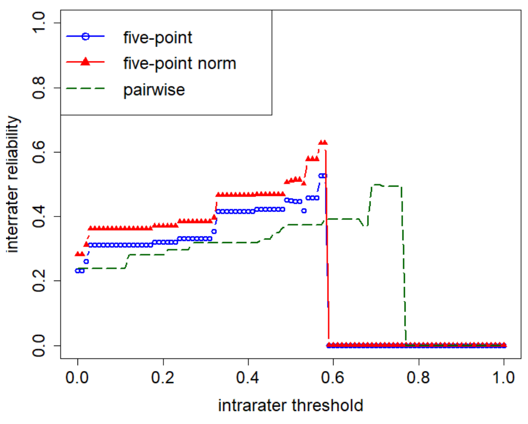


Figure 3: Improvements in inter-rater reliability using *intra-rater consistency* filter.

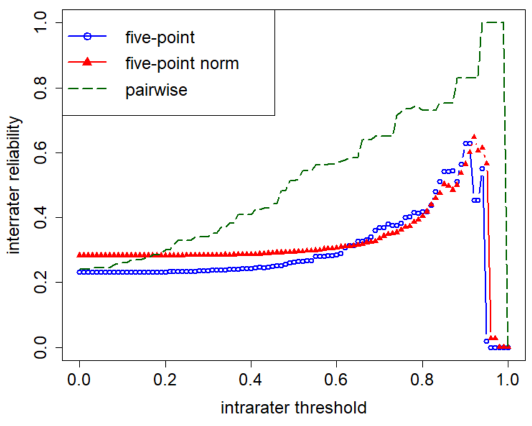


Figure 4: Improvements in inter-rater reliability using *item variance* filter.

4.2 Rater and Item Variance

The succeeding analysis is based on two assumptions: first, that human raters vary in that they do not provide equally good judgments of translation quality, and second, rating items vary in that some translations may be more difficult to judge than others. This allows to investigate the influence of rater variance and item variance on inter-rater reliability by an ablation analysis where low-quality judges and difficult translations are filtered out.

Using intra-rater reliability as an index of how well human raters judge translation quality, Figure 3 shows a filtering process whereby human raters with α scores lower than a moving threshold are dropped from the analysis. As the reliability threshold is increased from 0 to 1, overall inter-rater reliability is measured. Figure 4 shows a similar filtering process implemented using variance in translation scores. Item variances are normalized on a scale from 0 to 1 and subtracted from 1 to produce an item variance threshold. As the threshold increases, overall inter-rater reliability is likewise measured as high-variance items are progressively dropped from the analysis.

As the plots demonstrate, inter-rater reliability generally increases with consistency and variance filtering. For consistency filtering, Figure 3 shows how the inter-rater reliability of the 5-point task experiences greater increases than the pairwise task with lower filtering thresholds, especially in the normalized case. This may be attributed to the fact that more participants in the 5-point task had low intra-rater reliability. Pairwise tasks, on the other hand, require higher thresholds before large gains are observed in overall inter-rater reliability. This is because more participants in the pairwise task had relatively high intra-rater reliability. In the normalized 5-point task, selecting a threshold of 0.49 as a cutoff for intra-rater reliability retains 8 participants with an inter-rater reliability of 0.5059. For the pairwise task, a threshold of 0.66 leaves 5 participants with an inter-rater reliability of 0.3912.

The opposite phenomenon is observed in the case of variance filtering. As seen in Figure 4, the overall inter-rater reliability of the pairwise task quickly overtakes that of the 5-point task, with and without normalization. This may be attributed to how, in the pairwise setup, more items can be a source of disagreement among human judges. Ambiguous cases, that will be discussed in §4.3, may result in higher item variance. This problem is not as pronounced in the 5-point task, where judges must simply judge individual translations. It may be surmised that this item variance accounts for why, on average, judges in the pairwise task demonstrate higher intra-rater reliability than those in the 5-point task, yet the overall inter-rater reliability of the pairwise task is lower. By selecting a variance threshold such that at least 70% of items are retained in the analysis, the improved inter-rater reliabilities were 0.3236 for the 5-point task and 0.3519 for the pairwise task.

4.3 Qualitative Analysis

On completion of the rating task, we asked the participants for a *subjective* judgment of difficulty on a scale from 1 (very difficult) to 10 (very easy). On average, the pairwise rating task (mean 5.69) was perceived slightly easier than the 5-point rating task (mean 4.8). They also had to state which aspects of the tasks they found difficult: The biggest challenge for 5-point ratings seemed to be the weighing of different error types and the rating of long sentences with very few, but essential errors. For pairwise ratings, difficulties lie in distinguishing between similar, or similarly bad translations. Both tasks showed difficulties with ungrammatical or incomprehensible sources.

Comparing items with high and low agreement across raters allows conclusions about *objective* difficulty. We assume that high inter-rater agreement indicates an ease of judgment, while difficulties in judgment are manifested in low agreement. A list of examples is given in the supplementary material. For 5-point ratings, difficulties arise with ungrammatical sources and omissions, whereas obvious mistakes in the target, such as over-literal translations, make judgment easier. Preference judgments tend to be harder when both translations contain errors and are similar. When there is a tie, the pairwise rating framework does not allow to indicate whether both translations are of high or low quality. Since there is no normalization strategy for pairwise ratings, individual biases or rating schemes can hence have a larger negative impact on the inter-rater agreement.

5 Learnability of a Reward Estimator from MT Ratings

5.1 Learning a Reward Estimator

The numbers of ratings that can be obtained directly from human raters in a reasonable amount of time is tiny compared to the millions of sentences used for standard NMT training. By learning a reward estimator on the collection of human ratings, we seek to generalize to unseen translations. The model for this reward estimator should ideally work without time-consuming feature extraction so it can be deployed in direct interaction with a learning NMT system, estimating rewards on the fly, and most importantly generalize well so it can guide the NMT towards good local optima.

Learning from Cardinal Feedback. The inputs to the reward estimation model are sources $\mathbf{x}$ and their translations $\mathbf{y}$. Given cardinal judgments for these inputs, a regression model with parameters ψ is trained to minimize the mean squared error (MSE) for a set of n predicted rewards $\hat{r}$ and judgments r:

$$\mathcal{L}^{MSE}(\psi) = \frac{1}{n}\sum_{i=1}^{n}(r(\mathbf{y}_i) - \hat{r}_\psi(\mathbf{y}_i))^2.$$

In simulation experiments, where all translations can be compared to existing references, r may be computed by sentence-BLEU (sBLEU). For our human 5-point judgments, we first normalize the judgments per rater as described in §4, then average the judgments across raters and finally scale them linearly to the interval $[0.0, 1.0]$.

Learning from Pairwise Preference Feedback. When pairwise preferences are given instead of cardinal judgments, the Bradley-Terry model allows us to train an estimator of r. Following Christiano et al. (2017), let $\hat{P}_\psi[\mathbf{y^1} \succ \mathbf{y^2}]$ be the probability that any translation $\mathbf{y^1}$ is preferred over any other translation $\mathbf{y^2}$ by the reward estimator:

$$\hat{P}_\psi[\mathbf{y}^1 \succ \mathbf{y}^2] = \frac{\exp \hat{r}_\psi(\mathbf{y^1})}{\exp \hat{r}_\psi(\mathbf{y^1}) + \exp \hat{r}_\psi(\mathbf{y^2})}.$$

Let $Q[\mathbf{y^1} \succ \mathbf{y^2}]$ be the probability that translation $\mathbf{y_1}$ is preferred over translation $\mathbf{y_2}$ by a gold standard, e.g. the human raters or in comparison to a reference translation. With this supervision signal we formulate a pairwise (PW) training loss for the reward estimation model with parameters ψ:

$$\mathcal{L}^{PW}(\psi) = -\frac{1}{n}\sum_{i=1}^{n} Q[\mathbf{y_i^1} \succ \mathbf{y_i^2}] \log \hat{P}_\psi[\mathbf{y_i^1} \succ \mathbf{y_i^2}] + Q[\mathbf{y_i^2} \succ \mathbf{y_i^1}] \log \hat{P}_\psi[\mathbf{y_i^2} \succ \mathbf{y_i^1}].$$

For simulation experiments — where we lack a genuine supervision for preferences — we compute Q comparing the sBLEU scores for both translations, i.e. translation preferences are modeled according to their difference in sBLEU:

$$Q[\mathbf{y^1} \succ \mathbf{y^2}] = \frac{\exp \text{sBLEU}(\mathbf{y^1})}{\exp \text{sBLEU}(\mathbf{y^1}) + \exp \text{sBLEU}(\mathbf{y^2})}.$$

When obtaining preference jugdments directly from raters, $Q[\mathbf{y^1} \succ \mathbf{y^2}]$ is simply the relative frequency of $\mathbf{y^1}$ being preferred over $\mathbf{y^2}$ by a rater.

5.2 Experiments

Data. The 1,000 ratings collected as described in §3 are leveraged to train regression models and pairwise preference models. In addition, we train models on simulated rewards (sBLEU) for a comparison with arguably "clean" feedback for the

Model	Feedback	ρ
MSE	Simulated	-0.2571
PW	Simulated	-0.1307
MSE	Human	-0.2193
PW	Human	-0.1310
MSE	Human filt.	-0.2341
PW	Human filt.	-0.1255

Table 2: Spearman's rank correlation ρ between estimated rewards and TER for models trained with *simulated* rewards and *human* rewards (also filtered subsets).

same set of translations. In order to augment this very small collection of ratings, we leverage the available out-of-domain bitext as auxiliary training data. We sample translations for a subset of the out-of-domain sources and store sBLEU scores as rewards, collecting 90k out-of-domain training samples in total (see supplementary material for details). During training, each mini-batch is sampled from the auxiliary data with probability p_{aux}, from the original training data with probability $1 - p_{aux}$. Adding this auxiliary data as a regularization through multi-task learning prevents the model from overfitting to the small set of human ratings. In the experiments p_{aux} was tuned to 0.8.

Architecture. We choose the following neural architecture for the reward estimation (details see supplementary material): Inputs are padded source and target subword embeddings, which are each processed with a biLSTM (Hochreiter and Schmidhuber, 1997). Their outputs are concatenated for each time step, further fed to a 1D-convolution with max-over-time pooling and subsequently a leaky ReLU (Maas et al., 2013) output layer. This architecture can be seen as a biLSTM-enhanced bilingual extension to the convolutional model for sentence classification proposed by Kim (2014). It has the advantage of not requiring any feature extraction but still models n-gram features on an abstract level.

Evaluation Method. The quality of the reward estimation models is tested by measuring Spearman's ρ with TER on a held-out test set of 1,314 translations following the standard in sQE evaluations. Hyperparameters are tuned on another 1,200 TED translations.

Results. Table 2 reports the results of reward estimators trained on simulated and human rewards. When trained from cardinal rewards, the model of simulated scores performs slightly better than the model of human ratings. This advantage is lost when moving to preference judgments, which might be explained by the fact that the softmax over sBLEUs with respect to a single reference is just not as expressive as the preference probabilities obtained from several raters. Filtering by participants (retaining 8 participants for cardinal rewards and 5 for preference jugdments, see Section 4) improves the correlation further for cardinal rewards, but slightly hurts for preference judgments. The overall correlation scores are relatively low — especially for the PW models — which we suspect is due to overfitting to the small set of training data. From these experiments we conclude that when it comes to estimating translation quality, cardinal human jugdments are more useful than pairwise preference jugdments.

6 Reinforcement Learning from Direct and Estimated Rewards in MT

6.1 NMT Objectives

Supervised Learning. Most commonly, NMT models are trained with Maximum Likelihood Estimation (MLE) on a parallel corpus of source and target sequences $D = \{(\mathbf{x}^{(s)}, \mathbf{y}^{(s)})\}_{s=1}^{S}$:

$$\mathcal{L}^{MLE}(\boldsymbol{\theta}) = \sum_{s=1}^{S} \log p_\theta(\mathbf{y}^{(s)}|\mathbf{x}^{(s)}).$$

The MLE objective requires reference translations and is agnostic to rewards. In the experiments it is used to train the out-of-domain baseline model as a warm start for reinforcement learning from in-domain rewards.

Reinforcement Learning from Estimated or Simulated Direct Rewards. Deploying NMT in a reinforcement learning scenario, the goal is to maximize the expectation of a reward r over all source and target sequences (Wu et al., 2016), leading to the following REINFORCE (Williams, 1992) objective:

$$\begin{aligned}\mathcal{R}^{RL}(\boldsymbol{\theta}) =& \mathbb{E}_{p(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}\left[r(\mathbf{y})\right] & (1)\\ &\approx \sum_{s=1}^{S}\sum_{i=1}^{k} p_{\boldsymbol{\theta}}^{\tau}(\tilde{\mathbf{y}}_{\mathbf{i}}^{(\mathbf{s})}|\mathbf{x}^{(\mathbf{s})})\, r(\tilde{\mathbf{y}}_{\mathbf{i}}) & (2)\end{aligned}$$

The reward r can either come from a reward estimation model (*estimated reward*) or be computed with respect to a reference in a simulation setting (*simulated direct reward*). In order to counteract high variance in the gradient updates, the running average of rewards is subtracted from r for learning. In practice, Equation 1 is approximated with k samples from $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ (see Equation 2). When $k = 1$, this is equivalent to the expected loss minimization in Sokolov et al. (2016a,b); Kreutzer et al. (2017), where the system interactively learns from online bandit feedback. For $k > 1$ this is similar to the minimum-risk training for NMT proposed in Shen et al. (2016). Adding a temperature hyper-parameter $\tau \in (0.0, \infty]$ to the softmax over the model output $\mathbf{o}$ allows us to control the sharpness of the sampling distribution $p^{\tau}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(\mathbf{o}/\tau)$, i.e. the amount of exploration during training. With temperature $\tau < 1$, the model's entropy decreases and samples closer to the one-best output are drawn. We seek to keep the exploration low to prevent the NMT to produce samples that lie far outside the training domain of the reward estimator.

Off-Policy Learning from Direct Rewards. When rewards can not be obtained for samples from a learning system, but were collected for a static deterministic system (e.g. in a production environment), we are in an *off-policy learning* scenario. The challenge is to improve the MT system from a log $L = \{(\mathbf{x}^{(h)}, \mathbf{y}^{(h)}, r(\mathbf{y}^{(h)}))\}_{h=1}^{H}$ of rewarded translations. Following Lawrence et al. (2017) we define the following off-policy learning (OPL) objective to learn from logged rewards:

$$\mathcal{R}^{OPL}(\boldsymbol{\theta}) = \frac{1}{H}\sum_{h=1}^{H} r(\mathbf{y}^{(h)})\, \bar{p}_{\boldsymbol{\theta}}(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}),$$

with reweighting over the current mini-batch B: $\bar{p}_{\boldsymbol{\theta}}(\mathbf{y}^{(h)}|\mathbf{x}^{(h)}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{y}^{(h)}|\mathbf{x}^{(h)})}{\sum_{b=1}^{B} p_{\boldsymbol{\theta}}(\mathbf{y}^{(b)}|\mathbf{x}^{(b)})}$.[2] In contrast to the RL objective, only logged translations are reinforced, i.e. there is no exploration in learning.

6.2 Experiments

Data. We use the WMT 2017 data[3] for training a general domain (here: *out-of-domain*) model for translations from German to English. The training data contains 5.9M sentence pairs, the development data 2,999 sentences (WMT 2016 test set) and the test data 3,004 sentences. For *in-domain* data, we choose the translations of TED talks[4] as used in IWSLT evaluation campaigns. The training data contains 153k, the development data 6,969, and the test data 6,750 parallel sentences.

	WMT			TED		
Model	BLEU	METEOR	BEER	BLEU	METEOR	BEER
WMT	27.2	31.8	60.08	27.0	30.7	59.48
TED	26.3	31.3	59.49	34.3	34.6	64.94

Table 3: Results on test data for in- and out-of-domain *fully-supervised* models. Both are trained with MLE, the TED model is obtained by fine-tuning the WMT model in TED data.

Architecture. Our NMT model is a standard subword-based encoder-decoder architecture with attention (Bahdanau et al., 2015). An encoder Recurrent Neural Network (RNN) reads in the source sentence and a decoder RNN generates the target sentence conditioned on the encoded source. We implemented RL and OPL objectives in Neural Monkey (Helcl and Libovický, 2017).[5] The NMT has a bidirectional encoder and a single-layer decoder with 1,024 GRUs each, and subword embeddings of size 500 for a shared vocabulary of subwords obtained from 30k byte-pair merges (Sennrich et al., 2016). For model selection we use greedy decoding, for test set evaluation beam search with a beam of width 10. We sample $k = 5$ translations for RL models and set the softmax temperature $\tau = 0.5$. Further hyperparameters are given in the supplementary material.

Evaluation Method. Trained models are evaluated with respect to BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011) using MULTEVAL (Clark et al., 2011) and BEER (Stanojević and Sima'an, 2014) to cover a diverse set of automatic measures for translation quality.[6] We test for statistical significance with approximate randomization (Noreen, 1989).

[2] Lawrence et al. (2017) propose reweighting over the whole log, but this is infeasible for NMT. Here $B \ll H$.

[3] Pre-processed data available at `http://www.statmt.org/wmt17/translation-task.html`.

[4] Pre-processing and data splits as described in `https://github.com/rizar/actor-critic-public/tree/master/exp/ted`.

[5] The code is available in the Neural Monkey fork `https://github.com/juliakreutzer/bandit-neuralmonkey/tree/acl2018`.

[6] Since tendencies of improvement turn out to be consistent across metrics, we only discuss BLEU in the text.

Model	Rewards		BLEU	METEOR	BEER
Baseline	-	-	27.0	30.7	59.48
RL	D	S	$32.5^{\star}_{\pm 0.01}$	$33.7^{\star}_{\pm 0.01}$	$63.47^{\star}_{\pm 0.10}$
OPL	D	S	$27.5^{\star}$	$30.9^{\star}$	$59.62^{\star}$
RL+MSE	E	S	$28.2^{\star}_{\pm 0.09}$	$31.6^{\star}_{\pm 0.04}$	$60.23^{\star}_{\pm 0.14}$
RL+PW	E	S	$27.8^{\star}_{\pm 0.01}$	$31.2^{\star}_{\pm 0.01}$	$59.83^{\star}_{\pm 0.04}$
OPL	D	H	$27.5^{\star}$	$30.9^{\star}$	$59.72^{\star}$
RL+MSE	E	H	$28.1^{\star}_{\pm 0.01}$	$31.5^{\star}_{\pm 0.01}$	$60.21^{\star}_{\pm 0.12}$
RL+PW	E	H	$27.8^{\star}_{\pm 0.09}$	$31.3^{\star}_{\pm 0.09}$	$59.88^{\star}_{\pm 0.23}$
RL+MSE	E	F	$28.1^{\star}_{\pm 0.20}$	$31.6^{\star}_{\pm 0.10}$	$60.29^{\star}_{\pm 0.13}$

Table 4: Results on TED test data for training with *estimated* (E) and *direct* (D) rewards from *simulation* (S), *humans* (H) and *filtered* (F) human ratings. Significant ($p \leq 0.05$) differences to the baseline are marked with *. For RL experiments we show three runs with different random seeds, mean and standard deviation in subscript.

The out-of-domain model is trained with MLE on WMT. The task is now to improve the generalization of this model to the TED domain. Table 3 compares the out-of-domain baseline with domain-adapted models that were further trained on TED in a fully-supervised manner (*supervised fine-tuning* as introduced by Freitag and Al-Onaizan (2016); Luong and Manning (2015)). The supervised domain-adapted model serves as an upper bound for domain adaptation with human rewards: if we had references, we could improve up to 7 BLEU. What if references are not available, but we can obtain rewards for sample translations?

Results for RL from Simulated Rewards. First we simulate "clean" and deterministic rewards by comparing sample translations to references using GLEU (Wu et al., 2016) for RL, and smoothed sBLEU for estimated rewards and OPL. Table 4 lists the results for this simulation experiment in rows 2-5 (S). If unlimited clean feedback was given (RL with direct simulated rewards), improvements of over 5 BLEU can be achieved. When limiting the amount of feedback to a log of 800 translations, the improvements over the baseline are only marginal (OPL). When replacing the direct reward by the simulated reward estimators from §5, i.e. having unlimited amounts of approximately clean rewards, however, improvements of 1.2 BLEU for MSE estimators (RL+MSE) and 0.8 BLEU for pairwise estimators (RL+PW) are found. This suggests that the reward estimation model helps to tackle the challenge of generalization over a small set of ratings.

Results for RL from Human Rewards. Knowing what to expect in an ideal setting with non-noisy feedback, we now move to the experiments with human feedback. OPL is trained with the logged normalized, averaged and re-scaled human reward (see §5). RL is trained with the direct reward provided by the reward estimators trained on human rewards from §5. Table 4 shows the results for training with human rewards in rows 6-8: The improvements for OPL are very similar to OPL with simulated rewards, both suffering from overfitting. For RL we observe that the MSE-based reward estimator (RL+MSE) leads to significantly higher improvements as a the pairwise reward estimator (RL+PW) — the same trend as for simulated ratings. Finally, the improvement of 1.1 BLEU over the baseline showcases that we are able to improve NMT with only a small number of human rewards. Learning from estimated filtered 5-point ratings, does not significantly improve over these results, since the improvement of the reward estimator is only marginal (see § 5).

7 Conclusion

In this work, we sought to find answers to the questions of how cardinal and ordinal feedback differ in terms of reliability, learnability and effectiveness for RL training of NMT, with the goal of improving NMT with human bandit feedback. Our rating study, comparing 5-point and preference ratings, showed that their reliability is comparable, whilst cardinal ratings are easier to learn and to generalize from, and also more suitable for RL in our experiments.

Our work reports improvements of NMT leveraging actual human bandit feedback for RL, leaving the safe harbor of simulations. Our experiments show that improvements of over 1 BLEU are achievable by learning from a dataset that is tiny in machine translation proportions. Since this type of feedback, in contrast to post-edits and references, is fast and cheap to elicit from non-professionals, our results bear a great potential for future applications on larger scale.

Acknowledgments.

This work was supported in part by DFG Research Grant RI 2221/4-1, and by an internship program of the IWR at Heidelberg University.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Toulon, France.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipanakar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research* 14:3207–3260.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika* 39(3-4):324–345.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NIPS)*. Long Beach, CA, USA.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*. Portland, OR, USA.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT)*. Edinburgh, Scotland.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR* abs/1612.06897.

Spence Green, Sida I. Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and Christopher D. Manning. 2014. Human effort and machine learnability in computer aided translation. In *Proceedings the onference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar.

Jindřich Helcl and Jindřich Libovický. 2017. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics* (107):5–17.

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

Nan Jiang and Lihong Li. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. New York, NY, USA.

Kshitij Judah, Saikat Roy, Alan Fern, and Thomas G. Dietterich. 2019. Reinforcement learning via practice and critique advice. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. Atlanta, GA, USA.

Sham Kakade. 2001. A natural policy gradient. In *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Canada.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation (WMT)*. Copenhagen, Denmark.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar.

W. Bradley Knox and Peter Stone. 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the International Conference on Knowledge Capture (K-CAP)*. Redondo Beach, CA, USA.

Vijay R. Konda and John N. Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Canada.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. 2018. Can neural machine translation be improved with user feedback? In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Industry Track (NAACL-HLT)*. New Orleans, LA, USA.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada.

Klaus Krippendorff. 2013. *Content Analysis. An Introduction to Its Methodology*. Sage, third edition.

Richard Larsen and Morris Marx. 2012. *An Introduction to Mathematical Statistics and Its Applications*. Prentice Hall, fifth edition.

Carolin Lawrence, Artem Sokolov, and Stefan Riezler. 2017. Counterfactual learning from bandit feedback under deterministic logging: A case study in statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Copenhagen, Denmark.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*. Da Nang, Vietnam.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Atlanta, GA, USA.

James MacGlashan, Mark K. Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Sydney, Australia.

André Martins, Marcin Junczys-Dowmunt, Fabio Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017. Pushing the limits of translation quality estimation. *Transactions of the Association for Computational Linguistics (TACL)* 5:205–218.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. New York, NY, USA.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518:529–533.

Khanh Nguyen, Hal Daumé, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated feedback. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Copenhagen, Denmark.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction.* Wiley.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, PA, USA.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304.

Patrick M. Pilarski, Michael R. Dawson, Thomas Degris, Farbod Fahimi, Jason P. Carey, and Richard S. Sutton. 2011. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *Proceedings of the IEEE International Conference on Rehabilitation Robotics*. Zürich, Switzerland.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation (WMT)*. Lisbon, Portugal.

Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. 2001. Off-policy temporal-difference learning with function approximation. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*. Williams College, MA, USA.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. San Francisco, CA, USA.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust region policy optimization. In *Proceedings of the 31st International Conferene on Machine Learning (ICML)*. Lille, France.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2010. Learnability, stability and uniform convergence. *Journal of Machine Learning Research* 11:2635–2670.

Shiqi Shen, Yong Cheng, Zongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529:484–489.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016a. Learning structured predictors from bandit feedback for interactive NLP. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016b. Stochastic structured prediction under bandit feedback. In *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Spain.

Miloš Stanojević and Khalil Sima'an. 2014. Fitting sentence level translation evaluation with many dense features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processings Systems (NIPS)*. Vancouver, Canada.

Adith Swaminathan and Thorsten Joachims. 2015a. Counterfactual risk minimization: Learning from logged bandit feedback. In *International Conference on Machine Learning (ICML)*. Lille, France.

Adith Swaminathan and Thorsten Joachims. 2015b. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada.

Philip S. Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*. New York, NY, USA.

Louis Leon Thurstone. 1927. A law of comparative judgement. *Psychological Review* 34:278–286.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *CoRR* abs/1704.06933.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*. San Francisco, CA, USA.

Accelerating Neural Transformer via an Average Attention Network

Biao Zhang[1,2], **Deyi Xiong**[3] and **Jinsong Su**[1,2*]
Xiamen University, Xiamen, China 361005[1]
Beijing Advanced Innovation Center for Language Resources[2]
Soochow University, Suzhou, China 215006[3]
zb@stu.xmu.edu.cn, dyxiong@suda.edu.cn, jssu@xmu.edu.cn

Abstract

With parallelizable attention networks, the neural Transformer is very fast to train. However, due to the auto-regressive architecture and self-attention in the decoder, the decoding procedure becomes slow. To alleviate this issue, we propose an average attention network as an alternative to the self-attention network in the decoder of the neural Transformer. The average attention network consists of two layers, with an average layer that models dependencies on previous positions and a gating layer that is stacked over the average layer to enhance the expressiveness of the proposed attention network. We apply this network on the decoder part of the neural Transformer to replace the original target-side self-attention model. With masking tricks and dynamic programming, our model enables the neural Transformer to decode sentences over four times faster than its original version with almost no loss in training time and translation performance. We conduct a series of experiments on WMT17 translation tasks, where on 6 different language pairs, we obtain robust and consistent speed-ups in decoding.[1]

1 Introduction

The past few years have witnessed the rapid development of neural machine translation (NMT), which translates a source sentence into the target language with an encoder-attention-decoder framework (Sutskever et al., 2014; Bahdanau et al., 2015). Under this framework, various advanced neural architectures have been explored as the backbone network for translation, ranging from recurrent neural networks (RNN) (Sutskever et al., 2014; Luong et al., 2015), convolutional neural networks (CNN) (Gehring et al., 2017a,b) to full attention networks without recurrence and convolution (Vaswani et al., 2017). Particularly, the neural Transformer, relying solely on attention networks, has refreshed state-of-the-art performance on several language pairs (Vaswani et al., 2017).

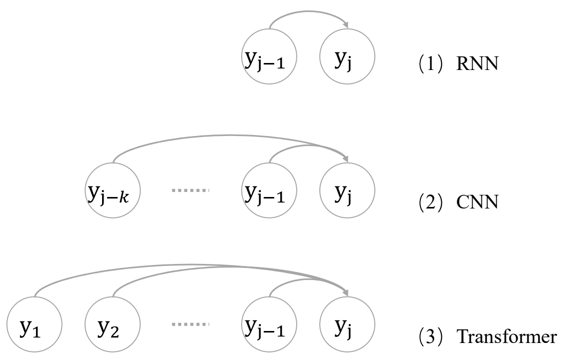


Figure 1: Illustration of the decoding procedure under different neural architectures. We show which previous target words are required to predict the current target word y_j in different NMT architectures. k indicates the filter size of the convolution layer.

Most interestingly, the neural Transformer is capable of being fully parallelized at the training phase and modeling intra-/inter-dependencies of source and target sentences within a short path. The parallelization property enables training NMT very quickly, while the dependency modeling property endows the Transformer with strong ability in inducing sentence semantics as well as translation correspondences. However, the decoding of the Transformer cannot enjoy the speed strength of parallelization due to the auto-regressive generation schema in the decoder. And the self-attention

*Corresponding author.
[1]Source code is available at https://github.com/bzhangXMU/transformer-aan.

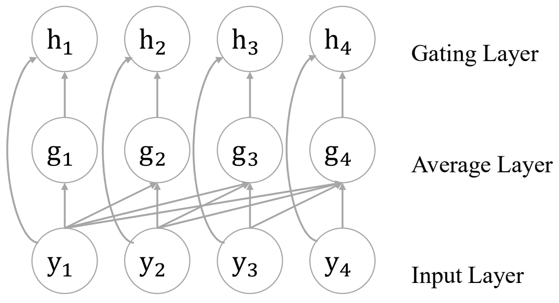


Figure 2: Visualization of the proposed model. For clarity, we show an example with only four words.

network in the decoder even further slows it.

We explain this using Figure 1, where we provide a comparison to RNN- and CNN-based NMT systems. To capture dependencies from previously predicted target words, the self-attention in the neural Transformer requires to calculate adaptive attention weights on all these words (Figure 1 (3)). By contrast, CNN only requires previous k target words (Figure 1 (2)), while RNN merely 1 (Figure 1 (1)). Due to the auto-regressive generation schema, decoding inevitably follows a sequential manner in the Transformer. Therefore the decoding procedure cannot be parallelized. Furthermore, the more target words are generated, the more time the self-attention in the decoder will take to model dependencies. Therefore, preserving the training efficiency of the Transformer on the one hand and accelerating its decoding on the other hand becomes a new and serious challenge.

In this paper, we propose an average attention network (AAN) to handle this challenge. We show the architecture of AAN in Figure 2, which consists of two layers: an *average layer* and *gating layer*. The average layer summarizes history information via a cumulative average operation over previous positions. This is equivalent to a simple attention network where original adaptively computed attention weights are replaced with averaged weights. Upon this layer, we stack a feed forward gating layer to improve the model's expressiveness in describing its inputs.

We use AAN to replace the self-attention part of the neural Transformer's decoder. Considering the characteristic of the cumulative average operation, we develop a masking method to enable parallel computation just like the original self-attention network in the training. In this way, the whole AAN model can be trained totally in parallel so that the training efficiency is ensured. As for the decoding, we can substantially accelerate it by feeding only the previous hidden state to the Transformer decoder just like RNN does. This is achieved with a dynamic programming method.

In spite of its simplicity, our model is capable of modeling complex dependencies. This is because AAN regards each previous word as an equal contributor to current word representation. Therefore, no matter how long the input is, our model can always build up connection signals with previous inputs, which we argue is very crucial for inducing long-range dependencies for machine translation.

We examine our model on WMT17 translation tasks. On 6 different language pairs, our model achieves a speed-up of over 4 times with almost no loss in both translation quality and training speed. In-depth analyses further demonstrate the convergency and advantages of translating long sentences of the proposed AAN.

2 Related Work

GRU (Chung et al., 2014) or LSTM (Hochreiter and Schmidhuber, 1997) RNNs are widely used for neural machine translation to deal with long-range dependencies as well as the gradient vanishing issue. A major weakness of RNNs lies at its sequential architecture that completely disables parallel computation. To cope with this problem, Gehring et al. (2017a) propose to use CNN-based encoder as an alternative to RNN, and Gehring et al. (2017b) further develop a completely CNN-based NMT system. However, shallow CNN can only capture local dependencies. Hence, CNN-based NMT normally develops deep archictures to model long-distance dependencies. Different from these studies, Vaswani et al. (2017) propose the Transformer, a neural architecture that abandons recurrence and convolution. It fully relies on attention networks to model translation. The properties of parallelization and short dependency path significantly improve the training speed as well as model performance for the Transformer. Unfortunately, as we have mentioned in Section 1, it suffers from decoding inefficiency.

The attention mechanism is originally proposed to induce translation-relevant source information for predicting next target word in NMT. It contributes a lot to make NMT outperform SMT. Recently, a variety of efforts are made to further improve its accuracy and capability. Luong et al.

(2015) explore several attention formulations and distinguish local attention from global attention. Zhang et al. (2016) treat RNN as an alternative to the attention to improve model's capability in dealing with long-range dependencies. Yang et al. (2017) introduce a recurrent cycle on the attention layer to enhance the model's memorization of previous translated source words. Zhang et al. (2017a) observe the weak discrimination ability of the attention-generated context vectors and propose a GRU-gated attention network. Kim et al. (2017) further model intrinsic structures inside attention through graphical models. Shen et al. (2017) introduce a direction structure into a self-attention network to integrate both long-range dependencies and temporal order information. Mi et al. (2016) and Liu et al. (2016) employ standard word alignment to supervise the automatically generated attention weights. Our work also focus on the evolution of attention network, but unlike previous work, we seek to simplify the self-attention network so as to accelerate the decoding procedure. The design of our model is partially inspired by the highway network (Srivastava et al., 2015) and the residual network (He et al., 2015).

In the respect of speeding up the decoding of the neural Transformer, Gu et al. (2018) change the auto-regressive architecture to speed up translation by directly generating target words without relying on any previous predictions. However, compared with our work, their model achieves the improvement in decoding speed at the cost of the drop in translation quality. Our model, instead, not only achieves a remarkable gain in terms of decoding speed, but also preserves the translation performance. Developing fast and efficient attention module for the Transformer, to the best of our knowledge, has never been investigated before.

3 The Average Attention Network

Given an input layer $\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m\}$, AAN first employs a cumulative-average operation to generate context-sensitive representation for each input embedding as follows (Figure 2 *Average Layer*):

$$\mathbf{g}_j = \text{FFN}\left(\frac{1}{j}\sum_{k=1}^{j}\mathbf{y}_k\right) \tag{1}$$

where $\text{FFN}(\cdot)$ denotes the position-wise feed-forward network proposed by Vaswani et al. (2017), and both $\mathbf{y}_k$ and $\mathbf{g}_j$ have a dimensionality of d. Intuitively, AAN replaces the original dynamically computed attention weights by the self-attention network in the decoder of the neural Transformer with simple and fixed average weights ($\frac{1}{j}$). In spite of its simplicity, the cumulative-average operation is very crucial for AAN because it builds up dependencies with previous input embeddings so that the generated representations are not independent of each other. Another benefit from the cumulative-average operation is that no matter how long the input is, the connection strength with each previous input embedding is invariant, which ensures the capability of AAN in modeling long-range dependencies.

We treat $\mathbf{g}_j$ as a contextual representation for the j-th input, and apply a feed-forward gating layer upon it as well as $\mathbf{y}_j$ to enrich the non-linear expressiveness of AAN:

$$\begin{aligned}\mathbf{i}_j, \mathbf{f}_j &= \sigma\left(\mathbf{W}\left[\mathbf{y}_j; \mathbf{g}_j\right]\right)\\ \tilde{\mathbf{h}}_j &= \mathbf{i}_j \odot \mathbf{y}_j + \mathbf{f}_j \odot \mathbf{g}_j\end{aligned} \tag{2}$$

where $[\cdot;\cdot]$ denotes concatenation operation, and $\odot$ indicates element-wise multiplication. $\mathbf{i}_j$ and $\mathbf{f}_j$ are the *input* and *forget* gate respectively. Via this gating layer, AAN can control how much past information can be preserved from previous context $\mathbf{g}_j$ and how much new information can be captured from current input $\mathbf{y}_j$. This helps our model to detect correlations inside input embeddings.

Following the architecture design in the neural Transformer (Vaswani et al., 2017), we employ a residual connection between the input layer and gating layer, followed by layer normalization to stabilize the scale of both output and gradient:

$$\mathbf{h}_j = \text{LayerNorm}\left(\mathbf{y}_j + \tilde{\mathbf{h}}_j\right) \tag{3}$$

We refer to the whole procedure formulated in Eq. (1~3) as original $\text{AAN}(\cdot)$ in following sections.

3.1 Parallelization in Training

A computation bottleneck of the original AAN described above is that the cumulative-average operation in Eq. (1) can only be performed sequentially. That is, this operation can not be parallelized. Fortunately, as the average is not a complex computation, we can use a masking trick to enable full parallelization of this operation.

We show the masking trick in Figure 3, where input embeddings are directly converted into their corresponding cumulative-averaged outputs

Model	Complexity	Sequential Operations	Maximum Path Length
Self-attention	$\mathcal{O}\left(n^2 \cdot d + n \cdot d^2\right)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Original AAN	$\mathcal{O}\left(n \cdot d^2\right)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Masked AAN	$\mathcal{O}\left(n^2 \cdot d + n \cdot d^2\right)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Table 1: Maximum path lengths, model complexity and minimum number of sequential operations for different models. n is the sentence length and d is the representation dimension.

$$\underbrace{\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{Bmatrix}}_{\text{Mask Matrix}} \times \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ \frac{y_1 + y_2}{2} \\ \frac{y_1 + y_2 + y_3}{3} \\ \frac{y_1 + y_2 + y_3 + y_4}{4} \end{pmatrix}$$

Figure 3: Visualization of parallel implementation for the cumulative-average operation enabled by a mask matrix. $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4\}$ are the input embeddings.

through a masking matrix. In this way, all the components inside AAN$(\cdot)$ can enjoy full parallelization, assuring its computational efficiency. We refer to this AAN as masked AAN.

3.2 Model Analysis

In this section, we provide a thorough analysis for AAN in comparison to the original self-attention model used by Vaswani et al. (2017). Unlike our AAN, the self-attention model leverages a scaled dot-product function rather than the average operation to compute attention weights:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = f(\mathbf{Y})$$
$$\text{Self-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V} \quad (4)$$

where $\mathbf{Y} \in \mathbb{R}^{n \times d}$ is the input matrix, $f(\cdot)$ is a mapping function and $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$ are the corresponding queries, keys and values. Following Vaswani et al. (2017), we compare both models in terms of computational complexity, minimum number of sequential operations required and maximum path length that a dependency signal between any two positions has to traverse in the network. Table 1 summarizes the comparison results.

Our AAN has a maximum path length of $\mathcal{O}(1)$, because it can directly capture dependencies between any two input embeddings. For the original AAN, the nature of its sequential computation enlarges its minimum number sequential operations to $\mathcal{O}(n)$. However, due to its lack of position-wise masked projection, it only consumes a computational complexity of $\mathcal{O}\left(n \cdot d^2\right)$. By contrast, both self-attention and masked AAN have a computational complexity of $\mathcal{O}\left(n^2 \cdot d + n \cdot d^2\right)$, and require only $\mathcal{O}(1)$ sequential operation. Theoretically, our masked AAN performs very similarly to the self-attention according to Table 1. We therefore use the masked version of AAN during training throughout all our experiments.

3.3 Decoding Acceleration

Differing noticeably from the self-attention in the Transformer, our AAN can be accelerated in the decoding phase via dynamic programming thanks to the simple average calculation. Particularly, we can decompose Eq. (1) into the following two steps:

$$\tilde{\mathbf{g}}_j = \tilde{\mathbf{g}}_{j-1} + \mathbf{y}_j \quad (5)$$
$$\mathbf{g}_j = \text{FFN}\left(\frac{\tilde{\mathbf{g}}_j}{j}\right) \quad (6)$$

where $\tilde{\mathbf{g}}_0 = \mathbf{0}$. In doing so, our model can compute the j-th input representation based on only one previous state $\tilde{\mathbf{g}}_{j-1}$, instead of relying on all previous states as the self-attention does. In this way, our model can be substantially accelerated during the decoding phase.

4 Neural Transformer with AAN

The neural Transformer models translation through an encoder-decoder framework, with each layer involving an attention network followed by a feed forward network (Vaswani et al., 2017). We apply our masked AAN to replace the self-attention network in its decoder part, and illustrate the overall architecture in Figure 4.

Given a source sentence $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the Transformer leverages its encoder to induce source-side semantics and dependencies so as to

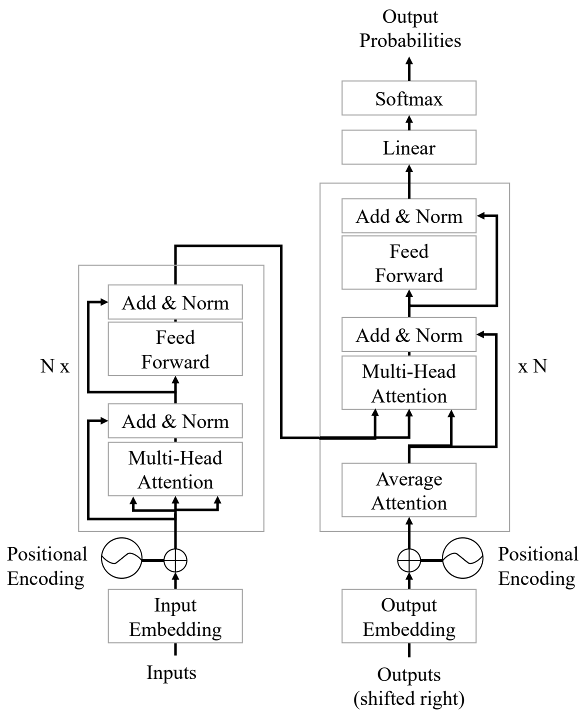


Figure 4: The new Transformer architecture with the proposed average attention network.

enable its decoder to recover the encoded information in a target language. The encoder is composed of a stack of $N = 6$ identical layers, each of which has two sub-layers:

$$\begin{aligned}\tilde{\mathbf{h}}^l &= \text{LayerNorm}\left(\mathbf{h}^{l-1} + \text{MHAtt}\left(\mathbf{h}^{l-1}, \mathbf{h}^{l-1}\right)\right)\\ \mathbf{h}^l &= \text{LayerNorm}\left(\tilde{\mathbf{h}}^l + \text{FFN}\left(\tilde{\mathbf{h}}^l\right)\right)\end{aligned} \tag{7}$$

where the superscript l indicates layer depth, and MHAtt denotes the multi-head attention mechanism proposed by Vaswani et al. (2017).

Based on the encoded source representation $\mathbf{h}^N$, the Transformer relies on its decoder to generate corresponding target translation $\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$. Similar to the encoder, the decoder also consists of a stack of $N = 6$ identical layers. For each layer in our architecture, the first sub-layer is our proposed average attention network, aiming at capturing target-side dependencies with previous predicted words:

$$\tilde{\mathbf{s}}^l = \text{AAN}\left(\mathbf{s}^{l-1}\right) \tag{8}$$

Carrying these dependencies, the decoder stacks another two sub-layers to seek translation-relevant source semantics for bridging the gap between the source and target language:

$$\begin{aligned}\mathbf{s}_c^l &= \text{LayerNorm}\left(\tilde{\mathbf{s}}^l + \text{MHAtt}\left(\tilde{\mathbf{s}}^l, \mathbf{h}^N\right)\right)\\ \mathbf{s}^l &= \text{LayerNorm}\left(\mathbf{s}_c^l + \text{FFN}\left(\mathbf{s}_c^l\right)\right)\end{aligned} \tag{9}$$

We use subscript c to denote the source-informed target representation. Upon the top layer of this decoder, translation is performed where a linear transformation and softmax activation are applied to compute the probability of the next token based on $\mathbf{s}^N$

To memorize position information, the Transformer augments its input layer $\mathbf{h}^0 = \mathbf{x}, \mathbf{s}^0 = \mathbf{y}$ with frequency-based positional encodings. The whole model is a large, single neural network, and can be trained on a large-scale bilingual corpus with a maximum likelihood objective. We refer readers to (Vaswani et al., 2017) for more details.

5 Experiments

5.1 WMT14 English-German Translation

We examine various aspects of our AAN on this translation task. The training data consist of 4.5M sentence pairs, involving about 116M English words and 110M German words. We used newstest2013 as the development set for model selection, and newstest2014 as the test set. We evaluated translation quality via case-sensitive BLEU metric (Papineni et al., 2002).

5.1.1 Model Settings

We applied byte pair encoding algorithm (Sennrich et al., 2016) to encode all sentences and limited the vocabulary size to 32K. All out-of-vocabulary words were mapped to an unique token "*unk*". We set the dimensionality d of all input and output layers to 512, and that of inner-FFN layer to 2048. We employed 8 parallel attention heads in both encoder and decoder layers. We batched sentence pairs together so that they were approximately of the same length, and each batch had roughly 25000 source and target tokens. During training, we used label smoothing with value $\epsilon_{ls} = 0.1$, attention dropout and residual dropout with a rate of $p = 0.1$. During decoding, we employed beam search algorithm and set the beam size to 4. Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ was used to tune model parameters, and the learning rate was varied under a warm-up strategy with $warmup\_steps = 4000$ (Vaswani et al., 2017).

Model	BLEU
Transformer	26.37
Our Model	26.31
Our Model w/o FFN	26.05
Our Model w/o Gate	25.91

Table 2: Case-sensitive tokenized BLEU score on WMT14 English-German translation. BLEU scores are calculated using *multi-bleu.perl*.

The maximum number of training steps was set to 100K. Weights of target-side embedding and output weight matrix were tied for all models. We implemented our model with masking tricks based on the open-sourced *thumt* (Zhang et al., 2017b)[2], and trained and evaluated all models on a single NVIDIA GeForce GTX 1080 GPU. For evaluation, we averaged last five models saved with an interval of 1500 training steps.

5.1.2 Translation Performance

Table 2 reports the translation results. On the same dataset, the Transformer yields a BLEU score of 26.37, while our model achieves 26.31. Both results are almost the same with no significant difference. Clearly, our model is capable of capturing complex translation correspondences so as to generate high-quality translations as effective as the Transformer.

We also show an ablation study in terms of the FFN$(\cdot)$ network in Eq. (1) and the gating layer in Eq. (2). Table 2 shows that without the FFN network, the performance of our model drops 0.26 BLEU points. This degeneration is enlarged to 0.40 BLEU points when the gating layer is not available. In order to reach comparable performance with the original Transformer, integrating both components is desired.

5.1.3 Analysis on Convergency

Different neural architectures might require different number of training steps to converge. In this section, we testify whether our AAN would reveal different characteristics with respect to convergency. We show the loss curve of both the Transformer and our model in Figure 5.

Surprisingly, both model show highly similar tendency, and successfully converge in the end. To train a high-quality translation system, our model consumes almost the same number of training steps as the Transformer. This strongly suggests

[2]https://github.com/thumt/THUMT

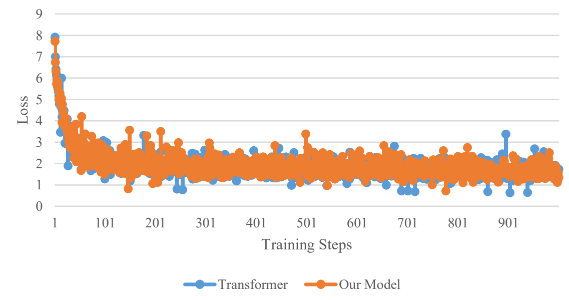


Figure 5: Convergence visualization. The horizontal axis denotes training steps scaled by 10^2, and the vertical axis indicates training loss. Roughly, our model converges similarly to the Transformer.

	Transformer	Our Model	$\triangle_r$
Training	0.2474	0.2464	1.00
Decoding			
beam=4	0.1804	0.0488	3.70
beam=8	0.3576	0.0881	4.06
beam=12	0.5503	0.1291	4.26
beam=16	0.7323	0.1700	4.31
beam=20	0.9172	0.2122	4.32

Table 3: Time required for training and decoding. *Training* denotes the number of global training steps processed per second; *Decoding* indicates the amount of time in seconds required for translating one sentence, which is averaged over the whole newstest2014 dataset. $\triangle_r$ shows the ratio between the Transformer and our model.

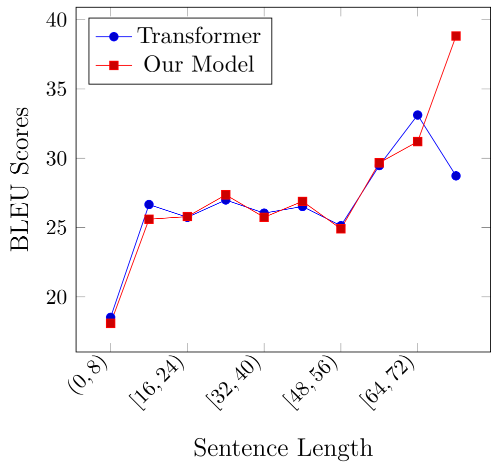
that replacing the self-attention network with our AAN does not have negative impact on the convergency of the entire model.

5.1.4 Analysis on Speed

In Section 3, we demonstrate in theory that our AAN is as efficient as the self-attention during training, but can be substantially accelerated during decoding. In this section, we provide quantitative evidences to examine this point.

We show the training and decoding speed of both the Transformer and our model in Table 3. During training, our model performs approximately 0.2464 training steps per second, while the Transformer processes around 0.2474. This indicates that our model shares similar computational strengths with the Transformer during training, which resonates with the computational analysis in Section 3.

When it comes to decoding procedure, the time of our model required to translate one sentence



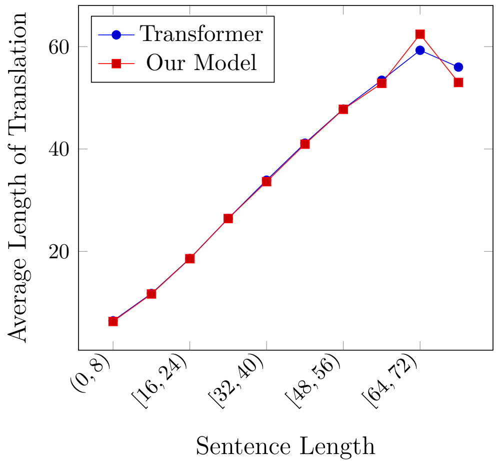


Figure 6: Translation statistics on WMT14 English-German test set (newstest14) with respect to the length of source sentences. The top figure shows tokenized BLEU score, and the bottom one shows the average length of translations, both visa-vis sentence length

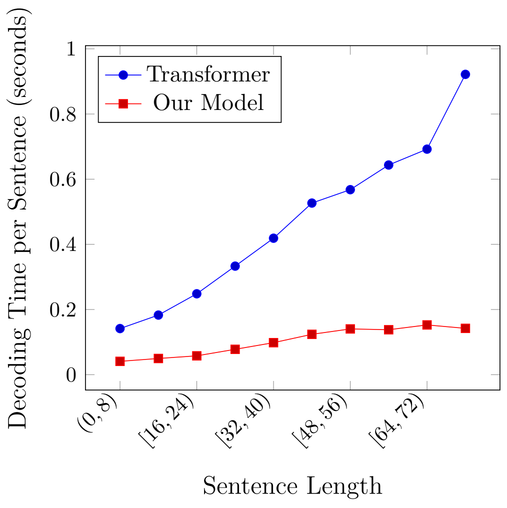


Figure 7: Average time required for translating one source sentence vs. the length of the source sentence. With the increase of sentence length, our model shows more clear and significant advantage over the Transformer in terms of the decoding speed.

is only a quarter of that of the Transformer, with beam size ranging from 4 to 20. Another noticeable feature is that as the beam size increases, the ratio of required decoding time between the Transformer and our model is consistently enlarged. This demonstrates empirically that our model, enhanced with the dynamic decoding acceleration algorithm (Section 3.3), can significantly improve the decoding speed of the Transformer.

5.1.5 Effects on Sentence Length

A serious common challenge for NMT is to translate long source sentences as handling long-distance dependencies and under-translation issues becomes more difficult for longer sentences. Our proposed AAN uses simple cumulative-average operations to deal with long-range dependencies. We want to examine the effectiveness of these operations on long sentence translation. For this, we provide the translation results along sentence length in Figure 6.

We find that both the Transformer and our model generate very similar translations in terms of BLEU score and translation length, and obtain rather promising performance on long source sentences. More specifically, our model yields relatively shorter translation length on the longest source sentences but significantly better translation quality. This suggests that in spite of the simplicity of the cumulative-average operations, our AAN can indeed capture long-range dependences desired for translating long source sentences.

Generally, the decoder takes more time for translating longer sentences. When it comes to the Transformer, this time issue of translating long sentences becomes notably severe as all previous predicted words must be included for estimating both self-attention weights and word prediction. We show the average time required for translating a source sentence with respect to its sentence length in Figure 7. Obviously, the decoding time of the Transformer grows dramatically with the increase of sentence length, while that of our model rises rather slowly. We contribute this great decoding advantage of our model over the Transformer to the average attention architecture which enables

	Case-sensitive BLEU				Case-insensitive BLEU			
	winner	Transformer	Our Model	$\triangle_d$	winner	Transformer	Our Model	$\triangle_d$
En→De	28.3	27.33	27.22	-0.11	28.9	27.92	27.80	-0.12
De→En	35.1	32.63	32.73	+0.10	36.5	34.06	34.13	+0.07
En→Fi	20.7	21.00	20.87	-0.13	21.1	21.54	21.47	-0.07
Fi→En	20.5	25.19	24.78	-0.41	21.4	26.22	25.74	-0.48
En→Lv	21.1	16.83	16.63	-0.20	21.6	17.42	17.23	-0.19
Lv→En	21.9	17.57	17.51	-0.06	22.9	18.48	18.30	-0.18
En→Ru	29.8	27.82	27.73	-0.09	29.8	27.83	27.74	-0.09
Ru→En	34.7	31.51	31.36	-0.15	35.6	32.59	32.36	-0.23
En→Tr	18.1	12.11	11.59	-0.52	18.4	12.56	12.03	-0.53
Tr→En	20.1	16.19	15.84	-0.35	20.9	16.93	16.57	-0.36
En→Cs	23.5	21.53	21.12	-0.41	24.1	22.07	21.66	-0.41
Cs→En	30.9	27.49	27.45	-0.04	31.9	28.41	28.33	-0.08

Table 4: Detokenized BLEU scores for WMT17 translation tasks. Results are reported with *multi-bleu-detok.perl*. "*winner*" denotes the translation results generated by the WMT17 winning systems. $\triangle_d$ indicates the difference between our model and the Transformer.

our model to perform next-word prediction by calculating information just from the previous hidden state, rather than considering all previous inputs like the self-attention in the Transformer's decoder.

5.2 WMT17 Translation Tasks

We further demonstrate the effectiveness of our model on six WMT17 translation tasks in both directions (12 translation directions in total). These tasks contain the following language pairs:

- **En-De**: The English-German language pair. This training corpus consists of 5.85M sentence pairs, with 141M English words and 135M German words. We used the concatenation of newstest2014, newstest2015 and newstest2016 as the development set, and the newstest2017 as the test set.

- **En-Fi**: The English-Finnish language pair. This training corpus consists of 2.63M sentence pairs, with 63M English words and 45M Finnish words. We used the concatenation of newstest2015, newsdev2015, newstest2016 and newstestB2016 as the development set, and the newstest2017 as the test set.

- **En-Lv**: The English-Latvian language pair. This training corpus consists of 4.46M sentence pairs, with 63M English words and 52M Latvian words. We used the newsdev2017 as the development set, and the newstest2017 as the test set.

- **En-Ru**: The English-Russian language pair. This training corpus consists of 25M sentence pairs, with 601M English words and 567M Russian words. We used the concatenation of newstest2014, newstest2015 and newstest2016 as the development set, and the newstest2017 as the test set.

- **En-Tr**: The English-Turkish language pair. This training corpus consists of 0.21M sentence pairs, with 5.2M English words and 4.6M Turkish words. We used the concatenation of newsdev2016 and newstest2016 as the development set, and newstest2017 as the test set.

- **En-Cs**: The English-Czech language pair. This training corpus consists of 52M sentence pairs, with 674M English words and 571M Czech words. We used the concatenation of newstest2014, newstest2015 and newstest2016 as the development set, and the newstest2017 as the test set.

Interestingly, these translation tasks involves training corpora with different scales (ranging from 0.21M to 52M sentence pairs). This help us thoroughly examine the ability of our model on different sizes of training data. All these preprocessed datasets are publicly available, and can be downloaded from WMT17 official website.[3]

We used the same modeling settings as in the WMT14 English-German translation task except for the number of training steps for En-Fi and En-Tr, which we set to 60K and 10K respectively. In addition, to compare with official results, we reported both case-sensitive and case-insensitive detokenized BLEU scores.

[3]http://data.statmt.org/wmt17/translation-task/preprocessed/

	Transformer	Our Model	$\triangle_r$
En→De	0.1411	0.02871	4.91
De→En	0.1255	0.02422	5.18
En→Fi	0.1289	0.02423	5.32
Fi→En	0.1285	0.02336	5.50
En→Lv	0.1850	0.03167	5.84
Lv→En	0.1980	0.03123	6.34
En→Ru	0.1821	0.03140	5.80
Ru→En	0.1595	0.02778	5.74
En→Tr	0.2078	0.02968	7.00
Tr→En	0.1886	0.03027	6.23
En→Cs	0.1150	0.02425	4.74
Cs→En	0.1178	0.02659	4.43

Table 5: Average seconds required for decoding one source sentence on WMT17 translation tasks.

5.2.1 Translation Results

Table 4 shows the overall results on 12 translation directions. We also provide the results from WMT17 winning systems[4]. Notice that unlike the Transformer and our model, these winner systems typically use model ensemble, system combination and large-scale monolingual corpus.

Although different languages have different linguistic and syntactic structures, our model consistently yields rather competitive results against the Transformer on all language pairs in both directions. Particularly, on the De→En translation task, our model achieves a slight improvement of 0.10/0.07 case-sensitive/case-insensitive BLEU points over the Transformer. The largest performance gap between our model and the Transformer occurs on the En→Tr translation task, where our model is lower than the Transformer by 0.52/0.53 case-sensitive/case-insensitive BLEU points. We conjecture that this difference may be due to the small training corpus of the En-Tr task. In all, these results suggest that our AAN is able to perform comparably to Transformer on different language pairs with different scales of training data.

We also show the decoding speed of both the Transformer and our model in Table 5. On all languages in both directions, our model yields significant and consistent improvements over the Transformer in terms of decoding speed. Our model decodes more than 4 times faster than the Transformer. Surprisingly, our model just consumes 0.02968 seconds to translate one source sentence on the En→Tr language pair, only a seventh of the decoding time of the Transformer. These results show that the benefit of decoding acceleration from the proposed average attention structure is language-invariant, and can be easily adapted to other translation tasks.

[4]http://matrix.statmt.org/matrix

6 Conclusion and Future Work

In this paper, we have described the average attention network that considerably alleviates the decoding bottleneck of the neural Transformer. Our model employs a cumulative average operation to capture important contextual clues from previous target words, and a feed forward gating layer to enrich the expressiveness of learned hidden representations. The model is further enhanced with a masking trick and a dynamic programming method to accelerate the Transformer's decoder. Extensive experiments on one WMT14 and six WMT17 language pairs demonstrate that the proposed average attention network is able to speed up the Transformer's decoder by over 4 times.

In the future, we plan to apply our model on other sequence to sequence learning tasks. We will also attempt to improve our model to enhance its modeling ability so as to consistently outperform the original neural Transformer.

7 Acknowledgments

The authors were supported by Beijing Advanced Innovation Center for Language Resources, National Natural Science Foundation of China (Nos. 61672440 and 61622209), the Fundamental Research Funds for the Central Universities (Grant No. ZK1024), and Scientific Research Project of National Language Committee of China (Grant No. YB135-49). Biao Zhang greatly acknowledges the support of the Baidu Scholarship. We also thank the reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2017a. A convolutional encoder model for neural machine translation. In *Proc. of ACL*, pages 123–135.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning. *Proc. of ICML*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. *Proc. of ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

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

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. *Proc. of ICLR*, abs/1702.00887.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proc. of ICLR*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proc. of COLING 2016*, pages 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, pages 1412–1421.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proc. of EMNLP*, pages 2283–2288, Austin, Texas. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*, pages 1715–1725.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2017. Neural machine translation with recurrent attention modeling. In *Proc. of EACL*, pages 383–387, Valencia, Spain. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Recurrent neural machine translation. *CoRR*, abs/1607.08725.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2017a. A gru-gated attention model for neural machine translation. *CoRR*, abs/1704.08430.

Jiacheng Zhang, Yanzhuo Ding, Shiqi Shen, Yong Cheng, Maosong Sun, Huan-Bo Luan, and Yang Liu. 2017b. THUMT: an open source toolkit for neural machine translation. *CoRR*, abs/1706.06415.

How Much Attention Do You Need?
A Granular Analysis of Neural Machine Translation Architectures

Tobias Domhan
Amazon
Berlin, Germany
domhant@amazon.com

Abstract

With recent advances in network architectures for Neural Machine Translation (NMT) recurrent models have effectively been replaced by either convolutional or self-attentional approaches, such as in the Transformer. While the main innovation of the Transformer architecture is its use of self-attentional layers, there are several other aspects, such as attention with multiple heads and the use of many attention layers, that distinguish the model from previous baselines. In this work we take a fine-grained look at the different architectures for NMT. We introduce an Architecture Definition Language (ADL) allowing for a flexible combination of common building blocks. Making use of this language, we show in experiments that one can bring recurrent and convolutional models very close to the Transformer performance by borrowing concepts from the Transformer architecture, but not using self-attention. Additionally, we find that self-attention is much more important for the encoder side than for the decoder side, where it can be replaced by a RNN or CNN without a loss in performance in most settings. Surprisingly, even a model without any target side self-attention performs well.

1 Introduction

Since the introduction of attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015) Neural Machine Translation (NMT) (Sutskever et al., 2014) has shown some impressive results. Initially, approaches to NMT mainly relied on Recurrent Neural Networks (RNNs) (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Luong et al., 2015; Wu et al., 2016) such as Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) or the Gated Rectified Unit (GRU) (Cho et al., 2014).

Recently, other approaches relying on convolutional networks (Kalchbrenner et al., 2016; Gehring et al., 2017) and self-attention (Vaswani et al., 2017) have been introduced. These approaches remove the dependency between source language time steps, leading to considerable speed-ups in training time and improvements in quality. The Transformer, however, contains other differences besides self-attention, including layer normalization across the entire model, multiple source attention mechanisms, a multi-head dot attention mechanism, and the use of residual feed-forward layers. This raises the question of how much each of these components matters.

To answer this question we first introduce a flexible Architecture Definition Language (ADL) (§2). In this language we standardize existing components in a consistent way making it easier to compare structural differences of architectures. Additionally, it allows us to efficiently perform a granular analysis of architectures, where we can evaluate the impact of individual components, rather than comparing entire architectures as a whole. This ability leads us to the following observations:

- Source attention on lower encoder layers brings no additional benefit (§4.2).
- Multiple source attention layers and residual feed-forward layers are key (§4.3).
- Self-attention is more important for the source than for the target side (§4.4).

2 Flexible Neural Machine Translation Architecture Combination

In order to experiment easily with different architecture variations we define a domain specific NMT Architecture Definition Language (ADL), consisting of combinable and nestable building blocks.

2.1 Neural Machine Translation

NMT is formulated as a sequence to sequence prediction task in which a source sentence $X = x_1, ..., x_n$ is translated auto-regressively into a target sentence $Y = y_1, ..., y_m$ one token at a time as

$$p(y_t|Y_{1:t-1}, X; \boldsymbol{\theta}) = \text{softmax}(\mathbf{W}_o \mathbf{z}^L + \mathbf{b}_o), \tag{1}$$

where $\mathbf{b}_o$ is a bias vector, $\mathbf{W}_o$ projects a model dependent hidden vector $\mathbf{z}^L$ of the Lth decoder layer to the dimension of the target vocabulary $\mathbf{V}_{trg}$ and $\boldsymbol{\theta}$ denotes the model parameters. Typically, during training $Y_{1:t-1}$ consists of the reference sequence tokens, rather then the predictions produced by the model, which is known as teacher-forcing. Training is done by minimizing the cross-entropy loss between the predicted and the reference sequence.

2.2 Architecture Definition Language

In the following we specify the ADL which can be used to define any standard NMT architecture and combinations thereof.

Layers The basic building block of the ADL is a layer l. Layers can be nested, meaning that a layer can consist of several sublayers. Layers optionally take set of named arguments $l(k_1{=}v_1, k_2{=}v_2, ...)$ with names k_1, k_2, ... and values v_1, v_2, ... or positional arguments $l(v_1, v_2, ...)$.

Layer definitions For each layer we have a corresponding layer definition based on the hidden states of the previous layer and any additional arguments. Specifically, each layer takes T hidden states $\mathbf{h}_1^i, ..., \mathbf{h}_T^i$, which in matrix form are $\mathbf{H}^i \in \mathbb{R}^{T \times d_i}$, and produces a new set of hidden states $\mathbf{h}_1^{i+1}, ..., \mathbf{h}_T^{i+1}$ or $\mathbf{H}^{i+1}$. While each layer can have a different number of hidden units d_i, in the following we assume them to stay constant across layers and refer to the model dimensionality as d_{model}. We distinguish the hidden states on the source side $\mathbf{U}^0, ..., \mathbf{U}^{L_s}$ from the hidden states of the target side $\mathbf{Z}^0, ..., \mathbf{Z}^L$. These are produced by the source and target embeddings and L_s source layers and L target layers.

Source attention layers play a special role in that their definition additionally makes use of any of the source hidden states $\mathbf{U}^0, ..., \mathbf{U}^{L_s}$.

Layer chaining Layers can be chained, feeding the output of one layer as the input to the next. We denote this as $l_1 \rightarrow l_2 \rightarrow ... \rightarrow l_L$. This is equivalent to writing $l_L(...\ l_2(l_1(\mathbf{H}^0)))$ if none of the layers is a source attention layer.

In layer chains layers may also contain layers that themselves take arguments. As an example $l_1(k{=}v) \rightarrow l_2 \rightarrow ... \rightarrow l_L$ is equivalent to $l_L(...\ l_2(l_1(\mathbf{H}^0, k{=}v)))$. Note that unlike in the layer definition hidden states are not explicitly stated in the layer chain, but rather implicitly defined through the preceding layers.

Encoder/Decoder structure A NMT model is fully defined through two layer chains, namely one describing the encoder and another describing the decoder. The first layer hidden states on the source $\mathbf{U}^0$ are defined through the source embedding as

$$\mathbf{u}_t^0 = \mathbf{E}_{src} \mathbf{x}_t \tag{2}$$

where $\mathbf{x}_t \in \{0, 1\}^{|\mathbf{V}_{src}|}$ is the one-hot representation of x_t and $\mathbf{E}_S \mathbf{x}_t \in \mathbb{R}^{e \times |\mathbf{V}_{src}|}$ an embedding matrix with embedding dimensionality e. Similarly, $\mathbf{Z}^0$ is defined through the target embedding matrix $\mathbf{E}_{tgt}$.

Given the final decoder hidden state $\mathbf{Z}^L$ the next word predictions are done according to Equation 1.

Layer repetition Networks often consist of substructures that are repeated several times. In order to support this we define a repetition layer as

$$\textit{repeat}(n, l) = l_1 \rightarrow l_2 \rightarrow ... \rightarrow l_n,$$

where l represents a layer chain and each one of $l_1, ..., l_n$ an instantiation of that layer chain with a separate set of weights.

2.3 Layer Definitions

In this section we will introduce the concrete layers and their definitions, which are available for composing NMT architectures. They are based on building blocks common to many current NMT models.

Dropout A dropout (Srivastava et al., 2014) layer, denoted as $\textit{dropout}(\mathbf{h}_t)$, can be applied to hidden states as a form of regularization.

Fixed positional embeddings Fixed positional embeddings (Vaswani et al., 2017) add information about the position in the sequence to the hidden states. With $\mathbf{h}_t \in \mathbb{R}^d$ the positional embedding layer is defined as

$$\begin{aligned} \textit{pos}(\mathbf{h}_t) &= \textit{dropout}(\sqrt{d} \cdot \mathbf{h}_t + \mathbf{p}_t) \\ p_{t,j} &= \sin(t/10000^{2j/d}) \\ p_{t,2j+1} &= \cos(t/10000^{2j/d}). \end{aligned}$$

Linear We define a linear projection layer as

$$\textit{linear}(\mathbf{h}_t, d_o) = \mathbf{W}\mathbf{h}_t + \mathbf{b},$$

where $\mathbf{W} \in \mathbb{R}^{d_o \times d_{in}}$.

Feed-forward Making use of the linear projection layer a feed-forward layer with ReLU activation and dropout is defined as

$$\textit{ff}(\mathbf{h}_t, d_o) = \textit{dropout}(\max(0, \textit{linear}(\mathbf{h}_t, d_o)))$$

and a version which temporarily upscales the number of hidden units, as done by Vaswani et al. (2017), can be defined as

$$\textit{ffl}(\mathbf{h}_t) = \textit{ff}(4d_{in}) \rightarrow \textit{linear}(d_{in})$$

where $\mathbf{h}_t \in \mathbb{R}^{d_{in}}$.

Convolution Convolutions run a small feed-forward network on a sliding window over the input. Formally, on the encoder side this is defined as

$$\textit{cnn}(\mathbf{H}, v, k) = v(\mathbf{W}[\mathbf{h}_{i-\lfloor k/2 \rfloor}; ...; \mathbf{h}_{i+\lfloor k/2 \rfloor}] + \mathbf{b})$$

where k is the kernel size, and v is a non-linearity. The input is padded so that the number of hidden states does not change.

To preserve the auto-regressive property of the decoder we need to make sure to never take future decoder time steps into account, which can be achieved by adding $k-1$ padding vectors $\mathbf{h}_{-k+1} = \mathbf{0}, \ldots, \mathbf{h}_{-1} = \mathbf{0}$ such that the decoder convolution is given as

$$\textit{cnn}(\mathbf{H}, v, k) = v(\mathbf{W}[\mathbf{h}_{t-k+1}; ...; \mathbf{h}_t] + \mathbf{b}).$$

The non-linearity v can either be a ReLU or a Gated Linear Unit (GLU) (Dauphin et al., 2016). With the GLU we set $d_i = 2d$ such that we can split $\mathbf{h} = [\mathbf{h}_A; \mathbf{h}_B] \in \mathbb{R}^{2d}$ and compute the non-linearity as

$$\textit{glu}([\mathbf{h}_A; \mathbf{h}_B]) = \mathbf{h}_A \otimes \sigma(\mathbf{h}_B).$$

Identity We define an identity layer as

$$\textit{id}(\mathbf{h}_t) = \mathbf{h}_t.$$

Concatenation To concatenate the output of p layer chains we define

$$\textit{concat}(\mathbf{h}_t, l_1, ..., l_p) = [l_1(\mathbf{h}_t); ...; l_p(\mathbf{h_t})].$$

Recurrent Neural Network An RNN layer is defined as

$$\begin{aligned} \textit{rnn}(\mathbf{h}_t) &= f_{\textit{rnn\_o}}(\mathbf{h}_t, \mathbf{s}_{t-1}) \\ \mathbf{s}_t &= f_{\textit{rnn\_h}}(\mathbf{h}_t, \mathbf{s}_{t-1}) \end{aligned}$$

where $f_{\textit{rnn\_o}}$ and $f_{\textit{rnn\_h}}$ could be defined through either a GRU (Cho et al., 2014) or a LSTM (Hochreiter and Schmidhuber, 1997) cell. In addition, a bidirectional RNN layer *birnn* is available, which runs one *rnn* in forward and another in reverse direction and concatenates both results.

Attention All attention mechanisms take a set of query vectors $\mathbf{q}_0, ..., \mathbf{q}_M$, key vectors $\mathbf{k}_0, ..., \mathbf{k}_N$ and value vectors $\mathbf{v}_0, ..., \mathbf{v}_N$ in order to produce one context vector per query, which is a linear combination of the value vectors. We define $\mathbf{Q} \in \mathbb{R}^{M \times d}$, $\mathbf{V} \in \mathbb{R}^{N \times d}$ and $\mathbf{K} \in \mathbb{R}^{N \times d}$ as the concatenation of these vectors. What is used as the query, key and value vectors depends on attention type and is defined below.

Dot product attention The scaled dot product attention (Vaswani et al., 2017) is defined as

$$\textit{dot\_att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, s) = \operatorname{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{s}}\right)\mathbf{V},$$

where the scaling factor s is implicitly set to d unless noted otherwise. Adding a projection to the queries, keys and values we get the projected dot attention as

$$\begin{gathered} \textit{proj\_dot\_att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, d_p, s) = \\ \textit{dot\_att}(\mathbf{Q}\mathbf{W}^Q, \mathbf{K}\mathbf{W}^K, \mathbf{V}\mathbf{W}^V, s) \end{gathered}$$

where d_p is dimensionality of the projected vectors such that $\mathbf{W}^Q \in \mathbb{R}^{d_q \times d_p}$, $\mathbf{W}^K \in \mathbb{R}^{d_k \times d_p}$ and $\mathbf{W}^V \in \mathbb{R}^{d_v \times d_p}$.

Vaswani et al. (2017) further introduces a multi-head attention, which applies multiple attentions at a reduced dimensionality. With h heads multi-head attention is computed as

$$\textit{mh\_dot\_att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, h, s) = [\mathbf{C}_0; ...; \mathbf{C}_h],$$

$$\mathbf{C}_i = \textit{proj\_dot\_att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, d/h, s).$$

Note that with $h = 1$ we recover the projected dot attention.

MLP attention The MLP attention (Bahdanau et al., 2014) computes the scores with a one-layer neural network as

$$\textit{mlp\_att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\mathbf{S})\,\mathbf{V},$$
$$S_{ij} = \mathbf{w}_o^T \tanh(\mathbf{W}_q\mathbf{q}_i + \mathbf{W}_k\mathbf{k}_j).$$

Source attention Using the source hidden vectors $\mathbf{U}$, the source attentions are computed as

$$\textit{mh\_dot\_src\_att}(\mathbf{H}, \mathbf{U}, h, s) =$$
$$\textit{mh\_dot\_att}(\mathbf{H}, \mathbf{U}, \mathbf{U}, h, s),$$
$$\textit{mlp\_src\_att}(\mathbf{H}, \mathbf{U}) = \textit{mlp\_att}(\mathbf{H}, \mathbf{U}, \mathbf{U}),$$
$$\textit{dot\_src\_att}(\mathbf{H}, \mathbf{U}, s) = \textit{mh\_dot\_att}(\mathbf{H}, \mathbf{U}, \mathbf{U}, 1, s).$$

Self-attention Self-attention (Vaswani et al., 2017) uses the hidden states as queries, keys and values such that

$$\textit{mh\_dot\_self\_att}(\mathbf{H}, s) = \textit{mh\_dot\_att}(\mathbf{H}, \mathbf{H}, \mathbf{H}, s).$$

Please note that on the target side one needs to make sure to preserve the auto-regressive property by only attending to hidden states at the current or past steps $\mathbf{h} < t$, which is achieved by masking the attention mechanism.

Layer normalization Layer normalization (Ba et al., 2016) uses the mean and standard deviation for normalization. It is computed as

$$\textit{norm}(\mathbf{h}_t) = \frac{\mathbf{g}}{\sigma_t} \otimes (\mathbf{h}_t - \mu_t) + \mathbf{b}$$
$$\mu_t = \frac{1}{d}\sum_{i=1}^{d} \mathbf{h}_{t,j} \quad \sigma_t = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(\mathbf{h}_{t,j} - \mu_j)^2}$$

where $\mathbf{g}$ and $\mathbf{b}$ are learned scale and shift parameters with the same dimensionality as $\mathbf{h}$.

Residual layer A residual layer adds the output of an arbitrary layer chain l to the current hidden states. We define this as

$$\textit{res}(\mathbf{h_t}, l) = \mathbf{h_t} + l(\mathbf{h_t}).$$

For convenience we also define

$$\textit{res\_d}(\mathbf{h}_t, l) = \textit{res}(l(\mathbf{h}_t) \rightarrow \textit{dropout}) \quad \text{and}$$
$$\textit{res\_nd}(\mathbf{h}_t, l) = \textit{res}(\text{norm} \rightarrow l(\mathbf{h}_t) \rightarrow \textit{dropout}).$$

2.4 Standard Architectures

Having defined the common building blocks we now show how standard NMT architectures can be constructed.

RNMT As RNNs have been around the longest in NMT, several smaller architecture variations exist. Similar to Wu et al. (2016) in the following we use a bi-directional RNN followed by a stack of uni-directional RNNs with residual connections on the encoder side. Using the ADL an n layer encoder can be expressed as

$$\mathbf{U}^{L_s} = \textit{dropout} \rightarrow \textit{birnn} \rightarrow \textit{repeat}(n-1, \textit{res\_d}(\textit{rnn})).$$

For the decoder we use the architecture by Luong et al. (2015), which first runs a stacked RNN and then combines the context provided by a single attention mechanism with the hidden state provided by the RNN. This can be expressed by

$$\mathbf{Z}^L = \textit{dropout} \rightarrow \textit{repeat}(n, \textit{res\_d}(\textit{rnn})) \rightarrow$$
$$\textit{concat}(\textit{id}, \textit{mlp\_att}) \rightarrow \textit{ff}.$$

If input feeding (Luong et al., 2015) is used the first layer hidden states are redefined as

$$\mathbf{z}_t^0 = [\mathbf{z}_{t-1}^L; \mathbf{E}_{tgt}\mathbf{y}_t].$$

Note that this inhibits any parallelism across decoder time steps. This is only an issue when using models other than RNNs, as RNNs already do not allow for parallelizing over decoder time steps.

ConvS2S Gehring et al. (2017) introduced a NMT model that fully relies on convolutions, both on the encoder and on the decoder side. The encoder is defined as

$$\mathbf{U}^{L_s} = \textit{pos} \rightarrow \textit{repeat}(n, \textit{res}(\textit{cnn}(\textit{glu}) \rightarrow \textit{dropout}))$$

and the decoder, which uses an unscaled single head dot attention is defined as

$$\mathbf{Z}^L = \textit{pos} \rightarrow \textit{res}(\textit{dropout} \rightarrow \textit{cnn}(\textit{glu}) \rightarrow \textit{dropout}$$
$$\rightarrow \textit{res}(\textit{dot\_src\_att}(s{=}1))).$$

Note that unlike (Gehring et al., 2017) we do not project the query vectors before the attention and do not add the embeddings to the attention values.

Transformer The Transformer (Vaswani et al., 2017) makes use of self-attention, instead of RNNs or Convolutional Neural Networks (CNNs), as the basic computational block. Note that we use a slightly updated residual structure as implemented by *tensor2tensor*[1] than proposed originally. Specifically, layer normalization is applied to the input of the residual block instead of applying it between blocks. The Transformer uses a combination of self-attention and feed-forward layers on the encoder and additionally source attention layers on the decoder side. When defining the Transformer encoder block as

$$t_{\text{enc}} = \mathit{res\_nd}(\mathit{mh\_dot\_self\_att}) \rightarrow \mathit{res\_nd}(\mathit{ffl}),$$

and the decoder block as

$$t_{\text{dec}} = \mathit{res\_nd}(\mathit{mh\_dot\_self\_att}) \rightarrow \mathit{res\_nd}(\mathit{mh\_dot\_src\_att}) \rightarrow \mathit{res\_nd}(\mathit{ffl}).$$

the Transformer encoder is given as

$$\mathbf{U}^{L_s} = \mathit{pos} \rightarrow \mathit{repeat}(n, t_{\text{enc}}) \rightarrow \mathit{norm}$$

and the decoder as

$$\mathbf{Z}^{L} = \mathit{pos} \rightarrow \mathit{repeat}(n, t_{\text{dec}}) \rightarrow \mathit{norm}.$$

3 Related Work

The dot attention mechanism, now heavily used in the Transformer models, was introduced by (Luong et al., 2015) as part of an exploration of different attention mechanisms for RNN based NMT models.

Britz et al. (2017) performed an extensive exploration of hyperparameters of RNN based NMT models. The variations explored include different attention mechanisms, RNN cells types and model depth.

Similar to our work, Schrimpf et al. (2017) define a language for exploring architectures. In this case the architectures are defined for RNN cells and not for the higher level model architecture. Using the language they perform an automatic search of RNN cell architectures.

For the application of image classification there have been several recent successful efforts of automatically searching for successful architectures (Zoph and Le, 2016; Negrinho and Gordon, 2017; Liu et al., 2017).

[1]https://github.com/tensorflow/tensor2tensor

4 Experiments

What follows is an extensive empirical analysis of current NMT architectures and how certain sublayers as defined through our ADL affect performance.

4.1 Setup

All experiments were run with an adapted version of SOCKEYE (Hieber et al., 2017), which can parse arbitrary model definitions that are expressed in the language described in Section 2.3. The code and configuration are available at https://github.com/awslabs/sockeye/tree/acl18 allowing researchers to easily replicate the experiments and to quickly try new NMT architectures by either making use of existing building blocks in novel ways or adding new ones.

In order to get data points on corpora of different sizes we ran experiments on both WMT and IWSLT data sets. For WMT we ran the majority of our experiments on the most recent WMT'17 data consisting of roughly 5.9 million training sentences for English-German (EN→DE) and 4.5 million sentences for Latvian-English (LV→EN). We used newstest2016 as validation data and report metrics calculated on newstest2017. For the smaller IWSLT'16 English-German corpus, which consists of roughly 200 thousand training sentences, we used TED.tst2013 as validation data and report numbers for TED.tst2014.

For both WMT'17 and IWSLT'16 we preprocessed all data using the Moses[2] tokenizer and apply Byte Pair Encoding (BPE) (Sennrich et al., 2015) with 32,000 merge operations. Unless noted otherwise we run each experiment three times with different random seeds and report the mean and standard deviation of the BLEU and METEOR (Lavie and Denkowski, 2009) scores across runs. Evaluation scores are based on tokenized sequences and calculated with MultEval (Clark et al., 2011).

Model	WMT'14
Vaswani et al. (2017)	27.3
Our Transformer$_{\text{base}}$ impl.	27.5

Table 1: BLEU scores on WMT'14 EN→DE.

In order to compare to previous work, we also ran an additional experiment on WMT'14 using the same data as Vaswani et al. (2017)

[2]https://github.com/moses-smt/mosesdecoder/

as provided in preprocessed form through *tensor2tensor*.[3] This data set consists of WMT'16 training data, which has been tokenized and byte pair encoded with 32,000 merge operations. Evaluation is done on tokenized and compound split newstest2014 data using multi-bleu.perl in order to get scores comparable to Vaswani et al. (2017). As seen in Table 1, our Transformer implementation achieves a score equivalent to the originally reported numbers.

On the smaller IWSLT data we use $d_{model} = 512$ and on WMT $d_{model} = 256$ for all models. Models are trained with 6 encoder and 6 decoder blocks, where in the Transformer model a layer refers to a full encoder or decoder block. All convolutional layers use a kernel of size 3 and a ReLU activation, unless noted otherwise. RNNs use LSTM cells. For training we use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0002. The learning rate is decayed by a factor of 0.7, whenever the validation perplexity does not improve for 8 consecutive checkpoints, where a checkpoint is created every 4,000 updates on WMT and 1,000 updates on IWSLT. All models use label smoothing (Szegedy et al., 2016) with $\epsilon_{ls} = 0.1$.

4.2 What to attend to?

Source attention is typically based on the top encoder block. With multiple source attention layers one could hypothesize that it could be beneficial to allow attention encoder blocks other than the top encoder block. It might for example be beneficial for lower decoder blocks to use encoder blocks from the same level as they represent the same level of abstraction. Inversely, assuming that the translation is done in a coarse to fine manner it might help to first use the uppermost encoder block and use gradually lower level representations.

Encoder block	IWSLT	WMT'17
upper	25.4 ± 0.2	27.6 ± 0.0
increasing	25.4 ± 0.1	27.3 ± 0.1
decreasing	25.3 ± 0.2	27.1 ± 0.1

Table 2: BLEU scores when varying the encoder block used in the source attention mechanism of a Transformer on the EN→DE IWSLT and WMT'17 datasets.

[3] https://github.com/tensorflow/tensor2tensor/blob/765d33bb/tensor2tensor/data_generators/translate_ende.py

The result of modifying the source attention mechanism to use different encoder blocks is shown in Table 2. The variations include using the result of the encoder Transformer block at the same level as the decoder Transformer block (*increasing*) and using the upper encoder Transformer block in the first decoder block and then gradually using the lower blocks (*decreasing*).

We can see that attention on the upper encoder block performs best and no gains can be observed by attention on different encoder layers in the source attention mechanism.

4.3 Network Structure

The Transformer sets itself apart from both standard RNN models and convolutional model by more than just the multi-head self-attention blocks.

RNN to Transformer The differences to the RNN include the multiple source attention layers, multi-head attention, layer normalization and the residual upscaling feed-forward layers. Additionally, RNN models typically use single head MLP attention instead of the dot attention. This raises the question of what aspect contributes most to the performance of the Transformer.

Table 3 shows the result of taking an RNN and step by step changing the architecture to be similar to the Transformer architecture. We start with a standard RNN architecture with MLP attention similar to Luong et al. (2015) as described in Section 2.4 with and without input feeding denoted as RNMT.

Next, we take a model with a residual connection around the encoder bi-RNN such that the encoder is defined as

$$\mathit{dropout} \rightarrow \mathit{res\_d}(\mathit{birnn}) \rightarrow \mathit{repeat}(5, \mathit{res\_d}(\mathit{rnn})).$$

The decoder uses a residual single head dot attention and no input feeding and is defined as

$$\mathit{dropout} \rightarrow \mathit{repeat}(6, \mathit{res\_d}(\mathit{rnn})) \rightarrow \\ \mathit{res\_d}(\mathit{dot\_src\_att}) \rightarrow \mathit{res\_d}(\mathit{ffl}).$$

We denote this model as RNN in Table 3. This model is then changed to use multi-head attention (mh), positional embeddings (pos), layer normalization on the inputs of the residual blocks (norm), an attention mechanism in a residual block after every RNN layer with multiple (multi-att) and a single head (multi-add-1h), and finally a residual

Model	IWSLT EN→DE	WMT'17 EN→DE		WMT'17 LV→EN	
	BLEU	BLEU	METEOR	BLEU	METEOR
Transformer	25.4 ± 0.1	27.6 ± 0.0	47.2 ± 0.1	18.5 ± 0.0	51.3 ± 0.1
RNMT	23.2 ± 0.2	25.5 ± 0.2	45.1 ± 0.1	-	-
- input feeding	23.1 ± 0.2	24.6 ± 0.1	43.8 ± 0.2	-	-
RNN	22.8 ± 0.2	23.8 ± 0.1	43.3 ± 0.1	15.2 ± 0.1	45.9 ± 0.1
+ mh	23.7 ± 0.4	24.4 ± 0.1	43.9 ± 0.1	16.0 ± 0.1	47.1 ± 0.1
+ pos	23.9 ± 0.2	24.1 ± 0.1	43.5 ± 0.2	-	-
+ norm	23.7 ± 0.1	24.0 ± 0.2	43.2 ± 0.1	15.2 ± 0.1	46.3 ± 0.2
+ multi-att-1h	24.5 ± 0.0	25.2 ± 0.1	44.9 ± 0.1	16.6 ± 0.2	49.1 ± 0.2
/ multi-att	24.4 ± 0.3	25.5 ± 0.0	45.3 ± 0.0	17.0 ± 0.2	49.4 ± 0.1
+ ff	25.1 ± 0.1	26.7 ± 0.1	46.4 ± 0.2	17.8 ± 0.1	50.5 ± 0.1

Table 3: Transforming an RNN into a Transformer style architecture. + shows the incrementally added variation. / denotes an alternative variation to which the subsequent + is relative to.

upscaling feed-forward layer is added after each attention block (ff). The final architecture of the encoder after applying these variations is

$$pos \rightarrow res\_nd(birnn) \rightarrow res\_nd(ffl) \rightarrow \\ repeat(5, res\_nd(rnn) \rightarrow res\_nd(ffl) \rightarrow norm$$

and of the decoder

$$pos \rightarrow repeat(6, res\_nd(rnn) \rightarrow \\ res\_nd(mh\_dot\_src\_att) \rightarrow res\_nd(ffl)) \rightarrow norm.$$

Comparing this to the Transformer as defined in Section 2.4 we note that the model is identical to the Transformer, except that each self-attention has been replaced by an RNN or bi-RNN.

Table 3 shows that not using input feeding has a negative effect on the result, which however can be compensated by the explored model variations. With just a single attention mechanism the model benefits from multiple attention heads. The gains are even larger when an attention mechanism is added to every layer. With multiple source attention mechanisms the benefit of multiple heads decreases. Layer normalization on the inputs of the residual blocks has a small negative effect in all settings and metrics. As RNNs can learn to encode positional information positional embeddings are not strictly necessary. Indeed, we can observe no gains but rather even a small drop in BLEU and METEOR for WMT'17 EN→DE when using them. Adding feed-forward layers leads to large and consistent performance boost. While the final model, which is a Transformer model where each self-attention has been replaced by an RNN, is able to make up for a large amount of the difference between the baseline and the Transformer, it is still outperformed by the Transformer. The largest gains come from multiple attention mechanisms and residual feed-forward layers.

CNN to Transformer While the convolutional models have much more in common with the Transformer than the RNN based models, there are still some notable differences. Like the Transformer, convolutional models have no dependency between decoder time steps during training, use multiple source attention mechanisms and use a slightly different residual structure, as seen in Section 2.4. The Transformer uses a multi-head scaled dot attention while the ConvS2S model uses an unscaled single head dot attention. Other differences include the use of layer normalization as well as residual feed-forward blocks in the Transformer.

The result of making a CNN based architecture more and more similar to the Transformer can be seen in Table 4. As a baseline we use a simple residual CNN structure with a residual single head dot attention. This is denoted as CNN in Table 4. On the encoder side we have

$$pos \rightarrow repeat(6, res\_d(cnn))$$

and for the decoder

$$pos \rightarrow repeat(6, res\_d(cnn) \rightarrow res\_d(dot\_src\_att)).$$

This is similar to, but slightly simpler than, the ConvS2S model described in Section 2.4. In the experiments we explore both the GLU and ReLU as non-linearities for the CNN.

Adding layer normalization (norm), multi-head attention (mh) and upsampling residual feed-forward layers (ff) we arrive at a model that is

	IWSLT EN-DE	WMT'17 EN→DE		WMT'17 LV→EN	
Model	BLEU	BLEU	METEOR	BLEU	METEOR
Transformer	25.4 ± 0.1	27.6 ± 0.0	47.2 ± 0.1	18.5 ± 0.0	51.3 ± 0.1
CNN GLU	24.3 ± 0.4	25.0 ± 0.3	44.4 ± 0.2	16.0 ± 0.5	47.4 ± 0.4
+ norm	24.1 ± 0.1	-	-	-	-
+ mh	24.2 ± 0.2	25.4 ± 0.1	44.8 ± 0.1	16.1 ± 0.1	47.6 ± 0.2
+ ff	25.3 ± 0.1	26.8 ± 0.1	46.0 ± 0.1	16.4 ± 0.2	47.9 ± 0.2
CNN ReLU	23.6 ± 0.3	23.9 ± 0.1	43.4 ± 0.1	15.4 ± 0.1	46.4 ± 0.3
+ norm	24.3 ± 0.1	24.3 ± 0.2	43.6 ± 0.1	16.0 ± 0.2	47.1 ± 0.5
+ mh	24.2 ± 0.2	24.9 ± 0.1	44.4 ± 0.1	16.1 ± 0.1	47.5 ± 0.2
+ ff	25.3 ± 0.3	26.9 ± 0.1	46.1 ± 0.0	16.4 ± 0.2	47.9 ± 0.1

Table 4: Transforming a CNN based model into a Transformer style architecture.

identical to a Transformer where the self-attention layers have been replaced by CNNs. This means that we have the following architecture on the encoder

$$pos \rightarrow repeat(6, res\_nd(cnn) \rightarrow res\_nd(ffl)) \rightarrow norm.$$

Whereas for the decoder we have

$$pos \rightarrow repeat(6, res\_nd(cnn) \\ \rightarrow res\_nd(mh\_dot\_src\_att) \rightarrow res\_nd(ffl)) \rightarrow norm.$$

While in the baseline the GLU activation works better than the ReLU activation, when layer normalization, multi-head attention attention and residual feed-forward layers are added, the performance is similar. Except for IWSLT multi-head attention gives consistent gains over single head attention. The largest gains can however be observed by the addition of residual feed-forward layers. The performance of the final model, which is very similar to a Transformer where each self-attention has been replaced by a CNN, matches the performance of the Transformer on IWSLT EN→DE but is still 0.7 BLEU points worse on WMT'17 EN→DE and two BLEU points on WMT'17 LV→EN.

4.4 Self-attention variations

At the core of the Transformer are self-attentional layers, which take the role previously occupied by RNNs and CNNs. Self-attention has the advantage that any two positions are directly connected and that, similar to CNNs, there are no dependencies between consecutive time steps so that the computation can be fully parallelized across time. One disadvantage is that relative positional information is not directly represented and one needs to rely on the different heads to make up for this. In a CNN information is constrained to a local window which grows linearly with depth. Relative positions are therefore taken into account. While an RNN keeps an internal state, which can be used in future time steps, it is unclear how well this works for very long range dependencies (Koehn and Knowles, 2017; Bentivogli et al., 2016). Additionally, having a dependency on the previous hidden state inhibits any parallelization across time.

Given the different advantages and disadvantages we selectively replace self-attention on the encoder and decoder side in order to see where the model benefits most from self-attention.

We take the encoder and decoder block defined in Section 2.4 and try out different layers in place of the self-attention. Concretely, we have

$$t_{\text{enc}} = res\_nd(\mathbf{x}_{\text{enc}}) \rightarrow res\_nd(ffl),$$

on the encoder side and

$$t_{\text{dec}} = res\_nd(\mathbf{x}_{\text{dec}}) \rightarrow \\ res\_nd(mh\_dot\_src\_att) \rightarrow res\_nd(ffl).$$

on the decoder side. Table 5 shows the result of replacing $\mathbf{x}_{\text{enc}}$ and $\mathbf{x}_{\text{dec}}$ with either self-attention, a CNN with ReLU activation or an RNN. Notice that with self-attention used in both $\mathbf{x}_{\text{enc}}$ and $\mathbf{x}_{\text{dec}}$ we recover the Transformer model. Additionally, we remove the residual block on the decoder side entirely (*none*). This results in a decoder block which only has information about the previous target word y_t through the word embedding that is fed as the input to the first layer. The decoder block is reduced to

$$t_{\text{dec}} = res\_nd(mh\_dot\_src\_att) \rightarrow res\_nd(ffl).$$

		IWSLT EN→DE	WMT'17 EN→DE		WMT'17 LV→EN	
Encoder	Decoder	BLEU	BLEU	METEOR	BLEU	METEOR
self-att	self-att	25.4 ± 0.2	27.6 ± 0.0	47.2 ± 0.1	18.3 ± 0.0	51.1 ± 0.1
self-att	RNN	25.1 ± 0.1	27.4 ± 0.1	47.0 ± 0.1	18.4 ± 0.2	51.1 ± 0.1
self-att	CNN	25.4 ± 0.4	27.6 ± 0.2	46.7 ± 0.1	18.0 ± 0.3	50.3 ± 0.3
RNN	self-att	25.8 ± 0.1	27.2 ± 0.1	46.7 ± 0.1	17.8 ± 0.1	50.6 ± 0.1
CNN	self-att	25.7 ± 0.1	26.6 ± 0.3	46.3 ± 0.1	16.8 ± 0.4	49.4 ± 0.4
RNN	RNN	25.1 ± 0.1	26.7 ± 0.1	46.4 ± 0.2	17.8 ± 0.1	50.5 ± 0.1
CNN	CNN	25.3 ± 0.3	26.9 ± 0.1	46.1 ± 0.0	16.4 ± 0.2	47.9 ± 0.2
self-att	*combined*	25.1 ± 0.2	27.6 ± 0.2	47.2 ± 0.2	18.3 ± 0.1	51.1 ± 0.1
self-att	*none*	23.7 ± 0.2	25.3 ± 0.2	43.1 ± 0.1	15.9 ± 0.1	45.1 ± 0.2

Table 5: Different variations of the encoder and decoder self-attention layer.

In addition to that, we try a combination where the first and fourth block use self-attention, the second and fifth an RNN, the third and sixth a CNN (*combined*).

Replacing the self-attention on both the encoder and the decoder side with an RNN or CNN results in a degradation of performance. In most settings, such as WMT'17 EN→DE for both variations and WMT'17 LV→EN for the RNN, the performance is comparable when replacing the decoder side self-attention. For the encoder however, except for IWSLT, we see a drop in performance of up to 1.5 BLEU points when not using self-attention. Therefore, self-attention seems to be more important on the encoder side than on the decoder side. Despite the disadvantage of having a limited context window, the CNN performs as well as self-attention on the decoder side on IWLT and WMT'17 EN→DE in terms of BLEU and only slightly worse in terms of METEOR. The combination of the three mechanisms (*combined*) on the decoder side performs almost identical to the full Transformer model, except for IWSLT where it is slightly worse.

It is surprising how well the model works without any self-attention as the decoder essentially looses any information about the history of generated words. Translations are entirely based on the previous word, provided through the target side word embedding, and the current position, provided through the positional embedding.

5 Conclusion

We described an ADL for specifying NMT architectures based on composable building blocks. Instead of committing to a single architecture, the language allows for combining architectures on a granular level. Using this language we explored how specific aspects of the Transformer architecture can successfully be applied to RNNs and CNNs. We performed an extensive evaluation on IWSLT EN→DE, WMT'17 EN→DE and LV→EN, reporting both BLEU and METEOR over multiple runs in each setting.

We found that RNN based models benefit from multiple source attention mechanisms and residual feed-forward blocks. CNN based models on the other hand can be improved through layer normalization and also feed-forward blocks. These variations bring the RNN and CNN based models close to the Transformer. Furthermore, we showed that one can successfully combine architectures. We found that self-attention is much more important on the encoder side than it is on the decoder side, where even a model without self-attention performed surprisingly well. For the data sets we evaluated on, models with self-attention on the encoder side and either an RNN or CNN on the decoder side performed competitively to the Transformer model in most cases.

We make our implementation available so that it can be used for exploring novel architecture variations.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *ArXiv preprint arXiv:1712.05690*.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2-3):105–115.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Renato Negrinho and Geoff Gordon. 2017. Deeparchitect: Automatically designing and training deep architectures. *arXiv preprint arXiv:1704.08792*.

Martin Schrimpf, Stephen Merity, James Bradbury, and Richard Socher. 2017. A flexible approach to automated rnn architecture generation. *arXiv preprint arXiv:1712.07316*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Weakly Supervised Semantic Parsing with Abstract Examples

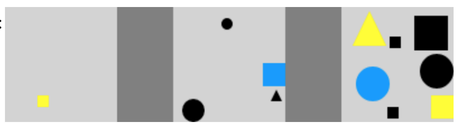
Omer Goldman[*], **Veronica Latcinnik**[*], **Udi Naveh**[*], **Amir Globerson**, **Jonathan Berant**
Tel-Aviv University
{omergoldman@mail,veronical@mail,
ehudnave@mail,gamir@post,joberant@cs}.tau.ac.il

Abstract

Training semantic parsers from weak supervision (denotations) rather than strong supervision (programs) complicates training in two ways. First, a large *search* space of potential programs needs to be explored at training time to find a correct program. Second, *spurious* programs that accidentally lead to a correct denotation add noise to training. In this work we propose that in closed worlds with clear semantic types, one can substantially alleviate these problems by utilizing an abstract representation, where tokens in both the language utterance and program are lifted to an abstract form. We show that these abstractions can be defined with a handful of lexical rules and that they result in sharing between different examples that alleviates the difficulties in training. To test our approach, we develop the first semantic parser for CNLVR, a challenging visual reasoning dataset, where the search space is large and overcoming spuriousness is critical, because denotations are either TRUE or FALSE, and thus random programs are likely to lead to a correct denotation. Our method substantially improves performance, and reaches 82.5% accuracy, a 14.7% absolute accuracy improvement compared to the best reported accuracy so far.

1 Introduction

The goal of semantic parsing is to map language utterances to executable programs. Early work on statistical learning of semantic parsers utilized supervised learning, where training examples included pairs of language utterances and programs (Zelle and Mooney, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2005, 2007). However, collecting such training examples at scale has quickly turned out to be difficult, because expert annotators who are familiar with formal languages are required. This has led to a body of work on weakly-supervised semantic parsing (Clarke et al., 2010; Liang et al., 2011; Krishnamurthy and Mitchell, 2012; Kwiatkowski et al., 2013; Berant et al., 2013; Cai and Yates, 2013; Artzi and Zettlemoyer, 2013). In this setup, training examples correspond to utterance-denotation pairs, where a *denotation* is the result of executing a program against the environment (see Fig. 1). Naturally, collecting denotations is much easier, because it can be performed by non-experts.

I :


k :[[{y_loc: ..., color: 'Black', type: 'square', x_loc: ... size: 20}, ...}]]
x :*There is a small yellow item not touching any wall*
y :`True`
z :`Exist(Filter(ALL_ITEMS,` λx`.And(And(IsYellow(`x`), IsSmall(`x`)), Not(IsTouchingWall(`x`, Side.Any)))))`

Figure 1: Overview of our visual reasoning setup for the CNLVR dataset. Given an image rendered from a KB k and an utterance x, our goal is to parse x to a program z that results in the correct denotation y. Our training data includes (x, k, y) triplets.

Training semantic parsers from denotations rather than programs complicates training in two ways: (a) *Search*: The algorithm must learn to search through the huge space of programs at training time, in order to find the correct program. This is a difficult search problem due to the combinatorial nature of the search space. (b) *Spurious-*

[*] Authors equally contributed to this work.

ness: Incorrect programs can lead to correct denotations, and thus the learner can go astray based on these programs. Of the two mentioned problems, spuriousness has attracted relatively less attention (Pasupat and Liang, 2016; Guu et al., 2017).

Recently, the Cornell Natural Language for Visual Reasoning corpus (CNLVR) was released (Suhr et al., 2017), and has presented an opportunity to better investigate the problem of spuriousness. In this task, an image with boxes that contains objects of various shapes, colors and sizes is shown. Each image is paired with a complex natural language statement, and the goal is to determine whether the statement is true or false (Fig. 1). The task comes in two flavors, where in one the input is the image (pixels), and in the other it is the knowledge-base (KB) from which the image was synthesized. Given the KB, it is easy to view CNLVR as a semantic parsing problem: our goal is to translate language utterances into programs that will be executed against the KB to determine their correctness (Johnson et al., 2017b; Hu et al., 2017). Because there are only two return values, it is easy to generate programs that execute to the right denotation, and thus spuriousness is a major problem compared to previous datasets.

In this paper, we present the first semantic parser for CNLVR. Semantic parsing can be coarsely divided into a *lexical* task (i.e., mapping words and phrases to program constants), and a *structural* task (i.e., mapping language composition to program composition operators). Our core insight is that in closed worlds with clear semantic types, like spatial and visual reasoning, we can manually construct a small lexicon that clusters language tokens and program constants, and create a partially abstract representation for utterances and programs (Table 1) in which the lexical problem is substantially reduced. This scenario is ubiquitous in many semantic parsing applications such as calendar, restaurant reservation systems, housing applications, etc: the formal language has a compact semantic schema and a well-defined typing system, and there are canonical ways to express many program constants.

We show that with abstract representations we can share information across examples and better tackle the search and spuriousness challenges. By pulling together different examples that share the same abstract representation, we can identify programs that obtain high reward across multiple examples, thus reducing the problem of spuriousness. This can also be done at search time, by augmenting the search state with partial programs that have been shown to be useful in earlier iterations. Moreover, we can annotate a small number of abstract utterance-program pairs, and automatically generate training examples, that will be used to warm-start our model to an initialization point in which search is able to find correct programs.

We develop a formal language for visual reasoning, inspired by Johnson et al. (2017b), and train a semantic parser over that language from weak supervision, showing that abstract examples substantially improve parser accuracy. Our parser obtains an accuracy of 82.5%, a 14.7% absolute accuracy improvement compared to state-of-the-art. All our code is publicly available at `https://github.com/udiNaveh/nlvr_tau_nlp_final_proj`.

2 Setup

Problem Statement Given a training set of N examples $\{(x_i, k_i, y_i)\}_{i=1}^N$, where x_i is an utterance, k_i is a KB describing objects in an image and $y_i \in \{\text{TRUE}, \text{FALSE}\}$ denotes whether the utterance is true or false in the KB, our goal is to learn a semantic parser that maps a new utterance x to a program z such that when z is executed against the corresponding KB k, it yields the correct denotation y (see Fig. 1).

Programming language The original KBs in CNLVR describe an image as a set of objects, where each object has a color, shape, size and location in absolute coordinates. We define a programming language over the KB that is more amenable to spatial reasoning, inspired by work on the CLEVR dataset (Johnson et al., 2017b). This programming language provides access to functions that allow us to check the size, shape, and color of an object, to check whether it is touching a wall, to obtain sets of items that are above and below a certain set of items, etc.[1] More formally, a program is a sequence of tokens describing a possibly recursive sequence of function applications in prefix notation. Each token is either a function with fixed arity (all functions have either one or two arguments), a constant, a variable or a λ term used to define Boolean functions. Functions, constants and variables have one of the following

[1]We leave the problem of learning the programming language functions from the original KB for future work.

x:	*"There are exactly 3 yellow squares touching the wall."*
z:	`Equal(3, Count(Filter(ALL_ITEMS, λx. And (And (IsYellow(x), IsSquare(x), IsTouchingWall(x))))))`
$\bar{x}$:	*"There are C-QuantMod C-Num C-Color C-Shape touching the wall."*
$\bar{z}$:	`C-QuantMod(C-Num, Count(Filter(ALL_ITEMS, λx. And (And (IsC-Color(x), IsC-Shape(x), IsTouchingWall(x))))))`

Table 1: An example for an utterance-program pair (x, z) and its abstract counterpart $(\bar{x}, \bar{z})$

x:	*"There is a small yellow item not touching any wall."*
z:	`Exist(Filter(ALL_ITEMS, λx.And(And(IsYellow(x), IsSmall(x)), Not(IsTouchingWall(x, Side.Any)))))`
x:	*"One tower has a yellow base."*
z:	`GreaterEqual(1, Count(Filter(ALL_ITEMS, λx.And(IsYellow(x), IsBottom(x)))))`

Table 2: Examples for utterance-program pairs. Commas and parenthesis provided for readability only.

atomic types: `Int`, `Bool`, `Item`, `Size`, `Shape`, `Color`, `Side` (sides of a box in the image); or a composite type `Set(?)`, and `Func(?,?)`. Valid programs have a return type `Bool`. Tables 1 and 2 provide examples for utterances and their correct programs. The supplementary material provides a full description of all program tokens, their arguments and return types.

Unlike CLEVR, CNLVR requires substantial set-theoretic reasoning (utterances refer to various aspects of sets of items in one of the three boxes in the image), which required extending the language described by Johnson et al. (2017b) to include set operators and lambda abstraction. We manually sampled 100 training examples from the training data and estimate that roughly 95% of the utterances in the training data can be expressed with this programming language.

3 Model

We base our model on the semantic parser of Guu et al. (2017). In their work, they used an encoder-decoder architecture (Sutskever et al., 2014) to define a distribution $p_\theta(z \mid x)$. The utterance x is encoded using a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) that creates a contextualized representation h_i for every utterance token x_i, and the decoder is a feed-forward network combined with an attention mechanism over the encoder outputs (Bahdanau et al., 2015). The feed-forward decoder takes as input the last K tokens that were decoded.

More formally the probability of a program is the product of the probability of its tokens given the history: $p_\theta(z \mid x) = \prod_t p_\theta(z_t \mid x, z_{1:t-1})$, and the probability of a decoded token is computed as follows. First, a Bi-LSTM encoder converts the input sequence of utterance embeddings into a sequence of forward and backward states $h_1^{\{F,B\}}, \ldots, h_{|x|}^{\{F,B\}}$. The utterance representation $\hat{x}$ is $\hat{x} = [h_{|x|}^F; h_1^B]$. Then decoding produces the program token-by-token:

$$q_t = \text{ReLU}(W_q[\hat{x}; \hat{v}; z_{t-K-1:t-1}]),$$

$$\alpha_{t,i} \propto \exp(q_t^\top W_\alpha h_i)\ ,\ c_t = \sum_i \alpha_{t,i} h_i,$$

$$p_\theta(z_t \mid x, z_{1:t-1}) \propto \exp(\phi_{z_t}^\top W_s[q_t; c_t]),$$

where ϕ_z is an embedding for program token z, $\hat{v}$ is a bag-of-words vector for the tokens in x, $z_{i:j} = (z_i, \ldots, z_j)$ is a history vector of size K, the matrices W_q, W_α, W_s are learned parameters (along with the LSTM parameters and embedding matrices), and ';' denotes concatenation.

Search: Searching through the large space of programs is a fundamental challenge in semantic parsing. To combat this challenge we apply several techniques. First, we use beam search at decoding time and when training from weak supervision (see Sec. 4), similar to prior work (Liang et al., 2017; Guu et al., 2017). At each decoding step we maintain a beam B of program prefixes of length n, expand them exhaustively to programs of length $n+1$ and keep the top-$|B|$ program prefixes with highest model probability.

Second, we utilize the semantic typing system to only construct programs that are syntactically valid, and substantially prune the program search space (similar to type constraints in Krishnamurthy et al. (2017); Xiao et al. (2016); Liang et al. (2017)). We maintain a stack that keeps track of the expected semantic type at each decoding step. The stack is initialized with the type `Bool`. Then, at each decoding step, only tokens that return the semantic type at the top of the stack are allowed, the stack is popped, and if the decoded token is a function, the semantic types of its arguments are pushed to the stack. This dramatically reduces the search space and guarantees that only syntactically valid programs will be produced. Fig. 2 illustrates the state of the stack when decoding a program for an input utterance.

x : *One tower has a yellow base.*

z :		`EqualInt`	`1`	`Count`	`Filter`	`ALL_ITEMS`	λx	`And`	`IsYellow`	x	`IsBottom`	x
s :		`Int`			`Set`			`Bool`	`Item`			
	`Bool`	`Int`	`Int`	`Set`	`BoolFunc`	`BoolFunc`	`Bool`	`Bool`	`Bool`	`Bool`	`Item`	

Figure 2: An example for the state of the type stack s while decoding a program z for an utterance x.

Given the constrains on valid programs, our model $p'_\theta(z \mid x)$ is defined as:

$$\prod_t \frac{p_\theta(z_t \mid x, z_{1:t-1}) \cdot \mathbb{1}(z_t \mid z_{1:t-1})}{\sum_{z'} p_\theta(z' \mid x, z_{1:t-1}) \cdot \mathbb{1}(z' \mid z_{1:t-1})},$$

where $\mathbb{1}(z_t \mid z_{1:t-1})$ indicates whether a certain program token is valid given the program prefix.

Discriminative re-ranking: The above model is a locally-normalized model that provides a distribution for every decoded token, and thus might suffer from the label bias problem (Andor et al., 2016; Lafferty et al., 2001). Thus, we add a globally-normalized re-ranker $p_\psi(z \mid x)$ that scores all $|B|$ programs in the final beam produced by $p'_\theta(z \mid x)$. Our globally-normalized model is:

$$p^g_\psi(z \mid x) \propto \exp(s_\psi(x, z)),$$

and is normalized over all programs in the beam. The scoring function $s_\psi(x, z)$ is a neural network with identical architecture to the locally-normalized model, except that (a) it feeds the decoder with the candidate program z and does not generate it. (b) the last hidden state is inserted to a feed-forward network whose output is $s_\psi(x, z)$. Our final ranking score is $p'_\theta(z|x)p^g_\psi(z \mid x)$.

4 Training

We now describe our basic method for training from weak supervision, which we extend upon in Sec. 5 using abstract examples. To use weak supervision, we treat the program z as a latent variable that is approximately marginalized. To describe the objective, define $R(z, k, y) \in \{0, 1\}$ to be one if executing program z on KB k results in denotation y, and zero otherwise. The objective is then to maximize $p(y \mid x)$ given by:

$$\begin{aligned}\sum_{z \in \mathcal{Z}} p'_\theta(z \mid x) p(y \mid z, k) &= \sum_{z \in \mathcal{Z}} p'_\theta(z \mid x) R(z, k, y) \\ &\approx \sum_{z \in B} p'_\theta(z \mid x) R(z, k, y)\end{aligned}$$

where $\mathcal{Z}$ is the space of all programs and $B \subset \mathcal{Z}$ are the programs found by beam search.

In most semantic parsers there will be relatively few z that generate the correct denotation y. However, in CNLVR, y is binary, and so spuriousness is a central problem. To alleviate it, we utilize a property of CNLVR: the same utterance appears 4 times with 4 different images.[2] If a program is spurious it is likely that it will yield the wrong denotation in one of those 4 images.

Thus, we can re-define each training example to be $(x, \{(k_j, y_j)\}_{j=1}^4)$, where each utterance x is paired with 4 different KBs and the denotations of the utterance with respect to these KBs. Then, we maximize $p(\{y_j\}_{j=1}^4 \mid x,)$ by maximizing the objective above, except that $R(z, \{k_j, y_j\}_{j=1}^4) = 1$ iff the denotation of z is correct for **all** four KBs. This dramatically reduces the problem of spuriousness, as the chance of randomly obtaining a correct denotation goes down from $\frac{1}{2}$ to $\frac{1}{16}$. This is reminiscent of Pasupat and Liang (2016), where random permutations of Wikipedia tables were shown to crowdsourcing workers to eliminate spurious programs.

We train the discriminative ranker analogously by maximizing the probability of programs with correct denotation $\sum_{z \in B} p^g_\psi(z \mid x) R(z, k, y)$.

This basic training method fails for CNLVR (see Sec. 6), due to the difficulties of search and spuriousness. Thus, we turn to learning from abstract examples, which substantially reduce these problems.

5 Learning from Abstract Examples

The main premise of this work is that in closed, well-typed domains such as visual reasoning, the main challenge is handling language compositionality, since questions may have a complex and nested structure. Conversely, the problem of mapping lexical items to functions and constants in the programming language can be substantially alleviated by taking advantage of the compact KB schema and typing system, and utilizing a

[2] We used the KBs in CNLVR, for which there are 4 KBs per utterance. When working over pixels there are 24 images per utterance, as 6 images were generated from each KB.

Utterance	Program	Cluster	#
"yellow"	`IsYellow`	`C-Color`	3
"big"	`IsBig`	`C-Size`	3
"square"	`IsSquare`	`C-Shape`	4
"3"	`3`	`C-Num`	2
"exactly"	`EqualInt`	`C-QuantMod`	5
"top"	`Side.Top`	`C-Location`	2
"above"	`GetAbove`	`C-SpaceRel`	6
		Total:	**25**

Table 3: Example mappings from utterance tokens to program tokens for the seven clusters used in the abstract representation. The rightmost column counts the number of mapping in each cluster, resulting in a total of 25 mappings.

small lexicon that maps prevalent lexical items into typed program constants. Thus, if we abstract away from the actual utterance into a partially abstract representation, we can combat the search and spuriousness challenges as we can generalize better across examples in small datasets.

Consider the utterances:

1. *"There are exactly 3 yellow squares touching the wall."*
2. *"There are at least 2 blue circles touching the wall."*

While the surface forms of these utterances are different, at an abstract level they are similar and it would be useful to leverage this similarity.

We therefore define an abstract representation for utterances and logical forms that is suitable for spatial reasoning. We define seven abstract clusters (see Table 3) that correspond to the main semantic types in our domain. Then, we associate each cluster with a small lexicon that contains language-program token pairs associated with this cluster. These mappings represent the canonical ways in which program constants are expressed in natural language. Table 3 shows the seven clusters we use, with an example for an utterance-program token pair from the cluster, and the number of mappings in each cluster. In total, 25 mappings are used to define abstract representations.

As we show next, abstract examples can be used to improve the process of training semantic parsers. Specifically, in sections 5.1-5.3, we use abstract examples in several ways, from generating new training data to improving search accuracy. The combined effect of these approaches is quite dramatic, as our evaluation demonstrates.

5.1 High Coverage via Abstract Examples

We begin by demonstrating that abstraction leads to rather effective coverage of the types of questions asked in a dataset. Namely, that many questions in the data correspond to a small set of abstract examples. We created abstract representations for all 3,163 utterances in the training examples by mapping utterance tokens to their cluster label, and then counted how many distinct abstract utterances exist. We found that as few as 200 abstract utterances cover roughly half of the training examples in the original training set.

The above suggests that knowing how to answer a small set of abstract questions may already yield a reasonable baseline. To test this baseline, we constructured a "rule-based" parser as follows. We manually annotated 106 abstract utterances with their corresponding abstract program (including alignment between abstract tokens in the utterance and program). For example, Table 1 shows the abstract utterance and program for the utterance *"There are exactly 3 yellow squares touching the wall"*. Note that the utterance *"There are at least 2 blue circles touching the wall"* will be mapped to the same abstract utterance and program.

Given this set of manual annotations, our rule-based semantic parser operates as follows: Given an utterance x, create its abstract representation $\bar{x}$. If it exactly matches one of the manually annotated utterances, map it to its corresponding abstract program $\bar{z}$. Replace the abstract program tokens with real program tokens based on the alignment with the utterance tokens, and obtain a final program z. If $\bar{x}$ does not match return TRUE, the majority label. The rule-based parser will fail for examples not covered by the manual annotation. However, it already provides a reasonable baseline (see Table 4). As shown next, manual annotations can also be used for generating new training data.

5.2 Data Augmentation

While the rule-based semantic parser has high precision and gauges the amount of structural variance in the data, it cannot generalize beyond observed examples. However, we can automatically generate non-abstract utterance-program pairs from the manually annotated abstract pairs and train a semantic parser with strong supervision that can potentially generalize better. E.g., consider the utterance *"There are exactly 3 yellow squares touching the wall"*, whose abstract representation is given in Table 1. It is clear that we can use this abstract pair to generate a program for a new utterance *"There are exactly 3 blue squares touching the wall"*. This program will be identical

Algorithm 1 Decoding with an Abstract Cache

```
1:  procedure DECODE($x, y, C, D$)
2:      // C is a map where the key is an abstract utterance
        and the value is a pair $(Z, \hat{R})$ of a list of abstract pro-
        grams $Z$ and their average rewards $\hat{R}$. $D$ is an integer.
3:      $\bar{x} \leftarrow$ Abstract utterance of x
4:      $\mathcal{A} \leftarrow D$ programs in $C[\bar{x}]$ with top reward values
5:      $B_1 \leftarrow$ compute beam of programs of length 1
6:      for $t = 2 \dots T$ do // Decode with cache
7:          $B_t \leftarrow$ construct beam from $B_{t-1}$
8:          $\mathcal{A}_t$ = truncate($\mathcal{A}, t$)
9:          $B_t$.add(de-abstract($\mathcal{A}_t$))
10:     for $z \in B_T$ do //Update cache
11:         Update rewards in $C[\bar{x}]$ using $(\bar{z}, R(z, y))$
12:     return $B_T \cup$ de-abstract($\mathcal{A}$).
```

to the program of the first utterance, with `IsBlue` replacing `IsYellow`.

More generally, we can sample any abstract example and instantiate the abstract clusters that appear in it by sampling pairs of utterance-program tokens for each abstract cluster. Formally, this is equivalent to a synchronous context-free grammar (Chiang, 2005) that has a rule for generating each manually-annotated abstract utterance-program pair, and rules for synchronously generating utterance and program tokens from the seven clusters.

We generated 6,158 (x, z) examples using this method and trained a standard sequence to sequence parser by maximizing $\log p'_\theta(z|x)$ in the model above. Although these are generated from a small set of 106 abstract utterances, they can be used to learn a model with higher coverage and accuracy compared to the rule-based parser, as our evaluation demonstrates.[3]

The resulting parser can be used as a standalone semantic parser. However, it can also be used as an initialization point for the weakly-supervised semantic parser. As we observe in Sec. 6, this results in further improvement in accuracy.

5.3 Caching Abstract Examples

We now describe a caching mechanism that uses abstract examples to combat search and spuriousness when training from weak supervision. As shown in Sec. 5.1, many utterances are identical at the abstract level. Thus, a natural idea is to keep track at training time of abstract utterance-program pairs that resulted in a correct denotation, and use this information to direct the search procedure.

[3]Training a parser directly over the 106 abstract examples results in poor performance due to the small number of examples.

Concretely, we construct a cache C that maps abstract utterances to all abstract programs that were decoded by the model, and tracks the average reward obtained for those programs. For every utterance x, after obtaining the final beam of programs, we add to the cache all abstract utterance-program pairs $(\bar{x}, \bar{z})$, and update their average reward (Alg. 1, line 10). To construct an abstract example $(\bar{x}, \bar{z})$ from an utterance-program pair (x, z) in the beam, we perform the following procedure. First, we create $\bar{x}$ by replacing utterance tokens with their cluster label, as in the rule-based semantic parser. Then, we go over every program token in z, and replace it with an abstract cluster if the utterance contains a token that is mapped to this program token according to the mappings from Table 3. This also provides an alignment from abstract program tokens to abstract utterance tokens that is necessary when utilizing the cache.

We propose two variants for taking advantage of the cache C. Both are shown in Algorithm 1.
1. Full program retrieval (Alg. 1, line 12): Given utterance x, construct an abstract utterance $\bar{x}$, retrieve the top D abstract programs $\mathcal{A}$ from the cache, compute the de-abstracted programs Z using alignments from program tokens to utterance tokens, and add the D programs to the *final* beam.
2. Program prefix retrieval (Alg. 1, line 9)**:** Here, we additionally consider prefixes of abstract programs to the beam, to further guide the search process. At each step t, let B_t be the beam of decoded programs at step t. For every abstract program $\bar{z} \in \mathcal{A}$ add the de-abstracted prefix $z_{1:t}$ to B_t and expand B_{t+1} accordingly. This allows the parser to potentially construct new programs that are not in the cache already. This approach combats both spuriousness and the search challenge, because we add promising program prefixes to the beam that might have fallen off of it earlier. Fig. 3 visualizes the caching mechanism.

A high-level overview of our entire approach for utilizing abstract examples at training time for both data augmentation and model training is given in Fig. 4.

6 Experimental Evaluation

Model and Training Parameters The Bi-LSTM state dimension is 30. The decoder has one hidden layer of dimension 50, that takes the

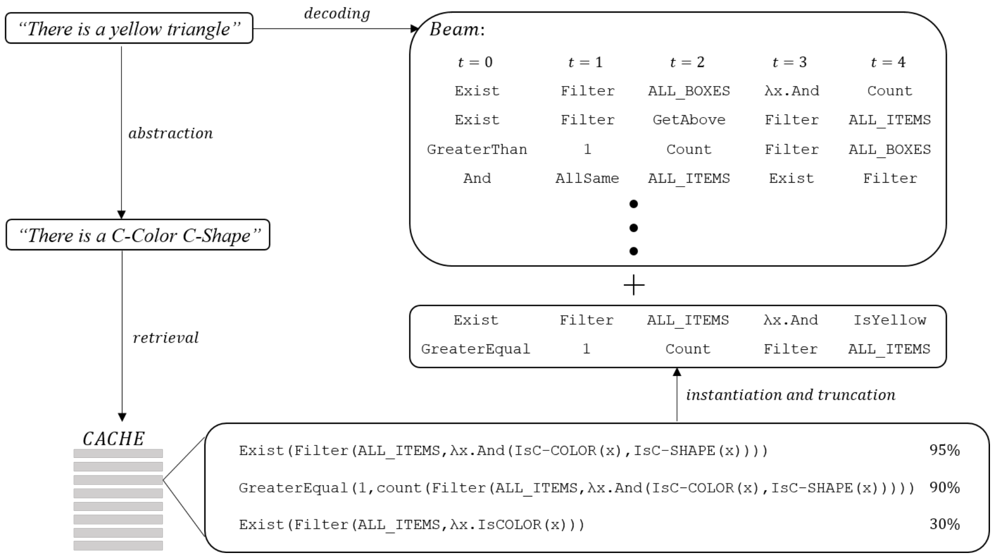

"There is a yellow triangle"
decoding
abstraction
"There is a C-Color C-Shape"
retrieval
CACHE
Beam:
t = 0
t = 1
t = 2
t = 3
t = 4
Exist
Filter
ALL_BOXES
λx.And
Count
Exist
Filter
GetAbove
Filter
ALL_ITEMS
GreaterThan
1
Count
Filter
ALL_BOXES
And
AllSame
ALL_ITEMS
Exist
Filter
Exist
Filter
ALL_ITEMS
λx.And
IsYellow
GreaterEqual
1
Count
Filter
ALL_ITEMS
instantiation and truncation
Exist(Filter(ALL_ITEMS,λx.And(IsC-COLOR(x),IsC-SHAPE(x)))) 95%
GreaterEqual(1,count(Filter(ALL_ITEMS,λx.And(IsC-COLOR(x),IsC-SHAPE(x))))) 90%
Exist(Filter(ALL_ITEMS,λx.IsCOLOR(x))) 30%

Figure 3: A visualization of the caching mechanism. At each decoding step, prefixes of high-reward abstract programs are added to the beam from the cache.

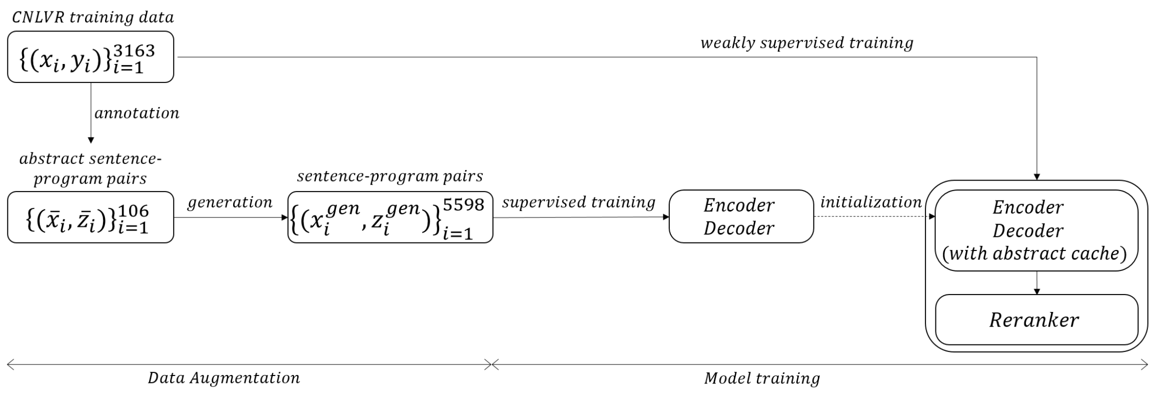

CNLVR training data
$\{(x_i, y_i)\}_{i=1}^{3163}$
weakly supervised training
annotation
abstract sentence-program pairs
$\{(\bar{x}_i, \bar{z}_i)\}_{i=1}^{106}$
generation
sentence-program pairs
$\{(x_i^{gen}, z_i^{gen})\}_{i=1}^{5598}$
supervised training
Encoder Decoder
initialization
Encoder Decoder (with abstract cache)
Reranker
Data Augmentation
Model training

Figure 4: An overview of our approach for utilizing abstract examples for data augmentation and model training.

last 4 decoded tokens as input as well as encoder states. Token embeddings are of dimension 12, beam size is 40 and $D = 10$ programs are used in Algorithm 1. Word embeddings are initialized from CBOW (Mikolov et al., 2013) trained on the training data, and are then optimized end-to-end. In the weakly-supervised parser we encourage exploration with meritocratic gradient updates with $\beta = 0.5$ (Guu et al., 2017). In the weakly-supervised parser we warm-start the parameters with the supervised parser, as mentioned above. For optimization, Adam is used (Kingma and Ba, 2014)), with learning rate of 0.001, and mini-batch size of 8.

Pre-processing Because the number of utterances is relatively small for training a neural model, we take the following steps to reduce sparsity. We lowercase all utterance tokens, and also use their lemmatized form. We also use spelling correction to replace words that contain typos. After pre-processing we replace every word that occurs less than 5 times with an UNK symbol.

Evaluation We evaluate on the public development and test sets of CNLVR as well as on the hidden test set. The standard evaluation metric is accuracy, i.e., how many examples are correctly classified. In addition, we report *consistency*, which is the proportion of utterances for which the decoded program has the correct denotation for all 4 images/KBs. It captures whether a model consistently produces a correct answer.

Baselines We compare our models to the MAJORITY baseline that picks the majority class (TRUE in our case). We also compare to the state-of-the-art model reported by Suhr et al. (2017)

Model	Dev. Acc.	Dev. Con.	Test-P Acc.	Test-P Con.	Test-H Acc.	Test-H Con.
MAJORITY	55.3	-	56.2	-	55.4	-
MAXENT	68.0	-	67.7	-	67.8	-
RULE	66.0	29.2	66.3	32.7	-	-
SUP.	67.7	36.7	66.9	38.3	-	-
SUP.+DISC	77.7	52.4	76.6	51.8	-	-
WEAKSUP.	84.3	66.3	81.7	60.1	-	-
W.+DISC	**85.7**	**67.4**	**84.0**	**65.0**	**82.5**	**63.9**

Table 4: Results on the development, public test (Test-P) and hidden test (Test-H) sets. For each model, we report both accuracy and consistency.

Model	Dev. Acc.	Dev. Con.
RANDOMER	53.2	7.1
−ABSTRACTION	58.2	17.6
−DATAAUGMENTATION	71.4	41.2
−BEAMCACHE	77.2	56.1
−EVERYSTEPBEAMCACHE	82.3	62.2
ONEEXAMPLEREWARD	58.2	11.2

Table 5: Results of ablations of our main models on the development set. Explanation for the nature of the models is in the body of the paper.

when taking the KB as input, which is a maximum entropy classifier (MAXENT). For our models, we evaluate the following variants of our approach:

- RULE: The rule-based parser from Sec. 5.1.
- SUP.: The supervised semantic parser trained on augmented data as in Sec. 5.2 ($5,598$ examples for training and 560 for validation).
- WEAKSUP.: Our full weakly-supervised semantic parser that uses abstract examples.
- +DISC: We add a discriminative re-ranker (Sec. 3) for both SUP. and WEAKSUP.

Main results Table 4 describes our main results. Our weakly-supervised semantic parser with re-ranking (W.+DISC) obtains 84.0 accuracy and 65.0 consistency on the public test set and 82.5 accuracy and 63.9 on the hidden one, improving accuracy by 14.7 points compared to state-of-the-art. The accuracy of the rule-based parser (RULE) is less than 2 points below MAXENT, showing that a semantic parsing approach is very suitable for this task. The supervised parser obtains better performance (especially in consistency), and with re-ranking reaches 76.6 accuracy, showing that generalizing from generated examples is better than memorizing manually-defined patterns. Our weakly-supervised parser significantly improves over SUP., reaching an accuracy of 81.7 before re-ranking, and 84.0 after re-ranking (on the public test set). Consistency results show an even crisper trend of improvement across the models.

6.1 Analysis

We analyze our results by running multiple ablations of our best model W.+DISC on the development set.

To examine the overall impact of our procedure, we trained a weakly-supervised parser from scratch without pre-training a supervised parser nor using a cache, which amounts to a re-implementation of the RANDOMER algorithm (Guu et al., 2017). We find that the algorithm is unable to bootstrap in this challenging setup and obtains very low performance. Next, we examined the importance of abstract examples, by pre-training only on examples that were manually annotated (utterances that match the 106 abstract patterns), but with no data augmentation or use of a cache (−ABSTRACTION). This results in performance that is similar to the MAJORITY baseline.

To further examine the importance of abstraction, we decoupled the two contributions, training once with a cache but without data augmentation for pre-training (−DATAAUGMENTATION), and again with pre-training over the augmented data, but without the cache (−BEAMCACHE). We found that the former improves by a few points over the MAXENT baseline, and the latter performs comparably to the supervised parser, that is, we are still unable to improve learning by training from denotations.

Lastly, we use a beam cache without line 9 in Alg. 1 (−EVERYSTEPBEAMCACHE). This already results in good performance, substantially higher than SUP. but is still 3.4 points worse than our best performing model on the development set.

Orthogonally, to analyze the importance of tying the reward of all four examples that share an utterance, we trained a model without this tying, where the reward is 1 iff the denotation is correct (ONEEXAMPLEREWARD). We find that spuriousness becomes a major issue and weakly-supervised learning fails.

Error Analysis We sampled 50 consistent and 50 inconsistent programs from the development set to analyze the weaknesses of our model. By and large, errors correspond to utterances that are more complex syntactically and semantically. In about half of the errors an object was described by two or more modifying clauses: *"there is a box with a yellow circle and three blue items"*; or nesting occurred: *"one of the gray boxes has exactly* *three objects one of which is a circle"*. In these cases the model either ignored one of the conditions, resulting in a program equivalent to *"there is a box with three blue items"* for the first case, or applied composition operators wrongly, outputting an equivalent to *"one of the gray boxes has exactly three circles"* for the second case. However, in some cases the parser succeeds on such examples and we found that 12% of the sampled utterances that were parsed correctly had a similar complex structure. Other, less frequent reasons for failure were problems with cardinality interpretation, i.e. ,*"there are 2"* parsed as *"exactly 2"* instead of *"at least 2"*; applying conditions to items rather than sets, e.g., *"there are 2 boxes with a triangle closely touching a corner"* parsed as *"there are 2 triangles closely touching a corner"*; and utterances with questionable phrasing, e.g., *"there is a tower that has three the same blocks color"*.

Other insights are that the algorithm tended to give higher probability to the top ranked program when it is correct (average probability 0.18), compared to cases when it is incorrect (average probability 0.08), indicating that probabilities are correlated with confidence. In addition, sentence length is not predictive for whether the model will succeed: average sentence length of an utterance is 10.9 when the model is correct, and 11.1 when it errs.

We also note that the model was successful with sentences that deal with spatial relations, but struggled with sentences that refer to the size of shapes. This is due to the data distribution, which includes many examples of the former case and fewer examples of the latter.

7 Related Work

Training semantic parsers from denotations has been one of the most popular training schemes for scaling semantic parsers since the beginning of the decade. Early work focused on traditional log-linear models (Clarke et al., 2010; Liang et al., 2011; Kwiatkowski et al., 2013), but recently denotations have been used to train neural semantic parsers (Liang et al., 2017; Krishnamurthy et al., 2017; Rabinovich et al., 2017; Cheng et al., 2017).

Visual reasoning has attracted considerable attention, with datasets such as VQA (Antol et al., 2015) and CLEVR (Johnson et al., 2017a). The advantage of CNLVR is that language utterances are both natural and compositional. Treating visual reasoning as an end-to-end semantic parsing problem has been previously done on CLEVR (Hu et al., 2017; Johnson et al., 2017b).

Our method for generating training data resembles data re-combination ideas in Jia and Liang (2016), where examples are generated automatically by replacing entities with their categories.

While spuriousness is central to semantic parsing when denotations are not very informative, there has been relatively little work on explicitly tackling it. Pasupat and Liang (2015) used manual rules to prune unlikely programs on the WIKITABLEQUESTIONS dataset, and then later utilized crowdsourcing (Pasupat and Liang, 2016) to eliminate spurious programs. Guu et al. (2017) proposed RANDOMER, a method for increasing exploration and handling spuriousness by adding randomness to beam search and a proposing a "meritocratic" weighting scheme for gradients. In our work we found that random exploration during beam search did not improve results while meritocratic updates slightly improved performance.

8 Discussion

In this work we presented the first semantic parser for the CNLVR dataset, taking structured representations as input. Our main insight is that in closed, well-typed domains we can generate abstract examples that can help combat the difficulties of training a parser from delayed supervision. First, we use abstract examples to semi-automatically generate utterance-program pairs that help warm-start our parameters, thereby reducing the difficult search challenge of finding correct programs with random parameters. Second, we focus on an abstract representation of examples, which allows us to tackle spuriousness and alleviate search, by sharing information about promising programs between different examples. Our approach dramatically improves performance on CNLVR, establishing a new state-of-the-art.

In this paper, we used a manually-built high-precision lexicon to construct abstract examples. This is suitable for well-typed domains, which are ubiquitous in the virtual assistant use case. In future work we plan to extend this work and automatically learn such a lexicon. This can reduce manual effort and scale to larger domains where there is substantial variability on the language side.

References

D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042* .

S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. 2015. Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*. pages 2425–2433.

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)* 1:49–62.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*.

J. Cheng, S. Reddy, V. Saraswat, and M. Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Association for Computational Linguistics (ACL)*.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Association for Computational Linguistics (ACL)*. pages 263–270.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*. pages 18–27.

K. Guu, P. Pasupat, E. Z. Liu, and P. Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

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

R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *International Conference on Computer Vision (ICCV)*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. 2017a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR)*.

J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick. 2017b. Inferring and executing programs for visual reasoning. In *International Conference on Computer Vision (ICCV)*.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 1062–1068.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

J. Krishnamurthy, P. Dasigi, and M. Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 754–765.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling data. In *International Conference on Machine Learning (ICML)*. pages 282–289.

C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Association for Computational Linguistics (ACL)*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*. pages 590–599.

T. Mikolov, K. Chen, G. Corrado, and Jeffrey. 2013. Efficient estimation of word representations in vector space. *arXiv* .

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

P. Pasupat and P. Liang. 2016. Inferring logical forms from denotations. In *Association for Computational Linguistics (ACL)*.

M. Rabinovich, M. Stern, and D. Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Association for Computational Linguistics (ACL)*.

A. Suhr, M. Lewis, J. Yeh, and Y. Artzi. 2017. A corpus of natural language for visual reasoning. In *Association for Computational Linguistics (ACL)*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3104–3112.

C. Xiao, M. Dymetman, and C. Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Association for Computational Linguistics (ACL)*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*. pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 678–687.

Improving a Neural Semantic Parser by Counterfactual Learning from Human Bandit Feedback

Carolin Lawrence
Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
Lawrence@cl.uni-heidelberg.de

Stefan Riezler
Computational Linguistics & IWR
Heidelberg University
69120 Heidelberg, Germany
riezler@cl.uni-heidelberg.de

Abstract

Counterfactual learning from human bandit feedback describes a scenario where user feedback on the quality of outputs of a historic system is logged and used to improve a target system. We show how to apply this learning framework to neural semantic parsing. From a machine learning perspective, the key challenge lies in a proper reweighting of the estimator so as to avoid known degeneracies in counterfactual learning, while still being applicable to stochastic gradient optimization. To conduct experiments with human users, we devise an easy-to-use interface to collect human feedback on semantic parses. Our work is the first to show that semantic parsers can be improved significantly by counterfactual learning from logged human feedback data.

1 Introduction

In semantic parsing, natural language utterances are mapped to machine readable parses which are complex and often tailored specifically to the underlying task. The cost and difficulty of manually preparing large amounts of such parses thus is a bottleneck for supervised learning in semantic parsing. Recent work (Liang et al. (2017); Mou et al. (2017); Peng et al. (2017); *inter alia*) has applied reinforcement learning to address the annotation bottleneck as follows: Given a question, the existence of a corresponding gold answer is assumed. A semantic parser produces multiple parses per question and corresponding answers are obtained. These answers are then compared against the gold answer and a positive reward is recorded if there is an overlap. The parser is then guided towards correct parses using the REINFORCE algorithm (Williams, 1992) which scales the gradient for the various parses by their obtained reward (see the left half of Figure 1). However, learning from question-answer pairs is only efficient if gold answers are cheap to obtain. For complex open-domain question-answering tasks, correct answers are not unique factoids, but open-ended lists, counts in large ranges, or fuzzily defined objects. For example, geographical queries against databases such as OpenStreetMap (OSM) can involve fuzzy operators such as "near" or "in walking distance" and thus need to allow for fuzziness in the answers as well. A possible solution lies in machine learning from even weaker supervision signals in form of human bandit feedback[1] where the semantic parsing system suggests exactly one parse for which feedback is collected from a human user. In this setup neither gold parse nor gold answer are known and feedback is obtained for only one system output per question.

The goal of our paper is to exploit this scenario of learning from human bandit feedback to train semantic parsers. This learning scenario perfectly fits commercial setups such as virtual personal assistants that embed a semantic parser. Commercial systems can easily log large amounts of interaction data between users and system. Once sufficient data has been collected, the log can then be used to improve the parser. This leads to a counterfactual learning scenario (Bottou et al., 2013) where we have to solve the counterfactual problem of how to improve a target system from logged feedback that was given to the outputs of a different historic system (see the right half of Figure 1).

In order to achieve our goal of counterfactual learning of semantic parsers from human bandit feedback, the following contributions are required:

[1]The term "bandit feedback" is inspired by the scenario of maximizing the reward for a sequence of pulls of arms of "one-armed bandit" slot machines.

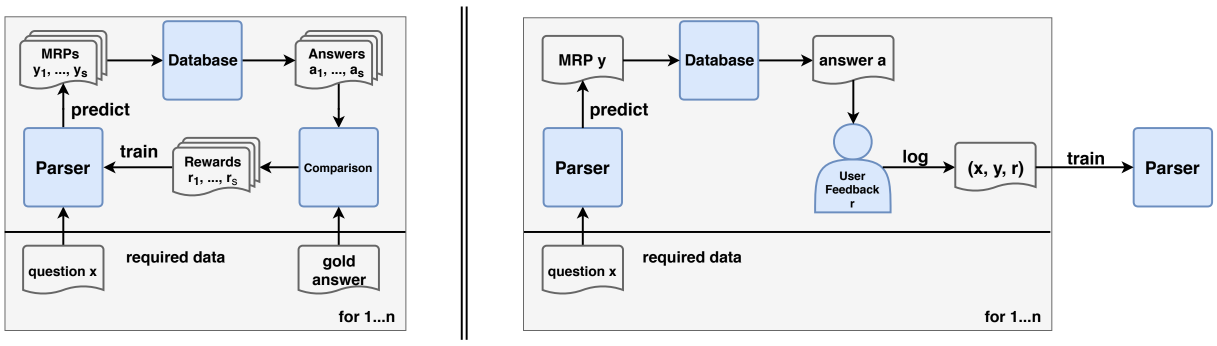


Figure 1: Left: Online reinforcement learning setup for semantic parsing setup where both questions and gold answers are available. The parser attempts to find correct machine readable parses (MRPs) by producing multiple parses, obtaining corresponding answers, and comparing them against the gold answer. Right: In our setup, a question does not have an associated gold answer. The parser outputs a single MRP and the corresponding answer is shown to a user who provides some feedback. Such triplets are collected in a log which can be used for offline training of a semantic parser. This scenario is called counterfactual since the feedback was logged for outputs from a system different from the target system to be optimized.

First, we need to construct an easy-to-use user interface that allows to collect feedback based on the parse rather than the answer. To this aim, we automatically convert the parse to a set of statements that can be judged as correct or incorrect by a human. This approach allows us to assign rewards at the token level, which in turn enables us to perform blame assignment in bandit learning and to learn from partially correct queries where tokens are reinforced individually. We show that users can provide such feedback for one question-parse pair in 16.4 seconds on average. This exemplifies that our approach is more efficient and cheaper than recruiting experts to annotate parses or asking workers to compile large answer sets.

Next, we demonstrate experimentally that counterfactual learning can be applied to neural sequence-to-sequence learning for semantic parsing. A baseline neural semantic parser is trained in fully supervised fashion, human bandit feedback from human users is collected in a log and subsequently used to improve the parser. The resulting parser significantly outperforms the baseline model as well as a simple bandit-to-supervised approach (B2S) where the subset of completely correct parses is treated as a supervised dataset. Finally, we repeat our experiments on a larger but simulated log to show that our gains can scale: the baseline system is improved by 7.45% in answer F1 score without ever seeing a gold standard parse.

Lastly, from a machine learning perspective, we have to solve problems of degenerate behavior in counterfactual learning by lifting the multiplicative control variate technique (Swaminathan and Joachims, 2015b; Lawrence et al., 2017b,a) to stochastic learning for neural models. This is done by reweighting target model probabilities over the logged data under a one-step-late model that decouples the normalization from gradient estimation and is thus applicable in stochastic (minibatch) gradient optimization.

2 Related Work

Semantic parsers have been successfully trained using neural sequence-to-sequence models with a cross-entropy objective and question-parse pairs (Jia and Liang, 2016; Dong and Lapata, 2016)) or question-answer pairs (Neelakantan et al., 2017). Improving semantic parsers using weak feedback has previously been studied (Goldwasser and Roth (2013); Artzi and Zettlemoyer (2013); *inter alia*). More recently, several works have applied policy gradient techniques such as REINFORCE (Williams, 1992) to train neural semantic parsers (Liang et al. (2017); Mou et al. (2017); Peng et al. (2017); *inter alia*). However, they assume the existence of the true target answers that can be used to obtain a reward for any number of output queries suggested by the system. It thus differs from a bandit setup where we assume that a reward is available for only one structure.

Our work most closely resembles the work of

Iyer et al. (2017) who do make the assumption of only being able to judge one output. They improve their parser using simulated and real user feedback. Parses with negative feedback are given to experts to obtain the correct parse. Corrected queries and queries with positive feedback are added to the training corpus and learning continues with a cross-entropy objective. We show that this bandit-to-supervision approach can be outperformed by offline bandit learning from partially correct queries. Yih et al. (2016) proposed a user interface for the Freebase database that enables a fast and easy creation of parses. However, in their setup the worker still requires expert knowledge about the Freebase database, whereas in our approach feedback can be collected freely and from any user interacting with the system.

From a machine learning perspective, related work can be found in the areas of counterfactual bandit learning (Dudik et al., 2011; Swaminathan and Joachims, 2015a), or equivalently, off-policy reinforcement learning (Precup et al., 2000; Jiang and Li, 2016). Our contribution is to modify the self-normalizing estimator (Kong, 1992; Precup et al., 2000; Swaminathan and Joachims, 2015b; Joachims et al., 2018) to be applicable to neural networks. Our work is similar to the counterfactual learning setup for machine translation introduced by Lawrence et al. (2017b). Following their insight, we also assume the logs were created deterministically, i.e. the logging policy always outputs the most likely sequence. Their framework was applied to statistical machine translation using linear models. We show how to generalize their framework to neural models and how to apply it to the task of neural semantic parsing in the OSM domain.

3 Neural Semantic Parsing

Our semantic parsing model is a state-of-the-art sequence-to-sequence neural network using an encoder-decoder setup (Cho et al., 2014; Sutskever et al., 2014) together with an attention mechanism (Bahdanau et al., 2015). We use the settings of Sennrich et al. (2017), where an input sequence $x = x_1, x_2, \dots x_{|x|}$ (a natural language question) is encoded by a Recurrent Neural Network (RNN), each input token has an associated hidden vector $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ where the former is created by a forward pass over the input, and the latter by a backward pass. $\overrightarrow{h}_i$ is obtained by recursively computing $f(x_i, \overrightarrow{h}_{i-1})$ where f is a Gated Recurrent Unit (GRU) (Chung et al., 2014), and $\overleftarrow{h}_i$ is computed analogously. The input sequence is reduced to a single vector $c = g(\{h_1, \dots, h_{|x|}\})$ which serves as the initialization of the decoder RNN. g calculates the average over all vectors h_i. At each time step t the decoder state is set by $s_t = q(s_{t-1}, y_{t-1}, c_t)$. q is a conditional GRU with an attention mechanism and c_t is the context vector computed by the attention mechanism. Given an output vocabulary $\mathcal{V}_y$ and the decoder state $s_t = \{s_1, \dots, s_{|\mathcal{V}_y|}\}$, a softmax output layer defines a probability distribution over $\mathcal{V}_y$ and the probability for a token y_j is:

$$\pi_w(y_j = t_o | y_{<j}, x) = \frac{\exp(s_{t_o})}{\sum_{v=1}^{|\mathcal{V}_y|} \exp(s_{t_v})}. \quad (1)$$

The model π_w can be seen as parameterized policy over an action space defined by the target language vocabulary. The probability for a full output sequence $y = y_1, y_2, \dots y_{|y|}$ is defined by

$$\pi_w(y|x) = \prod_{j=1}^{|y|} \pi_w(y_j | y_{<j}, x). \quad (2)$$

In our case, output sequences are linearized machine readable parses, called queries in the following. Given supervised data $\mathcal{D}_{sup} = \{(x_t, \bar{y}_t)\}_{t=1}^{n}$ of question-query pairs, where $\bar{y}_t$ is the true target query for x_t, the neural network can be trained using SGD and a cross-entropy (CE) objective:

$$\mathcal{L}_{CE} = -\frac{1}{n} \sum_{t=1}^{n} \sum_{j=1}^{|\bar{y}|} \log \pi_w(\bar{y}_j | \bar{y}_{<j}, x). \quad (3)$$

4 Counterfactual Learning from Deterministic Bandit Logs

Counterfactual Learning Objectives. We assume a policy π_w that, given an input $x \in \mathcal{X}$, defines a conditional probability distribution over possible outputs $y \in \mathcal{Y}(x)$. Furthermore, we assume that the policy is parameterized by w and its gradient can be derived. In this work, π_w is defined by the sequence-to-sequence model described in Section 3. We also assume that the model decomposes over individual output tokens, i.e. that the model produces the output token by token.

$\nabla_w \hat{R}_{\text{DPM}} = \frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t\|x_t)\nabla_w \log \pi_w(y_t\|x_t)$.
$\nabla_w \hat{R}_{\text{DPM+R}} = \frac{1}{n}\sum_{t=1}^{n}[\delta_t \bar{\pi}_w(y_t\|x_t)(\nabla_w \log \pi_w(y_t\|x_t) - \frac{1}{n}\sum_{u=1}^{n} \bar{\pi}_w(y_u\|x_u)\nabla \log \pi_w(y_u\|x_u))]$.
$\nabla_w \hat{R}_{\text{DPM+OSL}} = \frac{1}{m}\sum_{t=1}^{m} \delta_t \bar{\pi}_{w,w'}(y_t\|x_t)\nabla_w \log \pi_w(y_t\|x_t)$.
$\nabla_w \hat{R}_{\text{DPM+T}} = \frac{1}{n}\sum_{t=1}^{n} \prod_{j=1}^{\|y\|} \delta_j \pi_w(y_j\|x_t) \sum_{j=1}^{\|y\|} \nabla_w \log \pi_w(y_j\|x_t)$.
$\nabla_w \hat{R}_{\text{DPM+T+OSL}} = \frac{1}{m}\sum_{t=1}^{m} \prod_{j=1}^{\|y\|} \delta_j \bar{\pi}_{w,w'}(y_t\|x_t) \sum_{j=1}^{\|y\|} \nabla_w \log \pi_w(y_j\|x_t)$.

Table 1: Gradients of counterfactual objectives.

The counterfactual learning problem can be described as follows: We are given a data log of triples $\mathcal{D}_{log} = \{(x_t, y_t, \delta_t)\}_{t=1}^{n}$ where outputs y_t for inputs x_t were generated by a logging system under policy π_0, and loss values $\delta_t \in [-1, 0]$[2] were observed for the generated data points. Our goal is to optimize the expected reward (in our case: minimize the expected risk) for a target policy π_w given the data log $\mathcal{D}_{log}$. In case of deterministic logging, outputs are logged with propensity $\pi_0(y_t|x_t) = 1, \ t = 1, \ldots, n$. This results in a *deterministic propensity matching (DPM)* objective (Lawrence et al., 2017b), without the possibility to correct the sampling bias of the logging policy by inverse propensity scoring (Rosenbaum and Rubin, 1983):

$$\hat{R}_{\text{DPM}}(\pi_w) = \frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t|x_t). \qquad (4)$$

This objective can show degenerate behavior in that it overfits to the choices of the logging policy (Swaminathan and Joachims, 2015b; Lawrence et al., 2017a). This degenerate behavior can be avoided by reweighting using a multiplicative control variate (Kong, 1992; Precup et al., 2000; Jiang and Li, 2016; Thomas and Brunskill, 2016). The new objective is called the *reweighted deterministic propensity matching (DPM+R)* objective in Lawrence et al. (2017b):

$$\begin{aligned}\hat{R}_{\text{DPM+R}}(\pi_w) &= \frac{1}{n}\sum_{t=1}^{n} \delta_t \bar{\pi}_w(y_t|x_t) \qquad (5)\\ &= \frac{\frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t|x_t)}{\frac{1}{n}\sum_{t=1}^{n} \pi_w(y_t|x_t)}.\end{aligned}$$

Algorithms for optimizing the discussed objectives can be derived as gradient descent algorithms where gradients using the score function gradient estimator (Fu, 2006) are shown in Table 1.

[2]We use the terms loss and (negative) rewards interchangeably, depending on context.

Reweighting in Stochastic Learning. As shown in Swaminathan and Joachims (2015b) and Lawrence et al. (2017a), reweighting over the entire data log $\mathcal{D}_{log}$ is crucial since it avoids that high loss outputs in the log take away probability mass from low loss outputs. This multiplicative control variate has the additional effect of reducing the variance of the estimator, at the cost of introducing a bias of order $O(\frac{1}{n})$ that decreases as n increases (Kong, 1992). The desirable properties of this control variate cannot be realized in a stochastic (minibatch) learning setup since minibatch sizes large enough to retain the desirable reweighting properties are infeasible for large neural networks.

We offer a simple solution to this problem that nonetheless retains all desired properties of the reweighting. The idea is inspired by one-step-late algorithms that have been introduced for EM algorithms (Green, 1990). In the EM case, dependencies in objectives are decoupled by evaluating certain terms under parameter settings from previous iterations (thus: one-step-late) in order to achieve closed-form solutions. In our case, we decouple the reweighting from the parameterization of the objective by evaluating the reweighting under parameters w' from some previous iteration. This allows us to perform gradient descent updates and reweighting asynchronously. Updates are performed using minibatches, however, reweighting is based on the entire log, allowing us to retain the desirable properties of the control variate.

The new objective, called *one-step-late reweighted DPM* objective (DPM+OSL), optimizes $\pi_{w,w'}$ with respect to w for a minibatch of size m, with reweighting over the entire log of size n under parameters w':

$$\hat{R}_{\text{DPM+OSL}}(\pi_w) = \frac{1}{m}\sum_{t=1}^{m}\delta_t\bar{\pi}_{w,w'}(y_t|x_t) \quad (6)$$
$$= \frac{\frac{1}{m}\sum_{t=1}^{m}\delta_t\pi_w(y_t|x_t)}{\frac{1}{n}\sum_{t=1}^{n}\pi_{w'}(y_t|x_t)}.$$

If the renormalization is updated periodically, e.g. after every validation step, renormalizations under w or w' are not much different and will not hamper convergence. Despite losing the formal justification from the perspective of control variates, we found empirically that the OSL update schedule for reweighting is sufficient and does not deteriorate performance. The gradient for learning with OSL updates is given in Table 1.

Token-Level Rewards. For our application of counterfactual learning to human bandit feedback, we found another deviation from standard counterfactual learning to be helpful: For humans, it is hard to assign a graded reward to a query at a sequence level because either the query is correct or it is not. In particular, with a sequence level reward of 0 for incorrect queries, we do not know which part of the query is wrong and which parts might be correct. Assigning rewards at token-level will ease the feedback task and allow the semantic parser to learn from partially correct queries. Thus, assuming the underlying policy can decompose over tokens, a token level (DPM+T) reward objective can be defined:

$$\hat{R}_{\text{DPM+T}}(\pi_w) = \frac{1}{n}\sum_{t=1}^{n}\left(\prod_{j=1}^{|y|}\delta_j\pi_w(y_j|x_t)\right). \quad (7)$$

Analogously, we can define an objective that combines the token-level rewards and the minibatched reweighting (DPM+T+OSL):

$$\hat{R}_{\text{DPM+T+OSL}}(\pi_w) = \frac{\frac{1}{m}\sum_{t=1}^{m}\left(\prod_{j=1}^{|y|}\delta_j\pi_w(y_j|x_t)\right)}{\frac{1}{n}\sum_{t=1}^{n}\pi_{w'}(y_t|x_t)}. \quad (8)$$

Gradients for the DPM+T and DPM+T+OSL objectives are given in Table 1.

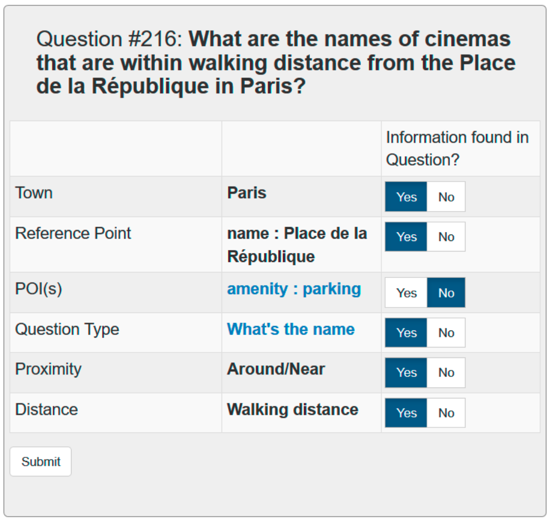
5 Semantic Parsing in the OpenStreetMap Domain

OpenStreetMap (OSM) is a geographical database in which volunteers annotate points of interests in the world. A point of interest consists of one or more associated GPS points. Further relevant information may be added at the discretion of the volunteer in the form of tags. Each tag consists of a key and an associated value, for example "*tourism : hotel*". The NLMAPS corpus was introduced by Haas and Riezler (2016) as a basis to create a natural language interface to the OSM database. It pairs English questions with machine readable parses, i.e. queries that can be executed against OSM.

Human Feedback Collection. The task of creating a natural language interface for OSM demonstrates typical difficulties that make it expensive to collect supervised data. The machine readable language of the queries is based on the OVERPASS query language which was specifically designed for the OSM database. It is thus not easily possible to find experts that could provide correct queries. It is equally difficult to ask workers at crowdsourcing platforms for the correct answer. For many questions, the answer set is too large to expect a worker to count or list them all in a reasonable amount of time and without errors. For example, for the question "*How many hotels are there in Paris?*" there are 951 hotels annotated in the OSM database. Instead we propose to automatically transform the query into a block of statements that can easily be judged as correct or incorrect by a human. The question and the created block of statements are embedded in a user interface with a form that can be filled out by users. Each statement is accompanied by a set of radio buttons where a user can select either "*Yes*" or "*No*". For a screenshot of the interface and an example see Figure 2.

In total there are 8 different types of statements. The presence of certain tokens in a query trigger different statement types. For example, the token "*area*" triggers the statement type "*Town*". The statement is then populated with the corresponding information from the query. In the case of "*area*", the following OSM value is used, e.g. "*Paris*". With this, the meaning of every query can be captured by a set of human-understandable statements. For a full overview of all statement types and their triggers see section B of the supplementary material.

OSM tags and keys are generally understandable. For example, the correct OSM tag for "*hotels*" is "*tourism : hotel*" and when searching for



Question #216: **What are the names of cinemas that are within walking distance from the Place de la République in Paris?**

		Information found in Question?
Town	**Paris**	[✓ Yes] No
Reference Point	**name : Place de la République**	[✓ Yes] No
POI(s)	**amenity : parking**	Yes [✓ No]
Question Type	**What's the name**	[✓ Yes] No
Proximity	**Around/Near**	[✓ Yes] No
Distance	**Walking distance**	[✓ Yes] No

Figure 2: The user interface for collecting feedback from humans with an example question and a correctly filled out form.

websites, the correct question type key would be "*website*". Nevertheless, for each OSM tag or key, we automatically search for the corresponding Wikipedia page on the OpenStreetMap Wiki[3] and extract the description for this tag or key. The description is made available to the user in form of a tool-tip that appears when hovering over the tag or key with the mouse. If a user is unsure if a OSM tag or key is correct, they can read this description to help in their decision making. Once the form is submitted, a script maps each statement back to the corresponding tokens in the original query. These tokens then receive negative or positive feedback based on the feedback the user provided for that statement.

Corpus Extension. Similar to the extension of the NLMAPS corpus by Lawrence and Riezler (2016) who include shortened questions which are more typically used by humans in search tasks, we present an automatic extension that allows a larger coverage of common OSM tags.[4] The basis for the extension is a hand-written, online freely available list[5] that links natural language expressions such as "*cash machine*" to appropriate OSM tags, in this case "*amenity : atm*". Using the list, we generate for each unique expression-tag pair a set of question-query pairs. These latter pairs contain several placeholders which will be filled automatically in a second step.

To fill the area placeholder `$LOC`, we sample from a list of 30 cities from France, Germany and the UK. `$POI` is the placeholder for a point of interest. We sample it from the list of objects which are located in the prior sampled city and which have a *name* key. The corresponding value belonging to the *name* key will be used to fill this spot. The placeholder `$QTYPE` is filled by uniformly sampling from the four primary question types available in the NLMAPS query language. On the natural language side they corresponded to "*How many*", "*Where*", "*Is there*" and `$KEY`. `$KEY` is a further parameter belonging to the primary question operator FINDKEY. It can be filled by any OSM key, such as *name*, *website* or *height*. To ensure that there will be an answer for the generated query, we first ran a query with the current tag ("*amenity : atm*") to find all objects fulfilling this requirement in the area of the already sampled city. From the list of returned objects and the keys that appear in association with them, we uniformly sampled a key. For `$DIST` we chose between the pre-defined options for walking distance and within city distance. The expressions map to corresponding values which define the size of a radius in which objects of interest (with tag "*amenity : atm*") will be located. If the walking distance was selected, we added "*in walking distance*" to the question. Otherwise no extra text was added to the question, assuming the within city distance to be the default. This sampling process was repeated twice.

Table 2 presents the corpus statistics, comparing NLMAPS to our extension. The automatic

	NLMAPS	NLMAPS V2
# question-query pairs	2,380	28,609
tokens	25,906	202,088
types	1,002	8,710
avg. sent. length	10.88	7.06
distinct tags	477	6,582

Table 2: Corpus statistics of the question-answering corpora NLMAPS and our extension NLMAPS V2 which additionally contains the search engine style queries (Lawrence and Riezler, 2016) and the automatic extensions of the most common OSM tags.

[3]https://wiki.openstreetmap.org/

[4]The extended dataset, called NLMAPS V2, will be released upon acceptance of the paper.

[5]http://wiki.openstreetmap.org/wiki/Nominatim/Special_Phrases/EN

extension, obviating the need for expensive manual work, allows a vast increase of question-query pairs by an order of magnitude. Consequently the number of tokens and types increase in a similar vein. However, the average sentence length drops. This comes as no surprise due to the nature of the rather simple hand-written list which contains never more than one tag for an element, resulting in simpler question structures. However, the main idea of utilizing this list is to extend the coverage to previously unknown OSM tags. With 6,582 distinct tags compared to the previous 477, this was clearly successful. Together with the still complex sentences from the original corpus, a semantic parser is now able to learn both complex questions and a large variety of tags. An experiment that empirically validates the usefulness of the automatically created data can be found in the supplementary material, section A.

6 Experiments

General Settings. In our experiments we use the sequence-to-sequence neural network package NEMATUS (Sennrich et al., 2017). Following the method used by Haas and Riezler (2016), we split the queries into individual tokens by taking a pre-order traversal of the original tree-like structure. For example, "*query(west(area(keyval('name','Paris')), nwr(keyval('railway','station'))),qtype(count))*" becomes "*query@2 west@2 area@1 keyval@2 name@0 Paris@s nwr@1 keyval@2 railway@0 station@s qtype@1 count@0*".

The SGD optimizer used is ADADELTA (Zeiler, 2012). The model employs 1,024 hidden units and word embeddings of size 1,000. The maximum sentence length is 200 and gradients are clipped if they exceed a value of 1.0. The stopping point is determined by validation on the development set and selecting the point at which the highest evaluation score is obtained. F1 validation is run after every 100 updates, and each update is made on the basis of a minibatch of size 80.

The evaluation of all models is based on the answers obtained by executing the most likely query obtained after a beam search with a beam of size 12. We report the F1 score which is the harmonic mean of precision and recall. Recall is defined as the percentage of fully correct answers divided by the set size. Precision is the percentage of correct answers out of the set of answers with non-empty strings. Statistical significance between models is measured using an approximate randomization test (Noreen, 1989).

Baseline Parser & Log Creation. Our experiment design assumes a baseline neural semantic parser that is trained in fully supervised fashion, and is to be improved by bandit feedback obtained for system outputs from the baseline system for given questions. For this purpose, we select 2,000 question-query pairs randomly from the full extended NLMAPS V2 corpus. We will call this dataset $\mathcal{D}_{sup}$. Using this dataset, a baseline semantic parser is trained in supervised fashion under a cross-entropy objective. It obtains an F1 score of 57.45% and serves as the logging policy π_0.

Furthermore we randomly split off 1,843 and 2,000 pairs for a development and test set, respectively. This leaves a set of 22,765 question-query pairs. The questions can be used as input and bandit feedback can be collected for the most likely output of the semantic parser. We refer to this dataset as $\mathcal{D}_{log}$.

To collect human feedback, we take the first 1,000 questions from $\mathcal{D}_{log}$ and use π_0 to parse these questions to obtain one output query for each. 5 question-query pairs are discarded because the suggested query is invalid. For the remaining question-query pairs, the queries are each transformed into a block of human-understandable statements and embedded into the user interface described in Section 5. We recruited 9 users to provide feedback for these question-query pairs. The resulting log is referred to as $\mathcal{D}_{human}$. Every question-query pair is purposely evaluated only once to mimic a realistic real-world scenario where user logs are collected as users use the system. In this scenario, it is also not possible to explicitly obtain several evaluations for the same question-query pair. Some examples of the received feedback can be found in the supplementary material, section C.

To verify that the feedback collection is efficient, we measured the time each user took from loading a form to submitting it. To provide feedback for one question-query pair, users took 16.4 seconds on average with a standard deviation of 33.2 seconds. The vast majority (728 instances) are completed in less than 10 seconds.

Learning from Human Bandit Feedback. An analysis of $\mathcal{D}_{human}$ shows that for 531 queries all corresponding statements were marked as correct. We consider a simple baseline that treats completely correct logged data as a supervised data set with which training continues using the cross-entropy objective. We call this baseline bandit-to-supervised conversion (B2S). Furthermore, we present experimental results using the log $\mathcal{D}_{human}$ for stochastic (minibatch) gradient descent optimization of the counterfactual objectives introduced in equations 4, 6, 7 and 8. For the token-level feedback, we map the evaluated statements back to the corresponding tokens in the original query and assign these tokens a feedback of 0 if the corresponding statement was marked as wrong and 1 otherwise. In the case of sequence-level feedback, the query receives a feedback of 1 if all statements are marked correct, 0 otherwise. For the OSL objectives, a separate experiment (see below) showed that updating the reweighting constant after every validation step promises the best trade-off between performance and speed.

Results, averaged over 3 runs, are reported in Table 3. The B2S model can slightly improve upon the baseline but not significantly. DPM improves further, significantly beating the baseline. Using the multiplicative control variate modified for SGD by OSL updates does not seem to help in this setup. By moving to token-level rewards, it is possible to learn from partially correct queries. These partially correct queries provide valuable information that is not present in the subset of correct answers employed by the previous models. Optimizing DPM+T leads to a slight improvement and combined with the multiplicative control variate, DPM+T+OSL yields an improvement of about 1.0 in F1 score upon the baseline. It beats both the baseline and the B2S model by a significant margin.

Learning from Large-Scale Simulated Feedback. We want to investigate whether the results scale if a larger log is used. Thus, we use π_0 to parse all 22,765 questions from $\mathcal{D}_{log}$ and obtain for each an output query. For sequence level rewards, we assign feedback of 1 for a query if it is identical to the true target query, 0 otherwise. We also simulate token-level rewards by iterating over the indices of the output and assigning a feedback of 1 if the same token appears at the current index for the true target query, 0 otherwise.

An analysis of $\mathcal{D}_{log}$ shows that 46.27% of the queries have a sequence level reward of 1 and are thus completely correct. This subset is used to train a bandit-to-supervised (B2S) model using the cross-entropy objective.

Experimental results for the various optimization setups, averaged over 3 runs, are reported in Table 4. We see that the B2S model outperforms the baseline model by a large margin, yielding an increase in F1 score by 6.24 points. Optimizing the DPM objective also yields a significant increase over the baseline, but its performance falls short of the stronger B2S baseline. Optimizing the DPM+OSL objective leads to a substantial improvement in F1 score over optimizing DPM but still falls slightly short of the strong B2S baseline. Token-level rewards are again crucial to beat the B2S baseline significantly. DPM+T is already able to significantly outperform B2S in this setup and DPM+T+OSL can improve upon this further.

Analysis. Comparing the baseline and DPM+T+OSL, we manually examined all queries in the test set where DPM+T+OSL ob-

		F1	Δ F1
1	baseline	57.45	
2	B2S	57.79 ± 0.18	+0.34
3	DPM^{1}	58.04 ± 0.04	+0.59
4	DPM+OSL	58.01 ± 0.23	+0.56
5	DPM+T^{1}	58.11 ± 0.24	+0.66
6	DPM+T+OSL^{1,2}	58.44 ± 0.09	+0.99

Table 3: Human Feedback: Answer F1 scores on the test set for the various setups, averaged over 3 runs. Statistical significance of system differences at $p < 0.05$ are indicated by experiment number in superscript.

		F1	Δ F1
1	baseline	57.45	
2	B2S^{1,3}	63.22 ± 0.27	+5.77
3	DPM^{1}	61.80 ± 0.16	+4.35
4	DPM+OSL^{1,3}	62.91 ± 0.05	+5.46
5	DPM+T^{1,2,3,4}	63.85 ± 0.2	+6.40
6	DPM+T+OSL^{1,2,3,4}	64.41 ± 0.05	+6.96

Table 4: Simulated Feedback: Answer F1 scores on the test set for the various setups, averaged over 3 runs. Statistical significance of system differences at $p < 0.05$ are indicated by experiment number in superscript.

Error Type	Human	Simulated
OSM Tag	90%	86.75%
Question Type	6%	8.43%
Structure	4%	4.82%

Table 5: Analysis of which type of errors DPM+T+OSL corrected on the test set compared to the baseline system for both human and simulated feedback experiments.

tained the correct answer and the baseline system did not (see Table 5). The analysis showed that the vast majority of previously wrong queries were fixed by correcting an OSM tag in the query. For example, for the question "*closest Florist from Manchester in walking distance*" the baseline system chose the tag "*landuse : retail*" in the query, whereas DPM+T+OSL learnt that the correct tag is "*shop : florist*". In some cases, the question type had to be corrected, e.g. the baseline's suggested query returned the location of a point of interest but DPM+T+OSL correctly returns the phone number. Finally, in a few cases DPM+T+OSL corrected the structure for a query, e.g. by searching for a point of interest in the east of an area rather than the south.

OSL Update Variation. Using the DPM+T+OSL objective and the simulated feedback setup, we vary the frequency of updating the reweighting constant. Results are reported in Table 6. Calculating the constant only once at the beginning leads to a near identical result in F1 score as not using OSL. The more frequent update strategies, once or four times per epoch, are more effective. Both strategies reduce variance further and lead to higher F1 scores. Updating four times per epoch compared to once per epoch, leads to a nominally higher performance in F1. It has the additional benefit that the re-calculation is done at the same time as the validation, leading to no additional slow down as executing the queries for the development set against the database takes longer than the re-calculation of the constant. Updating after every minibatch is infeasible as it slows down training too much. Compared to the previous setup, iterating over one epoch takes approximately an additional 5.5 hours.

	OSL Update	F1	Δ F1
1	no OSL (DPM+T)	63.85±0.2	
2	once	63.82±0.1	-0.03
3	every epoch	64.26±0.04	+0.41
4	every validation / 4x per epoch	64.41±0.05	+0.56
5	every minibatch	N/A	N/A

Table 6: Simulated Feedback: Answer F1 scores on the test set for DPM+T and DPM+T+OSL with varying OSL update strategies, averaged over 3 runs. Updating after every minibatch is infeasible as it significantly slows down learning. Statistical significance of system differences at $p < 0.05$ occur for experiment 4 over experiment 2.

7 Conclusion

We introduced a scenario for improving a neural semantic parser from logged bandit feedback. This scenario is important to avoid complex and costly data annotation for supervise learning, and it is realistic in commercial applications where weak feedback can be collected easily in large amounts from users. We presented robust counterfactual learning objectives that allow to perform stochastic gradient optimization which is crucial in working with neural networks. Furthermore, we showed that it is essential to obtain reward signals at the token-level in order to learn from partially correct queries. We presented experimental results using feedback collected from humans and a larger scale setup with simulated feedback. In both cases we show that a strong baseline using a bandit-to-supervised conversion can be significantly outperformed by a combination of a one-step-late reweighting and token-level rewards. Finally, our approach to collecting feedback can also be transferred to other domains. For example, (Yih et al., 2016) designed a user interface to help Freebase experts to efficiently create queries. This interface could be reversed: given a question and a query produced by a parser, the interface is filled out automatically and the user has to verify if the information fits.

Acknowledgments

The research reported in this paper was supported in part by DFG grant RI-2221/4-1.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, San Diego, CA.

Léon Bottou, Jonas Peters, Joaquin Qui nonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv e-prints*, 1412.3555.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Miroslav Dudik, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, New York, NY.

Michael C. Fu. 2006. Gradient estimation. *Handbook in Operations Research and Management Science*, 13.

Dan Goldwasser and Dan Roth. 2013. Learning from natural instructions. *Machine Learning*, 94(2).

Peter J. Green. 1990. On the use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society B*, 52(3).

Carolin Haas and Stefan Riezler. 2016. A corpus and semantic parser for multilingual natural language querying of openstreetmap. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Nan Jiang and Lihong Li. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, New York, New York, USA.

Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with logged bandit feedback. In *International Conference on Learning Representations (ICLR)*.

Augustine Kong. 1992. A note on importance sampling using standardized weights. Technical Report 348, Department of Statistics, University of Chicago, Illinois.

Carolin Lawrence, Pratik Gajane, and Stefan Riezler. 2017a. Counterfactual learning for machine translation: Degeneracies and solutions. In *Proceedings of the NIPS WhatIF Workshop*, Long Beach, CA.

Carolin Lawrence and Stefan Riezler. 2016. Nlmaps: A natural language interface to query openstreetmap. In *Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations (COLING)*, Osaka, Japan.

Carolin Lawrence, Artem Sokolov, and Stefan Riezler. 2017b. Counterfactual learning from bandit feedback under deterministic logging : A case study in statistical machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2017. Coupling distributed and symbolic execution for natural language queries. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *International Conference on Learning Representations (ICLR)*, Toulon, France.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017. Maximum margin reward networks for learning from explicit and implicit supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco, CA, USA.

Paul R. Rosenbaum and Donald B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1).

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada.

Adith Swaminathan and Thorsten Joachims. 2015a. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16.

Adith Swaminathan and Thorsten Joachims. 2015b. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada.

Philip Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, New York, NY.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 20.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *ArXiv e-prints*, 1212.5701.

AMR Dependency Parsing with a Typed Semantic Algebra

Jonas Groschwitz[*†] Matthias Lindemann[*] Meaghan Fowlie[*]
Mark Johnson[†] Alexander Koller[*]
[*] Saarland University, Saarbrücken, Germany [†] Macquarie University, Sydney, Australia
{jonasg|mlinde|mfowlie|koller}@coli.uni-saarland.de
mark.johnson@mq.edu.au

Abstract

We present a semantic parser for Abstract Meaning Representations which learns to parse strings into tree representations of the compositional structure of an AMR graph. This allows us to use standard neural techniques for supertagging and dependency tree parsing, constrained by a linguistically principled type system. We present two approximative decoding algorithms, which achieve state-of-the-art accuracy and outperform strong baselines.

1 Introduction

Over the past few years, Abstract Meaning Representations (AMRs, Banarescu et al. (2013)) have become a popular target representation for semantic parsing. AMRs are graphs which describe the predicate-argument structure of a sentence. Because they are graphs and not trees, they can capture reentrant semantic relations, such as those induced by control verbs and coordination. However, it is technically much more challenging to parse a string into a graph than into a tree. For instance, grammar-based approaches (Peng et al., 2015; Artzi et al., 2015) require the induction of a grammar from the training corpus, which is hard because graphs can be decomposed into smaller pieces in far more ways than trees. Neural sequence-to-sequence models, which do very well on string-to-tree parsing (Vinyals et al., 2014), can be applied to AMRs but face the challenge that graphs cannot easily be represented as sequences (van Noord and Bos, 2017a,b).

In this paper, we tackle this challenge by making the compositional structure of the AMR explicit. As in our previous work, Groschwitz et al. (2017), we view an AMR as consisting of atomic graphs representing the meanings of the individual words, which were combined compositionally using linguistically motivated operations for combining a head with its arguments and modifiers. We represent this structure as terms over the *AM algebra* as defined in Groschwitz et al. (2017). This previous work had no parser; here we show that the terms of the AM algebra can be viewed as dependency trees over the string, and we train a dependency parser to map strings into such trees, which we then evaluate into AMRs in a postprocessing step. The dependency parser relies on type information, which encodes the semantic valencies of the atomic graphs, to guide its decisions.

More specifically, we combine a neural supertagger for identifying the elementary graphs for the individual words with a neural dependency model along the lines of Kiperwasser and Goldberg (2016) for identifying the operations of the algebra. One key challenge is that the resulting term of the AM algebra must be semantically well-typed. This makes the decoding problem NP-complete. We present two approximation algorithms: one which takes the unlabeled dependency tree as given, and one which assumes that all dependencies are projective. We evaluate on two data sets, achieving state-of-the-art results on one and near state-of-the-art results on the other (Smatch f-scores of 71.0 and 70.2 respectively). Our approach clearly outperforms strong but non-compositional baselines.

Plan of the paper. After reviewing related work in Section 2, we explain the AM algebra in Section 3 and extend it to a dependency view in Section 4. We explain model training in Section 5 and decoding in Section 6. Section 7 evaluates a number of variants of our system.

2 Related Work

Recently, AMR parsing has generated considerable research activity, due to the availability of large-scale annotated data (Banarescu et al., 2013) and two successful SemEval Challenges (May, 2016; May and Priyadarshi, 2017).

Methods from dependency parsing have been shown to be very successful for AMR parsing. For instance, the JAMR parser (Flanigan et al., 2014, 2016) distinguishes *concept identification* (assigning graph fragments to words) from *relation identification* (adding graph edges which connect these fragments), and solves the former with a supertagging-style method and the latter with a graph-based dependency parser. Foland and Martin (2017) use a variant of this method based on an intricate neural model, yielding state-of-the-art results. We go beyond these approaches by explicitly modeling the compositional structure of the AMR, which allows the dependency parser to combine AMRs for the words using a small set of universal operations, guided by the types of these AMRs.

Other recent methods directly implement a dependency parser for AMRs, e.g. the transition-based model of Damonte et al. (2017), or postprocess the output of a dependency parser by adding missing edges (Du et al., 2014; Wang et al., 2015). In contrast to these, our model makes no strong assumptions on the dependency parsing algorithm that is used; here we choose that of Kiperwasser and Goldberg (2016).

The commitment of our parser to derive AMRs compositionally mirrors that of grammar-based AMR parsers (Artzi et al., 2015; Peng et al., 2015). In particular, there are parallels between the types we use in the AM algebra and CCG categories (see Section 3 for details). As a neural system, our parser struggles less with coverage issues than these, and avoids the complex grammar induction process these models require.

More generally, our use of semantic types to restrict our parser is reminiscent of Kwiatkowski et al. (2010), Krishnamurthy et al. (2017) and Zhang et al. (2017), and the idea of deriving semantic representations from dependency trees is also present in Reddy et al. (2017).

3 The AM algebra

A core idea of this paper is to parse a string into a graph by instead parsing a string into a dependency-style tree representation of the graph's compositional structure, represented as terms of the *Apply-Modify (AM) algebra* (Groschwitz et al., 2017).

The values of the AM algebra are *annotated s-graphs*, or *as-graphs*: directed graphs with node and edge labels in which certain nodes have been designated as *sources* (Courcelle and Engelfriet, 2012) and *annotated* with type information. Some examples of as-graphs are shown in Fig. 1. Each as-graph has exactly one *root*, indicated by the bold outline. The sources are indicated by red labels; for instance, G_{want} has an S-source and an O-source. The annotations, written in square brackets behind the red source names, will be explained below. We use these sources to mark open argument slots; for example, G_{sleep} in Fig. 1 represents an intransitive verb, missing its subject, which will be added at the S-source.

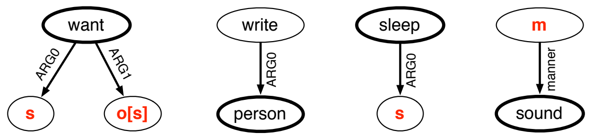


Figure 1: Elementary as-graphs G_{want}, G_{writer}, G_{sleep}, and G_{sound} for the words "want", "writer", "sleep", and "soundly" respectively.

The AM algebra can combine as-graphs with each other using two linguistically motivated operations: *apply* and *modify*. Apply (APP) adds an argument to a predicate. For example, we can add a subject – the graph G_{writer} in Fig. 1 – to the graph G_{VP} in Fig. 2d using APP_{S}, yielding the complete AMR in Fig. 2b. Linguistically, this is like filling the subject (S) slot of the predicate *wants to sleep soundly* with the argument *the writer*. In general, for a source a, $\text{APP}_a(G_P, G_A)$, combines the as-graph G_P representing a predicate, or head, with the as-graph G_A, which represents an argument. It does this by plugging the root node of G_A into the a-source u of G_P – that is, the node u of G_P marked with source a. The root of the resulting as-graph G is the root of G_P, and we remove the a marking on u, since that slot is now filled.

The modify operation (MOD) adds a modifier to a graph. For example, we can combine two elementary graphs from Fig. 1 with MOD_m (G_{sleep}, G_{sound}), yielding the graph in Fig. 2c. The M-source of the modifier G_{soundly} attaches to the root of G_{sleep}. The root of the result is the same as the root of G_{sleep} in the same sense that a verb phrase with an adverb modifier is still a verb phrase. In general, $\text{MOD}_a(G_H, G_M)$, combines a head G_H with a modifier G_M. It plugs the root of G_H into the a-source u of G_M. Although this may add incoming edges to the root of G_H, that node is still the root of the resulting graph G. We remove the a marking from G_M.

In both App and Mod, if there is any other source b which is present in both graphs, the nodes marked with b are unified with each other. For example, when G_{want} is o-applied to t_1 in Fig. 2d, the s-sources of the graphs for "want" and "sleep soundly" are unified into a single node, creating a reentrancy. This falls out of the definition of *merge* for s-graphs which formally underlies both operations (see (Courcelle and Engelfriet, 2012)).

Finally, the AM algebra uses *types* to restrict its operations. Here we define the *type* of an as-graph as the set of its sources with their annotations[1]; thus for example, in Fig. 1, the graph for "writer" has the empty type [ ], G_{sleep} has type $[\text{s}]$, and G_{want} has type $[\text{s}, \text{o}[\text{s}]]$. Each source in an as-graph specifies with its *annotation* the type of the as-graph which is plugged into it via App. In other words, for a source a, we may only a-apply G_P with G_A if the annotation of the a-source in G_P matches the type of G_A. For example, the o-source of G_{wants} (Fig. 1) requires that we plug in an as-graph of type $[\text{s}]$; observe that this means that the reentrancy in Fig. 2b is lexically specified by the control verb "want". All other source nodes in Fig. 1 have no annotation, indicating a type requirement of [ ].

Linguistically, modification is optional; we therefore want the modified graph to be derivationally just like the unmodified graph, in that exactly the same operations can apply to it. In a typed algebra, this means Mod should not change the type of the head. Mod_a therefore requires that the modifier G_M have no sources not already present in the head G_H, except a, which will be deleted anyway.

As in any algebra, we can build *terms* from constants (denoting elementary as-graphs) by recursively combining them with the operations of the AM algebra. By evaluating the operations bottom-up, we obtain an as-graph as the value of such a term; see Fig. 2 for an example. However, as discussed above, an operation in the term may be undefined due to a type mismatch. We call an AM-term *well-typed* if all its operations are defined. Every well-typed AM-term evaluates to an as-graph. Since the applicability of an AM operation depends only on the types, we also write $\tau = f(\tau_1, \tau_2)$ if as-graphs of type τ_1 and τ_2 can be combined with the operation f and the result has type τ.

[1]See (Groschwitz et al., 2017) for a more formally complete definition.

Relationship to CCG. There is close relationship between the types of the AM algebra and the categories of CCG. A type $[\text{s}, \text{o}]$ specifies that the as-graph needs to be applied to two arguments to be semantically complete, similar a CCG category such as $\text{S}\backslash\text{NP}/\text{NP}$, where a string needs to be applied to two NP arguments to be syntactically complete. However, AM types govern the combination of *graphs*, while CCG categories control the combination of strings. This relieves AM types of the need to talk about word order; there are no "forward" or "backward" slashes in AM types, and a smaller set of operations. Also, the AM algebra spells out raising and control phenomena more explicitly in the types.

4 Indexed AM terms

In this paper, we connect AM terms to the input string w for which we want to produce a graph. We do this in an *indexed AM term*, exemplified in Fig. 3a. We assume that every elementary as-graph G at a leaf represents the meaning of an individual word token w_i in w, and write $G[i]$ to annotate the leaf G with the index i of this token. This induces a connection between the nodes of the AMR and the tokens of the string, in that the label of each node was contributed by the elementary as-graph of exactly one token.

We define the *head index* of a subtree t to be the index of the token which contributed the root of the as-graph to which t evaluates. For a leaf with annotation i, the head index is i; for an App or Mod node, the head index is the head index of the left child, i.e. of the head argument. We annotate each App and Mod operation with the head index of the left and right subtree.

4.1 AM dependency trees

We can represent indexed AM terms more compactly as *AM dependency trees*, as shown in Fig. 3b. The nodes of such a dependency tree are the tokens of w. We draw an edge with label f from i to k if there is a node with label $f[i, k]$ in the indexed AM term. For example, the tree in 3b has an edge labeled Mod_m from 5 (G_{sleep}) to 6 (G_{soundly}) because there is a node in the term in 3a labeled $\text{Mod}_m[5, 6]$. The same AM dependency tree may represent multiple indexed AM terms, because the order of apply and modify operations is not specified in the dependency tree. However, it can be shown that all well-typed AM terms that map to

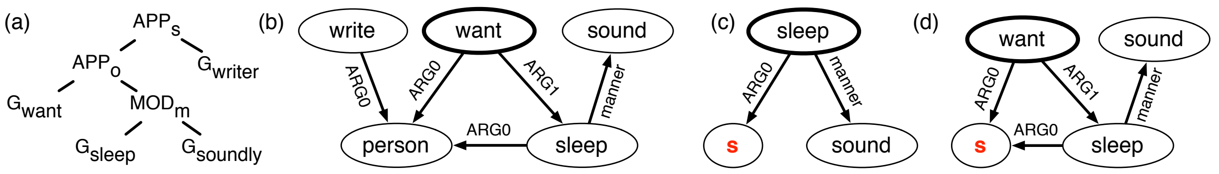


Figure 2: (a) An AM-term with its value (b), along with the values for its subexpressions (c) $t_1 = \text{MOD}_m(G_{\text{sleep}}, G_{\text{sound}})$ and (d) $t_2 = \text{APP}_o(G_{\text{want}}, t_1)$.

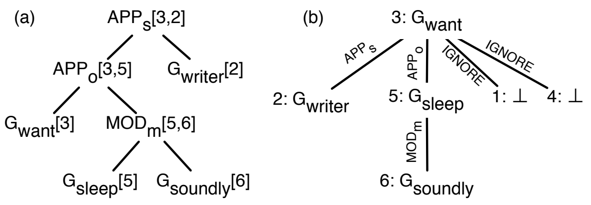


Figure 3: (a) An indexed AM term and (b) an AM dependency tree, linking the term in Fig. 2;a to the sentence “The writer wants to sleep soundly”.

the same AM dependency tree evaluate to the same as-graph. We define a *well-typed* AM dependency tree as one that represents a well-typed AM term.

Because not all words in the sentence contribute to the AMR, we include a mechanism for ignoring words in the input. As a special case, we allow the constant $\perp$, which represents a dummy as-graph (of type $\perp$) which we use as the semantic value of words without a semantic value in the AMR. We furthermore allow the edge label IGNORE in an AM dependency tree, where $\text{IGNORE}(\tau_1, \tau_2) = \tau_1$ if $\tau_2 = \perp$ and is undefined otherwise; in particular, an AM dependency tree with IGNORE edges is only well-typed if all IGNORE edges point into $\perp$ nodes. We keep all other operations $f(\tau_1, \tau_2)$ as is, i.e. they are undefined if either τ_1 or τ_2 is $\perp$, and never yield $\perp$ as a result. When reconstructing an AM term from the AM dependency tree, we skip IGNORE edges, such that the subtree below them will not contribute to the overall AMR.

4.2 Converting AMRs to AM terms

In order to train a model that parses sentences into AM dependency trees, we need to convert an AMR corpus – in which sentences are annotated with AMRs – into a treebank of AM dependency trees. We do this in three steps: first, we break each AMR up into elementary graphs and identify their roots; second, we assign sources and annotations to make elementary as-graphs out of them; and third, combine them into indexed AM terms.

For the first step, an aligner uses hand-written heuristics to identify the string token to which each node in the AMR corresponds (see Section C in the Supplementary Materials for details). We proceed in a similar fashion as the JAMR aligner (Flanigan et al., 2014), i.e. by starting from high-confidence token-node pairs and then extending them until the whole AMR is covered. Unlike the JAMR aligner, our heuristics ensure that exactly one node in each elementary graph is marked as the root, i.e. as the node where other graphs can attach their edges through APP and MOD. When an edge connects nodes of two different elementary graphs, we use the “blob decomposition” algorithm of Groschwitz et al. (2017) to decide to which elementary graph it belongs. For the example AMR in Fig. 2b, we would obtain the graphs in Fig. 1 (without source annotations). Note that ARG edges belong with the nodes at which they start, whereas the “manner” edge in G_{soundly} goes with its target.

In the second step we assign source names and annotations to the unlabeled nodes of each elementary graph. Note that the annotations are crucial to our system’s ability to generate graphs with reentrancies. We mostly follow the algorithm of Groschwitz et al. (2017), which determines necessary annotations based on the structure of the given graph. The algorithm chooses each source name depending on the incoming edge label. For instance, the two leaves of G_{want} can have the source labels S and O because they have incoming edges labeled ARG0 and ARG1. However, the Groschwitz algorithm is not deterministic: It allows object promotion (the sources for an ARG3 edge may be O3, O2, or O), unaccusative subjects (promoting the minimal object to S if the elementary graph contains an ARGi-edge ($i > 0$) but no ARG0-edge (Perlmutter, 1978)), and passive alternation (swapping O and S). To make our as-graphs more consistent, we prefer constants that promote objects as far as possible, use unaccusative subjects, and no passive alternation, but still allow constants that do not satisfy these conditions if necessary. This increased our Smatch score significantly.

Finally, we choose an arbitrary AM dependency tree that combines the chosen elementary as-graphs into the annotated AMR; in practice, the differences between the trees seem to be negligible.[2]

5 Training

We can now model the AMR parsing task as the problem of computing the best well-typed AM dependency tree t for a given sentence w. Because t is well-typed, it can be decoded into an (indexed) AM term and thence evaluated to an as-graph.

We describe t in terms of the elementary as-graphs $G[i]$ it uses for each token i and of its edges $f[i,k]$. We assume a node-factored, edge-factored model for the *score* $\omega(t)$ of t:

$$\omega(t) = \sum_{1 \le i \le n} \omega(G[i]) + \sum_{f[i,k] \in E} \omega(f[i,k]), \quad (1)$$

where the edge weight further decomposes into the sum $\omega(f[i,k]) = \omega(i \to k) + \omega(f \mid i \to k)$ of a score $\omega(i \to k)$ for the presence of an edge from i to k and a score $\omega(f \mid i \to k)$ for this edge having label f. Our aim is to compute the well-typed t with the highest score.

We present three models for ω: one for the graph scores and two for the edge scores. All of these are based on a two-layer bidirectional LSTM, which reads inputs $\mathbf{x} = (x_1, \dots, x_n)$ token by token, concatenating the hidden states of the forward and the backward LSTMs in each layer. On the second layer, we thus obtain vector representations $v_i = \text{BiLSTM}(\mathbf{x}, i)$ for the individual input tokens (see Fig. 4). Our models differ in the inputs $\mathbf{x}$ and the way they predict scores from the v_i.

5.1 Supertagging for elementary as-graphs

We construe the prediction of the as-graphs $G[i]$ for each input position i as a supertagging task (Lewis et al., 2016). The supertagger reads inputs $x_i = (w_i, p_i, c_i)$, where w_i is the word token, p_i its POS tag, and c_i is a character-based LSTM encoding of w_i. We use pretrained GloVe embeddings (Pennington et al., 2014) concatenated with learned embeddings for w_i, and learned embeddings for p_i.

To predict the score for each elementary as-graph out of a set of K options, we add a K-dimensional output layer as follows:

$$\omega(G[i]) = \log \text{softmax}(W \cdot v_i + b)$$

[2]Indeed, we conjecture that for a fixed set of constants and a fixed AMR, there is only one dependency tree.

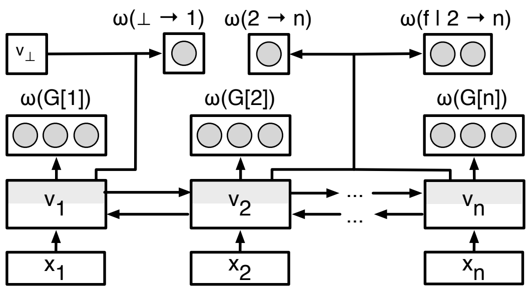


Figure 4: Architecture of the neural taggers.

and train the neural network using a cross-entropy loss function. This maximizes the likelihood of the elementary as-graphs in the training data.

5.2 Kiperwasser & Goldberg edge model

Predicting the edge scores amounts to a dependency parsing problem. We chose the dependency parser of Kiperwasser and Goldberg (2016), henceforth K&G, to learn them, because of its accuracy and its fit with our overall architecture. The K&G parser scores the potential edge from i to k and its label from the concatenations of v_i and v_k:

$$\begin{aligned} \text{MLP}_\theta(v) &= W_2 \cdot \tanh(W_1 \cdot v + b_1) + b_2 \\ \omega(i \to k) &= \text{MLP}_\text{E}(v_i \circ v_k) \\ \omega(f \mid i \to k) &= \text{MLP}_\text{LBL}(v_i \circ v_k) \end{aligned}$$

We use inputs $x_i = (w_i, p_i, \tau_i)$ including the type τ_i of the supertag $G[i]$ at position i, using trained embeddings for all three. At evaluation time, we use the best scoring supertag according to the model of Section 5.1. At training time, we sample from q, where $q(\tau_i) = (1 - \delta) + \delta \cdot p(\tau_i | p_i, p_{i-1})$, $q(\tau) = \delta \cdot p(\tau | p_i, p_{i-1})$ for any $\tau \neq \tau_i$ and δ is a hyperparameter controlling the bias towards the aligned supertag. We train the model using K&G's original DyNet implementation. Their algorithm uses a hinge loss function, which maximizes the score difference between the gold dependency tree and the best predicted dependency tree, and therefore requires parsing each training instance in each iteration. Because the AM dependency trees are highly non-projective, we replaced the projective parser used in the off-the-shelf implementation by the Chu-Liu-Edmonds algorithm implemented in the TurboParser (Martins et al., 2010), improving the LAS on the development set by 30 points.

5.3 Local edge model

We also trained a *local* edge score model, which uses a cross-entropy rather than a hinge loss and therefore avoids the repeated parsing at training time. Instead, we follow the intuition that every node in a dependency tree has at most one incoming edge, and train the model to score the correct incoming edge as high as possible. This model takes inputs $x_i = (w_i, p_i)$.

We define the edge and edge label scores as in Section 5.2, with tanh replaced by ReLU. We further add a learned parameter $v_\perp$ for the "LSTM embedding" of a nonexistent node, obtaining scores $\omega(\perp \rightarrow k)$ for k having no incoming edge.

To train $\omega(i \rightarrow k)$, we collect all scores for edges ending at the same node k into a vector $\omega(\bullet \rightarrow k)$. We then minimize the cross-entropy loss for the gold edge into k under softmax$(\omega(\bullet \rightarrow k))$, maximizing the likelihood of the gold edges. To train the labels $\omega(f \mid i \rightarrow k)$, we simply minimize the cross-entropy loss of the actual edge labels f of the edges which are present in the gold AM dependency trees.

The PyTorch code for this and the supertagger are available at `bitbucket.org/tclup/amr-dependency`.

6 Decoding

Given learned estimates for the graph and edge scores, we now tackle the challenge of computing the best well-typed dependency tree t for the input string w, under the score model (equation (1)). The requirement that t must be well-typed is crucial to ensure that it can be evaluated to an AMR graph, but as we show in the Supplementary Materials (Section A), makes the decoding problem NP-complete. Thus, an exact algorithm is not practical. In this section, we develop two different approximation algorithms for AM dependency parsing: one which assumes the (unlabeled) dependency tree structure as known, and one which assumes that the AM dependency tree is projective.

6.1 Projective decoder

The projective decoder assumes that the AM dependency tree is projective, i.e. has no crossing dependency edges. Because of this assumption, it can recursively combine adjacent substrings using dynamic programming. The algorithm is shown in Fig. 5 as a parsing schema (Shieber et al., 1995), which derives items of the form $([i, k], r, \tau)$ with scores s. An item represents a well-typed derivation of the substring from i to k with head index r, and which evaluates to an as-graph of type τ.

$$\frac{s = \omega(G[i]) \quad G \neq \perp}{([i, i+1], i, \tau(G)) : s} \ \text{Init}$$

$$\frac{([i, k], r, \tau) : s \quad s' = \omega(\perp[k])}{([i, k+1], r, \tau) : s + s'} \ \text{Skip-R}$$

$$\frac{([i, k], r, \tau) : s \quad s' = \omega(\perp[i-1])}{([i-1, k], r, \tau) : s + s'} \ \text{Skip-L}$$

$$\frac{\begin{array}{c}([i, j], r_1, \tau_1) : s_1 \quad ([j, k], r_2, \tau_2) : s_2 \\ \tau = f(\tau_1, \tau_2) \text{ defined} \quad s = \omega(f[r_1, r_2])\end{array}}{([i, k], r_1, \tau) : s_1 + s_2 + s} \ \text{Arc-R } [f]$$

$$\frac{\begin{array}{c}([i, j], r_1, \tau_1) : s_1 \quad ([j, k], r_2, \tau_2) : s_2 \\ \tau = f(\tau_2, \tau_1) \text{ defined} \quad s = \omega(f[r_2, r_1])\end{array}}{([i, k], r_2, \tau) : s_1 + s_2 + s} \ \text{Arc-L } [f]$$

Figure 5: Rules for the projective decoder.

The parsing schema consists of three types of rules. First, the Init rule generates an item for each graph fragment $G[i]$ that the supertagger predicted for the token w_i, along with the score and type of that graph fragment. Second, given items for adjacent substrings $[i, j]$ and $[j, k]$, the Arc rules apply an operation f to combine the indexed AM terms for the two substrings, with Arc-R making the left-hand substring the head and the right-hand substring the argument or modifier, and Arc-L the other way around. We ensure that the result is well-typed by requiring that the types can be combined with f. Finally, the Skip rules allow us to extend a substring such that it covers tokens which do not correspond to a graph fragment (i.e., their AM term is $\perp$), introducing IGNORE edges. After all possible items have been derived, we extract the best well-typed tree from the item of the form $([1, n], r, \tau)$ with the highest score, where $\tau = [\,]$.

Because we keep track of the head indices, the projective decoder is a bilexical parsing algorithm, and shares a parsing complexity of $O(n^5)$ with other bilexical algorithms such as the Collins parser. It could be improved to a complexity of $O(n^4)$ using the algorithm of Eisner and Satta (1999).

6.2 Fixed-tree decoder

The fixed-tree decoder computes the best unlabeled dependency tree t_r for w, using the edge scores $\omega(i \rightarrow k)$, and then computes the best AM dependency tree for w whose unlabeled version is t_r. The Chu-Liu-Edmonds algorithm produces a forest of dependency trees, which we want to combine into t_r. We choose the tree whose root r has the highest score for being the root of the AM dependency tree and make the roots of all others children of r.

At this point, the shape of t_r is fixed. We choose

$$\frac{s = \omega(G[i])}{(i, \emptyset, \tau(G)) : s}\ \text{Init}$$

$$\frac{(i, C_1, \tau_1) : s_1 \quad (k, Ch(k), \tau_2) : s_2 \quad k \in Ch(i) \backslash C_1 \quad \tau = f(\tau_1, \tau_2) \text{ defined} \quad s = \omega(f[i,k])}{(i, C_1 \cup \{k\}, \tau) : s_1 + s_2 + s}\ \text{Edge}[f]$$

Figure 6: Rules for the fixed-tree decoder.

supertags for the nodes and edge labels for the edges by traversing t_r bottom-up, computing types for the subtrees as we go along. Formally, we apply the parsing schema in Fig. 6. It uses items of the form $(i, C, \tau) : s$, where $1 \leq i \leq n$ is a node of t_r, C is the set of children of i for which we have already chosen edge labels, and τ is a type. We write $Ch(i)$ for the set of children of i in t_r.

The Init rule generates an item for each graph that the supertagger can assign to each token i in w, ensuring that every token is also assigned $\perp$ as a possible supertag. The Edge rule labels an edge from a parent node i in t_r to one of its children k, whose children already have edge labels. As above, this rule ensures that a well-typed AM dependency tree is generated by locally checking the types. In particular, if all types τ_2 that can be derived for k are incompatible with τ_1, we fall back to an item for k with $\tau_2 = \perp$ (which always exists), along with an IGNORE edge from i to k.

The complexity of this algorithm is $O(n \cdot 2^d \cdot d)$, where d is the maximal arity of the nodes in t_r.

7 Evaluation

We evaluate our models on the LDC2015E86 and LDC2017T10[3] datasets (henceforth "2015" and "2017"). Technical details and hyperparameters of our implementation can be found in Sections B to D of the Supplementary Materials.

7.1 Training data

The original LDC datasets pair strings with AMRs. We convert each AMR in the training and development set into an AM dependency tree, using the procedure of Section 4.2. About 10% of the training instances cannot be split into elementary as-graphs by our aligner; we removed these from the training data. Of the remaining AM dependency trees, 37% are non-projective.

Furthermore, the AM algebra is designed to handle short-range reentrancies, modeling grammatical phenomena such as control and coordination, as in the derivation in Fig. 2. It cannot easily handle the long-range reentrancies in AMRs which are caused by coreference, a non-compositional phenomenon.[4] We remove such reentrancies from our training data (about 60% of the roughly 20,000 reentrant edges). Despite this, our model performs well on reentrant edges (see Table 2).

7.2 Pre- and postprocessing

We use simple pre- and postprocessing steps to handle rare words and some AMR-specific patterns. In AMRs, named entities follow a pattern shown in Fig. 7. Here the named entity is of type "person", has a name edge to a "name" node whose children spell out the tokens of "Agatha Christie", and a link to a wiki entry. Before training, we replace each "name" node, its children, and the corresponding span in the sentence with a special `NAME` token, and we completely remove wiki edges. In this example, this leaves us with only a "person" and a `NAME` node. Further, we replace numbers and some date patterns with `NUMBER` and `DATE` tokens. On the training data this is straightforward, since names and dates are explicitly annotated in the AMR. At evaluation time, we detect dates and numbers with regular expressions, and names with Stanford CoreNLP (Manning et al., 2014). We also use Stanford CoreNLP for our POS tags.

Each elementary as-graph generated by the procedure of Section 4.2 has a unique node whose label corresponds most closely to the aligned word (e.g. the "want" node in G_{want} and the "write" node in G_{writer}). We replace these node labels with `LEX` in preprocessing, reducing the number of different elementary as-graphs from 28730 to 2370. We factor the supertagger model of Section 5.1 such that the unlexicalized version of $G[i]$ and the label for `LEX` are predicted separately.

At evaluation, we re-lexicalize all `LEX` nodes in the predicted AMR. For words that were frequent in the training data (at least 10 times), we take the supertagger's prediction for the label. For rarer words, we use simple heuristics, explained in the Supplementary Materials (Section D). For names, we just look up name nodes with their children and wiki entries observed for the name string in the training data, and for unseen names use the literal tokens as the name, and no wiki entry. Similarly,

[3]https://catalog.ldc.upenn.edu/LDC2017T10, identical to LDC2016E25.

[4]As Damonte et al. (2017) comment: "A valid criticism of AMR is that these two reentrancies are of a completely different type, and should not be collapsed together."

we collect the type for each encountered name (e.g. "person" for "Agatha Christie"), and correct it in the output if the tagger made a different prediction. We recover dates and numbers straightforwardly.

7.3 Supertagger accuracy

All of our models rely on the supertagger to predict elementary as-graphs; they differ only in the edge scores. We evaluated the accuracy of the supertagger on the converted development set (in which each token has a supertag) of the 2015 data set, and achieved an accuracy of 73%. The correct supertag is within the supertagger's 4 best predictions for 90% of the tokens, and within the 10 best for 95%.

Interestingly, supertags that introduce grammatical reentrancies are predicted quite reliably, although they are relatively rare in the training data. The elementary as-graph for subject control verbs (see G_{want} in Fig. 1) accounts for only 0.8% of supertags in the training data, yet 58% of its occurrences in the development data are predicted correctly (84% in 4-best). The supertag for VP coordination (with type $[\text{OP}1[\text{S}], \text{OP}2[\text{S}]]$) makes up for 0.4% of the training data, but 74% of its occurrences are recognized correctly (92% in 4-best). Thus the prediction of informative types for individual words is feasible.

7.4 Comparison to Baselines

Type-unaware fixed-tree baseline. The fixed-tree decoder is built to ensure well-typedness of the predicted AM dependency trees. To investigate to what extent this is required, we consider a baseline which just adds the individually highest-scoring supertags and edge labels to the unlabeled dependency tree t_u, ignoring types. This leads to AM dependency trees which are not well-typed for 75% of the sentences (we fall back to the largest well-typed subtree in these cases). Thus, an off-the-shelf dependency parser can reliably predict the tree structure of the AM dependency tree, but correct supertag and edge label assignment requires a decoder which takes the types into account.

JAMR-style baseline. Our elementary as-graphs differ from the elementary graphs used in JAMR-style algorithms in that they contain explicit source nodes, which restrict the way in which they can be combined with other as-graphs. We investigate the impact of this choice by implementing a strong JAMR-style baseline. We adapt the AMR-to-dependency conversion of Section 4.2 by removing all unlabeled nodes with source names from the

Model	2015	2017
Ours		
local edge + projective decoder	70.2±0.3	**71.0**±0.5
local edge + fixed-tree decoder	69.4±0.6	70.2±0.5
K&G edge + projective decoder	68.6±0.7	69.4±0.4
K&G edge + fixed-tree decoder	69.6±0.4	69.9±0.2
Baselines		
fixed-tree (type-unaware)	26.0±0.6	27.9±0.6
JAMR-style	66.1	66.2
Previous work		
CAMR (Wang et al., 2015)	66.5	-
JAMR (Flanigan et al., 2016)	67	-
Damonte et al. (2017)	64	-
van Noord and Bos (2017b)	68.5	**71.0**
Foland and Martin (2017)	**70.7**	-
Buys and Blunsom (2017)	-	61.9

Table 1: 2015 & 2017 test set Smatch scores

elementary graphs. For instance, the graph G_{want} in Fig. 1 now only consists of a single "want" node. We then aim to directly predict AMR edges between these graphs, using a variant of the local edge scoring model of Section 5.3 which learns scores for each edge in isolation. (The assumption for the original local model, that each node has only one incoming edge, does not apply here.)

When parsing a string, we choose the highest-scoring supertag for each word; there are only 628 different supertags in this setting, and 1-best supertagging accuracy is high at 88%. We then follow the JAMR parsing algorithm by predicting all edges whose score is over a threshold (we found -0.02 to be optimal) and then adding edges until the graph is connected. Because we do not predict which node is the root of the AMR, we evaluated this model as if it always predicted the root correctly, overestimating its score slightly.

7.5 Results

Table 1 shows the Smatch scores (Cai and Knight, 2013) of our models, compared to a selection of previously published results. Our results are averages over 4 runs with 95% confidence intervals (JAMR-style baselines are single runs). On the 2015 dataset, our best models (local + projective, K&G + fixed-tree) outperform all previous work, with the exception of the Foland and Martin (2017) model; on the 2017 set we match state of the art results (though note that van Noord and Bos (2017b) use 100k additional sentences of silver data). The fixed-tree decoder seems to work well with either edge model, but performance of the projective decoder drops with the K&G edge scores. It may be that, while the hinge loss used in the K&G edge scoring model is useful to finding the correct un-

	2015					2017		
Metric	W'15	F'16	D'17	PD	FTD	vN'17	PD	FTD
Smatch	67	67	64	**70**	**70**	**71**	**71**	70
Unlabeled	69	69	69	**73**	**73**	**74**	**74**	**74**
No WSD	64	68	65	**71**	70	**72**	**72**	70
Named Ent.	75	79	**83**	79	78	**79**	78	77
Wikification	0	**75**	64	71	72	65	**71**	**71**
Negations	18	45	48	**52**	**52**	**62**	57	55
Concepts	80	83	83	83	**84**	82	**84**	**84**
Reentrancies	41	42	41	**46**	44	**52**	49	46
SRL	60	60	56	**63**	61	**66**	64	62

Table 2: Details for the LDC2015E86 and LDC2017T10 test sets

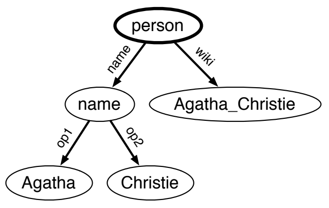


Figure 7: A named entity

labeled dependency tree in the fixed-tree decoder, scores for bad edges – which are never used when computing the hinge loss – are not trained accurately. Thus such edges may be erroneously used by the projective decoder.

As expected, the type-unaware baseline has low recall, due to its inability to produce well-typed trees. The fact that our models outperform the JAMR-style baseline so clearly is an indication that they indeed gain some of their accuracy from the type information in the elementary as-graphs, confirming our hypothesis that an explicit model of the compositional structure of the AMR can help the parser learn an accurate model.

Table 2 analyzes the performance of our two best systems (PD = projective, FTD = fixed-tree) in more detail, using the categories of Damonte et al. (2017), and compares them to Wang's, Flanigan's, and Damonte's AMR parsers on the 2015 set and , and van Noord and Bos (2017b) for the 2017 dataset. (Foland and Martin (2017) did not publish such results.) The good scores we achieve on reentrancy identification, despite removing a large amount of reentrant edges from the training data, indicates that our elementary as-graphs successfully encode phenomena such as control and coordination.

The projective decoder is given 4, and the fixed-tree decoder 6, supertags for each token. We trained the supertagging and edge scoring models of Section 5 separately; joint training did not help. Not sampling the supertag types τ_i during training of the K&G model, removing them from the input, and removing the character-based LSTM encodings c_i from the input of the supertagger, all reduced our models' accuracy.

7.6 Differences between the parsers

Although the Smatch scores for our two best models are close, they sometimes struggle with different sentences. The fixed-tree parser is at the mercy of the fixed tree; the projective parser cannot produce non-projective AM dependency trees. It is remarkable that the projective parser does so well, given the prevalence of non-projective trees in the training data. Looking at its analyses, we find that it frequently manages to find a projective tree which yields an (almost) correct AMR, by choosing supertags with unusual types, and by using modify rather than apply (or vice versa).

8 Conclusion

We presented an AMR parser which applies methods from supertagging and dependency parsing to map a string into a well-typed AM term, which it then evaluates into an AMR. The AM term represents the compositional semantic structure of the AMR explicitly, allowing us to use standard tree-based parsing techniques.

The projective parser currently computes the complete parse chart. In future work, we will speed it up through the use of pruning techniques. We will also look into more principled methods for splitting the AMRs into elementary as-graphs to replace our hand-crafted heuristics. In particular, advanced methods for alignments, as in Lyu and Titov (2018), seem promising. Overcoming the need for heuristics also seems to be a crucial ingredient for applying our method to other semantic representations.

Acknowledgements We would like to thank the anonymous reviewers for their comments. We thank Stefan Grünewald for his contribution to our PyTorch implementation, and want to acknowledge the inspiration obtained from Nguyen et al. (2017). We also extend our thanks to the organizers and participants of the Oslo CAS Meaning Construction workshop on Universal Dependencies. This work was supported by the DFG grant KO 2916/2-1 and a Macquarie University Research Excellence Scholarship for Jonas Groschwitz.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Jan Buys and Phil Blunsom. 2017. Oxford at SemEval-2017 task 9: Neural AMR parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 914–919.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge University Press.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th ACL*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

William Foland and James H. Martin. 2017. Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with amrs. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1516–1526.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 1223–1233.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Conference of the Association for Computational Linguistics (ACL)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics.

Jonathan May and Jay Priyadarshi. 2017. Semeval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics.

Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. A novel neural network model for joint POS tagging and graph-based dependency parsing. *arXiv preprint arXiv:1705.05952* .

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. In *Proceedings of the 19th Conference on Computational Language Learning*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

David M Perlmutter. 1978. Impersonal passives and the unaccusative hypothesis. In *annual meeting of the Berkeley Linguistics Society*. volume 4, pages 157–190.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 89–101. http://aclweb.org/anthology/D17-1009.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming* 24(1–2):3–36.

Rik van Noord and Johan Bos. 2017a. Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*.

Rik van Noord and Johan Bos. 2017b. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal* .

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2014. Grammar as a foreign language. *CoRR* abs/1412.7449.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A Transition-based Algorithm for AMR Parsing. In *Proceedings of NAACL-HLT*.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1214–1223. http://aclweb.org/anthology/D17-1125.

Sequence-to-sequence Models for Cache Transition Systems

Xiaochang Peng[1], **Linfeng Song**[1], **Daniel Gildea**[1], **Giorgio Satta**[2]
[1]University of Rochester [2]University of Padua
{xpeng,lsong10,gildea}@cs.rochester.edu,
satta@dei.unipd.it

Abstract

In this paper, we present a sequence-to-sequence based approach for mapping natural language sentences to AMR semantic graphs. We transform the sequence to graph mapping problem to a word sequence to transition action sequence problem using a special transition system called a cache transition system. To address the sparsity issue of neural AMR parsing, we feed feature embeddings from the transition state to provide relevant local information for each decoder state. We present a monotonic hard attention model for the transition framework to handle the strictly left-to-right alignment between each transition state and the current buffer input focus. We evaluate our neural transition model on the AMR parsing task, and our parser outperforms other sequence-to-sequence approaches and achieves competitive results in comparison with the best-performing models.[1]

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism where the meaning of a sentence is encoded as a rooted, directed graph. Figure 1 shows an example of an AMR in which the nodes represent the AMR concepts and the edges represent the relations between the concepts. AMR has been used in various applications such as text summarization (Liu et al., 2015), sentence compression (Takase et al., 2016), and event extraction (Huang et al., 2016).

[1]The implementation of our parser is available at https://github.com/xiaochang13/CacheTransition-Seq2seq

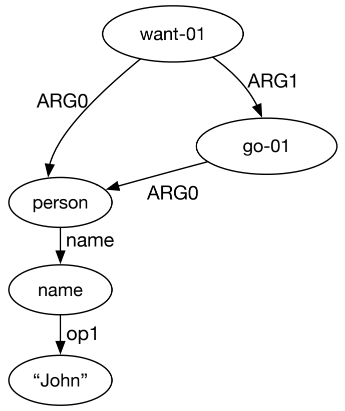


Figure 1: An example of AMR graph representing the meaning of: "John wants to go"

The task of AMR graph parsing is to map natural language strings to AMR semantic graphs. Different parsers have been developed to tackle this problem (Flanigan et al., 2014; Wang et al., 2015b,a; Peng et al., 2015; Artzi et al., 2015; Pust et al., 2015; van Noord and Bos, 2017). On the other hand, due to the limited amount of labeled data and the large output vocabulary, the sequence-to-sequence model has not been very successful on AMR parsing. Peng et al. (2017) propose a linearization approach that encodes labeled graphs as sequences. To address the data sparsity issue, low-frequency entities and tokens are mapped to special categories to reduce the vocabulary size for the neural models. Konstas et al. (2017) use self-training on a huge amount of unlabeled text to lower the out-of-vocabulary rate. However, the final performance still falls behind the best-performing models.

The best performing AMR parsers model graph structures directly. One approach to modeling graph structures is to use a transition system to build graphs step by step, as shown by the system of Wang and Xue (2017), which is currently the top performing system. This raises the question of whether the advantages of neural and transition-based system can be combined, as for example with the syntactic parser of Dyer et al. (2015), who use stack LSTMs to capture action history information in the transition state of the transition system. Ballesteros and Al-Onaizan (2017) apply stack-LSTM to transition-based AMR parsing and achieve competitive results, which shows that local transition state information is important for predicting transition actions.

Instead of linearizing the target AMR graph to a sequence structure, Buys and Blunsom (2017) propose a *sequence-to-action-sequence* approach where the reference AMR graph is replaced with an action derivation sequence by running a deterministic *oracle* algorithm on the training sentence, AMR graph pairs. They use a separate alignment probability to explicitly model the hard alignment from graph nodes to sentence tokens in the buffer.

Gildea et al. (2018) propose a special transition framework called a cache transition system to generate the set of semantic graphs. They adapt the stack-based parsing system by adding a working set, which they refer to as a cache, to the traditional stack and buffer. Peng et al. (2018) apply the cache transition system to AMR parsing and design refined action phases, each modeled with a separate feedforward neural network, to deal with some practical implementation issues.

In this paper, we propose a sequence-to-action-sequence approach for AMR parsing with cache transition systems. We want to take advantage of the sequence-to-sequence model to encode whole-sentence context information and the history action sequence, while using the transition system to constrain the possible output. The transition system can also provide better local context information than the linearized graph representation, which is important for neural AMR parsing given the limited amount of data.

More specifically, we use bi-LSTM to encode two levels of input information for AMR parsing: word level and concept level, each refined with more general category information such as lemmatization, POS tags, and concept categories.

We also want to make better use of the complex transition system to address the data sparsity issue for neural AMR parsing. We extend the hard attention model of Aharoni and Goldberg (2017), which deals with the nearly-monotonic alignment in the morphological inflection task, to the more general scenario of transition systems where the input buffer is processed from left-to-right. When we process the buffer in this ordered manner, the sequence of target transition actions are also strictly aligned left-to-right according to the input order. On the decoder side, we augment the prediction of output action with embedding features from the current transition state. Our experiments show that encoding information from the transition state significantly improves sequence-to-sequence models for AMR parsing.

2 Cache Transition Parser

We adopt the transition system of Gildea et al. (2018), which has been shown to have good coverage of the graphs found in AMR.

A **cache transition parser** consists of a stack, a cache, and an input buffer. The stack is a sequence σ of (integer, concept) pairs, as explained below, with the topmost element always at the rightmost position. The buffer is a sequence of ordered concepts β containing a suffix of the input concept sequence, with the first element to be read as a newly introduced concept/vertex of the graph. (We use the terms concept and vertex interchangeably in this paper.) Finally, the cache is a sequence of concepts $\eta = [v_1, \dots, v_m]$. The element at the leftmost position is called the first element of the cache, and the element at the rightmost position is called the last element.

Operationally, the functioning of the parser can be described in terms of configurations and transitions. A **configuration** of our parser has the form:

$$C = (\sigma, \eta, \beta, G_p)$$

where σ, η and β are as described above, and G_p is the partial graph that has been built so far. The initial configuration of the parser is $([], [\$, \dots, \$], [c_1, \dots, c_n], \emptyset)$, meaning that the stack and the partial graph are initially empty, and the cache is filled with m occurrences of the special symbol $\$$. The buffer is initialized with all the graph vertices constrained by the order of the input sentence. The final configuration is $([], [\$, \dots, \$], [], G)$, where the stack and the cache are as in the initial configuration and the buffer is empty. The constructed graph is the target AMR graph.

stack	cache	buffer	edges	actions taken
[]	[\$, \$, \$]	[Per, want-01, go-01]	$\emptyset$	—
[1, \$]	[\$, \$, Per]	[want-01, go-01]	$\emptyset$	*Shift*; *PushIndex*(1)
[1, \$]	[\$, \$, Per]	[want-01, go-01]	$\emptyset$	*Arc(1, -, NULL)*; *Arc(2, -, NULL)*
[1, \$, 1, \$]	[\$, Per, want-01]	[go-01]	$\emptyset$	*Shift*; *PushIndex*(1)
[1, \$, 1, \$]	[\$, Per, want-01]	[go-01]	E_1	*Arc(1, -, NULL)*; *Arc(2, L, ARG0)*
[1, \$, 1, \$, 1, \$]	[Per, want-01, go-01]	[]	E_1	*Shift*; *PushIndex*(1)
[1, \$, 1, \$, 1, \$]	[Per, want-01, go-01]	[]	E_2	*Arc(1, L, ARG0)*; *Arc(2, R, ARG1)*
[1, \$, 1, \$]	[\$, Per, want-01]	[]	E_2	*Pop*
[1, \$]	[\$, \$, Per]	[]	E_2	*Pop*
[]	[\$, \$, \$]	[]	E_2	*Pop*

Figure 2: Example run of the cache transition system constructing the graph for the sentence "John wants to go" with cache size of 3. The left four columns show the parser configurations after taking the actions shown in the last column. $E_1 = \{(Per, want\text{-}01, L\text{-}ARG0)\}$, $E_2 = \{(Per, want\text{-}01, L\text{-}ARG0), (Per, go\text{-}01, L\text{-}ARG0), (want\text{-}01, go\text{-}01, R\text{-}ARG1)\}$.

In the first step, which is called concept identification, we map the input sentence $w_{1:n'} = w_1, \ldots, w_{n'}$ to a sequence of concepts $c_{1:n} = c_1, \ldots, c_n$. We decouple the problem of concept identification from the transition system and initialize the buffer with a recognized concept sequence from another classifier, which we will introduce later. As the sequence-to-sequence model uses all possible output actions as the target vocabulary, this can significantly reduce the target vocabulary size. The **transitions** of the parser are specified as follows.

1. *Pop* pops a pair (i, v) from the stack, where the integer i records the position in the cache that it originally came from. We place concept v in position i in the cache, shifting the remainder of the cache one position to the right, and discarding the last element in the cache.
2. *Shift* signals that we will start processing the next input concept, which will become a new vertex in the output graph.
3. *PushIndex*(i) shifts the next input concept out of the buffer and moves it into the last position of the cache. We also take out the concept v_i appearing at position i in the cache and push it onto the stack σ, along with the integer i recording its original position in the cache.[2]
4. *Arc(i, d, l)* builds an arc with direction d and label l between the rightmost concept and the i-th concept in the cache. The label l is *NULL* if no arc is made and we use the action *NOARC* in this case. Otherwise we decompose the arc decision into two actions *ARC* and *d-l*. We consider all arc decisions between the rightmost cache concept and each of the other concepts in the cache. We can consider this phase as first making a binary decision whether there is an arc, and then predicting the label in case there is one, between each concept pair.

[2]Our transition design is different from Peng et al. (2018) in two ways: the PushIndex phase is initiated before making all the arc decisions; the newly introduced concept is placed at the last cache position instead of the leftmost buffer position, which essentially increases the cache size by 1.

Given the sentence "John wants to go" and the recognized concept sequence "Per want-01 go-01" (person name category *Per* for "John"), our cache transition parser can construct the AMR graph shown in Figure 1 using the run shown in Figure 2 with cache size of 3.

2.1 Oracle Extraction Algorithm

We use the following **oracle** algorithm (Nivre, 2008) to derive the sequence of actions that leads to the gold AMR graph for a cache transition parser with cache size m. The correctness of the oracle is shown by Gildea et al. (2018).

Let E_G be the set of edges of the gold graph G. We maintain the set of vertices that is not yet shifted into the cache as S, which is initialized with all vertices in G. The vertices are ordered according to their aligned position in the word sequence and the unaligned vertices are listed according to their order in the depth-first traversal of the graph. The oracle algorithm can look into

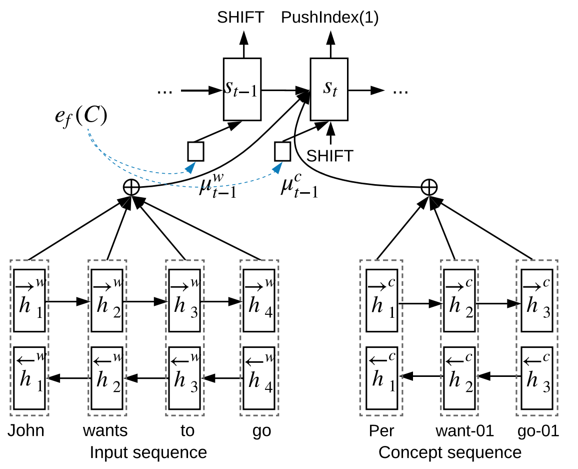


Figure 3: Sequence-to-sequence model with soft attention, encoding a word sequence and concept sequence separately by two BiLSTM encoders.

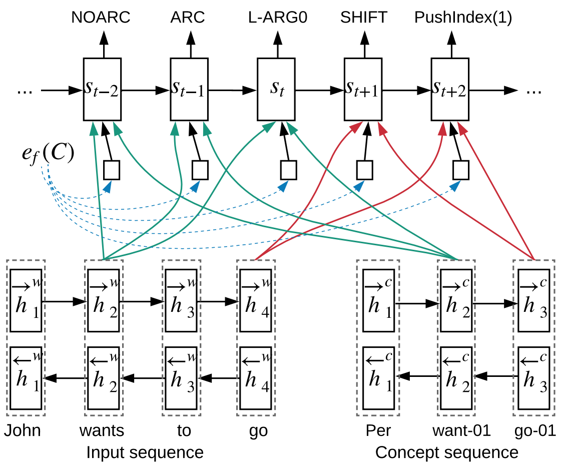


Figure 4: Sequence-to-sequence model with monotonic hard attention. Different colors show the changes of hard attention focus.

E_G to decide which transition to take next, or else to decide that it should fail. This decision is based on the mutually exclusive rules listed below.

1. ShiftOrPop phase: the oracle chooses transition *Pop*, in case there is no edge (v_m, v) in E_G such that vertex v is in S, or chooses transition *Shift* and proceeds to the next phase.
2. PushIndex phase: in this phase, the oracle first chooses a position i (as explained below) in the cache to place the candidate concept and removes the vertex at this position and places its index, vertex pair onto the stack. The oracle chooses transition *PushIndex(i)* and proceeds to the next phase.
3. ArcBinary, ArcLabel phases: between the rightmost cache concept and each concept in the cache, we make a binary decision about whether there is an arc between them. If there is an arc, the oracle chooses its direction and label. After arc decisions to $m-1$ cache concepts are made, we jump to the next step.
4. If the stack and buffer are both empty, and the cache is in the initial state, the oracle finishes with success, otherwise we proceed to the first step.

We use the equation below to choose the cache concept to take out in the step *PushIndex(i)*. For $j \in [|\beta|]$, we write β_j to denote the j-th vertex in β. We choose a vertex v_{i^*} in η such that:

$$i^* = \underset{i \in [m]}{\operatorname{argmax}} \min \{j \mid (v_i, \beta_j) \in E_G\}$$

In words, v_{i^*} is the concept from the cache whose closest neighbor in the buffer β is furthest forward in β. We move out of the cache vertex v_{i^*} and push it onto the stack, for later processing.

For each training example $(x_{1:n}, g)$, the transition system generates the output AMR graph g from the input sequence $x_{1:n}$ through an oracle sequence $a_{1:q} \in \Sigma_a^*$, where Σ_a is the union of all possible actions. We model the probability of the output with the action sequence:

$$P(a_{1:q}|x_{1:n}) = \prod_{t=1}^{q} P(a_t|a_1, \ldots, a_{t-1}, x_{1:n}; \theta)$$

which we estimate using a sequence-to-sequence model, as we will describe in the next section.

3 Soft vs Hard Attention for Sequence-to-action-sequence

Shown in Figure 3, our sequence-to-sequence model takes a word sequence $w_{1:n'}$ and its mapped concept sequence $c_{1:n}$ as the input, and the action sequence $a_{1:q}$ as the output. It uses two BiLSTM encoders, each encoding an input sequence. As the two encoders have the same structure, we only introduce the encoder for the word sequence in detail below.

3.1 BiLSTM Encoder

Given an input word sequence $w_{1:n'}$, we use a bidirectional LSTM to encode it. At each step j, the current hidden states $\overleftarrow{h}^w_j$ and $\overrightarrow{h}^w_j$ are generated from the previous hidden states $\overleftarrow{h}^w_{j+1}$ and $\overrightarrow{h}^w_{j-1}$, and the representation vector x_j of the current input word w_j:

$$\overleftarrow{h}^{w}_{j} = \text{LSTM}(\overleftarrow{h}^{w}_{j+1}, x_j)$$
$$\overrightarrow{h}^{w}_{j} = \text{LSTM}(\overrightarrow{h}^{w}_{j-1}, x_j)$$

The representation vector x_j is the concatenation of the embeddings of its word, lemma, and POS tag, respectively. Then the hidden states of both directions are concatenated as the final hidden state for word w_j:

$$h^w_j = [\overleftarrow{h}^{w}_{j}; \overrightarrow{h}^{w}_{j}]$$

Similarly, for the concept sequence, the final hidden state for concept c_j is:

$$h^c_j = [\overleftarrow{h}^{c}_{j}; \overrightarrow{h}^{c}_{j}]$$

3.2 LSTM Decoder with Soft Attention

We use an attention-based LSTM decoder (Bahdanau et al., 2014) with two attention memories H_w and H_c, where H_w is the concatenation of the state vectors of all input words, and H_c for input concepts correspondingly:

$$H_w = [h^w_1; h^w_2; \ldots; h^w_{n'}] \quad (1)$$
$$H_c = [h^c_1; h^c_2; \ldots; h^c_n] \quad (2)$$

The decoder yields an action sequence $a_1, a_2, \ldots, a_q$ as the output by calculating a sequence of hidden states $s_1, s_2 \ldots, s_q$ recurrently. While generating the t-th output action, the decoder considers three factors: (1) the previous hidden state of the LSTM model s_{t-1}; (2) the embedding of the previous generated action e_{t-1}; and (3) the previous context vectors for words μ^w_{t-1} and concepts μ^c_{t-1}, which are calculated using H_w and H_c, respectively. When $t = 1$, we initialize μ_0 as a zero vector, and set e_0 to the embedding of the start token "$\langle s \rangle$". The hidden state s_0 is initialized as:

$$s_0 = W_d[\overleftarrow{h}^{w}_{1}; \overrightarrow{h}^{w}_{n}; \overleftarrow{h}^{c}_{1}; \overrightarrow{h}^{c}_{n}] + b_d,$$

where W_d and b_d are model parameters.

For each time-step t, the decoder feeds the concatenation of the embedding of previous action e_{t-1} and the previous context vectors for words μ^w_{t-1} and concepts μ^c_{t-1} into the LSTM model to update its hidden state.

$$s_t = \text{LSTM}(s_{t-1}, [e_{t-1}; \mu^w_{t-1}; \mu^c_{t-1}]) \quad (3)$$

Then the attention probabilities for the word sequence and the concept sequence are calculated similarly. Take the word sequence as an example, $\alpha^w_{t,i}$ on $h^w_i \in H_w$ for time-step t is calculated as:

$$\epsilon_{t,i} = v_c^T \tanh(W_h h^w_i + W_s s_t + b_c)$$
$$\alpha^w_{t,i} = \frac{\exp(\epsilon_{t,i})}{\sum_{j=1}^{N} \exp(\epsilon_{t,j})}$$

W_h, W_s, v_c and b_c are model parameters. The new context vector $\mu^w_t = \sum_{i=1}^{n} \alpha^w_{t,i} h^w_i$. The calculation of μ^c_t follows the same procedure, but with a different set of model parameters.

The output probability distribution over all actions at the current state is calculated by:

$$P_{\Sigma_a} = \text{softmax}(V_a[s_t; \mu^w_t; \mu^c_t] + b_a), \quad (4)$$

where V_a and b_a are learnable parameters, and the number of rows in V_a represents the number of all actions. The symbol Σ_a is the set of all actions.

3.3 Monotonic Hard Attention for Transition Systems

When we process each buffer input, the next few transition actions are closely related to this input position. The buffer maintains the order information of the input sequence and is processed strictly left-to-right, which essentially encodes a monotone alignment between the transition action sequence and the input sequence.

As we have generated a concept sequence from the input word sequence, we maintain two hard attention pointers, l_w and l_c, to model monotonic attention to word and concept sequences respectively. The update to the decoder state now relies on a single position of each input sequence in contrast to Equation 3:

$$s_t = \text{LSTM}(s_{t-1}, [e_{t-1}; h^w_{l_w}; h^c_{l_c}]) \quad (5)$$

Control Mechanism. Both pointers are initialized as 0 and advanced to the next position deterministically. We move the concept attention focus l_c to the next position after arc decisions to all the other $m - 1$ cache concepts are made. We move the word attention focus l_w to its aligned position in case the new concept is aligned, otherwise we don't move the word focus. As shown in Figure 4, after we have made arc decisions from concept *want-01* to the other cache concepts, we move the concept focus to the next concept *go-01*. As this concept is aligned, we move the word focus to its aligned position *go* in the word sequence and skip the unaligned word *to*.

3.4 Transition State Features for Decoder

Another difference of our model with Buys and Blunsom (2017) is that we extract features from the current transition state configuration C_t:

$$e_f(C_t) = [e_{f_1}(C_t); e_{f_2}(C_t); \cdots ; e_{f_l}(C_t)]$$

where l is the number of features extracted from C_t and $e_{f_k}(C_t)$ $(k = 1, \ldots, l)$ represents the embedding for the k-th feature, which is learned during training. These feature embeddings are concatenated as $e_f(C_t)$, and fed as additional input to the decoder. For the soft attention decoder:

$$s_t = \text{LSTM}(s_{t-1}, [e_{t-1}; \mu^w_{t-1}; \mu^c_{t-1}; e_f(C_t)])$$

and for the hard attention decoder:

$$s_t = \text{LSTM}(s_{t-1}, [e_{t-1}; h^w_{l_w}; h^c_{l_c}; e_f(C_t)])$$

We use the following features in our experiments:

1. Phase type: indicator features showing which phase the next transition is.
2. ShiftOrPop features: *token features*[3] for the rightmost cache concept and the leftmost buffer concept. Number of dependencies to words on the right, and the top three dependency labels for them.
3. ArcBinary or ArcLabel features: token features for the rightmost concept and the current cache concept it makes arc decisions to. Word, concept and dependency distance between the two concepts. The labels for the two most recent outgoing arcs for these two concepts and their first incoming arc and the number of incoming arcs. Dependency label between the two positions if there is a dependency arc between them.
4. PushIndex features: token features for the leftmost buffer concept and all the concepts in the cache.

The phase type features are deterministic from the last action output. For example, if the last action output is *Shift*, the current phase type would be *PushIndex*. We only extract corresponding features for this phase and fill all the other feature types with *-NULL-* as placeholders. The features for other phases are similar.

[3]Concept, concept category at the specified position in concept sequence. And the word, lemma, POS tag at the aligned input position.

4 AMR Parsing

4.1 Training and Decoding

We train our models using the cross-entropy loss, over each oracle action sequence $a^*_1, \ldots, a^*_q$:

$$L = -\sum_{t=1}^{q} \log P(a^*_t | a^*_1, \ldots, a^*_{t-1}, X; \theta), \quad (6)$$

where X represents the input word and concept sequences, and θ is the model parameters. Adam (Kingma and Ba, 2014) with a learning rate of 0.001 is used as the optimizer, and the model that yields the best performance on the dev set is selected to evaluate on the test set. Dropout with rate 0.3 is used during training. Beam search with a beam size of 10 is used for decoding. Both training and decoding use a Tesla K20X GPU.

Hidden state sizes for both encoder and decoder are set to 100. The word embeddings are initialized from Glove pretrained word embeddings (Pennington et al., 2014) on Common Crawl, and are not updated during training. The embeddings for POS tags and features are randomly initialized, with the sizes of 20 and 50, respectively.

4.2 Preprocessing and Postprocessing

As the AMR data is very sparse, we collapse some subgraphs or spans into categories based on the alignment. We define some special categories such as named entities (*NE*), dates (*DATE*), single rooted subgraphs involving multiple concepts (*MULT*)[4], numbers (*NUMBER*) and phrases (*PHRASE*). The phrases are extracted based on the multiple-to-one alignment in the training data. One example phrase is *more than* which aligns to a single concept *more-than*. We first collapse spans and subgraphs into these categories based on the alignment from the JAMR aligner (Flanigan et al., 2014), which greedily aligns a span of words to AMR subgraphs using a set of heuristics. This categorization procedure enables the parser to capture mappings from continuous spans on the sentence side to connected subgraphs on the AMR side.

We use the semi-Markov model from Flanigan et al. (2016) as the concept identifier, which jointly segments the sentence into a sequence of spans and maps each span to a subgraph. During decoding, our output has categories, and we need to map

[4]For example, verbalization of "teacher" as "(person :ARG0-of teach-01)", or "minister" as "(person :ARG0-of (have-org-role-91 :ARG2 minister))".

	ShiftOrPop	PushIndex	ArcBinary	ArcLabel
Peng et al. (2018)	0.87	**0.87**	0.83	**0.81**
Soft+feats	0.93	0.84	0.91	0.75
Hard+feats	**0.94**	0.85	**0.93**	0.77

Table 1: Performance breakdown of each transition phase.

each category to the corresponding AMR concept or subgraph. We save a table Q which shows the original subgraph each category is collapsed from, and map each category to its original subgraph representation. We also use heuristic rules to generate the target-side AMR subgraph representation for *NE*, *DATE*, and *NUMBER* based on the source side tokens.

5 Experiments

We evaluate our system on the released dataset (LDC2015E86) for SemEval 2016 task 8 on meaning representation parsing (May, 2016). The dataset contains 16,833 training, 1,368 development, and 1,371 test sentences which mainly cover domains like newswire, discussion forum, etc. All parsing results are measured by Smatch (version 2.0.2) (Cai and Knight, 2013).

5.1 Experiment Settings

We categorize the training data using the automatic alignment and dump a template for date entities and frequent phrases from the multiple to one alignment. We also generate an alignment table from tokens or phrases to their candidate target-side subgraphs. For the dev and test data, we first extract the named entities using the Illinois Named Entity Tagger (Ratinov and Roth, 2009) and extract date entities by matching spans with the date template. We further categorize the dataset with the categories we have defined. After categorization, we use Stanford CoreNLP (Manning et al., 2014) to get the POS tags and dependencies of the categorized dataset. We run the oracle algorithm separately for training and dev data (with alignment) to get the statistics of individual phases. We use a cache size of 5 in our experiments.

5.2 Results

Individual Phase Accuracy We first evaluate the prediction accuracy of individual phases on the dev oracle data assuming gold prediction history. The four transition phases *ShiftOrPop*, *PushIndex*, *ArcBinary*, and *ArcLabel* account for 25%, 12.5%, 50.1%, and 12.4% of the total transition actions respectively. Table 1 shows the phase-wise accuracy of our sequence-to-sequence model. Peng et al. (2018) use a separate feedforward network to predict each phase independently. We use the same alignment from the SemEval dataset as in Peng et al. (2018) to avoid differences resulting from the aligner. *Soft+feats* shows the performance of our sequence-to-sequence model with soft attention and transition state features, while *Hard+feats* is using hard attention. We can see that the hard attention model outperforms the soft attention model in all phases, which shows that the single-pointer attention finds more relevant information than the soft attention on the relatively small dataset. The sequence-to-sequence models perform better than the feedforward model of Peng et al. (2018) on *ShiftOrPop* and *ArcBinary*, which shows that the whole-sentence context information is important for the prediction of these two phases. On the other hand, the sequence-to-sequence models perform worse than the feedforward models on *PushIndex* and *ArcLabel*. One possible reason is that the model tries to optimize the overall accuracy, while these two phases account for fewer than 25% of the total transition actions and might be less attended to during the update.

Impact of Different Components Table 2 shows the impact of different components for the sequence-to-sequence model. We can see that the transition state features play a very important role for predicting the correct transition action. This is because different transition phases have very different prediction behaviors and need different types of local information for the prediction. Relying on the sequence-to-sequence model alone does not perform well in disambiguating these choices, while the transition state can enforce direct constraints. We can also see that while the hard attention only attends to one position of the input, it performs slightly better than the soft attention model, while the time complexity is lower.

Impact of Different Cache Sizes The cache size of the transition system can be optimized as a trade-off between coverage of AMR graphs and the prediction accuracy. While larger cache size increases the coverage of AMR graphs, it complicates the prediction procedure with more cache decisions to make. From Table 3 we can see that

System	P	R	F
Soft	0.55	0.51	0.53
Soft+feats	0.69	0.63	0.66
Hard+feats	0.70	0.64	0.67

Table 2: Impact of various components for the sequence-to-sequence model (dev).

Cache Size	P	R	F
4	0.69	0.63	0.66
5	0.70	0.64	0.67
6	0.69	0.64	0.66

Table 3: Impact of cache size for the sequence-to-sequence model, hard attention (dev).

the hard attention model performs best with cache size 5. The soft attention model also achieves best performance with the same cache size.

Comparison with other Parsers Table 4 shows the comparison with other AMR parsers. The first three systems are some competitive neural models. We can see that our parser significantly outperforms the sequence-to-action-sequence model of Buys and Blunsom (2017). Konstas et al. (2017) use a linearization approach that linearizes the AMR graph to a sequence structure and use self-training on 20M unlabeled Gigaword sentences. Our model achieves better results without using additional unlabeled data, which shows that relevant information from the transition system is very useful for the prediction. Our model also outperforms the stack-LSTM model by Ballesteros and Al-Onaizan (2017), while their model is evaluated on the previous release of LDC2014T12.

System	P	R	F
Buys and Blunsom (2017)	–	–	0.60
Konstas et al. (2017)	0.60	0.65	0.62
Ballesteros and Al-Onaizan (2017)*	–	–	0.64
Damonte et al. (2017)	–	–	0.64
Peng et al. (2018)	0.69	0.59	0.64
Wang et al. (2015b)	0.64	0.62	0.63
Wang et al. (2015a)	0.70	0.63	0.66
Flanigan et al. (2016)	0.70	0.65	0.67
Wang and Xue (2017)	0.72	0.65	0.68
Ours soft attention	0.68	0.63	0.65
Ours hard attention	0.69	0.64	0.66

Table 4: Comparison to other AMR parsers. *Model has been trained on the previous release of the corpus (LDC2014T12).

System	P	R	F
Peng et al. (2018)	0.44	0.28	0.34
Damonte et al. (2017)	–	–	0.41
JAMR	0.47	0.38	0.42
Ours	0.58	0.34	0.43

Table 5: Reentrancy statistics.

We also show the performance of some of the best-performing models. While our hard attention achieves slightly lower performance in comparison with Wang et al. (2015a) and Wang and Xue (2017), it is worth noting that their approaches of using WordNet, semantic role labels and word cluster features are complimentary to ours. The alignment from the aligner and the concept identification identifier also play an important role for improving the performance. Wang and Xue (2017) propose to improve AMR parsing by improving the alignment and concept identification, which can also be combined with our system to improve the performance of a sequence-to-sequence model.

Dealing with Reentrancy Reentrancy is an important characteristic of AMR, and we evaluate the Smatch score only on the reentrant edges following Damonte et al. (2017). From Table 5 we can see that our hard attention model significantly outperforms the feedforward model of Peng et al. (2018) in predicting reentrancies. This is because predicting reentrancy is directly related to the *ArcBinary* phase of the cache transition system since it decides to make multiple arc decisions to the same vertex, and we can see from Table 1 that the hard attention model has significantly better prediction accuracy in this phase. We also compare the reentrancy results of our transition system with two other systems, Damonte et al. (2017) and JAMR, where these statistics are available. From Table 5, we can see that our cache transition system slightly outperforms these two systems in predicting reentrancies.

Figure 5 shows a reentrancy example where JAMR and the feedforward network of Peng et al. (2018) do not predict well, while our system predicts the correct output. JAMR fails to predict the reentrancy arc from *desire-01* to *i*, and connects the wrong arc from "live-01" to "-" instead of from "desire-01". The feedforward model of Peng et al. (2018) fails to predict the two arcs from *desire-01*

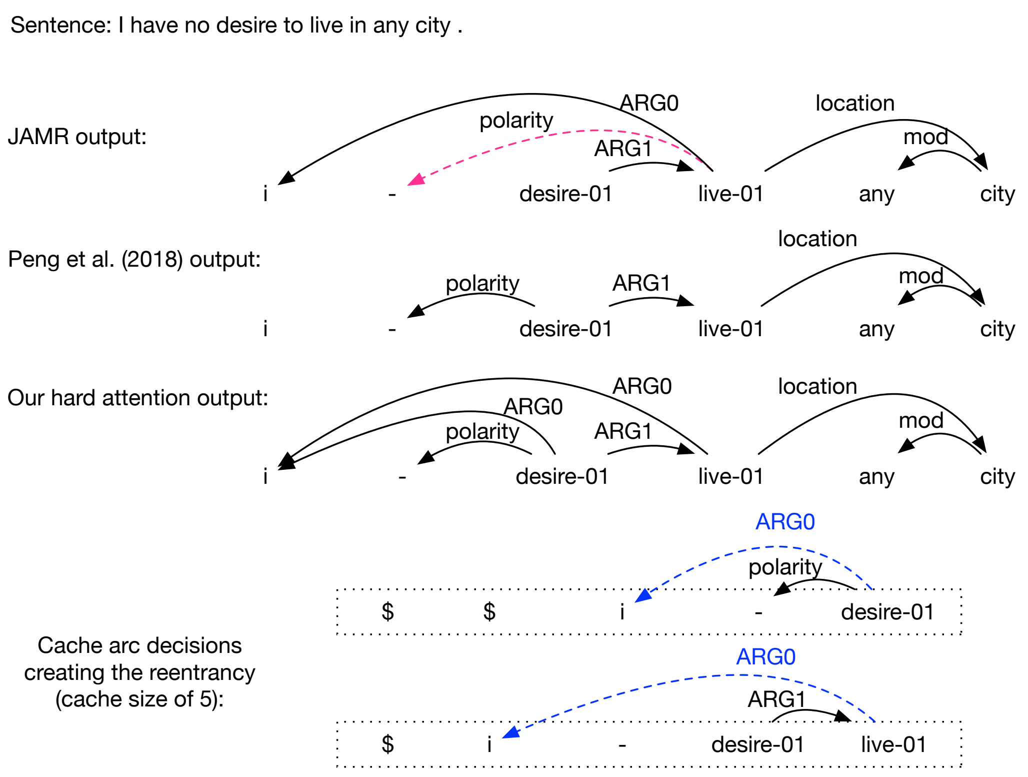

Sentence: I have no desire to live in any city .
JAMR output:
Peng et al. (2018) output:
Our hard attention output:
Cache arc decisions creating the reentrancy (cache size of 5):
ARG0
polarity
ARG1
location
mod
i
-
desire-01
live-01
any
city
$

Figure 5: An example showing how our system predicts the correct reentrancy.

and *live-01* to *i*. This error is because their feed-forward ArcBinary classifier does not model long-term dependency and usually prefers making arcs between words that are close and not if they are distant. Our classifier, which encodes both word and concept sequence information, can accurately predict the reentrancy through the two arc decisions shown in Figure 5. When *desire-01* and *live-01* are shifted into the cache respectively, the transition system makes a left-going arc from each of them to the same concept *i*, thus creating the reentrancy as desired.

6 Conclusion

In this paper, we have presented a sequence-to-action-sequence approach for cache transition systems and applied it to AMR parsing. To address the data sparsity issue for neural AMR parsing, we show that the transition state features are very helpful in constraining the possible output and improving the performance of sequence-to-sequence models. We also show that the monotonic hard attention model can be generalized to the transition-based framework and outperforms the soft attention model when limited data is available. While we are focused on AMR parsing in this paper, in future work our cache transition system and the presented sequence-to-sequence models can be potentially applied to other semantic graph parsing tasks (Oepen et al., 2015; Du et al., 2015; Zhang et al., 2016; Cao et al., 2017).

Acknowledgments

We gratefully acknowledge the assistance of Hao Zhang from Google, New York for the monotonic hard attention idea and the helpful comments and suggestions.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1215–1226.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 748–752.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2110–2120.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 536–546.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 927–931.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 334–343.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 1426–1436, Baltimore, Maryland.

Daniel Gildea, Giorgio Satta, and Xiaochang Peng. 2018. Cache transition systems for graph parsing. *Computational Linguistics*, 44(1):85–118.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 258–268.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *arXiv preprint arXiv:1705.09980*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado.

Xiaochang Peng, Daniel Gildea, and Giorgio Satta. 2018. AMR parsing with cache transition systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-18)*.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL-15)*, pages 32–41, Beijing, China.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the European Chapter of the ACL (EACL-17)*.

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

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154, Lisbon, Portugal. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of AMR parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 857–862, Beijing, China.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-15)*, pages 366–375, Denver, Colorado.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics*, 42(3):353–389.

Batch IS NOT Heavy: Learning Word Representations From All Samples

*Xin Xin[1], *Fajie Yuan[1], Xiangnan He[2], Joemon M.Jose[1]
[1]School of Computing Science, University of Glasgow, UK
[2]School of Computing, National University of Singapore
{x.xin.1,f.yuan.1}@research.gla.ac.uk, xiangnanhe@gmail.com

Abstract

Stochastic Gradient Descent (SGD) with negative sampling is the most prevalent approach to learn word representations. However, it is known that sampling methods are biased especially when the sampling distribution deviates from the true data distribution. Besides, SGD suffers from dramatic fluctuation due to the one-sample learning scheme. In this work, we propose AllVec that uses batch gradient learning to generate word representations from *all* training samples. Remarkably, the time complexity of AllVec remains at the same level as SGD, being determined by the number of positive samples rather than all samples. We evaluate AllVec on several benchmark tasks. Experiments show that AllVec outperforms sampling-based SGD methods with comparable efficiency, especially for small training corpora.

1 Introduction

Representing words using dense and real-valued vectors, aka word embeddings, has become the cornerstone for many natural language processing (NLP) tasks, such as document classification (Sebastiani, 2002), parsing (Huang et al., 2012), discourse relation recognition (Lei et al., 2017) and named entity recognition (Turian et al., 2010). Word embeddings can be learned by optimizing that words occurring in similar contexts have similar embeddings, i.e. the well-known distributional hypothesis (Harris, 1954). A representative method is skip-gram (SG) (Mikolov et al., 2013a,b), which realizes the hypothesis using a shallow neural network model. The other family of methods is count-based, such as GloVe (Pennington et al., 2014) and LexVec (Salle et al., 2016a,b), which exploit low-rank models such as matrix factorization (MF) to learn embeddings by reconstructing the word co-occurrence statistics.

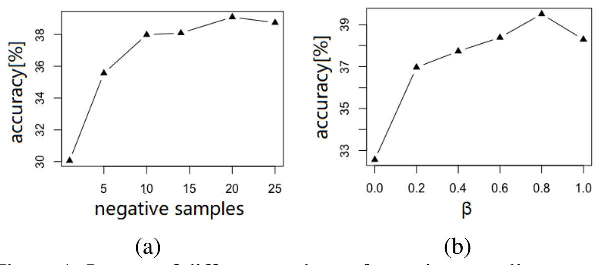


Figure 1: Impact of different settings of negative sampling on skip-gram for the word analogy task on Text8. Clearly, the accuracy depends largely on (a) the sampling size of negative words, and (b) the sampling distribution ($\beta = 0$ means the uniform distribution and $\beta = 1$ means the word frequency distribution).

By far, most state-of-the-art embedding methods rely on SGD and negative sampling for optimization. However, the performance of SGD is highly sensitive to the sampling distribution and the number of negative samples (Chen et al., 2018; Yuan et al., 2016), as shown in Figure 1. Essentially, sampling is biased, making it difficult to converge to the same loss with all examples, regardless of how many update steps have been taken. Moreover, SGD exhibits dramatic fluctuation and suffers from overshooting on local minimums (Ruder, 2016). These drawbacks of SGD can be attributed to its one-sample learning scheme, which updates parameters based on one training sample in each step.

To address the above-mentioned limitations of SGD, a natural solution is to perform exact (full) batch learning. In contrast to SGD, batch learning does not involve any sampling procedure and computes the gradient over all training samples. As such, it can easily converge to a better optimum in a more stable way. Nevertheless, a well-known

*The first two authors contributed equally to this paper and share the first-authorship.

difficulty in applying full batch learning lies in the expensive computational cost for large-scale data. Taking the word embedding learning as an example, if the vocabulary size is $|V|$, then evaluating the loss function and computing the full gradient takes $O(|V|^2 k)$ time, where k is the embedding size. This high complexity is unaffordable in practice, since $|V|^2$ can easily reach billion level or even higher.

In this paper, we introduce AllVec, an exact and efficient word embedding method based on full batch learning. To address the efficiency challenge in learning from all training samples, we devise a regression-based loss function for word embedding, which allows fast optimization with memorization strategies. Specifically, the acceleration is achieved by reformulating the expensive loss over all negative samples using a *partition* and a *decouple* operation. By decoupling and caching the bottleneck terms, we succeed to use all samples for each parameter update in a manageable time complexity which is mainly determined by the positive samples. The main contributions of this work are summarized as follows:

- We present a fine-grained weighted least square loss for learning word embeddings. Unlike GloVe, it explicitly accounts for all negative samples and reweights them with a frequency-aware strategy.
- We propose an efficient and exact optimization algorithm based on full batch gradient optimization. It has a comparable time complexity with SGD, but being more effective and stable due to the consideration of all samples in each parameter update.
- We perform extensive experiments on several benchmark datasets and tasks to demonstrate the effectiveness, efficiency, and convergence property of our AllVec method.

2 Related Work

2.1 Skip-gram with Negative Sampling

Mikolov et al. (2013a,b) proposed the skip-gram model to learn word embeddings. SG formulates the problem as a predictive task, aiming at predicting the proper context c for a target word w within a local window. To speed up the training process, it applies the negative sampling (Mikolov et al., 2013b) to approximate the full softmax. That is, each positive (w, c) pair is trained with n randomly sampled negative pairs (w, w_i). The sampled loss function of SG is defined as

$$L_{wc}^{SG} = \log \sigma(U_w \tilde{U}_c^T) + \sum_{i=1}^{n} \mathbf{E}_{w_i \sim P_n(w)} \log \sigma(-U_w \tilde{U}_{w_i}^T)$$

where U_w and $\tilde{U}_c$ denote the k-dimensional embedding vectors for word w and context c. $P_n(w)$ is the distribution from which negative context w_i is sampled.

Plenty of research has been done based on SG, such as the use of prior knowledge from another source (Kumar and Araki, 2016; Liu et al., 2015a; Bollegala et al., 2016), incorporating word type information (Cao and Lu, 2017; Niu et al., 2017), character level n-gram models (Bojanowski et al., 2016; Joulin et al., 2016) and jointly learning with topic models like LDA (Shi et al., 2017; Liu et al., 2015b).

2.2 Importance of the Sampling Distribution

Mikolov et al. (2013b) showed that the unigram distribution raised to the 3/4th power as $P_n(w)$ significantly outperformed both the unigram and the uniform distribution. This suggests that the sampling distribution (of negative words) has a great impact on the embedding quality. Furthermore, Chen et al. (2018) and Guo et al. (2018) recently found that replacing the original sampler with adaptive samplers could result in better performance. The adaptive samplers are used to find more informative negative examples during the training process. Compared with the original word-frequency based sampler, adaptive samplers adapt to both the target word and the current state of the model. They also showed that the fine-grained samplers not only speeded up the convergence but also significantly improved the embedding quality. Similar observations were also found in other fields like collaborative filtering (Yuan et al., 2016). While being effective, it is proven that negative sampling is a biased approximation and does not converges to the same loss as the full softmax — regardless of how many update steps have been taken (Bengio and Senécal, 2008; Blanc and Rendle, 2017).

2.3 Count-based Embedding Methods

Another line of research is the count-based embedding, such as GloVe (Pennington et al., 2014). GloVe performs a biased MF on the word-context co-occurrence statistics, which is a common approach in the field of collaborative filtering (Koren, 2008). However, GloVe only formulates the loss on positive entries of the co-occurrence matrix, meaning that negative signals about word-context co-occurrence are discarded. A remedy solution is LexVec (Salle et al., 2016a,b) which integrates negative sampling into MF. Some other methods (Li et al., 2015; Stratos et al., 2015; Ailem et al., 2017) also use MF to approximate the word-context co-occurrence statistics. Although predictive models and count-based models seem different at first glance, Levy and Goldberg (2014) proved that SG with negative sampling is implicitly factorizing a shifted pointwise mutual information (PMI) matrix, which means that the two families of embedding models resemble each other to a certain degree.

Our proposed method departs from all above methods by using the full batch gradient optimizer to learn from all (positive and negative) samples. We propose a fast learning algorithm to show that such batch learning is not "heavy" even with tens of billions of training examples.

3 AllVec Loss

In this work, we adopt the regression loss that is commonly used in count-based models (Pennington et al., 2014; Stratos et al., 2015; Ailem et al., 2017) to perform matrix factorization on word co-occurrence statistics. As highlighted, to retain the modeling fidelity, AllVec eschews using any sampling but optimizes the loss on all positive and negative word-context pairs.

Given a word w and a symmetric window of win contexts, the set of positive contexts can be obtained by sliding through the corpus. Let c denote a specific context, M_{wc} be the number of co-occurred (w,c) pairs in the corpus within the window. $M_{wc}=0$ means that the pair (w,c) has never been observed, i.e. the negative signal. r_{wc} is the association coefficient between w and c, which is calculated from M_{wc}. Specifically, we use r^{+}_{wc} to denote the ground truth value for positive (w,c) pairs and a constant value r^{-}(e.g., 0 or -1) for negative ones since there is no interaction between w and c in negative pairs. Finally, with all positive and negative pairs considered, a regular loss function can be given as Eq.(1), where V is the vocabulary and S is the set of positive pairs. α^{+}_{wc} and α^{-}_{wc} represent the weight for positive and negative (w,c) pairs, respectively.

$$L = \underbrace{\sum_{(w,c)\in S} \alpha^{+}_{wc}(r^{+}_{wc} - U_w\tilde{U}_c^T)^2}_{L_P} + \underbrace{\sum_{(w,c)\in(V\times V)\setminus S} \alpha^{-}_{wc}(r^{-} - U_w\tilde{U}_c^T)^2}_{L_N} \tag{1}$$

When it comes to r^{+}_{wc}, there are several choices. For example, GloVe applies the log of M_{wc} with bias terms for w and c. However, research from Levy and Goldberg (2014) showed that the SG model with negative sampling implicitly factorizes a shifted PMI matrix. The PMI value for a (w,c) pair can be defined as

$$PMI_{wc} = log\frac{P(w,c)}{P(w)P(c)} = log\frac{M_{wc}M_{**}}{M_{w*}M_{*c}} \tag{2}$$

where '*' denotes the summation of all corresponding indexes (e.g., $M_{w*}=\sum_{c\in V} M_{wc}$). Inspired by this connection, we set r^{+}_{wc} as the positive point-wise mutual information (PPMI) which has been commonly used in the NLP literature (Stratos et al., 2015; Levy and Goldberg, 2014). Sepcifically, PPMI is the positive version of PMI by setting the negative values to zero. Finally, r^{+}_{wc} is defined as

$$r^{+}_{wc} = PPMI_{wc} = \max(PMI_{wc}, 0) \tag{3}$$

3.1 Weighting Strategies

Regarding α^{+}_{wc}, we follow the design in GloVe, where it is defined as

$$\alpha^{+}_{wc} = \begin{cases} (M_{wc}/xmax)^{\rho} & M_{wc} < xmax \\ 1 & M_{wc} \geq xmax \end{cases} \tag{4}$$

As for the weight for negative instances α^{-}_{wc}, considering that there is no interaction between w and negative c, we set α^{-}_{wc} as α^{-}_{c} (or α^{-}_{w}), which means that the weight is determined by the word itself rather than the word-context interaction. Note that either $\alpha^{-}_{wc} = \alpha^{-}_{c}$ or $\alpha^{-}_{wc} = \alpha^{-}_{w}$ does not influence the complexity of AllVec learning algorithm described in the next section. The design of α^{-}_{c} is inspired by the frequency-based oversampling scheme in skip-gram and missing data reweighting in recommendation (He et al., 2016). The intuition is that a word with high frequency is more likely to be a true negative context word if there is no observed word-context interactions. Hence, to effectively differentiate the positive and negative examples, we assign a higher weight for the negative examples that have a higher word frequency, and a smaller weight for infrequent words. Formally, α^-_{wc} is defined as

$$\alpha^-_{wc} = \alpha^-_c = \alpha_0 \frac{M^\delta_{*c}}{\sum_{c\in V} M^\delta_{*c}} \tag{5}$$

where α_0 can be seen as a global weight to control the overall importance of negative samples. $\alpha_0 = 0$ means that no negative information is utilized in the training. The exponent δ is used for smoothing the weights. Specially, $\delta = 0$ means a uniform weight for all negative examples and $\delta = 1$ means that no smoothing is applied.

4 Fast Batch Gradient Optimization

Once specifying the loss function, the main challenge is how to perform an efficient optimization for Eq.(1). In the following, we develop a fast batch gradient optimization algorithm that is based on a *partition* reformulation for the loss and a *decouple* operation for the inner product.

4.1 Loss Partition

As can be seen, the major computational cost in Eq.(1) lies in the term L_N, because the size of $(V\times V)\setminus S$ is very huge, which typically contains over billions of negative examples. To this end, we show our first key design that separates the loss of negative samples into the difference between the loss on all samples and that on positive samples[1]. The loss partition serves as the prerequisite for the efficient computation of full batch gradients.

$$L_N=\sum_{w\in V}\sum_{c\in V}\alpha^-_c(r^- - U_w\tilde{U}^T_c)^2 - \sum_{(w,c)\in S}\alpha^-_c(r^- - U_w\tilde{U}^T_c)^2 \tag{6}$$

By replacing L_N in Eq.(1) with Eq.(6), we can obtain a new loss function with a more clear structure. We further simplify the loss function by merging the terms on positive examples. Finally, we achieve a reformulated loss

$$L = \underbrace{\sum_{w\in V}\sum_{c\in V}\alpha^-_c(r^- - U_w\tilde{U}^T_c)^2}_{L_A} + \underbrace{\sum_{(w,c)\in S}(\alpha^+_{wc} - \alpha^-_c)(\Delta - U_w\tilde{U}^T_c)^2}_{L_{P'}} + C \tag{7}$$

where $\Delta = (\alpha^+_{wc}r^+_{wc} - \alpha^-_c r^-)/(\alpha^+_{wc} - \alpha^-_c)$. It can be seen that the new loss function consists of two components: the loss L_A on the whole $V \times V$ training examples and $L_{P'}$ on positive examples. The major computation now lies in L_A which has a time complexity of $O(k|V|^2)$. In the following, we show how to reduce the huge volume of computation by a simple mathematical decouple.

[1]The idea here is similar to that used in (He et al., 2016; Li et al., 2016) for a different problem.

4.2 Decouple

To clearly show the *decouple* operation, we rewrite L_A as $\widetilde{L}_A$ by omitting the constant term $\alpha^-_c(r^-)^2$. Note that u_{wd} and $\tilde{u}_{cd}$ denote the d-th element in U_w and $\tilde{U}_c$, respectively.

$$\begin{aligned}\widetilde{L}_A = &\sum_{w\in V}\sum_{c\in V}\alpha^-_c\sum_{d=0}^{k}u_{wd}\tilde{u}_{cd}\sum_{d'=0}^{k}u_{wd'}\tilde{u}_{cd'}\\ &- 2r^-\sum_{w\in V}\sum_{c\in V}\alpha^-_c\sum_{d=0}^{k}u_{wd}\tilde{u}_{cd}\end{aligned} \tag{8}$$

Now we show our second key design that is based on a *decouple* manipulation for the inner product operation. Interestingly, we observe that the summation operator and elements in U_w and $\tilde{U}_c$ can be rearranged by the commutative property (Dai et al., 2007), as shown below.

$$\begin{aligned}\widetilde{L}_A = &\sum_{d=0}^{k}\sum_{d'=0}^{k}\sum_{w\in V}u_{wd}u_{wd'}\sum_{c\in V}\alpha^-_c\tilde{u}_{cd}\tilde{u}_{cd'}\\ &- 2r^-\sum_{d=0}^{k}\sum_{w\in V}u_{wd}\sum_{c\in V}\alpha^-_c\tilde{u}_{cd}\end{aligned} \tag{9}$$

An important feature in Eq.(9) is that the original inner product terms are disappeared, while in the new equation $\sum_{c\in V}\alpha^-_c\tilde{u}_{cd}\tilde{u}_{cd'}$ and $\sum_{c\in V}\alpha^-_c\tilde{u}_{cd}$ are "constant" values relative to $u_{wd}u_{wd'}$ and u_{wd} respectively. This means that they can be pre-calculated before training in each iteration. Specifically, we define $p^w_{dd'}$, $p^c_{dd'}$, q^w_d and q^c_d as the pre-calculated terms

$$\begin{aligned}p^w_{dd'} &= \sum_{w\in V}u_{wd}u_{wd'} & q^w_d &= \sum_{w\in V}u_{wd}\\ p^c_{dd'} &= \sum_{c\in V}\alpha^-_c\tilde{u}_{cd}\tilde{u}_{cd'} & q^c_d &= \sum_{c\in V}\alpha^-_c\tilde{u}_{cd}\end{aligned} \tag{10}$$

Then the computation of $\tilde{L}_A$ can be simplified to $\sum_{d=0}^{k}\sum_{d'=0}^{k}p^w_{dd'}p^c_{dd'} - 2r^- q^w_d q^c_d$.

It can be seen that the time complexity to compute all $p^w_{dd'}$ is $O(|V|k^2)$, and similarly, $O(|V|k^2)$ for $p^c_{dd'}$ and $O(|V|k)$ for q^w_d and q^c_d. With all terms pre-calculated before each iteration, the time complexity of computing $\tilde{L}_A$ is just $O(k^2)$. As a result, the total time complexity of computing L_A is decreased to $O(2|V|k^2+2|V|k+k^2) \approx O(2|V|k^2)$, which is much smaller than the original $O(k|V|^2)$. Moreover, it's worth noting that our efficient computation for $\widetilde{L}_A$ is strictly equal to its original value, which means AllVec does not introduce any approximation in evaluating the loss function.

Finally, we can derive the batch gradients for

u_{wd} and $\tilde{u}_{cd}$ as

$$\frac{\partial L}{\partial u_{wd}} = \sum_{d'=0}^{k} u_{wd'} p^{c}_{dd'} - \sum_{c \in I_w^+} \Lambda \cdot \tilde{u}_{cd} - r^- q^c_d$$

$$\frac{\partial L}{\partial \tilde{u}_{cd}} = \sum_{d'=0}^{k} \tilde{u}_{cd'} p^{w}_{dd'} \alpha_c^- - \sum_{w \in I_c^+} \Lambda \cdot u_{wd} - r^- \alpha_c^- q^w_d \tag{11}$$

where I_w^+ denotes the set of positive contexts for w, I_c^+ denotes the set of positive words for c and $\Lambda = (\alpha^+_{wc} - \alpha^-_c)(\Delta - U_w \tilde{U}_c^T)$. Algorithm 1 shows the training procedure of AllVec.

Algorithm 1 AllVec learning

```
Input: corpus $\mathbf{\Gamma}$, $win$, $\alpha_0$, $\delta$, $iter$, learning rate $\eta$
Output: embedding matrices $U$ and $\tilde{U}$
 1: Build vocabulary V from $\mathbf{\Gamma}$
 2: Obtain all positive $(w, c)$ and $M_{wc}$ from $\mathbf{\Gamma}$
 3: Compute all $r^+_{wc}$, $\alpha^+_{wc}$ and $\alpha^-_c$
 4: Initialize $U$ and $\tilde{U}$
 5: for $i = 1, ..., iter$ do
 6:     for $d \in \{0, .., k\}$ do
 7:         Compute and store $q^c_d$                 $\triangleright O(|V|k)$
 8:         for $d' \in \{0, .., k\}$ do
 9:             Compute and store $p^c_{dd'}$         $\triangleright O(|V|k^2)$
10:         end for
11:     end for
12:     for $w \in V$ do
13:         Compute $\Lambda$                         $\triangleright O(|S|k)$
14:         for $d \in \{0, .., k\}$ do
15:             Update $u_{wd}$                       $\triangleright O(|S|k + |V|k^2)$
16:         end for
17:     end for
18:     Repeat 6-17 for $\tilde{u}_{cd}$              $\triangleright O(2|S|k+2|V|k^2)$
19: end for
```

4.3 Time Complexity Analysis

In the following, we show that AllVec can achieve the same time complexity with negative sampling based SGD methods.

Given the sample size n, the total time complexity for SG is $O((n + 1)|S|k)$, where $n + 1$ denotes n negative samples and 1 positive example. Regarding the complexity of AllVec, we can see that the overall complexity of Algorithm 1 is $O(4|S|k + 4|V|k^2)$.

For the ease of discussion, we denote $\bar{c}$ as the average number of positive contexts for a word in the training corpus, i.e. $|S| = \bar{c}|V|$ ($\bar{c} \geq 1000$ in most cases). We then obtain the ratio

$$\frac{4|S|k + 4|V|k^2}{(n + 1)|S|k} = \frac{4}{n+1}(1 + \frac{k}{\bar{c}}) \tag{12}$$

where k is typically set from 100 to 300 (Mikolov et al., 2013a; Pennington et al., 2014), resulting in $k \leq \bar{c}$. Hence, we can give the lower and upper bound for the ratio:

$$\frac{4}{n+1} < \frac{4|S|k+4|V|k^2}{(n+1)|S|k} = \frac{4}{n+1}(1+\frac{k}{\bar{c}}) \leq \frac{8}{n+1} \tag{13}$$

The above analysis suggests that the complexity of AllVec is same as that of SGD with negative sample size between 3 and 7. In fact, considering that $\bar{c}$ is much larger than k in most datasets, the major cost of AllVec comes from the part $4|S|k$ (see Section 5.4 for details), which is linear with respect to the number of positive samples.

5 Experiments

We conduct experiments on three popular evaluation tasks, namely word analogy (Mikolov et al., 2013a), word similarity (Faruqui and Dyer, 2014) and QVEC (Tsvetkov et al., 2015).

Word analogy task. The task aims to answer questions like, "a is to b as c is to _?". We adopt the Google testbed[2] which contains 19,544 such questions in two categories: semantic and syntactic. The semantic questions are usually analogies about people or locations, like "king is to man as queen is to _?", while the syntactic questions focus on forms or tenses, e.g., "swimming is to swim as running to _?".

Word similarity tasks. We perform evaluation on six datasets, including MEN (Bruni et al., 2012), MC (Miller and Charles, 1991), RW (Luong et al., 2013), RG (Rubenstein and Goodenough, 1965), WS-353 Similarity (WSim) and Relatedness (WRel) (Finkelstein et al., 2001). We compute the spearman rank correlation between the similarity scores calculated based on the trained embeddings and human labeled scores.

QVEC. QVEC is an intrinsic evaluation metric of word embeddings based on the alignment to features extracted from manually crafted lexical resources. QVEC has shown strong correlation with the performance of embeddings in several semantic tasks (Tsvetkov et al., 2015).

We compare AllVec with the following word embedding methods.

- SG: This is the original skip-gram model with SGD and negative sampling (Mikolov et al., 2013a,b).
- SGA: This is the skip-gram model with an adaptive sampler (Chen et al., 2018).

[2]https://code.google.com/archive/p/word2vec/

Table 1: Corpora statistics.

Corpus	Tokens	Vocab	Size
Text8	17M	71K	100M
NewsIR	78M	83K	500M
Wiki-sub	0.8B	190K	4.0G
Wiki-all	2.3B	200K	13.4G

- GloVe: This method applies biased MF on the positive samples of word co-occurrence matrix (Pennington et al., 2014).
- LexVec: This method applies MF on the PPMI matrix. The optimization is done with negative sampling and mini-batch gradient descent (Salle et al., 2016b).

For all baselines, we use the original implementation released by the authors.

5.1 Datasets and Experimental Setup

We evaluate the performance of AllVec on four real-world corpora, namely Text8[3], NewsIR[4], Wiki-sub and Wiki-all. Wiki-sub is a subset of 2017 Wikipedia dump[5]. All corpora have been pre-processed by a standard pipeline (i.e. removing non-textual elements, lowercasing and tokenization). Table 1 summarizes the statistics of these corpora.

To obtain M_{wc} for positive (w, c) pairs, we follow GloVe where word pairs that are x words apart contribute $1/x$ to M_{wc}. The window size is set as $win = 8$. Regarding α_{wc}^{+}, we set $xmax = 100$ and $\rho = 0.75$. For a fair comparison, the embedding size k is set as 200 for all models and corpora. AllVec can be easily trained by AdaGrad (Zeiler, 2012) like GloVe or Newton-like (Bayer et al., 2017; Bradley et al., 2011) second order methods. For models based on negative sampling (i.e. SG, SGA and LexVec), the sample size is set as $n = 25$ for Text8, $n = 10$ for NewsIR and $n = 5$ for Wiki-sub and Wiki-all. The setting is also suggested by Mikolov et al. (2013b). Other detailed hyper-parameters are reported in Table 2.

5.2 Accuracy Comparison

We present results on the word analogy task in Table 2. As shown, AllVec achieves the highest total accuracy (Tot.) in all corpora, particularly in smaller corpora (Text8 and NewsIR). The reason is that in smaller corpora the number of positive (w, c) pairs is very limited, thus making use of negative examples will bring more benefits. Similar reason also explains the poor accuracy of GloVe in Text8, because GloVe does not consider negative samples. Even in the very large corpus (Wiki-all), ignoring negative samples still results in sub-optimal performance.

Our results also show that SGA achieves better performance than SG, which demonstrates the importance of a good sampling strategy. However, regardless what sampler (except the full softmax sampling) is utilized and how many updates are taken, sampling is still a biased approach. AllVec achieves the best performance because it is trained on the whole batch data for each parameter update rather than a fraction of sampled data.

Another interesting observation is AllVec performs better in semantic tasks in general. The reason is that our model utilizes global co-occurrence statistics, which capture more semantic signals than syntactic signals. While both AllVec and GloVe use global contexts, AllVec performs much better than GloVe in syntactic tasks. We argue that the main reason is because AllVec can distill useful signals from negative examples, while GloVe simply ignores all negative information. By contrast, local-window based methods, such as SG and SGA, are more effective to capture local sentence features, resulting in good performance on syntactic analogies. However, Rekabsaz et al. (2017) argues that these local-window based methods may suffer from the topic shifting issue.

Table 3 and Table 4 provide results in the word similarity and QVEC tasks. We can see that AllVec achieves the best performance in most tasks, which admits the advantage of batch learning with all samples. Interestingly, although GloVe performs well in semantic analogy tasks, it shows extremely worse results in word similarity and QVEC. The reason shall be the same as that it performs poorly in syntactic tasks.

5.3 Impact of α_c^-

In this subsection, we investigate the impact of the proposed weighting scheme for negative (context) words. We show the performance change of word analogy tasks on NewsIR in Figure 2 by tuning α_0 and δ. Results in other corpora show similar trends thus are omitted due to space limitation.

[3]http://mattmahoney.net/dc/text8.zip

[4]http://research.signalmedia.co/newsir16/signal-dataset.html

[5]https://dumps.wikimedia.org/enwiki/

Table 2: Results ("Tot." denotes total accuracy) on the word analogy task.

Corpus	Text8						NewsIR					
	para.			Sem.	Syn.	Tot.	para.			Sem.	Syn.	Tot.
SG	1e-4	8	25	47.51	32.26	38.60	1e-5	10	10	70.81	47.48	58.10
SGA	6e-3	-	-	48.10	33.78	39.74	6e-3	-	-	71.74	48.71	59.20
GloVe	10	15	1	45.11	26.89	34.47	50	8	1	78.79	41.58	58.52
LexVec	1e-4	25	-	51.87	31.78	40.14	1e-5	10	-	76.11	39.09	55.95
AllVec	350	0.75	-	56.66	32.42	**42.50**	100	0.8	-	78.47	48.33	**61.57**
	Wiki-sub						Wiki-all					
SG	1e-5	10	5	72.05	55.88	63.24	1e-5	10	5	73.91	61.91	67.37
SGA	6e-3	-	-	73.93	56.10	63.81	6e-3	-	-	75.11	61.94	67.92
GloVe	100	8	1	77.22	53.16	64.13	100	8	1	77.38	58.94	67.33
LexVec	1e-5	5	-	75.95	52.78	63.33	1e-5	5	-	76.31	56.83	65.48
AllVec	100	0.75	-	76.66	54.72	**64.75**	50	0.75	-	77.64	60.96	**68.52**

The parameter columns (para.) for each model are given from left to right as follows. SG: subsampling of frequent words, window size and the number of negative samples; SGA: λ (Chen et al., 2018) that controls the distribution of the rank, the other parameters are the same with SG; GloVe: $xmax$, window size and symmetric window; LexVec: subsampling of frequent words and the number of negative samples; AllVec: the negative weight α_0 and δ. Boldface denotes the highest total accuracy.

Figure 2(a) shows the impact of the overall weight α_0 by setting δ as 0.75 (inspired by the setting of skip-gram). Clearly, we observe that all results (including semantic, syntactic and total accuracy) have been greatly improved when α_0 increases from 0 to a larger value. As mentioned before, $\alpha_0 = 0$ means that no negative information is considered. This observation verifies that negative samples are very important for learning good embeddings. It also helps to explain why GloVe performs poorly on syntactic tasks. In addition, we find that in all corpora the optimal results are usually obtained when α_0 falls in the range of 50 to 400. For example, in the NewIR corpus as shown, AllVec achieves the best performance when $\alpha_0 = 100$. Figure 2(b) shows the impact of δ with $\alpha_0 = 100$. As mentioned before, $\delta = 0$ denotes a uniform value for all negative words and $\delta = 1$ denotes that no smoothing is applied to word frequency. We can see that the total accuracy is only around 55% when $\delta = 0$. By increasing its value, the performance is gradually improved, achieving the highest score when δ is around 0.8. Further increase of δ will degrade the total accuracy. This analysis demonstrates the effectiveness of the proposed negative weighting scheme.

5.4 Convergence Rate and Runtime

Figure 3(a) compares the convergence between AllVec and GloVe on NewsIR. Clearly, AllVec ex-

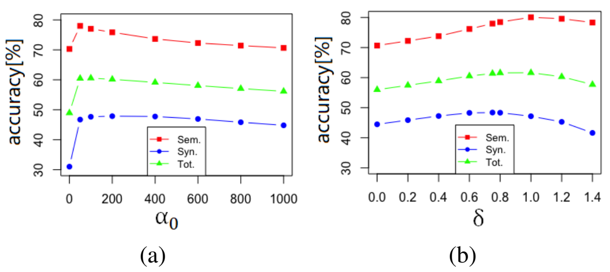


Figure 2: Effect of α_0 and δ on NewsIR.

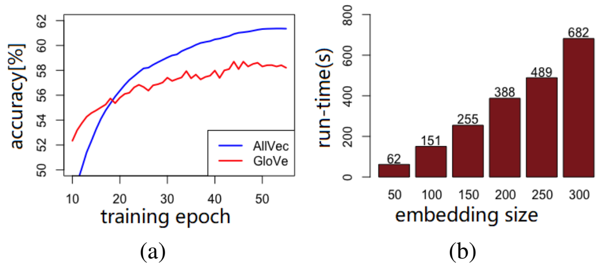


Figure 3: Convergence and runtime.

hibits a more stable convergence due to its full batch learning. In contrast, GloVe has a more dramatic fluctuation because of the one-sample learning scheme. Figure 3(b) shows the relationship between the embedding size k and runtime on NewsIR. Although the analysis in Section 4.3 demonstrates that the time complexity of AllVec is $O(4|S|k + 4|V|k^2)$, the actual runtime shows a near linear relationship with k. This is because $4|V|k^2/4|S|k = k/\bar{c}$, where $\bar{c}$ generally ranges from $1000 \sim 6000$ and k is set from 200 to 300 in practice. The above ratio explains the fact that $4|S|k$ dominates the complexity, which is linear

Table 3: Results on the word similarity task.

Corpus	Text8						NewsIR					
	MEN	MC	RW	RG	WSim	WRel	MEN	MC	RW	RG	WSim	WRel
SG	.6868	.6776	.3336	.6904	.7082	.6539	.7293	.7328	.3705	.7184	.7176	.6147
SGA	.6885	.6667	.3399	**.7035**	.7291	.6708	**.7409**	.7513	.3797	.7508	.7442	.6398
GloVe	.4999	.3349	.2614	.3367	.5168	.5115	.5839	.5637	.2487	.6284	.6029	.5329
LexVec	.6660	.6267	.2935	.6076	.7005	.6862	.7301	**.8403**	.3614	**.8341**	.7404	**.6545**
AllVec	**.6966**	**.6975**	**.3424**	.6588	**.7484**	**.7002**	.7407	.7642	**.4610**	.7753	**.7453**	.6322
	Wiki-sub						Wiki-all					
SG	**.7532**	.7943	.4250	.7555	.7627	**.6563**	.7564	.8083	.4311	.7678	**.7662**	.6485
SGA	.7465	.7983	.4296	.7623	**.7715**	.6560	**.7577**	.7940	.4379	.7683	.7110	.6488
GloVe	.6898	.6963	.3184	.7041	.6669	.5629	.7370	.7767	.3197	.7499	.7359	.6336
LexVec	.7318	.7591	.4225	.7628	.7292	.6219	.7256	**.8219**	.4383	.7797	.7548	.6091
AllVec	.7155	**.8305**	**.4667**	**.7945**	.7675	.6459	.7396	.7840	**.4966**	**.7800**	.7492	**.6518**

Table 4: Results on QVEC.

Qvec	Text8	NewsIR	Wiki-sub	Wiki-all
SG	.3999	.4182	.4280	.4306
SGA	.4062	.4159	**.4419**	.4464
GloVe	.3662	.3948	.4174	.4206
LexVec	**.4211**	.4172	.4332	.4396
AllVec	**.4211**	**.4319**	.4351	**.4489**

Table 5: Comparison of runtime.

Model	SI	Iter	Tot.
SG-3	259s	15	65m
SG-7	521s	15	131m
SG-10	715s	15	179m
AllVec	388s	50	322m

SG-n represents n negative samples for skip-gram, SI represents the runtime for a single iteration, and Tot. denotes the total runtime. All models are of embedding size 200 and trained with 8 threads.

with k and $|S|$.

We also compare the overall runtime of AllVec and SG on NewsIR and show the results in Table 5. As can be seen, the runtime of AllVec falls in the range of SG-3 and SG-7 in a single iteration, which confirms the theoretical analysis in Section 4.3. In contrast with SG, AllVec needs more iterations to converge. The reason is that each parameter in SG is updated many times during each iteration, although only one training example is used in each update. Despite this, the total run time of AllVec is still in a feasible range. Assuming the convergence is measured by the number of parameter updates, our AllVec yields a much faster convergence rate than the one-sample SG method.

In practice, the runtime of our model in each iteration can be further reduced by increasing the number of parallel workers. Although baseline methods like SG and GloVe can also be parallelized, the stochastic gradient steps in these methods unnecessarily influence each other as there is no exact way to separate these updates for different workers. In other words, the parallelization of SGD is not well suited to a large number of workers. In contrast, the parameter updates in AllVec are completely independent of each other, therefore AllVec does not have the update collision issue. This means we can achieve the embarrassing parallelization by simply separating the updates by words; that is, letting different workers update the model parameters for disjoint sets of words. As such, AllVec can provide a near linear scaling without any approximation since there is no potential conflicts between updates.

6 Conclusion

In this paper, we presented AllVec, an efficient batch learning based word embedding model that is capable to leverage all positive and negative training examples without any sampling and approximation. In contrast with models based on SGD and negative sampling, AllVec shows more stable convergence and better embedding quality by the all-sample optimization. Besides, both theoretical analysis and experiments demonstrate that AllVec achieves the same time complexity with the classic SGD models. In future, we will extend our proposed all-sample learning scheme to deep learning methods, which are more expressive than the shallow embedding model. Moreover, we will integrate prior knowledge, such as the words that are synonyms and antonyms, into the word embedding process. Lastly, we are interested in exploring the recent adversarial learning techniques to enhance the robustness of word embeddings.

Acknowledgements. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative. Joemon M.Jose and Xiangnan He are corresponding authors.

References

Melissa Ailem, Aghiles Salah, and Mohamed Nadif. 2017. Non-negative matrix factorization meets word embedding. In *SIGIR*, pages 1081–1084.

Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, pages 1341–1350.

Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, pages 713–722.

Guy Blanc and Steffen Rendle. 2017. Adaptive sampled softmax with kernel based sampling. *arXiv preprint arXiv:1712.00527*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *AAAI*, pages 2690–2696.

Joseph K Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. 2011. Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*, volume 1, pages 136–145.

Shaosheng Cao and Wei Lu. 2017. Improving word embeddings with convolutional feature learning and subword information. In *AAAI*, pages 3144–3151.

Long Chen, Fajie Yuan, Joemon M Jose, and Weinan Zhang. 2018. Improving negative sampling for word representation using self-embedded features. In *WSDM*, pages 99–107.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *SIGKDD*, pages 210–219.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at word-vectors. org. In *ACL*, pages 19–24.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414.

Guibing Guo, SC Ouyang, and Fajie Yuan. 2018. Approximating word ranking and negative sampling for word embedding. In *IJCAI*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, pages 873–882.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434.

Abhishek Kumar and Jun Araki. 2016. Incorporating relational knowledge into word representations using subspace regularization. In *ACL*, volume 2, pages 506–511.

Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. Swim: A simple word interaction model for implicit discourse relation recognition. In *IJCAI*, pages 4026–4032.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185.

Huayu Li, Richang Hong, Defu Lian, Zhiang Wu, Meng Wang, and Yong Ge. 2016. A relaxed ranking-based factor model for recommender system from implicit feedback. In *IJCAI*, pages 1683–1689.

Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. 2015. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *IJCAI*, pages 3650–3656.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015a. Learning semantic word embeddings based on ordinal knowledge constraints. In *ACL*, volume 1, pages 1501–1511.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015b. Topical word embeddings. In *AAAI*, pages 2418–2424.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *ACL*, volume 1, pages 2049–2058.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Hamed Zamani. 2017. Word embedding causes topic shifting; exploit global context! In *SIGIR*, pages 1105–1108.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Alexandre Salle, Marco Idiart, and Aline Villavicencio. 2016a. Enhancing the lexvec distributed word representation model using positional contexts and external memory. *arXiv preprint arXiv:1606.01283*.

Alexandre Salle, Marco Idiart, and Aline Villavicencio. 2016b. Matrix factorization using window sampling and negative sampling for improved word representations. *arXiv preprint arXiv:1606.00819*.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Bei Shi, Wai Lam, Shoaib Jameel, Steven Schockaert, and Kwun Ping Lai. 2017. Jointly learning word embeddings and latent topics. In *SIGIR*, pages 375–384.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *ACL*, volume 1, pages 1282–1291.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *EMNLP*, pages 2049–2054.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *CIKM*, pages 227–236. ACM.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Backpropagating through Structured Argmax using a SPIGOT

Hao Peng[◇] **Sam Thomson**[♣] **Noah A. Smith**[◇]
[◇] Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣] School of Computer Science, Carnegie Mellon University
{hapeng,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

Abstract

We introduce the **structured projection of intermediate gradients** optimization technique (SPIGOT), a new method for backpropagating through neural networks that include hard-decision structured predictions (e.g., parsing) in intermediate layers. SPIGOT requires no marginal inference, unlike structured attention networks (Kim et al., 2017) and some reinforcement learning-inspired solutions (Yogatama et al., 2017). Like so-called straight-through estimators (Hinton, 2012), SPIGOT defines gradient-like quantities associated with intermediate nondifferentiable operations, allowing backpropagation before and after them; SPIGOT's proxy aims to ensure that, after a parameter update, the intermediate structure will remain well-formed.

We experiment on two structured NLP pipelines: syntactic-then-semantic dependency parsing, and semantic parsing followed by sentiment classification. We show that training with SPIGOT leads to a larger improvement on the downstream task than a modularly-trained pipeline, the straight-through estimator, and structured attention, reaching a new state of the art on semantic dependency parsing.

1 Introduction

Learning methods for natural language processing are increasingly dominated by end-to-end differentiable functions that can be trained using gradient-based optimization. Yet traditional NLP often assumed modular stages of processing that formed a pipeline; e.g., text was tokenized, then tagged with parts of speech, then parsed into a phrase-structure or dependency tree, then semantically analyzed. Pipelines, which make "hard" (i.e., discrete) decisions at each stage, appear to be incompatible with neural learning, leading many researchers to abandon earlier-stage processing.

Inspired by findings that continue to see benefit from various kinds of linguistic or domain-specific preprocessing (He et al., 2017; Oepen et al., 2017; Ji and Smith, 2017), we argue that pipelines can be treated as layers in neural architectures for NLP tasks. Several solutions are readily available:

- Reinforcement learning (most notably the REINFORCE algorithm; Williams, 1992), and **structured attention** (SA; Kim et al., 2017). These methods replace argmax with a sampling or marginalization operation. We note two potential downsides of these approaches: (i) not all argmax-able operations have corresponding sampling or marginalization methods that are efficient, and (ii) inspection of intermediate outputs, which could benefit error analysis and system improvement, is more straightforward for hard decisions than for posteriors.
- The **straight-through estimator** (STE; Hinton, 2012) treats discrete decisions as if they were differentiable and simply passes through gradients. While fast and surprisingly effective, it ignores *constraints* on the argmax problem, such as the requirement that every word has exactly one syntactic parent. We will find, experimentally, that the quality of intermediate representations degrades substantially under STE.

This paper introduces a new method, the **structured projection of intermediate gradients** optimization technique (SPIGOT; §2), which defines a proxy for the gradient of a loss function with respect to the input to argmax. Unlike STE's gradient proxy, SPIGOT aims to respect the constraints in the argmax problem. SPIGOT can be applied with any intermediate layer that is expressible as a constrained maximization problem, and whose feasible set can be projected onto. We show empirically that SPIGOT works even when the maximization and the projection are done approximately.

We offer two concrete architectures that employ structured argmax as an intermediate layer: semantic parsing with syntactic parsing in the middle, and sentiment analysis with semantic parsing in the middle (§3). These architectures are trained using a joint objective, with one part using data for the intermediate task, and the other using data for the end task. The datasets are not assumed to overlap at all, but the parameters for the intermediate task are affected by both parts of the training data.

Our experiments (§4) show that our architecture improves over a state-of-the-art semantic dependency parser, and that SPIGOT offers stronger performance than a pipeline, SA, and STE. On sentiment classification, we show that semantic parsing offers improvement over a BiLSTM, more so with SPIGOT than with alternatives. Our analysis considers how the behavior of the intermediate parser is affected by the end task (§5). Our code is open-source and available at `https://github.com/Noahs-ARK/SPIGOT`.

2 Method

Our aim is to allow a (structured) argmax layer in a neural network to be treated almost like any other differentiable function. This would allow us to place, for example, a syntactic parser in the middle of a neural network, so that the forward calculation simply calls the parser and passes the parse tree to the next layer, which might derive syntactic features for the next stage of processing.

The challenge is in the *backward* computation, which is key to learning with standard gradient-based methods. When its output is discrete as we assume here, argmax is a piecewise constant function. At every point, its gradient is either zero or undefined. So instead of using the true gradient, we will introduce a *proxy* for the gradient of the loss function with respect to the inputs to argmax, allowing backpropagation to proceed through the argmax layer. Our proxy is designed as an improvement to earlier methods (discussed below) that completely ignore constraints on the argmax operation. It accomplishes this through a projection of the gradients.

We first lay out notation, and then briefly review max-decoding and its relaxation (§2.1). We define SPIGOT in §2.2, and show how to use it to backpropagate through NLP pipelines in §2.3.

Notation. Our discussion centers around two tasks: a structured *intermediate* task followed by an *end* task, where the latter considers the outputs of the former (e.g., syntactic-then-semantic parsing). Inputs are denoted as $\mathbf{x}$, and end task outputs as $\mathbf{y}$. We use $\mathbf{z}$ to denote intermediate structures derived from $\mathbf{x}$. We will often refer to the intermediate task as "decoding", in the structured prediction sense. It seeks an output $\hat{\mathbf{z}} = \operatorname{argmax}_{\mathbf{z}\in\mathcal{Z}} S$ from the feasible set $\mathcal{Z}$, maximizing a (learned, parameterized) scoring function S for the structured intermediate task. L denotes the loss of the end task, which may or may not also involve structured predictions. We use $\Delta^{k-1} = \{\mathbf{p} \in \mathbb{R}^k \mid \mathbf{1}^\top\mathbf{p} = 1, \mathbf{p} \geq \mathbf{0}\}$ to denote the $(k-1)$-dimensional simplex. We denote the domain of binary variables as $\mathbb{B} = \{0, 1\}$, and the unit interval as $\mathbb{U} = [0, 1]$. By projection of a vector $\mathbf{v}$ onto a set $\mathcal{A}$, we mean the closest point in $\mathcal{A}$ to $\mathbf{v}$, measured by Euclidean distance: $\operatorname{proj}_{\mathcal{A}}(\mathbf{v}) = \operatorname{argmin}_{\mathbf{v}'\in\mathcal{A}} \|\mathbf{v}' - \mathbf{v}\|_2$.

2.1 Relaxed Decoding

Decoding problems are typically decomposed into a collection of "parts", such as arcs in a dependency tree or graph. In such a setup, each element of $\mathbf{z}$, z_i, corresponds to one possible part, and z_i takes a boolean value to indicate whether the part is included in the output structure. The scoring function S is assumed to decompose into a vector $\mathbf{s}(\mathbf{x})$ of part-local, input-specific scores:

$$\hat{\mathbf{z}} = \operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} S(\mathbf{x}, \mathbf{z}) = \operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} \mathbf{z}^\top \mathbf{s}(\mathbf{x}) \tag{1}$$

In the following, we drop $\mathbf{s}$'s dependence on $\mathbf{x}$ for clarity.

In many NLP problems, the output space $\mathcal{Z}$ can be specified by linear constraints (Roth and Yih, 2004):

$$\mathbf{A}\begin{bmatrix}\mathbf{z}\\ \boldsymbol{\psi}\end{bmatrix} \leq \mathbf{b}, \tag{2}$$

where $\boldsymbol{\psi}$ are auxiliary variables (also scoped by argmax), together with integer constraints (typically, each $z_i \in \mathbb{B}$).

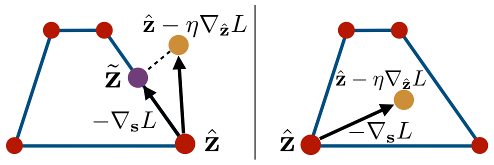

$\hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L$
$\tilde{\mathbf{z}}$
$-\nabla_{\mathbf{s}}L$
$\hat{\mathbf{z}}$
$\hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L$
$-\nabla_{\mathbf{s}}L$
$\hat{\mathbf{z}}$

Figure 1: The original feasible set $\mathcal{Z}$ (red vertices), is relaxed into a convex polytope $\mathcal{P}$ (the area encompassed by blue edges). Left: making a gradient update to $\hat{\mathbf{z}}$ makes it step outside the polytope, and it is projected back to $\mathcal{P}$, resulting in the projected point $\tilde{\mathbf{z}}$. $\nabla_{\mathbf{s}}L$ is then along the edge. Right: updating $\hat{\mathbf{z}}$ keeps it within $\mathcal{P}$, and thus $\nabla_{\mathbf{s}}L = \eta\nabla_{\hat{\mathbf{z}}}L$.

The problem in Equation 1 can be NP-complete in general, so the $\{0,1\}$ constraints are often relaxed to $[0,1]$ to make decoding tractable (Martins et al., 2009). Then the discrete combinatorial problem over $\mathcal{Z}$ is transformed into the optimization of a linear objective over a convex polytope $\mathcal{P}=\{\mathbf{p} \in \mathbb{R}^d | \mathbf{A}\mathbf{p} \leq \mathbf{b}\}$, which is solvable in polynomial time (Bertsimas and Tsitsiklis, 1997). This is not necessary in some cases, where the argmax can be solved exactly with dynamic programming.

2.2 From STE to SPIGOT

We now view structured argmax as an activation function that takes a vector of input-specific part-scores $\mathbf{s}$ and outputs a solution $\hat{\mathbf{z}}$. For backpropagation, to calculate gradients for parameters of $\mathbf{s}$, the chain rule defines:

$$\nabla_{\mathbf{s}}L = \mathbf{J}\ \nabla_{\hat{\mathbf{z}}}L, \tag{3}$$

where the Jacobian matrix $\mathbf{J} = \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{s}}$ contains the derivative of each element of $\hat{\mathbf{z}}$ with respect to each element of $\mathbf{s}$. Unfortunately, argmax is a piecewise constant function, so its Jacobian is either zero (almost everywhere) or undefined (in the case of ties).

One solution, taken in *structured attention*, is to replace the argmax with marginal inference and a softmax function, so that $\hat{\mathbf{z}}$ encodes probability distributions over parts (Kim et al., 2017; Liu and Lapata, 2018). As discussed in §1, there are two reasons to avoid this modification. Softmax can only be used when marginal inference is feasible, by sum-product algorithms for example (Eisner, 2016; Friesen and Domingos, 2016); in general marginal inference can be #P-complete. Further, a soft intermediate layer will be less amenable to inspection by anyone wishing to understand and improve the model.

In another line of work, argmax is augmented with a strongly-convex penalty on the solutions (Martins and Astudillo, 2016; Amos and Kolter, 2017; Niculae and Blondel, 2017; Niculae et al., 2018; Mensch and Blondel, 2018). However, their approaches require solving a relaxation even when exact decoding is tractable. Also, the penalty will bias the solutions found by the decoder, which may be an undesirable conflation of computational and modeling concerns.

A simpler solution is the STE method (Hinton, 2012), which replaces the Jacobian matrix in Equation 3 by the identity matrix. This method has been demonstrated to work well when used to "backpropagate" through hard threshold functions (Bengio et al., 2013; Friesen and Domingos, 2018) and categorical random variables (Jang et al., 2016; Choi et al., 2017).

Consider for a moment what we would do if $\hat{\mathbf{z}}$ were a vector of parameters, rather than intermediate predictions. In this case, we are seeking points in $\mathcal{Z}$ that minimize L; denote that set of minimizers by $\mathcal{Z}^*$. Given $\nabla_{\hat{\mathbf{z}}}L$ and step size η, we would update $\hat{\mathbf{z}}$ to be $\hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L$. This update, however, might not return a value in the feasible set $\mathcal{Z}$, or even (if we are using a linear relaxation) the relaxed set $\mathcal{P}$.

SPIGOT therefore introduces a *projection* step that aims to keep the "updated" $\hat{\mathbf{z}}$ in the feasible set. Of course, we do not directly update $\hat{\mathbf{z}}$; we continue backpropagation through $\mathbf{s}$ and onward to the parameters. But the projection step nonetheless alters the parameter updates in the way that our proxy for "$\nabla_{\mathbf{s}}L$" is defined.

The procedure is defined as follows:

$$\hat{\mathbf{p}} = \hat{\mathbf{z}} - \eta\nabla_{\hat{\mathbf{z}}}L, \tag{4a}$$
$$\tilde{\mathbf{z}} = \mathrm{proj}_{\mathcal{P}}(\hat{\mathbf{p}}), \tag{4b}$$
$$\nabla_{\mathbf{s}}L \triangleq \hat{\mathbf{z}} - \tilde{\mathbf{z}}. \tag{4c}$$

First, the method makes an "update" to $\hat{\mathbf{z}}$ as if it contained parameters (Equation 4a), letting $\hat{\mathbf{p}}$ denote the new value. Next, $\hat{\mathbf{p}}$ is projected back onto the (relaxed) feasible set (Equation 4b), yielding a feasible new value $\tilde{\mathbf{z}}$. Finally, the gradients with respect to $\mathbf{s}$ are computed by Equation 4c.

Due to the convexity of $\mathcal{P}$, the projected point $\tilde{\mathbf{z}}$ will always be unique, and is guaranteed to be no farther than $\hat{\mathbf{p}}$ from any point in $\mathcal{Z}^*$ (Luenberger and Ye, 2015).[1] Compared to STE, SPIGOT in-

[1]Note that this property follows from $\mathcal{P}$'s convexity, and we do not assume the convexity of L.

volves a projection and limits $\nabla_{\mathbf{s}} L$ to a smaller space to satisfy constraints. See Figure 1 for an illustration.

When efficient exact solutions (such as dynamic programming) are available, they can be used. Yet, we note that SPIGOT does not assume the argmax operation is solved exactly.

2.3 Backpropagation through Pipelines

Using SPIGOT, we now devise an algorithm to "backpropagate" through NLP pipelines. In these pipelines, an intermediate task's output is fed into an end task for use as features. The parameters of the complete model are divided into two parts: denote the parameters of the intermediate task model by $\boldsymbol{\phi}$ (used to calculate $\mathbf{s}$), and those in the end task model as $\boldsymbol{\theta}$.[2] As introduced earlier, the end-task loss function to be minimized is L, which depends on both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

Algorithm 1 describes the forward and backward computations. It takes an end task training pair $\langle \mathbf{x}, \mathbf{y} \rangle$, along with the intermediate task's feasible set $\mathcal{Z}$, which is determined by $\mathbf{x}$. It first runs the intermediate model and decodes to get intermediate structure $\hat{\mathbf{z}}$, just as in a standard pipeline. Then forward propagation is continued into the end-task model to compute loss L, using $\hat{\mathbf{z}}$ to define input features. Backpropagation in the end-task model computes $\nabla_{\boldsymbol{\theta}} L$ and $\nabla_{\hat{\mathbf{z}}} L$, and $\nabla_{\mathbf{s}} L$ is then constructed using Equations 4. Backpropagation then continues into the intermediate model, computing $\nabla_{\boldsymbol{\phi}} L$.

Due to its flexibility, SPIGOT is applicable to many training scenarios. When there is no $\langle \mathbf{x}, \mathbf{z} \rangle$ training data for the intermediate task, SPIGOT can be used to induce latent structures for the end-task (Yogatama et al., 2017; Kim et al., 2017; Choi et al., 2017, *inter alia*). When intermediate-task training data *is* available, one can use SPIGOT to adopt joint learning by minimizing an interpolation of L (on end-task data $\langle \mathbf{x}, \mathbf{y} \rangle$) and an intermediate-task loss function $\widetilde{L}$ (on intermediate task data $\langle \mathbf{x}, \mathbf{z} \rangle$). This is the setting in our experiments; note that we do not assume any overlap in the training examples for the two tasks.

Algorithm 1 Forward and backward computation with SPIGOT.

```
 1: procedure SPIGOT($\mathbf{x}, \mathbf{y}, \mathcal{Z}$)
 2:     Construct $\mathbf{A}, \mathbf{b}$ such that $\mathcal{Z} = \{\mathbf{p} \in \mathbb{Z}^d \mid \mathbf{A}\mathbf{p} \leq \mathbf{b}\}$
 3:     $\mathcal{P} \leftarrow \{\mathbf{p} \in \mathbb{R}^d \mid \mathbf{A}\mathbf{p} \leq \mathbf{b}\}$        ▷ Relaxation
 4:     Forwardprop and compute $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x})$
 5:     $\hat{\mathbf{z}} \leftarrow \text{argmax}_{\mathbf{z} \in \mathcal{Z}} \mathbf{z}^\top \mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x})$   ▷ Intermediate decoding
 6:     Forwardprop and compute $L$ given $\mathbf{x}$, $\mathbf{y}$, and $\hat{\mathbf{z}}$
 7:     Backprop and compute $\nabla_{\boldsymbol{\theta}} L$ and $\nabla_{\hat{\mathbf{z}}} L$
 8:     $\tilde{\mathbf{z}} \leftarrow \text{proj}_{\mathcal{P}}(\hat{\mathbf{z}} - \eta \nabla_{\hat{\mathbf{z}}} L)$        ▷ Projection
 9:     $\nabla_{\mathbf{s}} L \leftarrow \hat{\mathbf{z}} - \tilde{\mathbf{z}}$
10:     Backprop and compute $\nabla_{\boldsymbol{\phi}} L$
11: end procedure
```

3 Solving the Projections

In this section we discuss how to compute approximate projections for the two intermediate tasks considered in this work, arc-factored unlabeled dependency parsing and first-order semantic dependency parsing.

In early experiments we observe that for both tasks, projecting with respect to *all* constraints of their original formulations using a generic quadratic program solver was prohibitively slow. Therefore, we construct relaxed polytopes by considering only a subset of the constraints.[3] The projection then decomposes into a series of singly constrained quadratic programs (QP), each of which can be efficiently solved in linear time.

The two approximate projections discussed here are used in backpropagation only. In the forward pass, we solve the decoding problem using the models' original decoding algorithms.

Arc-factored unlabeled dependency parsing. For unlabeled dependency trees, we impose $[0, 1]$ constraints and single-headedness constraints.[4]

Formally, given a length-n input sentence, excluding self-loops, an arc-factored parser considers $d = n(n-1)$ candidate arcs. Let $i \rightarrow j$ denote an arc from the ith token to the jth, and $\sigma(i \rightarrow j)$ denote its index. We construct the relaxed feasible set by:

$$\mathcal{P}_{\text{DEP}} = \left\{ \mathbf{p} \in \mathbb{U}^d \;\middle|\; \sum_{i \neq j} p_{\sigma(i \rightarrow j)} = 1, \forall j \right\}, \quad (5)$$

i.e., we consider each token j individually, and force single-headedness by constraining the number of arcs incoming to j to sum to 1. Algorithm 2 summarizes the procedure to project onto $\mathcal{P}_{\text{DEP}}$.

[2]Nothing prohibits tying across pre-argmax parameters and post-argmax parameters; this separation is notationally convenient but not at all necessary.

[3]A parallel work introduces an active-set algorithm to solve the same class of quadratic programs (Niculae et al., 2018). It might be an efficient approach to solve the projections in Equation 4b, which we leave to future work.

[4] It requires $O(n^2)$ auxiliary variables and $O(n^3)$ additional constraints to ensure well-formed tree structures (Martins et al., 2013).

Line 3 forms a singly constrained QP, and can be solved in $O(n)$ time (Brucker, 1984).

Algorithm 2 Projection onto the relaxed polytope $\mathcal{P}_{\text{DEP}}$ for dependency tree structures. Let bold $\boldsymbol{\sigma}(\cdot \to j)$ denote the index set of arcs incoming to j. For a vector $\mathbf{v}$, we use $\mathbf{v}_{\boldsymbol{\sigma}(\cdot\to j)}$ to denote vector $[v_k]_{k\in\boldsymbol{\sigma}(\cdot\to j)}$.

```
1: procedure DEPPROJ($\hat{\mathbf{p}}$)
2:     for $j = 1, 2, \dots, n$ do
3:         $\tilde{\mathbf{z}}_{\boldsymbol{\sigma}(\cdot\to j)} \leftarrow \text{proj}_{\triangle^{n-2}}\left(\hat{\mathbf{p}}_{\boldsymbol{\sigma}(\cdot\to j)}\right)$
4:     end for
5:     return $\tilde{\mathbf{z}}$
6: end procedure
```

First-order semantic dependency parsing. Semantic dependency parsing uses labeled bilexical dependencies to represent sentence-level semantics (Oepen et al., 2014, 2015, 2016). Each dependency is represented by a labeled directed arc from a head token to a modifier token, where the arc label encodes broadly applicable semantic relations. Figure 2 diagrams a semantic graph from the DELPH-IN MRS-derived dependencies (DM), together with a syntactic tree.

We use a state-of-the-art semantic dependency parser (Peng et al., 2017) that considers three types of parts: heads, unlabeled arcs, and labeled arcs. Let $\sigma(i \xrightarrow{\ell} j)$ denote the index of the arc from i to j with semantic role ℓ. In addition to $[0,1]$ constraints, we constrain that the predictions for labeled arcs sum to the prediction of their associated unlabeled arc:

$$\mathcal{P}_{\text{SDP}} \left\{ \mathbf{p} \in \mathbb{U}^d \,\middle|\, \sum_{\ell} p_{\sigma(i \xrightarrow{\ell} j)} = p_{\sigma(i\to j)}, \forall i \neq j \right\}. \tag{6}$$

This ensures that exactly one label is predicted if and only if its arc is present. The projection onto $\mathcal{P}_{\text{SDP}}$ can be solved similarly to Algorithm 2. We drop the determinism constraint imposed by Peng et al. (2017) in the backward computation.

4 Experiments

We empirically evaluate our method with two sets of experiments: using syntactic tree structures in semantic dependency parsing, and using semantic dependency graphs in sentiment classification.

4.1 Syntactic-then-Semantic Parsing

In this experiment we consider an intermediate syntactic parsing task, followed by semantic dependency parsing as the end task. We first briefly review the neural network architectures for the two models (§4.1.1), and then introduce the datasets (§4.1.2) and baselines (§4.1.3).

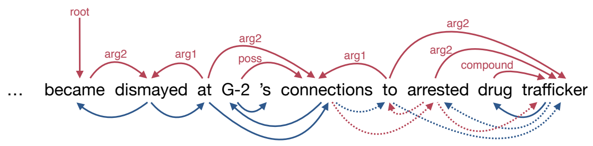

root
arg2
arg1
arg2
poss
arg1
arg2
arg2
compound
... became dismayed at G-2 's connections to arrested drug trafficker

Figure 2: A development instance annotated with both gold DM semantic dependency graph (red arcs on the top), and gold syntactic dependency tree (blue arcs at the bottom). A pretrained syntactic parser predicts the same tree as the gold; the semantic parser backpropagates into the intermediate syntactic parser, and changes the dashed blue arcs into dashed red arcs (§5).

4.1.1 Architectures

Syntactic dependency parser. For intermediate syntactic dependencies, we use the unlabeled arc-factored parser of Kiperwasser and Goldberg (2016). It uses bidirectional LSTMs (BiLSTM) to encode the input, followed by a multilayer-perceptron (MLP) to score each potential dependency. One notable modification is that we replace their use of Chu-Liu/Edmonds' algorithm (Chu and Liu, 1965; Edmonds, 1967) with the Eisner algorithm (Eisner, 1996, 2000), since our dataset is in English and mostly projective.

Semantic dependency parser. We use the basic model of Peng et al. (2017) (denoted as NEURBOPARSER) as the end model. It is a first-order parser, and uses local factors for heads, unlabeled arcs, and labeled arcs. NEURBOPARSER does not use syntax. It first encodes an input sentence with a two-layer BiLSTM, and then computes part scores with two-layer tanh-MLPs. Inference is conducted with AD^3 (Martins et al., 2015). To add syntactic features to NEURBOPARSER, we concatenate a token's contextualized representation to that of its syntactic head, predicted by the intermediate parser. Formally, given length-n input sentence, we first run a BiLSTM. We use the concatenation of the two hidden representations $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]$ at each position j as the contextualized token representations. We then concatenate

$\mathbf{h}_j$ with the representation of its head $\mathbf{h}_{\text{HEAD}(j)}$ by

$$\widetilde{\mathbf{h}}_j = [\mathbf{h}_j; \mathbf{h}_{\text{HEAD}(j)}] = \left[\mathbf{h}_j; \sum_{i \neq j} \hat{z}_{\sigma(i \to j)}\ \mathbf{h}_i\right], \tag{7}$$

where $\hat{\mathbf{z}} \in \mathbb{B}^{n(n-1)}$ is a binary encoding of the tree structure predicted by by the intermediate parser. We then use $\widetilde{\mathbf{h}}_j$ anywhere $\mathbf{h}_j$ would have been used in NEURBOPARSER. In backpropagation, we compute $\nabla_{\hat{\mathbf{z}}} L$ with an automatic differentiation toolkit (DyNet; Neubig et al., 2017).

We note that this approach can be generalized to convolutional neural networks over graphs (Mou et al., 2015; Duvenaud et al., 2015; Kipf and Welling, 2017, *inter alia*), recurrent neural networks along paths (Xu et al., 2015; Roth and Lapata, 2016, *inter alia*) or dependency trees (Tai et al., 2015). We choose to use concatenations to control the model's complexity, and thus to better understand which parts of the model work.

We refer the readers to Kiperwasser and Goldberg (2016) and Peng et al. (2017) for further details of the parsing models.

Training procedure. Following previous work, we minimize structured hinge loss (Tsochantaridis et al., 2004) for both models. We jointly train both models from scratch, by randomly sampling an instance from the union of their training data at each step. In order to isolate the effect of backpropagation, we do not share any parameters between the two models.[5] Implementation details are summarized in the supplementary materials.

4.1.2 Datasets

- For semantic dependencies, we use the English dataset from SemEval 2015 Task 18 (Oepen et al., 2015). Among the three formalisms provided by the shared task, we consider DELPH-IN MRS-derived dependencies (DM) and Prague Semantic Dependencies (PSD).[6] It includes §00–19 of the WSJ corpus as training data, §20 and §21 for development and in-domain test data, resulting in a 33,961/1,692/1,410 train/dev./test split, and 1,849 out-of-domain test instances from the Brown corpus.[7]
- For syntactic dependencies, we use the Stanford Dependency (de Marneffe and Manning, 2008) conversion of the the Penn Treebank WSJ portion (Marcus et al., 1993). To avoid data leak, we depart from standard split and use §20 and §21 as development and test data, and the remaining sections as training data. The number of training/dev./test instances is 40,265/2,012/1,671.

	DM		PSD	
Model	UF	LF	UF	LF
NEURBOPARSER	–	89.4	–	77.6
FREDA3	–	90.4	–	78.5
PIPELINE	91.8	90.8	88.4	78.1
SA	91.6	90.6	87.9	78.1
STE	92.0	91.1	**88.9**	**78.9**
SPIGOT	**92.4**	**91.6**	88.6	**78.9**

(a) F_1 on in-domain test set.

	DM		PSD	
Model	UF	LF	UF	LF
NEURBOPARSER	–	84.5	–	75.3
FREDA3	–	85.3	–	76.4
PIPELINE	87.4	85.8	85.5	75.6
SA	87.3	85.6	84.9	75.9
STE	87.7	86.4	**85.8**	76.6
SPIGOT	**87.9**	**86.7**	85.5	**77.1**

(b) F_1 on out-of-domain test set.

Table 1: Semantic dependency parsing performance in both unlabeled (UF) and labeled (LF) F_1 scores. Bold font indicates the best performance. Peng et al. (2017) does not report UF.

4.1.3 Baselines

We compare to the following baselines:

- A pipelined system (PIPELINE). The pretrained parser achieves 92.9 test unlabeled attachment score (UAS).[8]

[5] Parameter sharing has proved successful in many related tasks (Collobert and Weston, 2008; Søgaard and Goldberg, 2016; Ammar et al., 2016; Swayamdipta et al., 2016, 2017, *inter alia*), and could be easily combined with our approach.

[6] We drop the third (PAS) because its structure is highly predictable from parts-of-speech, making it less interesting.

[7] The organizers remove, e.g., instances with cyclic graphs, and thus only a subset of the WSJ corpus is included. See Oepen et al. (2015) for details.

[8] Note that this number is not comparable to the parsing literature due to the different split. As a sanity check, we found in preliminary experiments that the same parser archi-

- Structured attention networks (SA; Kim et al., 2017). We use the inside-outside algorithm (Baker, 1979) to populate $\mathbf{z}$ with arcs' marginal probabilities, use log-loss as the objective in training the intermediate parser.
- The straight-through estimator (STE; Hinton, 2012), introduced in §2.2.

4.1.4 Empirical Results

Table 1 compares the semantic dependency parsing performance of SPIGOT to all five baselines. FREDA3 (Peng et al., 2017) is a state-of-the-art variant of NEURBOPARSER that is trained using multitask learning to jointly predict three different semantic dependency graph formalisms. Like the basic NEURBOPARSER model that we build from, FREDA3 does not use any syntax. Strong DM performance is achieved in a more recent work by using joint learning and an ensemble (Peng et al., 2018), which is beyond fair comparisons to the models discussed here.

We found that using syntactic information improves semantic parsing performance: using pipelined syntactic head features brings 0.5–1.4% absolute labeled F_1 improvement to NEURBOPARSER. Such improvements are smaller compared to previous works, where dependency path and syntactic relation features are included (Almeida and Martins, 2015; Ribeyre et al., 2015; Zhang et al., 2016), indicating the potential to get better performance by using more syntactic information, which we leave to future work.

Both STE and SPIGOT use hard syntactic features. By allowing backpropation into the intermediate syntactic parser, they both consistently outperform PIPELINE. On the other hand, when marginal syntactic tree structures are used, SA outperforms PIPELINE only on the out-of-domain PSD test set, and improvements under other cases are not observed.

Compared to STE, SPIGOT outperforms STE on DM by more than 0.3% absolute labeled F_1, both in-domain and out-of-domain. For PSD, SPIGOT achieves similar performance to STE on in-domain test set, but has a 0.5% absolute labeled F_1 improvement on out-of-domain data, where syntactic parsing is less accurate.

tecture achieves 93.5 UAS when trained and evaluated with the standard split, close to the results reported by Kiperwasser and Goldberg (2016).

4.2 Semantic Dependencies for Sentiment Classification

Our second experiment uses semantic dependency graphs to improve sentiment classification performance. We are not aware of any efficient algorithm that solves marginal inference for semantic dependency graphs under determinism constraints, so we do not include a comparison to SA.

4.2.1 Architectures

Here we use NEURBOPARSER as the intermediate model, as described in §4.1.1, but with no syntactic enhancements.

Sentiment classifier. We first introduce a baseline that does not use any structural information. It learns a one-layer BiLSTM to encode the input sentence, and then feeds the sum of all hidden states into a two-layer ReLU-MLP.

To use semantic dependency features, we concatenate a word's BiLSTM-encoded representation to the averaged representation of its heads, together with the corresponding semantic roles, similarly to that in Equation 7.[9] Then the concatenation is fed into an affine transformation followed by a ReLU activation. The rest of the model is kept the same as the BiLSTM baseline.

Training procedure. We use structured hinge loss to train the semantic dependency parser, and log-loss for the sentiment classifier. Due to the discrepancy in the training data size of the two tasks (33K vs. 7K), we pre-train a semantic dependency parser, and then adopt joint training together with the classifier. In the joint training stage, we randomly sample 20% of the semantic dependency training instances each epoch. Implementations are detailed in the supplementary materials.

4.2.2 Datasets

For semantic dependencies, we use the DM dataset introduced in §4.1.2.

We consider a binary classification task using the Stanford Sentiment Treebank (Socher et al., 2013). It consists of roughly 10K movie review sentences from Rotten Tomatoes. The full dataset includes a rating on a scale from 1 to 5 for each constituent (including the full sentences), resulting in more than 200K instances. Following previous work (Iyyer et al., 2015), we only use full-sentence

[9]In a well-formed semantic dependency graph, a token may have multiple heads. Therefore we use average instead of the sum in Equation 7.

Model	Accuracy (%)
BiLSTM	84.8
PIPELINE	85.7
STE	85.4
SPIGOT	**86.3**

Table 2: Test accuracy of sentiment classification on Stanford Sentiment Treebank. Bold font indicates the best performance.

instances, with neutral instances excluded (3s) and the remaining four rating levels converted to binary "positive" or "negative" labels. This results in a 6,920/872/1,821 train/dev./test split.

4.2.3 Empirical Results

Table 2 compares our SPIGOT method to three baselines. Pipelined semantic dependency predictions brings 0.9% absolute improvement in classification accuracy, and SPIGOT outperforms all baselines. In this task STE achieves slightly worse performance than a fixed pre-trained PIPELINE.

5 Analysis

We examine here how the intermediate model is affected by the end-task training signal. Is the end-task signal able to "overrule" intermediate predictions?

We use the syntactic-then-semantic parsing model (§4.1) as a case study. Table 3 compares a pipelined system to one jointly trained using SPIGOT. We consider the development set instances where both syntactic and semantic annotations are available, and partition them based on whether the two systems' syntactic predictions agree (SAME), or not (DIFF). The second group includes sentences with much lower syntactic parsing accuracy (91.3 vs. 97.4 UAS), and SPIGOT further reduces this to 89.6. Even though these changes hurt syntactic parsing accuracy, they lead to a 1.1% absolute gain in labeled F_1 for semantic parsing. Furthermore, SPIGOT has an overall less detrimental effect on the intermediate parser than STE: using SPIGOT, intermediate dev. parsing UAS drops to 92.5 from the 92.9 pipelined performance, while STE reduces it to 91.8.

We then take a detailed look and categorize the changes in intermediate trees by their correlations with the semantic graphs. Specifically, when a modifier m's head is changed from h to h' in the

Split	# Sent.	Model	UAS	DM
SAME	1011	PIPELINE	97.4	94.0
		SPIGOT	97.4	94.3
DIFF	681	PIPELINE	91.3	88.1
		SPIGOT	89.6	89.2

Table 3: Syntactic parsing performance (in unlabeled attachment score, UAS) and DM semantic parsing performance (in labeled F_1) on different groups of the development data. Both systems predict the same syntactic parses for instances from SAME, and they disagree on instances from DIFF (§5).

tree, we consider three cases: (a) h' is a head of m in the semantic graph; (b) h' is a modifier of m in the semantic graph; (c) h is the modifier of m in the semantic graph. The first two reflect modifications to the syntactic parse that rearrange semantically linked words to be neighbors. Under (c), the semantic parser removes a syntactic dependency that reverses the direction of a semantic dependency. These cases account for 17.6%, 10.9%, and 12.8%, respectively (41.2% combined) of the total changes. Making these changes, of course, is complicated, since they often require *other* modifications to maintain well-formedness of the tree. Figure 2 gives an example.

6 Related Work

Joint learning in NLP pipelines. To avoid cascading errors, much effort has been devoted to joint decoding in NLP pipelines (Habash and Rambow, 2005; Cohen and Smith, 2007; Goldberg and Tsarfaty, 2008; Lewis et al., 2015; Zhang et al., 2015, *inter alia*). However, joint inference can sometimes be prohibitively expensive. Recent advances in representation learning facilitate exploration in the joint learning of multiple tasks by sharing parameters (Collobert and Weston, 2008; Blitzer et al., 2006; Finkel and Manning, 2010; Zhang and Weiss, 2016; Hashimoto et al., 2017, *inter alia*).

Differentiable optimization. Gould et al. (2016) review the generic approaches to differentiation in bi-level optimization (Bard, 2010; Kunisch and Pock, 2013). Amos and Kolter (2017) extend their efforts to a class of subdifferentiable quadratic programs. However, they both require that the intermediate objective has an invertible Hessian, limiting their application in NLP. In another line of work, the steps of a gradient-based optimization procedure are unrolled into a single computation graph (Stoyanov et al., 2011; Domke, 2012; Goodfellow et al., 2013; Brakel et al., 2013). This comes at a high computational cost due to the second-order derivative computation during backpropagation. Moreover, constrained optimization problems (like many NLP problems) often require projection steps within the procedure, which can be difficult to differentiate through (Belanger and McCallum, 2016; Belanger et al., 2017).

7 Conclusion

We presented SPIGOT, a novel approach to backpropagating through neural network architectures that include discrete structured decisions in intermediate layers. SPIGOT devises a proxy for the gradients with respect to argmax's inputs, employing a projection that aims to respect the constraints in the intermediate task. We empirically evaluate our method with two architectures: a semantic parser with an intermediate syntactic parser, and a sentiment classifier with an intermediate semantic parser. Experiments show that SPIGOT achieves stronger performance than baselines under both settings, and outperforms state-of-the-art systems on semantic dependency parsing. Our implementation is available at `https://github.com/Noahs-ARK/SPIGOT`.

Acknowledgments

We thank the ARK, Julian Michael, Minjoon Seo, Eunsol Choi, and Maxwell Forbes for their helpful comments on an earlier version of this work, and the anonymous reviewers for their valuable feedback. This work was supported in part by NSF grant IIS-1562364.

References

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proc. of SemEval*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL* 4:431–444.

Brandon Amos and J. Zico Kolter. 2017. OptNet: Differentiable optimization as a layer in neural networks. In *Proc. of ICML*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*.

Jonathan F. Bard. 2010. *Practical Bilevel Optimization: Algorithms and Applications*. Springer.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proc. of ICML*.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proc. of ICML*.

Yoshua Bengio, Nicholas Lonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432.

Dimitris Bertsimas and John Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

Philémon Brakel, Dirk Stroobandt, and Benjamin Schrauwen. 2013. Training energy-based models for time-series imputation. *Journal of Machine Learning Research* 14:2771–2797.

Peter Brucker. 1984. An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters* 3(3):163 – 166.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-LSTMs. arXiv:1707.02786.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14:1396–1400.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proc. of EMNLP*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Justin Domke. 2012. Generic methods for optimization-based modeling. In *Proc. of AISTATS*.

David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proc. of NIPS*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards* 71B:233–240.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, Springer Netherlands, pages 29–61.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop. In *Proceedings of the EMNLP Workshop on Structured Prediction for NLP*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Jenny Rose Finkel and Christopher D. Manning. 2010. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proc. of ACL*.

Abram L. Friesen and Pedro M. Domingos. 2016. The sum-product theorem: A foundation for learning tractable models. In *Proc. of ICML*.

Abram L. Friesen and Pedro M. Domingos. 2018. Deep learning as a mixed convex-combinatorial optimization problem. In *Proc. of ICLR*.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proc. of ACL*.

Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Multi-prediction deep Boltzmann machines. In *Proc. of NIPS*.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. arXiv:1607.05447.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proc. ACL*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proc. of EMNLP*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proc. of ACL*.

Geoffrey Hinton. 2012. Neural networks for machine learning. *Coursera* video lectures.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-Softmax. arXiv:1611.01144.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proc. of ACL*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proc. of ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*.

Karl Kunisch and Thomas Pock. 2013. A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences* 6(2):938–983.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proc. of EMNLP*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *TACL* 6:63–75.

David G. Luenberger and Yinyu Ye. 2015. *Linear and Nonlinear Programming*. Springer.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19(2):313–330.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. of ICML*.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. 2015. AD3: Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research* 16:495–545.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

Arthur Mensch and Mathieu Blondel. 2018. Differentiable dynamic programming for structured prediction and attention. *arXiv:1802.03676* .

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proc. of EMNLP*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Proc. of NIPS*.

Vlad Niculae, Andr F. T. Martins, Mathieu Blondel, and Claire Cardie. 2018. SparseMAP: Differentiable sparse structured inference. arXiv:1802.04223.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Lilja vrelid, Jari Bjrne, Richard Johansson, Emanuele Lapponi, Filip Ginter, and Erik Velldal. 2017. The 2017 shared task on extrinsic parser evaluation. towards a reusable community infrastructure. In *Proc. of the 2017 Shared Task on Extrinsic Parser Evaluation*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

Corentin Ribeyre, Éric Villemonte De La Clergerie, and Djamé Seddah. 2015. Because syntax does matter: Improving predicate-argument structures parsing using syntactic features. In *Proc. of NAACL*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of NAACL*.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proc. of ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. 2011. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proc. of AISTATS*.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. arXiv:1706.09528.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL*.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. of EMNLP*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proc. of ICLR*.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proc. NAACL*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Learning How to Actively Learn: A Deep Imitation Learning Approach

Ming Liu **Wray Buntine** **Gholamreza Haffari**
Faculty of Information Technology, Monash University
`{ming.m.liu, wray.buntine, gholamreza.haffari} @ monash.edu`

Abstract

Heuristic-based active learning (AL) methods are limited when the data distribution of the underlying learning problems vary. We introduce a method that learns an AL *policy* using *imitation learning* (IL). Our IL-based approach makes use of an efficient and effective *algorithmic expert*, which provides the policy learner with good actions in the encountered AL situations. The AL strategy is then learned with a feedforward network, mapping situations to most informative query datapoints. We evaluate our method on two different tasks: text classification and named entity recognition. Experimental results show that our IL-based AL strategy is more effective than strong previous methods using heuristics and reinforcement learning.

1 Introduction

For many real-world NLP tasks, labeled data is rare while unlabelled data is abundant. Active learning (AL) seeks to learn an accurate model with minimum amount of annotation cost. It is inspired by the observation that a model can get better performance if it is allowed to choose the data points on which it is trained. For example, the learner can identify the areas of the space where it does not have enough knowledge, and query those data points which bridge its knowledge gap.

Traditionally, AL is performed using engineered heuristics in order to estimate the usefulness of unlabeled data points as queries to an annotator. Recent work (Fang et al., 2017; Bachman et al., 2017; Woodward and Finn, 2017) have focused on learning the AL querying strategy, as engineered heuristics are not flexible to exploit characteristics inherent to a given problem. The basic idea is to cast AL as a decision process, where the most informative unlabeled data point needs to be selected based on the history of previous queries. However, previous works train for the AL policy by a reinforcement learning (RL) formulation, where the rewards are provided at the end of sequences of queries. This makes learning the AL policy difficult, as the policy learner needs to deal with the credit assignment problem. Intuitively, the learner needs to observe many pairs of query sequences and the resulting end-rewards to be able to associate single queries with their utility scores.

In this work, we formulate learning AL strategies as an imitation learning problem. In particular, we consider the popular pool-based AL scenario, where an AL agent is presented with a pool of unlabelled data. Inspired by the Dataset Aggregation (DAGGER) algorithm (Ross et al., 2011), we develop an effective AL policy learning method by designing an efficient and effective algorithmic expert, which provides the AL agent with good decisions in the encountered states. We then use a deep feedforward network to learn the AL policy to associate states to actions. Unlike the RL approach, our method can get observations and actions directly from the expert's trajectory. Therefore, our trained policy can make better rankings of unlabelled datapoints in the pool, leading to more effective AL strategies.

We evaluate our method on text classification and named entity recognition. The results show our method performs better than strong AL methods using heuristics and reinforcement learning, in that it boosts the performance of the underlying model with fewer labelling queries. An open source implementation of our model is available at: `https://github.com/Grayming/ALIL`.

2 Pool-based AL as a Decision Process

We consider the popular pool-based AL setting where we are given a small set of initial labeled data and a large pool of unlabelled data, and a budget for getting the annotation of some unlabelled data by querying an oracle, e.g. a human annotator. The goal is to intelligently pick those unlabelled data for which if the annotations were available, the performance of the underlying re-trained model would be improved the most.

The main challenge in AL is how to identify and select the most beneficial unlabelled data points. Various *heuristics* have been proposed to guide the unlabelled data selection (Settles, 2010). However, there is no one AL heuristic which performs best for all problems. The goal of this paper is to provide an approach to *learn* an AL strategy which is best suited for the problem at hand, instead of resorting to ad-hoc heuristics.

The AL strategy can be learned by attempting to actively learn on tasks sampled from a distribution over the tasks (Bachman et al., 2017). The idea is to *simulate* the AL scenario on *instances* of the problem created using available labeled data, where the label of some part of the data is kept hidden. This allows to have an *automatic* oracle to reveal the labels of the queried data, resulting in an efficient way to quickly evaluate a hypothesised AL strategy. Once the AL strategy is learned on simulations, it is then applied to real AL scenarios. The more related are the tasks in the real scenario to those used to train the AL strategy, the more effective the AL strategy would be.

We are interested to train a model $m_{\boldsymbol{\phi}}$ which maps an input $\boldsymbol{x} \in \mathcal{X}$ to its label $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}$, where $\mathcal{Y}_{\boldsymbol{x}}$ is the set of labels for the input $\boldsymbol{x}$ and $\boldsymbol{\phi}$ is the parameter vector of the underling model. For example, in the named entity recognition (NER) task, the input is a sentence and the output is its label sequence, e.g. in the IBO format. Let $D = \{(\boldsymbol{x}, \boldsymbol{y})\}$ be a support set of labeled data, which is randomly partitioned into labeled D^{lab}, unlabelled D^{unl}, and evaluation D^{evl} datasets. Repeated random partitioning creates multiple instances of the AL problem. At each time step t of an AL problem, the algorithm interacts with the oracle and queries the label of a datapoint $\boldsymbol{x}_t \in D^{unl}_t$. As the result of this *action*, the followings happen:

- The automatic oracle reveals the label $\boldsymbol{y}_t$;
- The labeled and unlabelled datasets are updated to include and exclude the recently queried data point, respectively;
- The underlying model is re-trained based on the enlarged labeled data to update $\boldsymbol{\phi}$; and
- The AL algorithm receives a reward $-loss(m_{\boldsymbol{\phi}}, D^{evl})$, which is the negative loss of the current trained model on the evaluation set, defined as

$$loss(m_{\boldsymbol{\phi}}, D^{evl}) := \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D^{evl}} loss(m_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{y})$$

  where $loss(\boldsymbol{y}', \boldsymbol{y})$ is the loss incurred due to predicting $\boldsymbol{y}'$ instead of the ground truth $\boldsymbol{y}$.

More formally, a pool-based AL problem is a Markov decision process (MDP), denoted by $(S, A, Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_t), R)$ where S is the state space, A is the set of actions, $Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_t)$ is the transition function, and R is the reward function. The state $\boldsymbol{s}_t \in S$ at time t consists of the labeled D^{lab}_t and unlabelled D^{unl}_t datasets paired with the parameters of the currently trained model ϕ_t. An action $a_t \in A$ corresponds to the selection of a query datapoint, and the reward function $R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) := -loss(m_{\boldsymbol{\phi}_t}, D^{evl})$.

We aim to find the optimal AL *policy* prescribing which datapoint needs to be queried in a given state to get the most benefit. The optimal policy is found by maximising the following objective over the parameterised policies:

$$\mathbb{E}_{(D^{lab}, D^{unl}, D^{evl}) \sim \mathcal{D}} \left[\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\sum_{t=1}^{\mathcal{B}} R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) \right] \right] \quad (1)$$

where $\pi_{\boldsymbol{\theta}}$ is the *policy network* parameterised by $\boldsymbol{\theta}$, $\mathcal{D}$ is a distribution over possible AL problem instances, and $\mathcal{B}$ is the maximum number of queries made in an AL run, a.k.a. an *episode*. Following (Bachman et al., 2017), we maximise the sum of the rewards after each time step to encourage the *anytime* behaviour, i.e. the model should perform well after each label query.

3 Deep Imitation Learning to Train the AL Policy

The question remains as how can we train the policy network to maximise the training objective in eqn 1. Typical learning approaches resort to deep reinforcement learning (RL) and provide training signal at the end of each episode to learn the optimal policy (Fang et al., 2017; Bachman et al., 2017) e.g., using policy gradient methods. These approaches, however, need a large number of training episodes to learn a reasonable policy as they need to deal with the credit assignment problem, i.e. discovery of the utility of individual actions in the sequence based on the achieved reward at the end of the episode. This exacerbates the difficulty of finding a good AL policy.

We formulate learning for the AL policy as an imitation learning problem. At each state, we provide the AL agent with *a correct* action which is computed by an *algorithmic expert*. The AL agent uses the sequence of states observed in an episode paired with the expert's sequence of actions to update its policy. This directly addresses the credit assignment problem, and reduces the complexity of the problem compared to the RL approaches. In what follows, we describe the ingredients of our deep imitation learning (IL) approach, which is summarised in Algorithm 1.

Algorithmic Expert. At a given AL state $\boldsymbol{s}_t$, our algorithmic expert computes an action by evaluating the current pool of unlabeled data. More concretely, for each $\boldsymbol{x}' \in D^{pool}_{rnd}$ and its correct label $\boldsymbol{y}'$, the underlying model $m_{\boldsymbol{\phi}_t}$ is re-trained to get $m^{\boldsymbol{x}'}_{\boldsymbol{\phi}_t}$, where $D^{pool}_{rnd} \subset D^{unl}_t$ is a small subset of the current large pool of unlabeled data. The expert action is then computed as:

$$\arg\min_{\boldsymbol{x}' \in D^{pool}_{rnd}} loss(m^{\boldsymbol{x}'}_{\boldsymbol{\phi}_t}(\boldsymbol{x}), D^{evl}). \qquad (2)$$

In other words, our algorithmic expert tries a subset of actions to *roll-out* one step from the current state, in order to *efficiently* compute a reasonable action. Searching for the *optimal* action would be $\mathcal{O}(|D^{unl}|^{\mathcal{B}})$, which is computationally challenging due to (i) the large action set, and (ii) the exponential dependence on the length of the roll out. We will see in the experiments that our method efficiently learns effective AL policies.

Policy Network. Our policy network is a feed-forward network with two fully-connected hidden layers. It receives the current AL state, and provides a *preference* score for a given unlabeled data point, allowing to select the most beneficial one corresponding to the highest score. The input to our policy network consists of three parts: (i) a fixed dimensional representation of the content and the predicted label of the unlabeled data point under consideration, (ii) a fixed-dimensional representation of the content and the labels of the labeled dataset, and (iii) a fixed-dimensional representation of the content of the unlabeled dataset.

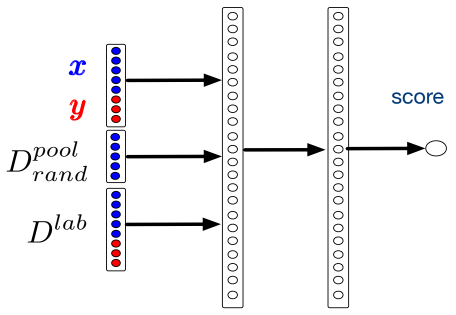


Figure 1: The policy network and its inputs.

Imitation Learning Algorithm. A typical approach to imitation learning (IL) is to train the policy network so that it mimics the expert's behaviour given training data of the encountered states (input) and actions (output) performed by the expert. The policy network's prediction affects future inputs during the execution of the policy. This violates the crucial independent and identically distributed (iid) assumption, inherent to most statistical supervised learning approaches for learning a mapping from states to actions.

We make use of Dataset Aggregation (DAGGER) (Ross et al., 2011), an iterative algorithm for IL which addresses the *non*-iid nature of the encountered states during the AL process (see Algorithm 1). In round τ of DAGGER, the learned policy network $\hat{\pi}_\tau$ is applied to the AL problem to collect a sequence of states which are paired with the expert actions. The collected pair of states and actions are aggregated to the dataset of such pairs M, collected from the previous iterations of the algorithm. The policy network is then re-trained on the aggregated set, resulting in $\hat{\pi}_{\tau+1}$ for the next iteration of the algorithm. The intuition is to build up the set of states that the algorithm is likely to encounter during its execution, in order to increase the generalization of the policy network. To better leverage the training signal from the algorithmic expert, we allow the algorithm to collect state-action pairs according to a modified policy which is a mixture of $\hat{\pi}_\tau$ and the expert policy $\tilde{\pi}^*_\tau$, i.e.

$$\pi_\tau = \beta_\tau \tilde{\pi}^* + (1 - \beta_\tau)\hat{\pi}_\tau$$

where $\beta_\tau \in [0, 1]$ is a mixing coefficient. This amounts to tossing a coin with parameter β_τ in each iteration of the algorithm to decide one of these two policies for data collection.

Re-training the Policy Network. To train our policy network, we turn the preference scores to probabilities, and optimise the parameters such that the probability of the action prescribed by the expert is maximized. More specifically, let $\mathcal{M} := \{(\boldsymbol{s}_i, \boldsymbol{a}_i)\}_{i=1}^{I}$ be the collected states paired with their expert's prescribed actions. Let D_i^{pool} be the set of unlabelled datapoints in the pool within the state, and $\boldsymbol{a}_i$ denote the datapoint selected by the expert in the set. Our training objective is $\sum_{i=1}^{I} \log Pr(\boldsymbol{a}_i|D_i^{pool})$ where

$$Pr(\boldsymbol{a}_i|D_i^{pool}) := \frac{\exp \hat{\pi}(\boldsymbol{a}_i; \boldsymbol{s}_i)}{\sum_{\boldsymbol{x} \in D_i^{pool}} \exp \hat{\pi}(\boldsymbol{x}; \boldsymbol{s}_i)}.$$

The above can be interpreted as the probability of $\boldsymbol{a}_i$ being the best action among all possible actions in the state. Following (Mnih et al., 2015), we randomly sample multiple[1] mini-batches from the *replay memory* $\mathcal{M}$, in addition to the current round's stat-action pair, in order to retrain the policy network. For each mini-batch, we make one SGD step to update the policy, where the gradients of the network parameters are calculated using the backpropagation algorithm.

Transferring the Policy. We now apply the policy learned on the source task to AL in the target task. We expect the learned policy to be effective for target tasks which are related to the source task in terms of the data distribution and characteristics. Algorithm 2 illustrates the policy transfer. The pool-based AL scenario in Algorithm 2 is cold-start; however, extending to incorporate initially available labeled data is straightforward.

4 Experiments

We conduct experiments on text classification and named entity recognition (NER). The AL scenarios include cross-domain sentiment classification, cross-lingual authorship profiling, and cross-lingual named entity recognition (NER), whereby an AL policy trained on a source domain/language is transferred to the target domain/language.

We compare our proposed AL method using imitation learning (ALIL) with the followings:

- *Random sampling*: The query datapoint is chosen randomly.

[1]In our experiments, we use 10 mini-bathes, each of which of size 100.

Algorithm 1 Learn active learning policy via imitation learning

```
Input: large labeled data $D$, max episodes $T$, budget $\mathcal{B}$,
    sample size $K$, the coin parameter $\beta$
Output: The learned policy
 1: $M \leftarrow \emptyset$                          ▷ the aggregated dataset
 2: initialise $\hat{\pi}_1$ with a random policy
 3: for $\tau$=1, ..., T do
 4:     $D^{lab}, D^{unl}, D^{evl} \leftarrow$ dataPartition($D$)
 5:     $\boldsymbol{\phi}_1 \leftarrow$ trainModel($D^{lab}$)
 6:     $c \leftarrow$ coinToss($\beta$)
 7:     for $t \in 1, \dots, \mathcal{B}$ do
 8:         $D^{pool}_{rnd} \leftarrow$ sampleUniform($D^{unl}, K$)
 9:         $\boldsymbol{s}_t \leftarrow (D^{lab}, D^{pool}_{rnd}, \boldsymbol{\phi}_t)$
10:         $\boldsymbol{a}_t \leftarrow \arg\min_{\boldsymbol{x}' \in D^{pool}_{rnd}} loss(m^{\boldsymbol{x}'}_{\boldsymbol{\phi}_t}, D^{evl})$
11:         if $c$ is head then                      ▷ the expert
12:             $\boldsymbol{x}_t \leftarrow \boldsymbol{a}_t$
13:         else                                     ▷ the policy
14:             $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x}' \in D^{pool}_{rnd}} \hat{\pi}_\tau(\boldsymbol{x}'; \boldsymbol{s}_t)$
15:         end if
16:         $D^{lab} \leftarrow D^{lab} + \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}$
17:         $D^{unl} \leftarrow D^{unl} - \{\boldsymbol{x}_t\}$
18:         $M \leftarrow M + \{(\boldsymbol{s}_t, \boldsymbol{a}_t)\}$
19:         $\boldsymbol{\phi}_{t+1} \leftarrow$ retrainModel($\boldsymbol{\phi}_t, D^{lab}$)
20:     end for
21:     $\hat{\pi}_{\tau+1} \leftarrow$ retrainPolicy($\hat{\pi}_\tau, M$)
22: end for
23: return $\hat{\pi}_{T+1}$
```

Algorithm 2 Active learning by policy transfer

```
Input: unlabeled pool $D^{unl}$, budget $\mathcal{B}$, policy $\hat{\pi}$
Output: labeled dataset and trained model
 1: $D^{lab} \leftarrow \emptyset$
 2: initialise $\boldsymbol{\phi}$ randomly
 3: for $t \in 1, \dots, \mathcal{B}$ do
 4:     $\boldsymbol{s}_t \leftarrow (D^{lab}, D^{unl}, \boldsymbol{\phi})$
 5:     $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x}' \in D^{unl}} \hat{\pi}(\boldsymbol{x}'; \boldsymbol{s}_t)$
 6:     $\boldsymbol{y}_t \leftarrow$ askAnnotation($\boldsymbol{x}_t$)
 7:     $D^{lab} \leftarrow D^{lab} + \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}$
 8:     $D^{unl} \leftarrow D^{unl} - \{\boldsymbol{x}_t\}$
 9:     $\phi \leftarrow$ retrainModel($\boldsymbol{\phi}, D^{lab}$)
10: end for
11: return $D^{lab}$ and $\boldsymbol{\phi}$
```

- *Diversity sampling*: The query datapoint is $\arg\min_{\boldsymbol{x}} \sum_{\boldsymbol{x}' \in D^{lab}}$ Jaccard$(\boldsymbol{x}, \boldsymbol{x}')$, where the Jaccard coefficient between the unigram features of the two given texts is used as the similarity measure.

- *Uncertainty-based sampling*: For text classification, we use the datapoint with the highest predictive entropy, $\arg\max_{\boldsymbol{x}} -\sum_y p(y|\boldsymbol{x}, D^{lab}) \log p(y|\boldsymbol{x}, D^{lab})$ where $p(\boldsymbol{y}|\boldsymbol{x}, D^{lab})$ comes from the underlying model. We further use a state-of-the-art extension of this method, called *uncertainty with rationals* (Sharma et al., 2015), which not only considers uncertainty but also looks whether

		doc. (src/tgt)	
src	**tgt**	**number**	**avg. len. (tokens)**
elec.	music dev.	27k/1k	35/20
book	movie	24k/2k	140/150
en	sp	3.6k/4.2k	1.15k/1.35k
en	pt	3.6k/1.2k	1.15k/1.03k

Table 1: The data sets used in sentiment classification (top part) and gender profiling (bottom part).

the unlabelled document contains sentiment words or phrases that were returned as rationales for any of the existing labeled documents. For NER, we use the Total Token Entropy (TTE) as the uncertainty sampling method, $\arg\max_{\boldsymbol{x}} -\sum_{i=1}^{|\boldsymbol{x}|}\sum_{y_i} p(y_i|\boldsymbol{x}, D^{lab}) \log p(y_i|\boldsymbol{x}, D^{lab})$ which has been shown to be the best heuristic for this task among 17 different heuristics (Settles and Craven, 2008).

- *PAL*: A reinforcement learning based approach (Fang et al., 2017), which makes use a deep Q-network to make the selection decision for stream-based active learning.

4.1 Text Classification

Datasets and Setup. The first task is sentiment classification, in which product reviews express either positive or negative sentiment. The data comes from the Amazon product reviews (McAuley and Yang, 2016); see Table 1 for data statistics.

The second task is Authorship Profiling, in which we aim to predict the gender of the text author. The data comes from the gender profiling task in PAN 2017 (Rangel et al., 2017), which consists of a large Twitter corpus in multiple languages: English (en), Spanish (es) and Portuguese (pt). For each language, all tweets collected from a user constitute one document; Table 1 shows data statistics. The multilingual embeddings for this task come from off-the-shelf CCA-trained embeddings (Ammar et al., 2016) for twelve languages, including English, Spanish and Portuguese. We fix these word embeddings during training of both the policy and the underlying classification model.

For training, 10% of the source data is used as the evaluation set for computing the best action in imitation learning. We run $T = 100$ episodes with the budget $\mathcal{B} = 100$ documents in each episode, set the sample size $K = 5$, and fix the mixing coefficient $\beta_\tau = 0.5$. For testing, we take 90% of the target data as the unlabeled pool, and the remaining 10% as the test set. We show the test accuracy w.r.t. the number of labelled documents selected in the AL process.

As the underlying model $m_{\boldsymbol{\phi}}$, we use a fast and efficient text classifier based on convolutional neural networks. More specifically, we apply 50 convolutional filters with ReLU activation on the embedding of all words in a document $\boldsymbol{x}$, where the width of the filters is 3. The filter outputs are averaged to produce a 50-dimensional document representation $\boldsymbol{h}(\boldsymbol{x})$, which is then fed into a softmax to predict the class.

Representing state-action. The input to the policy network, i.e. the feature vector representing a state-action pair, includes: the candidate document represented by the convolutional net $\boldsymbol{h}(\boldsymbol{x})$, the distribution over the document's class labels $m_{\boldsymbol{\phi}}(\boldsymbol{x})$, the sum of all document vector representations in the labeled set $\sum_{\boldsymbol{x}'\in D^{lab}} \boldsymbol{h}(\boldsymbol{x}')$, the sum of all document vectors in the random pool of unlabelled data $\sum_{\boldsymbol{x}'\in D^{pool}_{rnd}} \boldsymbol{h}(\boldsymbol{x}')$, and the empirical distribution of class labels in the labeled dataset.

Results. Fig 2 shows the results on product sentiment prediction and authorship profiling, in cross-domain and cross-lingual AL scenarios[2]. Our ALIL method consistently outperforms both heuristic-based and RL-based (PAL) (Fang et al., 2017) approaches across all tasks. ALIL tends to convergence faster than other methods, which indicates its policy can quickly select the most informative datapoints. Interestingly, the uncertainty and diversity sampling heuristics perform worse than random sampling on sentiment classification. We speculate this may be due to these two heuristics not being able to capture the polarity information during the data selection process. PAL performs on-par with uncertainty with rationals on musical device, both of which outperform the traditional diversity and uncertainty sampling heuristics. Interestingly, PAL is outperformed by random sampling on movie reviews, and by the traditional uncertainty sampling heuristic on authorship profiling tasks. We attribute this to ineffectiveness of the RL-based approach for learning a reasonable AL query strategy.

We further investigate combining the transfer of the policy network with the transfer of the underlying classifier. That is, we first train a classi-

[2]Uncertainty with rationale cannot be done for authorship profiling as the rationales come from a sentiment dictionary.

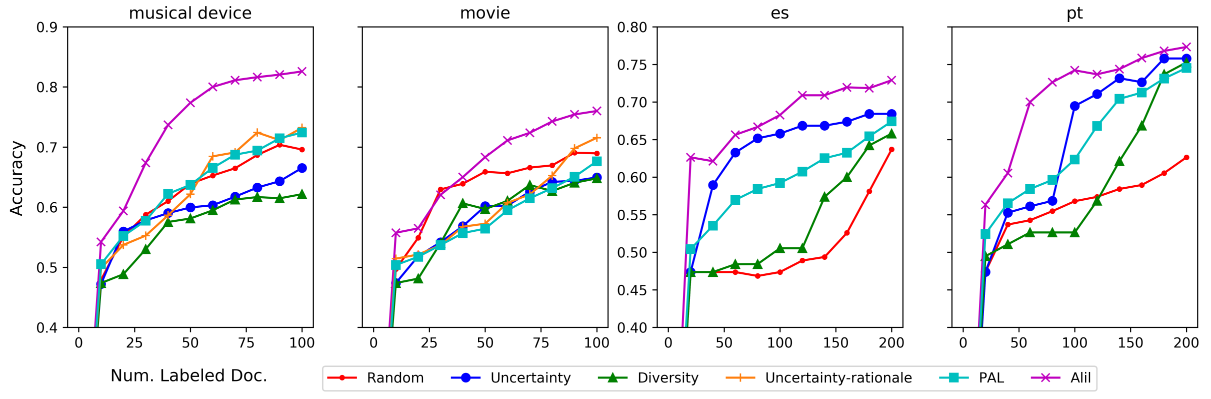


Figure 2: The performance of different active learning methods for cross domain sentiment classification (left two plots) and cross lingual authorship profiling (right two plots).

fier on all of the annotated data from the source domain/language. Then, this classifier is ported to the target domain/language; for cross-language transfer, we make use of multilingual word embeddings. We start the AL process starting from the transferred classifier, referred to as the warm-start AL. We compare the performance of the *directly transferred* classifier with those obtained after the AL process in the warm-start and cold-start scenarios. The results are shown in Table 2. We have run the cold-start and warm-start AL for 25 times, and reported the average accuracy in Table 2. As seen from the results, both the cold and warm start AL settings outperform the direct transfer significantly, and the warm start consistently gets higher accuracy than the cold start. The difference between the results are statistically significant, with a p-value of .001, according to McNemar test[3] (Dietterich, 1998).

	musical	movie	es	pt
direct transfer	0.715	0.640	0.675	0.740
cold-start AL	0.800	0.760	0.728	0.773
warm-start AL	**0.825**	**0.765**	**0.730**	**0.780**

Table 2: Classifiers performance under three different transfer settings.

4.2 Named Entity Recognition

Data and setup We use NER corpora from the CONLL2002/2003 shared tasks, which include annotated text in English (en), German (de), Spanish (es), and Dutch (nl). The original annotation is based on IOB1, which we convert to the IO labelling scheme. Following Fang et al. (2017), we consider two experimental conditions: (i) the bilingual scenario where English is the source (used for policy training) and other languages are the target, and (ii) the multilingual scenario where one of the languages (except English) is the target and the remaining ones are the source used in joint training of the AL policy. The underlying model $m_{\boldsymbol{\phi}}$ is a conditional random field (CRF) treating NER as a sequence labelling task. The prediction is made using the Viterbi algorithm.

In the existing corpus partitions from CoNLL, each language has three subsets: **train**, **testa** and **testb**. During policy training with the source language(s), we combine these three subsets, shuffle, and re-split them into simulated training, unlabelled pool, and evaluation sets in every episode. We run $N = 100$ episodes with the budget $\mathcal{B} = 200$, and set the sample size $k = 5$. When we transfer the policy to the target language, we do one episode and select $\mathcal{B}$ datapoints from **train** (treated as the pool of unlabeled data) and report F1 scores on **testa**.

Representing state-action. The input to the policy network includes the representation of the candidate sentence using the sum of its words' embeddings $\boldsymbol{h}(\boldsymbol{x})$, the representation of the labelling marginals using the label-level convolutional network $\text{cnn}_{\text{lab}}(\mathbb{E}_{m_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{y}])$ (Fang et al., 2017), the representation of sentences in the labeled data $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in D^{lab}} \boldsymbol{h}(\boldsymbol{x}')$, the representation of sentences in the random pool of unlabelled data $\sum_{\boldsymbol{x}'\in D^{pool}_{rnd}} \boldsymbol{h}(\boldsymbol{x}')$, the representation of ground-truth labels in the labeled data $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in D^{lab}} \text{cnn}_{\text{lab}}(\boldsymbol{y}')$ using the empirical distributions, and the confidence of the sequential pre-

[3]As the contingency table needed for the McNemar test, we have used the average counts across the 25 runs.

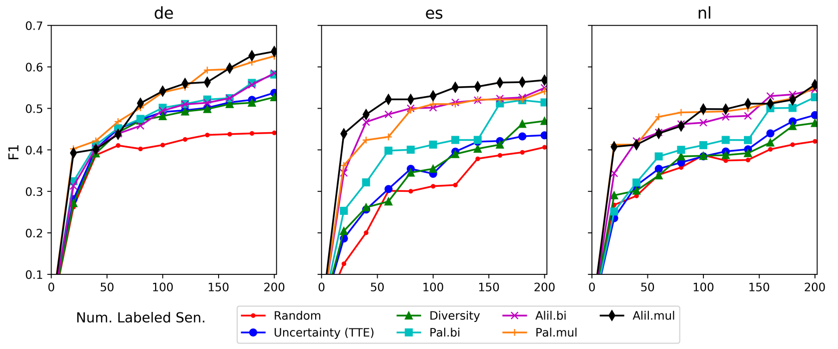


Figure 3: The performance of active learning methods on the bilingual and multilingual settings for three target languages: German (de), Spanish (es) and Dutch (nl).

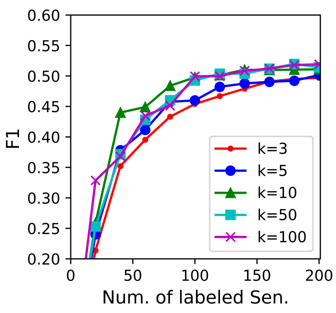


Figure 4: The learning curves of agents with different K on Spanish (es) NER.

diction $\sqrt[|\boldsymbol{x}|]{\max_{\boldsymbol{y}} m_{\boldsymbol{\phi}}(\boldsymbol{y}|\boldsymbol{x})}$, where $|\boldsymbol{x}|$ denotes the length of the sentence $\boldsymbol{x}$. For the word embeddings, we use off-the-shelf CCA trained multilingual embeddings (Ammar et al., 2016) with 40 dimensions; we fix these during policy training.

Results. Fig. 3 shows the results for three target languages. In addition to the strong heuristic-based methods, we compare our imitation learning approach (ALIL) with the reinforcement learning approach (PAL) (Fang et al., 2017), in both bilingual (bi) and multilingual (mul) transfer settings. Across all three languages, ALIL.bi and ALIL.mul outperform the heuristic methods, including Uncertainty Sampling based on TTE. This is expected as the uncertainty sampling largely relies on a high quality underlying model, and diversity sampling ignores the labelling information. In the bilingual case, ALIL.bi outperforms PAL.bi on Spanish (es) and Dutch (nl), and performs similarly on German (de). In the multilingual case, ALIL.mul achieves the best performance on Spanish, and performs competitively with PAL.mul on German and Dutch.

4.3 Analysis

Insight on the selected data. We compare the data selected by ALIL to other methods. This will confirm that ALIL learns policies which are suitable for the problem at hand, without resorting to a fixed engineered heuristics. For this analysis, we report the mean reciprocal rank (MRR) of the data points selected by the ALIL policy under rankings of the unlabelled pool generated by the uncertainty and diversity sampling. Furthermore, we measure the fraction of times the decisions made by the ALIL policy agrees with those which would

	movie sentiment	gender pt	NER es
acc Unc.	0.06	0.58	0.51
MRR Unc.	0.083	0.674	0.551
acc Div.	0.05	0.52	0.45
MRR Div.	0.057	0.593	0.530
acc PAL	0.15	0.56	0.52

Table 3: The first four rows show MRR and accuracy of instances returned by ALIL under the rankings of uncertainty and diversity sampling, the last row give average accuracy of instances under PAL.

have been made by the heuristic methods, which is measured by the accuracy (acc). Table 3 report these measures. As we can see, for sentiment classification since uncertainty and diversity sampling perform badly, ALIL has a big disagreement with them on the selected data points. While for gender classification on Portuguese and NER on Spanish, ALIL shows much more agreement with other three heuristics.

Lastly, we compare chosen queries by ALIL to those by PAL, to investigate the extent of the agreement between these two methods. This is simply measure by the fraction of identical query data points among the total number of queries (i.e. accuracy). Since PAL is stream-based and sensitive to the order in which it receives the data points, we report the *average* accuracy taken over multiple runs with random input streams. The expected accuracy numbers are reported in Table 3. As seen, ALIL has higher overlap with PAL than the heuristic-based methods, in terms of the selected queries.

Sensitivity to K. As seen in Algorithm 1, we resort to an approximate algorithmic expert, which selects the best action in a random subset of the

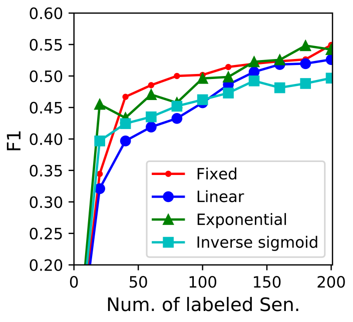


Figure 5: The learning curves of agents with different schedules for β before the trajectory on Spanish (es) NER.

pool of unlabelled data with size K, in order to make the policy training efficient. Note that, in policy training, setting K to one and the size of the unlabelled data pool correspond to stream-based and pool-based AL scenarios, respectively. By changing K to values between these two extremes, we can analyse the effect of the quality of the algorithmic expert on the trained policy; Figure 4 shows the results. A larger candidate set may correspond to a better learned policy, needed to be traded off with the training time growing linearly with K. Interestingly, even small candidate sets lead to strong AL policies as increasing K beyond 10 does not change the performance significantly.

Dynamically changing β. In our algorithm, β plays an important role as it trades off exploration versus exploitation. In the above experiments, we fix it to 0.5; however, we can change its value throughout trajectory collection as a function of τ (see Algorithm 1). We investigate schedules which tend to put more emphasis on exploration and exploitation towards the beginning and end of data collection, respectively. We investigate the following schedules: (i) linear $\beta_\tau = \max(0.5, 1 - 0.01\tau)$, (ii) exponential $\beta_\tau = 0.9^\tau$, and (iii) and inverse sigmoid $\beta_\tau = \frac{5}{5+exp(\tau/5)}$, as a function of iterations. Fig. 5 shows the comparisons of these schedules. The learned policy seems to perform competitively with either a fixed or an exponential schedule. We have also investigated tossing the coin in each step *within* the trajectory roll out, but found that it is more effective to have it *before* the full trajectory roll out (as currently done in Algorithm 1).

5 Related Work

Traditional active learning algorithms rely on various heuristics (Settles, 2010), such as uncertainty sampling (Settles and Craven, 2008; Houlsby et al., 2011), query-by-committee (Gilad-Bachrach et al., 2006), and diversity sampling (Brinker, 2003; Joshi et al., 2009; Yang et al., 2015). Apart from these, different heuristics can be combined, thus creating integrated strategy which consider one or more heuristics at the same time. Combined with transfer learning, pre-existing labeled data from related tasks can help improve the performance of an active learner (Xiao and Guo, 2013; Kale and Liu, 2013; Huang and Chen, 2016; Konyushkova et al., 2017). More recently, deep reinforcement learning is used as the framework for *learning* active learning algorithms, where the active learning cycle is considered as a decision process. (Woodward and Finn, 2017) extended one shot learning to active learning and combined reinforcement learning with a deep recurrent model to make labeling decisions. (Bachman et al., 2017) introduced a policy gradient based method which jointly learns data representation, selection heuristic as well as the model prediction function. (Fang et al., 2017) designed an active learning algorithm based on a deep Q-network, in which the action corresponds to binary annotation decisions applied to a stream of data. The learned policy can then be transferred between languages or domains.

Imitation learning (IL) refers to an agent's acquisition of skills or behaviours by observing an expert's trajectory in a given task. It helps reduce sequential prediction tasks into supervised learning by employing a (near) optimal oracle at training time. Several IL algorithms has been proposed in sequential prediction tasks, including SEARA (Daumé et al., 2009), AggreVaTe (Ross and Bagnell, 2014), DaD (Venkatraman et al., 2015), LOLS(Chang et al., 2015), DeeplyAggreVaTe (Sun et al., 2017). Our work is closely related to Dagger (Ross et al., 2011), which can guarantee to find a good policy by addressing the dependency nature of encountered states in a trajectory.

6 Conclusion

In this paper, we have proposed a new method for learning active learning algorithms using deep imitation learning. We formalize pool-based active learning as a Markov decision process, in which active learning corresponds to the selection decision of the most informative data points from the pool. Our efficient algorithmic expert provides state-action pairs from which effective active learning policies can be learned. We show that the algorithmic expert allows direct policy learning, while at the same time, the learned policies transfer successfully between domains and languages, demonstrating improvement over previous heuristic and reinforcement learning approaches.

Acknowledgments

We would like to thank the feedback from anonymous reviewers. G. H. is grateful to Trevor Cohn for interesting discussions. This work was supported by computational resources from the Multimodal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) at Monash University.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. 2017. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 301–310, International Convention Centre, Sydney, Australia. PMLR.

Klaus Brinker. 2003. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 59–66.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2058–2066.

Ido Dagan and Sean P Engelson. 1995. Selective sampling in natural language learning. In *Proceedings of IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*.

Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. 2006. Query by committee made real. In *Advances in neural information processing systems*, pages 443–450.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Sheng-Jun Huang and Songcan Chen. 2016. Transfer learning with active queries from source domain. In *IJCAI*, pages 1592–1598.

Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. 2009. Multi-class active learning for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2372–2379. IEEE.

David Kale and Yan Liu. 2013. Accelerating active learning with transfer learning. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1085–1090. IEEE.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2017. Learning active learning from data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4225–4235. Curran Associates, Inc.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Kamal Nigam and Andrew McCallum. 1998. Pool-based active learning for text classification. In *Conference on Automated Learning and Discovery (CONALD)*.

Francisco Rangel, Paolo Rosso, Martin Potthast, and Benno Stein. 2017. Overview of the 5th author profiling task at PAN 2017: Gender and language variety identification in twitter. *Working Notes Papers of the CLEF*.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Claude E Shannon. 1948. A note on the concept of entropy. *Bell System Tech. J*, 27:379–423.

Manali Sharma, Di Zhuang, and Mustafa Bilgic. 2015. Active learning with rationales for text classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 441–451.

Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. 2017. Deeply aggrevated: Differentiable imitation learning for sequential prediction. *arXiv preprint arXiv:1703.01030*.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030.

Mark Woodward and Chelsea Finn. 2017. Active one-shot learning. *arXiv preprint arXiv:1702.06559*.

Min Xiao and Yuhong Guo. 2013. Online active learning for cost sensitive domain adaptation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 1–9.

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127.

Training Classifiers with Natural Language Explanations

Braden Hancock
Computer Science Dept.
Stanford University
bradenjh@cs.stanford.edu

Paroma Varma
Electrical Engineering Dept.
Stanford University
paroma@stanford.edu

Stephanie Wang
Computer Science Dept.
Stanford University
steph17@stanford.edu

Martin Bringmann
OccamzRazor
San Francisco, CA
martin@occamzrazor.com

Percy Liang
Computer Science Dept.
Stanford University
pliang@cs.stanford.edu

Christopher Ré
Computer Science Dept.
Stanford University
chrismre@cs.stanford.edu

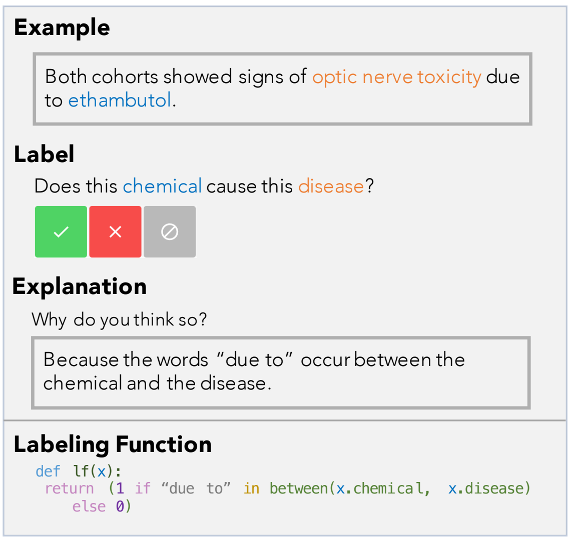
Abstract

Training accurate classifiers requires many labels, but each label provides only limited information (one bit for binary classification). In this work, we propose `BabbleLabble`, a framework for training classifiers in which an annotator provides a natural language explanation for each labeling decision. A semantic parser converts these explanations into programmatic labeling functions that generate noisy labels for an arbitrary amount of unlabeled data, which is used to train a classifier. On three relation extraction tasks, we find that users are able to train classifiers with comparable F1 scores from 5–100 faster by providing explanations instead of just labels. Furthermore, given the inherent imperfection of labeling functions, we find that a simple rule-based semantic parser suffices.

1 Introduction

The standard protocol for obtaining a labeled dataset is to have a human annotator view each example, assess its relevance, and provide a label (e.g., positive or negative for binary classification). However, this only provides one bit of information per example. This invites the question: how can we get more information per example, given that the annotator has already spent the effort reading and understanding an example?

Previous works have relied on identifying relevant parts of the input such as labeling features (Druck et al., 2009; Raghavan et al., 2005; Liang et al., 2009), highlighting rationale phrases in text (Zaidan and Eisner, 2008; Arora and Nyberg, 2009), or marking relevant regions in images (Ahn et al., 2006). But there are certain types of information which cannot be easily reduced to annotating a portion of the input, such as the absence of a certain word, or the presence of at least two words. In this work, we tap into the power of natural language and allow annotators to provide supervision to a classifier via *natural language explanations*.


Example

Both cohorts showed signs of optic nerve toxicity due to ethambutol.

Label

Does this chemical cause this disease?

Explanation

Why do you think so?

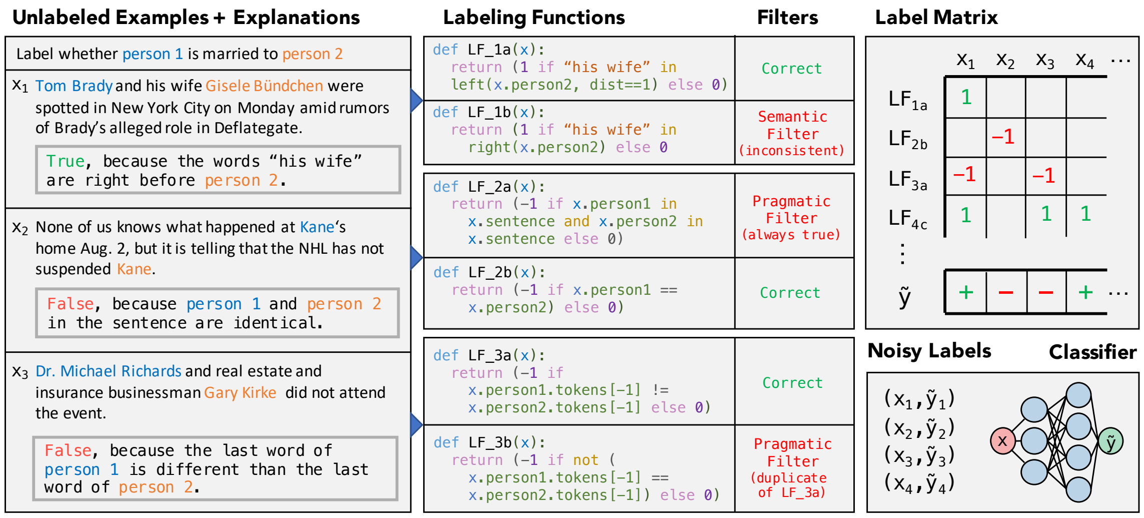
Because the words "due to" occur between the chemical and the disease.

Labeling Function

```
def lf(x):
  return (1 if "due to" in between(x.chemical, x.disease)
      else 0)
```

Figure 1: In `BabbleLabble`, the user provides a natural language explanation for each labeling decision. These explanations are parsed into labeling functions that convert unlabeled data into a large labeled dataset for training a classifier.

Specifically, we propose a framework in which annotators provide a natural language explanation for each label they assign to an example (see Figure 1). These explanations are parsed into logical forms representing labeling functions (LFs), functions that heuristically map examples to labels (Ratner et al., 2016). The labeling functions are


Unlabeled Examples + Explanations
Label whether person 1 is married to person 2
x_1 Tom Brady and his wife Gisele Bündchen were spotted in New York City on Monday amid rumors of Brady's alleged role in Deflategate.
True, because the words "his wife" are right before person 2.
x_2 None of us knows what happened at Kane's home Aug. 2, but it is telling that the NHL has not suspended Kane.
False, because person 1 and person 2 in the sentence are identical.
x_3 Dr. Michael Richards and real estate and insurance businessman Gary Kirke did not attend the event.
False, because the last word of person 1 is different than the last word of person 2.
Labeling Functions
Filters
Correct
Semantic Filter (inconsistent)
Pragmatic Filter (always true)
Correct
Correct
Pragmatic Filter (duplicate of LF_3a)
Label Matrix
Noisy Labels
Classifier

```
def LF_1a(x):
  return (1 if "his wife" in
  left(x.person2, dist==1) else 0)

def LF_1b(x):
  return (1 if "his wife" in
    right(x.person2) else 0

def LF_2a(x):
  return (-1 if x.person1 in
    x.sentence and x.person2 in
    x.sentence else 0)

def LF_2b(x):
  return (-1 if x.person1 ==
    x.person2) else 0)

def LF_3a(x):
  return (-1 if
    x.person1.tokens[-1] !=
    x.person2.tokens[-1] else 0)

def LF_3b(x):
  return (-1 if not (
    x.person1.tokens[-1] ==
    x.person2.tokens[-1]) else 0)
```

$$\begin{array}{c|cccc}
 & x_1 & x_2 & x_3 & x_4 \; \cdots \\
\hline
LF_{1a} & 1 & & & \\
LF_{2b} & & -1 & & \\
LF_{3a} & -1 & & -1 & \\
LF_{4c} & 1 & & 1 & 1 \\
\vdots & & & & \\
\tilde{y} & + & - & - & + \; \cdots
\end{array}$$

$(x_1, \tilde{y}_1)$
$(x_2, \tilde{y}_2)$
$(x_3, \tilde{y}_3)$
$(x_4, \tilde{y}_4)$

Figure 2: Natural language explanations are parsed into candidate labeling functions (LFs). Many incorrect LFs are filtered out automatically by the filter bank. The remaining functions provide heuristic labels over the unlabeled dataset, which are aggregated into one noisy label per example, yielding a large, noisily-labeled training set for a classifier.

then executed on many unlabeled examples, resulting in a large, weakly-supervised training set that is then used to train a classifier.

Semantic parsing of natural language into logical forms is recognized as a challenging problem and has been studied extensively (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011; Liang, 2016). One of our major findings is that in our setting, even a simple rule-based semantic parser suffices for three reasons: First, we find that the majority of incorrect LFs can be automatically filtered out either semantically (e.g., is it consistent with the associated example?) or pragmatically (e.g., does it avoid assigning the same label to the entire training set?). Second, LFs near the gold LF in the space of logical forms are often just as accurate (and sometimes even more accurate). Third, techniques for combining weak supervision sources are built to tolerate some noise (Alfonseca et al., 2012; Takamatsu et al., 2012; Ratner et al., 2018). The significance of this is that we can deploy the same semantic parser across tasks without task-specific training. We show how we can tackle a real-world biomedical application with the same semantic parser used to extract instances of spouses.

Our work is most similar to that of Srivastava et al. (2017), who also use natural language explanations to train a classifier, but with two important differences. First, they jointly train a task-specific semantic parser and classifier, whereas we use a simple rule-based parser. In Section 4, we find that in our weak supervision framework, the rule-based semantic parser and the perfect parser yield nearly identical downstream performance. Second, while they use the logical forms of explanations to produce *features* that are fed directly to a classifier, we use them as *functions* for labeling a much larger training set. In Section 4, we show that using functions yields a 9.5 F1 improvement (26% relative improvement) over features, and that the F1 score scales with the amount of available unlabeled data.

We validate our approach on two existing datasets from the literature (extracting spouses from news articles and disease-causing chemicals from biomedical abstracts) and one real-world use case with our biomedical collaborators at OccamzRazor to extract protein-kinase interactions related to Parkinson's disease from text. We find empirically that users are able to train classifiers with comparable F1 scores up to two orders of magnitude faster when they provide natural language explanations instead of individual labels. Our code and data can be found at `https://github.com/HazyResearch/babble`.

2 The `BabbleLabble` Framework

The `BabbleLabble` framework converts natural language explanations and unlabeled data into a noisily-labeled training set (see Figure 2). There are three key components: a semantic parser, a filter bank, and a label aggregator. The semantic

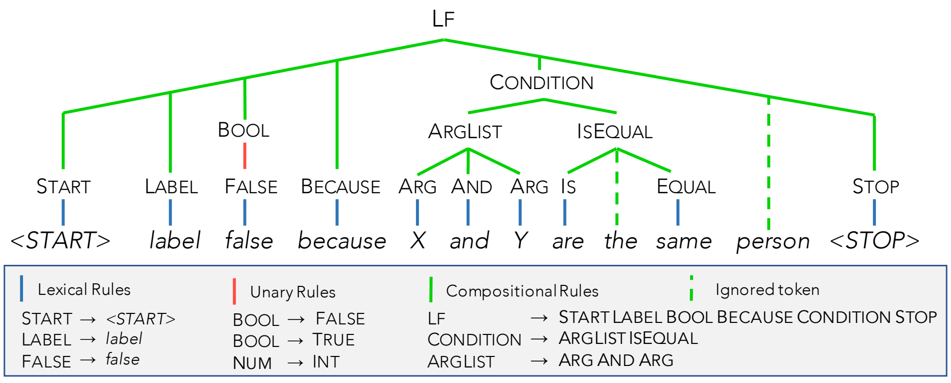


Figure 3: Valid parses are found by iterating over increasingly large subspans of the input looking for matches among the right hand sides of the rules in the grammar. Rules are either lexical (converting tokens into symbols), unary (converting one symbol into another symbol), or compositional (combining many symbols into a single higher-order symbol). A rule may optionally ignore unrecognized tokens in a span (denoted here with a dashed line).

parser converts natural language explanations into a set of logical forms representing labeling functions (LFs). The filter bank removes as many incorrect LFs as possible without requiring ground truth labels. The remaining LFs are applied to unlabeled examples to produce a matrix of labels. This label matrix is passed into the label aggregator, which combines these potentially conflicting and overlapping labels into one label for each example. The resulting labeled examples are then used to train an arbitrary discriminative model.

2.1 Explanations

To create the input explanations, the user views a subset S of an unlabeled dataset D (where $|S| \ll |D|$) and provides for each input $x_i \in S$ a label y_i and a natural language explanation e_i, a sentence explaining why the example should receive that label. The explanation e_i generally refers to specific aspects of the example (e.g., in Figure 2, the location of a specific string "his wife").

2.2 Semantic Parser

The semantic parser takes a natural language explanation e_i and returns a set of LFs (logical forms or labeling functions) $\{f_1, \ldots, f_k\}$ of the form $f_i : \mathcal{X} \to \{-1, 0, 1\}$ in a binary classification setting, with 0 representing abstention. We emphasize that the goal of this semantic parser is *not* to generate the single correct parse, but rather to have coverage over many potentially useful LFs.[1]

[1]Indeed, we find empirically that an incorrect LF nearby the correct one in the space of logical forms actually has higher end-task accuracy 57% of the time (see Section 4.2).

We choose a simple rule-based semantic parser that can be used without any training. Formally, the parser uses a set of rules of the form $\alpha \to \beta$, where α can be replaced by the token(s) in β (see Figure 3 for example rules). To identify candidate LFs, we recursively construct a set of valid parses for each span of the explanation, based on the substitutions defined by the grammar rules. At the end, the parser returns all valid parses (LFs in our case) corresponding to the entire explanation.

We also allow an arbitrary number of tokens in a given span to be ignored when looking for a matching rule. This improves the ability of the parser to handle unexpected input, such as unknown words or typos, since the portions of the input that *are* parseable can still result in a valid parse. For example, in Figure 3, the word "person" is ignored.

All predicates included in our grammar (summarized in Table 1) are provided to annotators, with minimal examples of each in use (Appendix A). Importantly, all rules are domain independent (e.g., all three relation extraction tasks that we tested used the same grammar), making the semantic parser easily transferrable to new domains. Additionally, while this paper focuses on the task of relation extraction, in principle the `BabbleLabble` framework can be applied to other tasks or settings by extending the grammar with the necessary primitives (e.g., adding primitives for rows and columns to enable explanations about the alignments of words in tables). To guide the construction of the grammar, we collected 500 explanations for the `Spouse` domain from workers

Predicate	Description
`bool, string, int, float, tuple, list, set`	Standard primitive data types
`and, or, not, any, all, none`	Standard logic operators
$=, \neq, <, \leq, >, \geq$	Standard comparison operators
`lower, upper, capital, all_caps`	Return True for strings of the corresponding case
`starts_with, ends_with, substring`	Return True if the first string starts/ends with or contains the second
`person, location, date, number, organization`	Return True if a string has the corresponding NER tag
`alias`	A frequently used list of words may be predefined and referred to with an alias
`count, contains, intersection`	Operators for checking size, membership, or common elements of a `list/set`
`map, filter`	Apply a functional primitive to each member of `list/set` to transform or filter the elements
`word_distance, character_distance`	Return the distance between two strings by words or characters
`left, right, between, within`	Return as a string the text that is left/right/within some distance of a string or between two designated strings

Table 1: Predicates in the grammar supported by `BabbleLabble`'s rule-based semantic parser.

on Amazon Mechanical Turk and added support for the most commonly used predicates. These were added before the experiments described in Section 4. Altogether the grammar contains 200 rule templates.

2.3 Filter Bank

The input to the filter bank is a set of candidate LFs produced by the semantic parser. The purpose of the filter bank is to discard as many incorrect LFs as possible without requiring additional labels. It consists of two classes of filters: semantic and pragmatic.

Recall that each explanation e_i is collected in the context of a specific labeled example (x_i, y_i). The *semantic filter* checks for LFs that are inconsistent with their corresponding example; formally, any LF f for which $f(x_i) \neq y_i$ is discarded. For example, in the first explanation in Figure 2, the word "right" can be interpreted as either "immediately" (as in "right before") or simply "to the right." The latter interpretation results in a function that is inconsistent with the associated example (since "his wife" is actually to the left of person 2), so it can be safely removed.

The *pragmatic filters* removes LFs that are constant, redundant, or correlated. For example, in Figure 2, `LF_2a` is constant, as it labels every example positively (since all examples contain two people from the same sentence). `LF_3b` is redundant, since even though it has a different syntax tree from `LF_3a`, it labels the training set identically and therefore provides no new signal.

Finally, out of all LFs from the same explanation that pass all the other filters, we keep only the most specific (lowest coverage) LF. This prevents multiple correlated LFs from a single example from dominating.

As we show in Section 4, over three tasks, the filter bank removes 86% of incorrect parses, and the incorrect ones that remain have average end-task accuracy within 2.5% of the corresponding correct parses.

2.4 Label Aggregator

The label aggregator combines multiple (potentially conflicting) suggested labels from the LFs and combines them into a single probabilistic label per example. Concretely, if m LFs pass the filter bank and are applied to n examples, the label aggregator implements a function $f : \{-1, 0, 1\}^{m \times n} \to [0, 1]^n$.

A naive solution would be to use a simple majority vote, but this fails to account for the fact that LFs can vary widely in accuracy and coverage. Instead, we use data programming (Ratner et al., 2016), which models the relationship between the true labels and the output of the labeling functions as a factor graph. More specifically, given the true labels $Y \in \{-1, 1\}^n$ (latent) and label matrix $\Lambda \in \{-1, 0, 1\}^{m \times n}$ (observed) where $\Lambda_{i,j} = \text{LF}_i(x_j)$, we define two types of factors representing labeling propensity and accuracy:

$$\phi_{i,j}^{\text{Lab}}(\Lambda, Y) = \mathbb{1}\{\Lambda_{i,j} \neq 0\} \tag{1}$$
$$\phi_{i,j}^{\text{Acc}}(\Lambda, Y) = \mathbb{1}\{\Lambda_{i,j} = y_j\}. \tag{2}$$

Denoting the vector of factors pertaining to a given data point x_j as $\phi_j(\Lambda, Y) \in \mathbb{R}^m$, define the model:

$$p_w(\Lambda, Y) = Z_w^{-1} \exp\Big(\sum_{j=1}^{n} w \cdot \phi_j(\Lambda, Y)\Big), \tag{3}$$

Spouse (person 1, person 2)	Example	They include Joan Ridsdale, a 62-year-old payroll administrator from County Durham who was hit with a €16,000 tax bill when her husband Gordon died.
	Explanation	True, because the phrase "her husband" is within three words of person 2.
Disease (chemical, disease)	Example	Young women on replacement estrogens for ovarian failure after cancer therapy may also have increased risk of endometrial carcinoma and should be examined periodically.
	Explanation	True, because "risk of" comes before the disease.
Protein (protein, kinase)	Example	Here we show that c-Jun N-terminal kinases JNK1, JNK2 and JNK3 phosphorylate tau at many serine/threonine-prolines, as assessed by the generation of the epitopes of phosphorylation-dependent anti-tau antibodies.
	Explanation	True, because at least one of the words 'phosphorylation', 'phosphorylate', 'phosphorylated', 'phosphorylates' is found in the sentence and the number of words between the protein and kinase is smaller than 8."

Figure 4: An example and explanation for each of the three datasets.

where $w \in \mathbb{R}^{2m}$ is the weight vector and Z_w is the normalization constant. To learn this model without knowing the true labels Y, we minimize the negative log marginal likelihood given the observed labels Λ:

$$\hat{w} = \arg\min_{w} -\log \sum_{Y} p_w(\Lambda, Y) \tag{4}$$

using SGD and Gibbs sampling for inference, and then use the marginals $p_{\hat{w}}(Y \mid \Lambda)$ as probabilistic training labels.

Intuitively, we infer accuracies of the LFs based on the way they overlap and conflict with one another. Since noisier LFs are more likely to have high conflict rates with others, their corresponding accuracy weights in w will be smaller, reducing their influence on the aggregated labels.

2.5 Discriminative Model

The noisily-labeled training set that the label aggregator outputs is used to train an arbitrary discriminative model. One advantage of training a discriminative model on the task instead of using the label aggregator as a classifier directly is that the label aggregator only takes into account those signals included in the LFs. A discriminative model, on the other hand, can incorporate features that were not identified by the user but are nevertheless informative.[2] Consequently, even examples for which all LFs abstained can still be classified correctly. On the three tasks we evaluate, using the discriminative model averages 4.3 F1 points higher than using the label aggregator directly.

For the results reported in this paper, our discriminative model is a simple logistic regression classifier with generic features defined over dependency paths.[3] These features include unigrams, bigrams, and trigrams of lemmas, dependency labels, and part of speech tags found in the siblings, parents, and nodes between the entities in the dependency parse of the sentence. We found this to perform better on average than a biLSTM, particularly for the traditional supervision baselines with small training set sizes; it also provided easily interpretable features for analysis.

[2]We give an example of two such features in Section 4.3.

[3]https://github.com/HazyResearch/treedlib

Task	Train	Dev	Test	% Pos.
Spouse	22195	2796	2697	8%
Disease	6667	773	4101	20%
Protein	5546	1011	1058	22%

Table 2: The total number of unlabeled training examples (a pair of annotated entities in a sentence), labeled development examples (for hyperparameter tuning), labeled test examples (for assessment), and the fraction of positive labels in the test split.

3 Experimental Setup

We evaluate the accuracy of BabbleLabble on three relation extraction tasks, which we refer to as Spouse, Disease, and Protein. The goal of each task is to train a classifier for predicting whether the two entities in an example are participating in the relationship of interest, as described below.

3.1 Datasets

Statistics for each dataset are reported in Table 2, with one example and one explanation for each given in Figure 4 and additional explanations shown in Appendix B.

In the Spouse task, annotators were shown a sentence with two highlighted names and asked to label whether the sentence suggests that the two people are spouses. Sentences were pulled from the Signal Media dataset of news articles (Corney

	BL				TS			
# Inputs	30	30	60	150	300	1,000	3,000	10,000
`Spouse`	50.1	15.5	15.9	16.4	17.2	22.8	41.8	55.0
`Disease`	42.3	32.1	32.6	34.4	37.5	41.9	44.5	-
`Protein`	47.3	39.3	42.1	46.8	51.0	57.6	-	-
Average	46.6	28.9	30.2	32.5	35.2	40.8	43.2	55.0

Table 3: F1 scores obtained by a classifier trained with `BabbleLabble` (BL) using 30 explanations or with traditional supervision (TS) using the specified number of individually labeled examples. `BabbleLabble` achieves the same F1 score as traditional supervision while using fewer user inputs by a factor of over 5 (`Protein`) to over 100 (`Spouse`).

et al., 2016). Ground truth data was collected from Amazon Mechanical Turk workers, accepting the majority label over three annotations. The 30 explanations we report on were sampled randomly from a pool of 200 that were generated by 10 graduate students unfamiliar with `BabbleLabble`.

In the `Disease` task, annotators were shown a sentence with highlighted names of a chemical and a disease and asked to label whether the sentence suggests that the chemical causes the disease. Sentences and ground truth labels came from a portion of the 2015 BioCreative chemical-disease relation dataset (Wei et al., 2015), which contains abstracts from PubMed. Because this task requires specialized domain expertise, we obtained explanations by having someone unfamiliar with `BabbleLabble` translate from Python to natural language labeling functions from an existing publication that explored applying weak supervision to this task (Ratner et al., 2018).

The `Protein` task was completed in conjunction with OccamzRazor, a neuroscience company targeting biological pathways of Parkinson's disease. For this task, annotators were shown a sentence from the relevant biomedical literature with highlighted names of a protein and a kinase and asked to label whether or not the kinase influences the protein in terms of a physical interaction or phosphorylation. The annotators had domain expertise but minimal programming experience, making `BabbleLabble` a natural fit for their use case.

3.2 Experimental Settings

Text documents are tokenized with spaCy.[4] The semantic parser is built on top of the Python-based implementation SippyCup.[5] On a single core, parsing 360 explanations takes approximately two seconds. We use existing implementations of the label aggregator, feature library, and discriminative classifier described in Sections 2.4–2.5 provided by the open-source project Snorkel (Ratner et al., 2018).

Hyperparameters for all methods we report were selected via random search over thirty configurations on the same held-out development set. We searched over learning rate, batch size, L_2 regularization, and the subsampling rate (for improving balance between classes).[6] All reported F1 scores are the average value of 40 runs with random seeds and otherwise identical settings.

4 Experimental Results

We evaluate the performance of `BabbleLabble` with respect to its rate of improvement by number of user inputs, its dependence on correctly parsed logical forms, and the mechanism by which it utilizes logical forms.

4.1 High Bandwidth Supervision

In Table 3 we report the average F1 score of a classifier trained with `BabbleLabble` using 30 explanations or traditional supervision with the indicated number of labels. On average, it took the same amount of time to collect 30 explanations as 60 labels.[7] We observe that in all three tasks, `BabbleLabble` achieves a given F1 score with far fewer user inputs than traditional supervision, by

[4]https://github.com/explosion/spaCy

[5]https://github.com/wcmac/sippycup

[6]Hyperparameter ranges: learning rate (1e-2 to 1e-4), batch size (32 to 128), L_2 regularization (0 to 100), subsampling rate (0 to 0.5)

[7]Zaidan and Eisner (2008) also found that collecting annotator rationales in the form of highlighted substrings from the sentence only doubled annotation time.

	Pre-filters		Discarded		Post-filters	
	LFs	Correct	Sem.	Prag.	LFs	Correct
Spouse	153	10%	19	115	19	84%
Disease	104	23%	41	36	27	89%
Protein	122	14%	44	58	20	85%

Table 4: The number of LFs generated from 30 explanations (pre-filters), discarded by the filter bank, and remaining (post-filters), along with the percentage of LFs that were correctly parsed from their corresponding explanations.

	BL-FB	BL	BL+PP
Spouse	15.7	50.1	49.8
Disease	39.8	42.3	43.2
Protein	38.2	47.3	47.4
Average	31.2	46.6	46.8

Table 5: F1 scores obtained using BabbleLabble with no filter bank (BL-FB), as normal (BL), and with a perfect parser (BL+PP) simulated by hand.

as much as 100 times in the case of the Spouse task. Because explanations are applied to many unlabeled examples, each individual input from the user can implicitly contribute many (noisy) labels to the learning algorithm.

We also observe, however, that once the number of labeled examples is sufficiently large, traditional supervision once again dominates, since ground truth labels are preferable to noisy ones generated by labeling functions. However, in domains where there is much more unlabeled data available than labeled data (which in our experience is most domains), we can gain in supervision efficiency from using BabbleLabble.

Of those explanations that did not produce a correct LF, 4% were caused by the explanation referring to unsupported concepts (e.g., one explanation referred to "the subject of the sentence," which our simple parser doesn't support). Another 2% were caused by human errors (the correct LF for the explanation was inconsistent with the example). The remainder were due to unrecognized paraphrases (e.g., the explanation said "the order of appearance is X, Y" instead of a supported phrasing like "X comes before Y").

4.2 Utility of Incorrect Parses

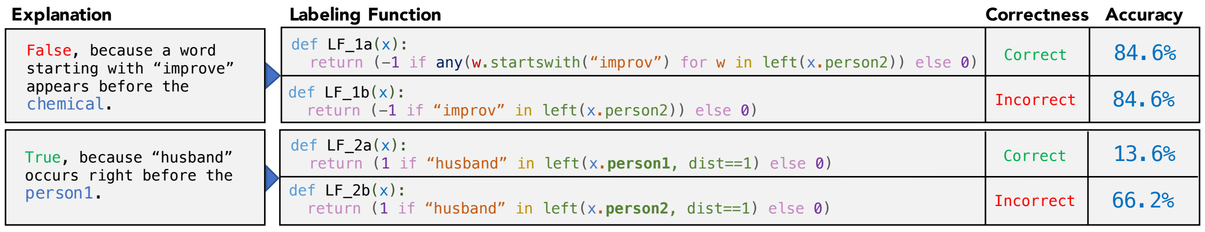
In Table 4, we report LF summary statistics before and after filtering. LF correctness is based on exact match with a manually generated parse for each explanation. Surprisingly, the simple heuristic-based filter bank successfully removes over 95% of incorrect LFs in all three tasks, resulting in final LF sets that are 86% correct on average. Furthermore, among those LFs that pass through the filter bank, we found that the average difference in end-task accuracy between correct and incorrect parses is less than 2.5%. Intuitively, the filters are effective because it is quite difficult for an LF to be parsed from the explanation, label its own example correctly (passing the semantic filter), and not label all examples in the training set with the same label or identically to another LF (passing the pragmatic filter).

We went one step further: using the LFs that would be produced by a perfect semantic parser as starting points, we searched for "nearby" LFs (LFs differing by only one predicate) with higher end-task accuracy on the test set and succeeded 57% of the time (see Figure 5 for an example). In other words, when users provide explanations, the signals they describe provide good starting points, but they are actually unlikely to be optimal. This observation is further supported by Table 5, which shows that the filter bank is necessary to remove clearly irrelevant LFs, but with that in place, the simple rule-based semantic parser and a perfect parser have nearly identical average F1 scores.

4.3 Using LFs as Functions or Features

Once we have relevant logical forms from user-provided explanations, we have multiple options for how to use them. Srivastava et al. (2017) propose using these logical forms as features in a linear classifier. We choose instead to use them as functions for weakly supervising the creation of a larger training set via data programming (Ratner et al., 2016). In Table 6, we compare the two approaches directly, finding that the the data programming approach outperforms a feature-based one by 9.5 F1 points with the rule-based parser, and by 4.5 points with a perfect parser.

We attribute this difference primarily to the ability of data programming to utilize unlabeled data. In Figure 6, we show how the data programming approach improves with the number of unlabeled examples, even as the number of LFs remains constant. We also observe qualitatively that data programming exposes the classifier to additional patterns that are correlated with our explanations but not mentioned directly. For example, in the Disease task, two of the features weighted most



Explanation	Labeling Function	Correctness	Accuracy
False, because a word starting with "improve" appears before the chemical.	`def LF_1a(x): return (-1 if any(w.startswith("improv") for w in left(x.person2)) else 0)`	Correct	84.6%
	`def LF_1b(x): return (-1 if "improv" in left(x.person2)) else 0)`	Incorrect	84.6%
True, because "husband" occurs right before the person1.	`def LF_2a(x): return (1 if "husband" in left(x.person1, dist==1) else 0)`	Correct	13.6%
	`def LF_2b(x): return (1 if "husband" in left(x.person2, dist==1) else 0)`	Incorrect	66.2%

Figure 5: Incorrect LFs often still provide useful signal. On top is an incorrect LF produced for the `Disease` task that had the same accuracy as the correct LF. On bottom is a correct LF from the `Spouse` task and a more accurate incorrect LF discovered by randomly perturbing one predicate at a time as described in Section 4.2. (Person 2 is always the second person in the sentence).

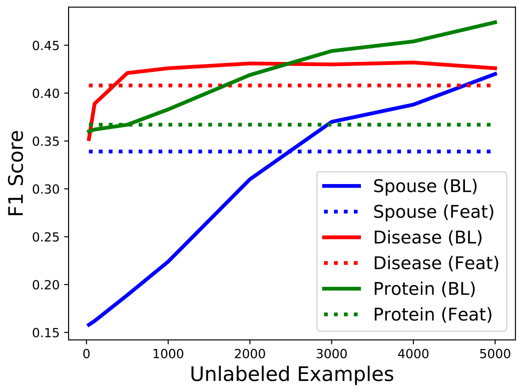


Figure 6: When logical forms of natural language explanations are used as functions for data programming (as they are in `BabbleLabble`), performance can improve with the addition of *unlabeled* data, whereas using them as features does not benefit from unlabeled data.

	BL-DM	BL	BL+PP	Feat	Feat+PP
`Spouse`	46.5	50.1	49.8	33.9	39.2
`Disease`	39.7	42.3	43.2	40.8	43.8
`Protein`	40.6	47.3	47.4	36.7	44.0
Average	42.3	46.6	46.8	37.1	42.3

Table 6: F1 scores obtained using explanations as functions for data programming (BL) or features (Feat), optionally with no discriminative model (-DM) or using a perfect parser (+PP).

highly by the discriminative model were the presence of the trigrams "could produce a" or "support diagnosis of" between the chemical and disease, despite none of these words occurring in the explanations for that task. In Table 6 we see a 4.3 F1 point improvement (10%) when we use the discriminative model that can take advantage of these features rather than applying the LFs directly to the test set and making predictions based on the output of the label aggregator.

5 Related Work and Discussion

Our work has two themes: modeling natural language explanations/instructions and learning from weak supervision. The closest body of work is on "learning from natural language." As mentioned earlier, Srivastava et al. (2017) convert natural language explanations into classifier features (whereas we convert them into labeling functions). Goldwasser and Roth (2011) convert natural language into concepts (e.g., the rules of a card game). Ling and Fidler (2017) use natural language explanations to assist in supervising an image captioning model. Weston (2016); Li et al. (2016) learn from natural language feedback in a dialogue. Wang et al. (2017) convert natural language definitions to rules in a semantic parser to build up progressively higher-level concepts.

We lean on the formalism of semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang, 2016). One notable trend is to learn semantic parsers from weak supervision (Clarke et al., 2010; Liang et al., 2011), whereas our goal is to obtain weak supervision signal from semantic parsers.

The broader topic of weak supervision has received much attention; we mention some works most related to relation extraction. In distant supervision (Craven et al., 1999; Mintz et al., 2009) and multi-instance learning (Riedel et al., 2010; Hoffmann et al., 2011), an existing knowledge base is used to (probabilistically) impute a training set. Various extensions have focused on aggregating a variety of supervision sources by learning generative models from noisy labels (Alfonseca et al., 2012; Takamatsu et al., 2012; Roth and Klakow, 2013; Ratner et al., 2016; Varma et al., 2017).

Finally, while we have used natural language explanations as *input* to train models, they can also be *output* to interpret models (Krening et al., 2017; Lei et al., 2016). More generally, from a machine learning perspective, labels are the primary asset, but they are a low bandwidth signal between annotators and the learning algorithm. Natural language opens up a much higher-bandwidth communication channel. We have shown promising results in relation extraction (where one explanation can be "worth" 100 labels), and it would be interesting to extend our framework to other tasks and more interactive settings.

Reproducibility

The code, data, and experiments for this paper are available on the CodaLab platform at https://worksheets.codalab.org/worksheets/0x900e7e41deaa4ec5b2fe41dc50594548/.

Acknowledgments

We gratefully acknowledge the support of the following organizations: DARPA under No. N66001-15-C-4043 (SIMPLEX), No. FA8750-17-2-0095 (D3M), No. FA8750-12-2-0335 (XDATA), and No. FA8750-13-2-0039 (DEFT), DOE under No. 108845, NIH under No. U54EB020405 (Mobilize), ONR under No. N000141712266 and No. N000141310129, AFOSR under No. 580K753, the Intel/NSF CPS Security grant No. 1505728, the Michael J. Fox Foundation for Parkinsons Research under Grant No. 14672, the Secure Internet of Things Project, Qualcomm, Ericsson, Analog Devices, the Moore Foundation, the Okawa Research Grant, American Family Insurance, Accenture, Toshiba, the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747, the Stanford Finch Family Fellowship, the Joseph W. and Hon Mai Goodman Stanford Graduate Fellowship, an NSF CAREER Award IIS-1552635, and the members of the Stanford DAWN project: Facebook, Google, Intel, Microsoft, NEC, Teradata, and VMware.

We thank Alex Ratner and the developers of Snorkel for their assistance with data programming, as well as the many members of the Hazy Research group and Stanford NLP group who provided feedback and tested early prototypes. Thanks as well to the OccamzRazor team: Tarik Koc, Benjamin Angulo, Katharina S. Volz, and Charlotte Brzozowski.

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, DOE, NIH, ONR, AFOSR, NSF, or the U.S. Government.

References

L. V. Ahn, R. Liu, and M. Blum. 2006. Peekaboom: a game for locating objects in images. In *Conference on Human Factors in Computing Systems (CHI)*. pages 55–64.

E. Alfonseca, K. Filippova, J. Delort, and G. Garrido. 2012. Pattern learning for relation extraction with a hierarchical topic model. In *Association for Computational Linguistics (ACL)*. pages 54–59.

S. Arora and E. Nyberg. 2009. Interactive annotation learning with indirect feature voting. In *Association for Computational Linguistics (ACL)*. pages 55–60.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*. pages 18–27.

D. Corney, D. Albakour, M. Martinez-Alvarez, and S. Moussa. 2016. What do a million news articles look like? In *NewsIR@ ECIR*. pages 42–47.

M. Craven, J. Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*. pages 77–86.

G. Druck, B. Settles, and A. McCallum. 2009. Active learning by labeling features. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 81–90.

D. Goldwasser and D. Roth. 2011. Learning from natural instructions. In *International Joint Conference on Artificial Intelligence (IJCAI)*. pages 1794–1800.

R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Association for Computational Linguistics (ACL)*. pages 541–550.

S. Krening, B. Harrison, K. M. Feigh, C. L. Isbell, M. Riedl, and A. Thomaz. 2017. Learning from explanations using sentiment and advice in RL. *IEEE Transactions on Cognitive and Developmental Systems* 9(1):44–55.

T. Lei, R. Barzilay, and T. Jaakkola. 2016. Rationalizing neural predictions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Li, A. H. Miller, S. Chopra, M. Ranzato, and J. Weston. 2016. Learning through dialogue interactions. *arXiv preprint arXiv:1612.04936* .

P. Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM* 59.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning from measurements in exponential families. In *International Conference on Machine Learning (ICML)*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*. pages 590–599.

H. Ling and S. Fidler. 2017. Teaching machines to describe images via natural language feedback. In *Advances in Neural Information Processing Systems (NIPS)*.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics (ACL)*. pages 1003–1011.

H. Raghavan, O. Madani, and R. Jones. 2005. Interactive feature selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*. volume 5, pages 841–846.

A. J. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. R'e. 2018. Snorkel: Rapid training data creation with weak supervision. In *Very Large Data Bases (VLDB)*.

A. J. Ratner, C. M. D. Sa, S. Wu, D. Selsam, and C. R'e. 2016. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3567–3575.

S. Riedel, L. Yao, and A. McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. pages 148–163.

B. Roth and D. Klakow. 2013. Combining generative and discriminative model scores for distant supervision. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 24–29.

S. Srivastava, I. Labutov, and T. Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1528–1537.

S. Takamatsu, I. Sato, and H. Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Association for Computational Linguistics (ACL)*. pages 721–729.

P. Varma, B. He, D. Iter, P. Xu, R. Yu, C. D. Sa, and C. R'e. 2017. Socratic learning: Augmenting generative models to incorporate latent subsets in training data. *arXiv preprint arXiv:1610.08123* .

S. I. Wang, S. Ginn, P. Liang, and C. D. Manning. 2017. Naturalizing a programming language via interactive learning. In *Association for Computational Linguistics (ACL)*.

C. Wei, Y. Peng, R. Leaman, A. P. Davis, C. J. Mattingly, J. Li, T. C. Wiegers, and Z. Lu. 2015. Overview of the biocreative V chemical disease relation (cdr) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*. pages 154–166.

J. E. Weston. 2016. Dialog-based language learning. In *Advances in Neural Information Processing Systems (NIPS)*. pages 829–837.

O. F. Zaidan and J. Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*. pages 658–666.

A Predicate Examples

Below are the predicates in the rule-based semantic parser grammar, each of which may have many supported paraphrases, only one of which is listed here in a minimal example.

Logic
```
and:  X is true and Y is true
or:  X is true or Y is true
not:  X is not true
any:  Any of X or Y or Z is true
all:  All of X and Y and Z are true
none:  None of X or Y or Z is true
```

Comparison
```
=:  X is equal to Y
≠:  X is not Y
<:  X is smaller than Y
≤:  X is no more than Y
>:  X is larger than Y
≥:  X is at least Y
```

Syntax
```
lower:  X is lowercase
upper:  X is upper case
capital:  X is capitalized
all_caps:  X is in all caps
starts_with:  X starts with "cardio"
ends_with:  X ends with "itis"
substring:  X contains "-induced"
```

Named-entity Tags
```
person:  A person is between X and Y
location:  A place is within two words of X
date:  A date is between X and Y
number:  There are three numbers in the sentence
organization:  An organization is right after X
```

Lists
```
list:  (X, Y) is in Z
set:  X, Y, and Z are true
count:  There is one word between X and Y
contains:  X is in Y
intersection:  At least two of X are in Y
map:  X is at the start of a word in Y
filter:  There are three capitalized words to the left of X
alias:  A spouse word is in the sentence (“spouse” is a predefined list from the user)
```

Position
```
word_distance:  X is two words before Y
char_distance:  X is twenty characters after Y
left:  X is before Y
right:  X is after Y
between:  X is between Y and Z
within:  X is within five words of Y
```

B Sample Explanations

The following are a sample of the explanations provided by users for each task.

`Spouse`
Users referred to the first person in the sentence as "X" and the second as "Y".

```
Label true because "and" occurs between X and Y and "marriage" occurs
one word after person1.

Label true because person Y is preceded by ‘beau’.

Label false because the words "married", "spouse", "husband", and
"wife" do not occur in the sentence.

Label false because there are more than 2 people in the sentence and
"actor" or "actress" is left of person1 or person2.
```

`Disease`

```
Label true because the disease is immediately after the chemical and
'induc' or 'assoc' is in the chemical name.

Label true because a word containing 'develop' appears somewhere
before the chemical, and the word 'following' is between the disease
and the chemical.

Label true because "induced by", "caused by", or "due to" appears
between the chemical and the disease."

Label false because "none", "not", or "no" is within 30 characters to
the left of the disease.
```

`Protein`

```
Label true because "Ser" or "Tyr" are within 10 characters of the
protein.

Label true because the words "by" or "with" are between the protein
and kinase and the words "no", "not" or "none" are not in between
the protein and kinase and the total number of words between them is
smaller than 10.

Label false because the sentence contains "mRNA", "DNA", or "RNA".

Label false because there are two "," between the protein and the
kinase with less than 30 characters between them.
```

Did the Model Understand the Question?

Pramod K. Mudrakarta University of Chicago pramodkm@uchicago.edu

Ankur Taly Google Brain

Mukund Sundararajan Google

Kedar Dhamdhere Google

{ataly,mukunds,kedar}@google.com

Abstract

We analyze state-of-the-art deep learning models for three tasks: question answering on (1) images, (2) tables, and (3) passages of text. Using the notion of *attribution* (word importance), we find that these deep networks often ignore important question terms. Leveraging such behavior, we perturb questions to craft a variety of adversarial examples. Our strongest attacks drop the accuracy of a visual question answering model from 61.1% to 19%, and that of a tabular question answering model from 33.5% to 3.3%. Additionally, we show how attributions can strengthen attacks proposed by Jia and Liang (2017) on paragraph comprehension models. Our results demonstrate that attributions can augment standard measures of accuracy and empower investigation of model performance. When a model is accurate but for the wrong reasons, attributions can surface erroneous logic in the model that indicates inadequacies in the test data.

1 Introduction

Recently, deep learning has been applied to a variety of question answering tasks. For instance, to answer questions about images (e.g. (Kazemi and Elqursh, 2017)), tabular data (e.g. (Neelakantan et al., 2017)), and passages of text (e.g. (Yu et al., 2018)). Developers, end-users, and reviewers (in academia) would all like to understand the capabilities of these models.

The standard way of measuring the goodness of a system is to evaluate its error on a test set. High accuracy is indicative of a good model only if the test set is representative of the underlying real-world task. Most tasks have large test and training sets, and it is hard to manually check that they are representative of the real world.

In this paper, we propose techniques to analyze the sensitivity of a deep learning model to question words. We do this by applying *attribution* (as discussed in section 3), and generating adversarial questions. Here is an illustrative example: recall Visual Question Answering (Agrawal et al., 2015) where the task is to answer questions about images. Consider the question "*how symmetrical are the white bricks on either side of the building?*" (corresponding image in Figure 1). The system that we study gets the answer right ("*very*"). But, we find (using an attribution approach) that the system relies on only a few of the words like "how" and "bricks". Indeed, we can construct adversarial questions about the same image that the system gets wrong. For instance, "*how spherical are the white bricks on either side of the building?*" returns the same answer ("*very*"). A key premise of our work is that most humans have expertise in question answering. Even if they cannot manually check that a dataset is representative of the real world, they can identify important question words, and anticipate their function in question answering.

1.1 Our Contributions

We follow an analysis workflow to understand three question answering models. There are two steps. First, we apply Integrated Gradients (henceforth, IG) (Sundararajan et al., 2017) to attribute the systems' predictions to words in the questions. We propose visualizations of attributions to make analysis easy. Second, we identify weaknesses (e.g., relying on unimportant words) in the networks' logic as exposed by the attributions, and leverage them to craft adversarial questions.

A key contribution of this work is an *overstability* test for question answering networks. Jia and Liang (2017) showed that reading comprehension networks are overly stable to semantics-altering edits to the passage. In this work, we find that such overstability also applies to questions. Furthermore, this behavior can be seen in visual and tabular question answering networks as well. We use attributions to a define a general-purpose test for measuring the extent of the overstability (sections 4.3 and 5.3). It involves measuring how a network's accuracy changes as words are systematically dropped from questions.

We emphasize that, in contrast to model-independent adversarial techniques such as that of Jia and Liang (2017), our method exploits the strengths and weaknesses of the model(s) at hand. This allows our attacks to have a high success rate. Additionally, using insights derived from attributions we were able to improve the attack success rate of Jia and Liang (2017) (section 6.2). Such extensive use of attributions in crafting adversarial examples is novel to the best of our knowledge.

Next, we provide an overview of our results. In each case, we evaluate a pre-trained model on new inputs. We keep the networks' parameters intact.

Visual QA (section 4): The task is to answer questions about images. We analyze the deep network in Kazemi and Elqursh (2017). We find that the network ignores many question words, relying largely on the image to produce answers. For instance, we show that the model retains more than 50% of its original accuracy even when every word that is not "color" is deleted from all questions in the validation set. We also show that the model under-relies on important question words (e.g. nouns) and attaching content-free prefixes (e.g., "in not many words, ...") to questions drops the accuracy from 61.1% to 19%.

QA on tables (section 5): We analyze a system called Neural Programmer (henceforth, NP) (Neelakantan et al., 2017) that answers questions on tabular data. NP determines the answer to a question by selecting a sequence of operations to apply on the accompanying table (akin to an SQL query; details in section 5). We find that these operation selections are more influenced by content-free words (e.g., "in", "at", "the", etc.) in questions than important words such as nouns or adjectives. Dropping all content-free words reduces the validation accuracy of the network from 33.5%[1] to 28.5%. Similar to Visual QA, we show that attaching content-free phrases (e.g., "in not a lot of words") to the question drops the network's accuracy from 33.5% to 3.3%. We also find that NP often gets the answer right for the wrong reasons. For instance, for the question "*which nation earned the most gold medals?*", one of the operations selected by NP is "first" (pick the first row of the table). Its answer is right only because the table happens to be arranged in order of rank. We quantify this weakness by evaluating NP on the set of perturbed tables generated by Pasupat and Liang (2016) and find that its accuracy drops from 33.5% to 23%. Finally, we show an extreme form of overstability where the table itself induces a large bias in the network regardless of the question. For instance, we found that in tables about Olympic medal counts, NP was predisposed to selecting the "prev" operator.

Reading comprehension (Section 6): The task is to answer questions about paragraphs of text. We analyze the network by Yu et al. (2018). Again, we find that the network often ignores words that should be important. Jia and Liang (2017) proposed attacks wherein sentences are added to paragraphs that ought not to change the network's answers, but sometimes do. Our main finding is that these attacks are more likely to succeed when an added sentence includes all the question words that the model found important (for the original paragraph). For instance, we find that attacks are 50% more likely to be successful when the added sentence includes top-attributed nouns in the question. This insight should allow the construction of more successful attacks and better training data sets.

In summary, we find that all networks ignore important parts of questions. One can fix this by either improving training data, or introducing an inductive bias. Our analysis workflow is helpful in both cases. It would also make sense to expose end-users to attribution visualizations. Knowing which words were ignored, or which operations the words were mapped to, can help the user decide whether to trust a system's response.

2 Related Work

We are motivated by Jia and Liang (2017). As they discuss, "*the extent to which [reading comprehension systems] truly understand language remains unclear*". The contrast between Jia and Liang

[1]This is the single-model accuracy that we obtained on training the Neural Programmer network. The accuracy reported in the paper is 34.1%.

(2017) and our work is instructive. Their main contribution is to fix the evaluation of reading comprehension systems by augmenting the test set with adversarially constructed examples. (As they point out in Section 4.6 of their paper, this does not necessarily fix the model; the model may simply learn to circumvent the specific attack underlying the adversarial examples.) Their method is independent of the specification of the model at hand. They use crowdsourcing to craft passage perturbations intended to fool the network, and then query the network to test their effectiveness.

In contrast, we propose improving the analysis of question answering systems. Our method peeks into the logic of a network to identify high-attribution question terms. Often there are several important question terms (e.g., nouns, adjectives) that receive tiny attribution. We leverage this weakness and perturb questions to craft targeted attacks. While Jia and Liang (2017) focus exclusively on systems for the reading comprehension task, we analyze one system each for three different tasks. Our method also helps improve the efficacy Jia and Liang (2017)'s attacks; see table 4 for examples. Our analysis technique is specific to deep-learning-based systems, whereas theirs is not.

We could use many other methods instead of Integrated Gradients (IG) to attribute a deep network's prediction to its input features (Baehrens et al., 2010; Simonyan et al., 2013; Shrikumar et al., 2016; Binder et al., 2016; Springenberg et al., 2014). One could also use model agnostic techniques like Ribeiro et al. (2016b). We choose IG for its ease and efficiency of implementation (requires just a few gradient-calls) and its axiomatic justification (see Sundararajan et al. (2017) for a detailed comparison with other attribution methods).

Recently, there have been a number of techniques for crafting and defending against adversarial attacks on image-based deep learning models (cf. Goodfellow et al. (2015)). They are based on oversensitivity of models, i.e., tiny, imperceptible perturbations of the image to change a model's response. In contrast, our attacks are based on models' over-reliance on few question words even when other words should matter.

We discuss task-specific related work in corresponding sections (sections 4 to 6).

3 Integrated Gradients (IG)

We employ an attribution technique called Integrated Gradients (IG) (Sundararajan et al., 2017) to isolate question words that a deep learning system uses to produce an answer.

Formally, suppose a function $F : \mathbb{R}^n \rightarrow [0, 1]$ represents a deep network, and an input $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. An attribution of the prediction at input x relative to a baseline input x' is a vector $A_F(x, x') = (a_1, \ldots, a_n) \in \mathbb{R}^n$ where a_i is the *contribution* of x_i to the prediction $F(x)$. One can think of F as the probability of a specific response. $x_1, \ldots, x_n$ are the question words; to be precise, they are going to be vector representations of these terms. The attributions $a_1, \ldots, a_n$ are the influences/blame-assignments to the variables $x_1, \ldots, x_n$ on the probability F.

Notice that attributions are defined relative to a special, uninformative input called the *baseline*. In this paper, we use an empty question as the baseline, that is, a sequence of word embeddings corresponding to padding value. Note that the context (image, table, or passage) of the baseline x' is set to be that of x; only the question is set to empty. We now describe how IG produces attributions.

Intuitively, as we interpolate between the baseline and the input, the prediction moves along a trajectory, from uncertainty to certainty (the final probability). At each point on this trajectory, one can use the gradient of the function F with respect to the input to attribute the change in probability back to the input variables. IG simply aggregates the gradients of the probability with respect to the input along this trajectory using a path integral.

Definition 1 (Integrated Gradients) *Given an input x and baseline x', the integrated gradient along the i^{th} dimension is defined as follows.*

$$\mathsf{IG}_i(x, x') ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \, d\alpha$$

(here $\frac{\partial F(x)}{\partial x_i}$ is the gradient of F along the i^{th} dimension at x).

Sundararajan et al. (2017) discuss several properties of IG. Here, we informally mention a few desirable ones, deferring the reader to Sundararajan et al. (2017) for formal definitions.

IG satisfies the condition that the attributions sum to the difference between the probabilities at the input and the baseline. We call a variable uninfluential if all else fixed, varying it does not change the output probability. IG satisfies the property that uninfluential variables do not get any attribution. Conversely, influential variables always get some attribution. Attributions for a linear combination of two functions F_1 and F_2 are a linear combination of the attributions for F_1 and F_2. Finally, IG satisfies the condition that symmetric variables get equal attributions.

In this work, we validate the use of IG empirically via question perturbations. We observe that perturbing high-attribution terms changes the networks' response (sections 4.4 and 5.5). Conversely, perturbing terms that receive a low attribution does not change the network's response (sections 4.3 and 5.3). We use these observations to craft attacks against the network by perturbing instances where generic words (e.g., "a", "the") receive high attribution or contentful words receive low attribution.

4 Visual Question Answering

4.1 Task, model, and data

The Visual Question Answering Task (Agrawal et al., 2015; Teney et al., 2017; Kazemi and Elqursh, 2017; Ben-younes et al., 2017; Zhu et al., 2016) requires a system to answer questions about images (fig. 1). We analyze the deep network from Kazemi and Elqursh (2017). It achieves 61.1% accuracy on the validation set (the state of the art (Fukui et al., 2016) achieves 66.7%). We chose this model for its easy reproducibility.

The VQA 1.0 dataset (Agrawal et al., 2015) consists of 614,163 questions posed over 204,721 images (3 questions per image). The images were taken from COCO (Lin et al., 2014), and the questions and answers were crowdsourced.

The network in Kazemi and Elqursh (2017) treats question answering as a classification task wherein the classes are 3000 most frequent answers in the training data. The input question is tokenized, embedded and fed to a multi-layer LSTM. The states of the LSTM attend to a featurized version of the image, and ultimately produce a probability distribution over the answer classes.

4.2 Observations

We applied IG and attributed the top selected answer class to input question words. The baseline for a given input instance is the image and an empty question[2]. We omit instances where the top answer class predicted by the network remains the same even when the question is emptied (i.e., the baseline input). This is because IG attributions are not informative when the input and the baseline have the same prediction.

Question: how symmetrical are the white bricks on either side of the building
Prediction: very
Ground truth: very

Figure 1: Visual QA (Kazemi and Elqursh, 2017): Visualization of attributions (word importances) for a question that the network gets right. Red indicates high attribution, blue negative attribution, and gray near-zero attribution. The colors are determined by attributions normalized w.r.t the maximum magnitude of attributions among the question's words.

A visualization of the attributions is shown in fig. 1. Notice that very few words have high attribution. We verified that altering the low attribution words in the question does not change the network's answer. For instance, the following questions still return "*very*" as the answer: "*how spherical are the white bricks on either side of the building*", "*how soon are the bricks fading on either side of the building*", "*how fast are the bricks speaking on either side of the building*".

On analyzing attributions across examples, we find that most of the highly attributed words are words such as "*there*", "*what*", "*how*", "*doing*"– they are usually the less important words in questions. In section 4.3 we describe a test to measure the extent to which the network depends on such words. We also find that informative words in the question (e.g., nouns) often receive very low attribution, indicating a weakness on part of the network. In Section 4.4, we describe various attacks that exploit this weakness.

4.3 Overstability test

To determine the set of question words that the network finds most important, we isolate words that most frequently occur as top attributed words in questions. We then drop all words except these and compute the accuracy.

Figure 2 shows how the accuracy changes as the size of this isolated set is varied from 0 to 5305.

[2]We do not black out the image in our baseline as our objective is to study the influence of just the question words for a given image

We find that just one word is enough for the model to achieve more than 50% of its final accuracy. That word is “color”.

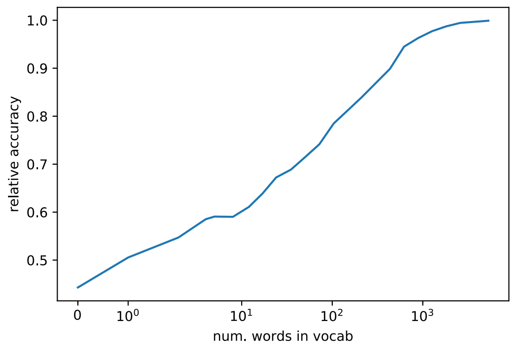


Figure 2: VQA network (Kazemi and Elqursh, 2017): Accuracy as a function of vocabulary size, relative to its original accuracy. Words are chosen in the descending order of how frequently they appear as top attributions. The X-axis is on logscale, except near zero where it is linear.

Note that even when empty questions are passed as input to the network, its accuracy remains at about 44.3% of its original accuracy. This shows that the model is largely reliant on the image for producing the answer.

The accuracy increases (almost) monotonically with the size of the isolated set. The top 6 words in the isolated set are “color”, “many”, “what”, “is”, “there”, and “how”. We suspect that generic words like these are used to determine the type of the answer. The network then uses the type to choose between a few answers it can give for the image.

4.4 Attacks

Attributions reveal that the network relies largely on generic words in answering questions (section 4.3). This is a weakness in the network’s logic. We now describe a few attacks against the network that exploit this weakness.

Subject ablation attack

In this attack, we replace the subject of a question with a specific noun that consistently receives low attribution across questions. We then determine, among the questions that the network originally answered correctly, what percentage result in the same answer after the ablation. We repeat this process for different nouns; specifically, “fits”, “childhood”, “copyrights”, “mornings”, “disorder”, “importance”, “topless”, “critter”, “jumper”, “tweet”, and average the result.

Prefix	Accuracy
in not a lot of words	35.5%
in not many words	32.5%
what is the answer to	31.7%
Union of all three	**19%**
Baseline prefix	
tell me	51.3%
answer this	55.7%
answer this for me	49.8%
Union of baseline prefixes	**46.9%**

Table 1: VQA network (Kazemi and Elqursh, 2017): Accuracy for prefix attacks; original accuracy is 61.1%.

We find that, among the set of questions that the network originally answered correctly, 75.6% of the questions return the same answer despite the subject replacement.

Prefix attack

In this attack, we attach content-free phrases to questions. The phrases are manually crafted using generic words that the network finds important (section 4.3). Table 1 (top half) shows the resulting accuracy for three prefixes —“*in not a lot of words*”, “*what is the answer to*”, and “*in not many words*”. All of these phrases nearly halve the model’s accuracy. The union of the three attacks drops the model’s accuracy from 61.1% to 19%.

We note that the attributions computed for the network were crucial in crafting the prefixes. For instance, we find that other prefixes like “*tell me*”, “*answer this*” and “*answer this for me*” do not drop the accuracy by much; see table 1 (bottom half). The union of these three ineffective prefixes drops the accuracy from 61.1% to only 46.9%. Per attributions, words present in these prefixes are not deemed important by the network.

4.5 Related work

Agrawal et al. (2016) analyze several VQA models. Among other attacks, they test the models on question fragments of telescopically increasing length. They observe that VQA models often arrive at the same answer by looking at a small fragment of the question. Our stability analysis in section 4.3 explains, and intuitively subsumes this; indeed, several of the top attributed words appear in the prefix, while important words like “color” often occur in the middle of the question. Our analysis enables additional attacks, for instance, replacing question subject with low attribution nouns. Ribeiro et al. (2016a) use a model explanation technique to illustrate overstability for two examples. They do not quantify their analysis at scale. Kafle and Kanan (2017); Zhang et al. (2016) examine the VQA data, identify deficiencies, and propose data augmentation to reduce over-representation of certain question/answer types. Goyal et al. (2016) propose the VQA 2.0 dataset, which has pairs of similar images that have different answers on the same question. We note that our method can be used to improve these datasets by identifying inputs where models ignore several words. Huang et al. (2017) evaluate robustness of VQA models by appending *questions* with semantically similar questions. Our prefix attacks in section 4.4 are in a similar vein and perhaps a more natural and targeted approach. Finally, Fong and Vedaldi (2017) use saliency methods to produce image perturbations as adversarial examples; our attacks are on the question.

5 Question Answering over Tables

5.1 Task, model, and data

We now analyze question answering over tables based on the WikiTableQuestions benchmark dataset (Pasupat and Liang, 2015). The dataset has 22033 questions posed over 2108 tables scraped from Wikipedia. Answers are either contents of table cells or some table aggregations. Models performing QA on tables translate the question into a structured program (akin to an SQL query) which is then executed on the table to produce the answer. We analyze a model called Neural Programmer (NP) (Neelakantan et al., 2017). NP is the state of the art among models that are weakly supervised, i.e., supervised using the final answer instead of the correct structured program. It achieves 33.5% accuracy on the validation set.

NP translates the input into a structured program consisting of four operator and table column selections. An example of such a program is "reset (score), reset (score), min (score), print (name)", where the output is the name of the person who has the lowest score.

5.2 Observations

We applied IG to attribute operator and column selection to question words. NP preprocesses inputs and whenever applicable, appends symbols tm_token, cm_token to questions that signify matches between a question and the accompanying table. These symbols are treated the same as question words. NP also computes priors for column selection using question-table matches. These vectors, tm and cm, are passed as additional inputs to the neural network. In the baseline for IG, we use an empty question, and zero vectors for column selection priors[3].

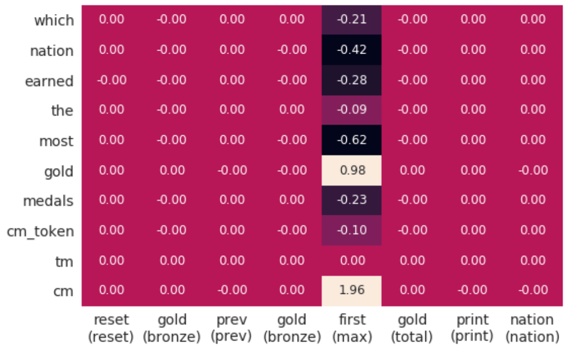


Figure 3: Visualization of attributions. Question words, preprocessing tokens and column selection priors on the Y-axis. Along the X-axis are operator and column selections with their baseline counterparts in parentheses. Operators and columns not affecting the final answer, and those which are same as their baseline counterparts, are given zero attribution.

We visualize the attributions using an alignment matrix; they are commonly used in the analysis of translation models (fig. 3). Observe that the operator "first" is used when the question is asking for a superlative. Further, we see that the word "gold" is a trigger for this operator. We investigate implications of this behavior in the following sections.

5.3 Overstability test

Similar to the test we did for Visual QA (section 4.3), we check for overstability in NP by looking at accuracy as a function of the vocabulary size. We treat table match annotations tm_token, cm_token and the out-of-vocab token (unk) as part of the vocabulary. The results are in fig. 4. We see that the curve is similar to that of Visual QA (fig. 2). Just 5 words (along with the column selection priors) are sufficient for the model to reach more than 50% of its final accuracy on the validation set. These five words are: "many", "number", "tm_token", "after", and "total".

5.4 Table-specific default programs

We saw in the previous section that the model relies on only a few words in producing correct answers. An extreme case of overstability is when

[3]Note that the table is left intact in the baseline

Operator sequence	#	Triggers	Insights
reset, reset, max, print	109	[*unk*, date, position, points, name, competition, notes, no, year, venue]	sports
reset, prev, max, print	68	[*unk*, rank, total, bronze, gold, silver, nation, name, date, no]	medal tallies
reset, reset, first, print	29	[name, *unk*, notes, year, nationality, rank, location, date, comments, hometown]	player rankings
reset, mfe, first, print	25	[notes, date, title, *unk*, role, genre, year, score, opponent, event]	awards
reset, reset, min, print	17	[year, height, *unk*, name, position, floors, notes, jan, jun, may]	building info.
reset, mfe, max, print	14	[opponent, date, result, location, rank, site, attendance, notes, city, listing]	politics
reset, next, first, print	10	[*unk*, name, year, edition, birth, death, men, time, women, type]	census

Table 2: Attributions to column names for table-specific default programs (programs returned by NP on empty input questions). See supplementary material, table 6 for the full list. These results are indication that the network is predisposed towards picking certain operators solely based on the table.

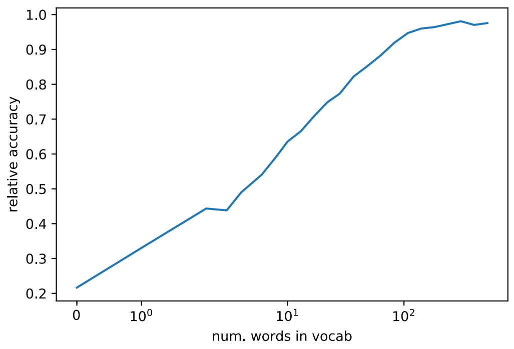


Figure 4: Accuracy as a function of vocabulary size. The words are chosen in the descending order of their frequency appearance as top attributions to question terms. The X-axis is on logscale, except near zero where it is linear. Note that just 5 words are necessary for the network to reach more than 50% of its final accuracy.

Attack phrase	Prefix	Suffix
in not a lot of words	20.6%	10.0%
if its all the same	21.8%	18.7%
in not many words	15.6%	11.2%
one way or another	23.5%	20.0%
Union of above attacks	**3.3%**	
Baseline		
please answer	32.3%	30.7%
do you know	31.2%	29.5%
Union of baseline prefixes	**27.1%**	

Table 3: Neural Programmer (Neelakantan et al., 2017): Left: Validation accuracy when attack phrases are concatenated to the question. (Original: 33.5%)

the operator sequences produced by the model are independent of the question. We find that if we supply an empty question as an input, i.e., the output is a function only of the table, then the distribution over programs is quite skewed. We call these programs table-specific default programs. On average, about 36.9% of the selected operators match their table-default counterparts, indicating that the model relies significantly on the table for producing an answer.

For each default program, we used IG to attribute operator and column selections to column names and show ten most frequently occurring ones across tables in the validation set (table 2).

Here is an insight from this analysis: NP uses the combination "reset, prev" to exclude the last row of the table from answer computation. The default program corresponding to "reset, prev, max, print" has attributions to column names such as *"rank"*, *"gold"*, *"silver"*, *"bronze"*, *"nation"*, *"year"*. These column names indicate medal tallies and usually have a *"total"* row. If the table happens not to have a *"total"* row, the model may produce an incorrect answer.

We now describe attacks that add or drop content-free words from the question, and cause NP to produce the wrong answer. These attacks leverage the attribution analysis.

5.5 Attacks

Question concatenation attacks

In these attacks, we either suffix or prefix content-free phrases to questions. The phrases are crafted using irrelevant trigger words for operator selections (supplementary material, table 5). We manually ensure that the phrases are content-free.

Table 3 describes our results. The first 4 phrases use irrelevant trigger words and result in a large drop in accuracy. For instance, the first phrase uses "not" which is a trigger for "next", "last", and "min", and the second uses "same" which is a trigger for "next" and "mfe". The four phrases combined results in the model's accuracy going down from 33.5% to 3.3%. The first two phrases alone drop the accuracy to 5.6%.

The next set of phrases use words that receive low attribution across questions, and are hence non-triggers for any operator. The resulting drop in accuracy on using these phrases is relatively low. Combined, they result in the model's accuracy dropping from 33.5% to 27.1%.

Stop word deletion attacks

We find that sometimes an operator is selected based on *stop words* like: "a", "at", "the", etc. For instance, in the question, "what ethnicity is at the top?", the operator "next" is triggered on the word "at". Dropping the word "at" from the question changes the operator selection and causes NP to return the wrong answer.

We drop stop words from questions in the validation dataset that were originally answered correctly and test NP on them. The stop words to be dropped were manually selected[4] and are shown in Figure 5 in the supplementary material.

By dropping stop words, the accuracy drops from 33.5% to 28.5%. Selecting operators based on stop words is not robust. In real world search queries, users often phrase questions without stop words, trading grammatical correctness for conciseness. For instance, the user may simply say "top ethnicity". It may be possible to defend against such examples by generating synthetic training data, and re-training the network on it.

Row reordering attacks

We found that NP often got the question right by leveraging artifacts of the table. For instance, the operators for the question "which nation earned the most gold medals" are "reset", "prev", "first" and "print". The "prev" operator essentially excludes the last row from the answer computation. It gets the answer right for two reasons: (1) the answer is not in the last row, and (2) rows are sorted by the values in the column "gold".

In general, a question answering system should not rely on row ordering in tables. To quantify the extent of such biases, we used a perturbed version of WikiTableQuestions validation dataset as described in Pasupat and Liang (2016)[5] and evaluated the existing NP model on it (there was no re-training involved here). We found that NP has only 23% accuracy on it, in constrast to an accuracy of 33.5% on the original validation dataset.

One approach to making the network robust to row-reordering attacks is to train against perturbed tables. This may also help the model generalize better. Indeed, Mudrakarta et al. (2018) note that the state-of-the-art strongly supervised[6] model on WikiTableQuestions (Krishnamurthy et al., 2017) enjoys a 7% gain in its final accuracy by leveraging perturbed tables during training.

[4]We avoided standard stop word lists (e.g. NLTK) as they contain contentful words (e.g "after") which may be important in some questions (e.g. "who ranked right after turkey?")

[5]based on data at https://nlp.stanford.edu/software/sempre/wikitable/dpd/

6 Reading Comprehension

6.1 Task, model, and data

The reading comprehension task involves identifying a span from a context paragraph as an answer to a question. The SQuAD dataset (Rajpurkar et al., 2016) for machine reading comprehension contains 107.7K query-answer pairs, with 87.5K for training, 10.1K for validation, and another 10.1K for testing. Deep learning methods are quite successful on this problem, with the state-of-the-art F1 score at 84.6 achieved by Yu et al. (2018); we analyze their model.

6.2 Analyzing adversarial examples

Recall the adversarial attacks proposed by Jia and Liang (2017) for reading comprehension systems. Their attack ADDSENT appends sentences to the paragraph that resemble an answer to the question without changing the ground truth. See the second column of table 4 for a few examples.

We investigate the effectiveness of their attacks using attributions. We analyze 100 examples generated by the ADDSENT method in Jia and Liang (2017), and find that an adversarial sentence is successful in fooling the model in two cases:

First, a contentful word in the question gets low/zero attribution and the adversarially added sentence modifies that word. E.g. in the question, "Who did Kubiak take the place of after Super Bowl XXIV?", the word "Super" gets low attribution. Adding "After *Champ* Bowl XXV, Crowton took the place of Jeff Dean" changes the prediction for the model. Second, a contentful word in the question that is not present in the context. For e.g. in the question "Where hotel did the Panthers stay at?", "hotel", is not present in the context. Adding "The Vikings stayed at Chicago hotel." changes the prediction for the model.

On the flip side, an adversarial sentence is unsuccessful when a contentful word in the question having high attribution is not present in the added sentence. E.g. for "Where according to gross state product does Victoria rank in Australia?", "Australia" receives high attribution. Adding "Accord-

[6]supervised on the structured program

Question	AddSent attack that does not work	Attack that works
Who was Count of Melfi	Jeff Dean was the mayor of Bracco.	Jeff Dean was the mayor of Melfi.
What country was Abhisit Vejjajiva prime minister of , despite having been born in Newcastle ?	Samak Samak was prime minister of the country of Chicago, despite having been born in Leeds.	Abhisit Vejjajiva was chief minister of the country of Chicago, despite having been born in Leeds.
Where according to gross state product does Victoria rank in Australia ?	According to net state product, Adelaide ranks 7 in New Zealand	According to net state product, Adelaide ranked 7 in Australia. (as a prefix)
When did the Methodist Protestant Church split from the Methodist Episcopal Church ?	The Presbyterian Catholics split from the Presbyterian Anglican in 1805.	The Methodist Protestant Church split from the Presbyterian Anglican in 1805. (as a prefix)
What period was 2.5 million years ago ?	The period of Plasticean era was 2.5 billion years ago.	The period of Plasticean era was 1.5 billion years ago. (as a prefix)

Table 4: AddSent attacks that failed to fool the model. With modifications to preserve nouns with high attributions, these are successful in fooling the model. Question words that receive high attribution are colored red (intensity indicates magnitude).

ing to net state product, Adelaide ranks 7 in New Zealand." does not fool the model. However, retaining "Australia" in the adversarial sentence does change the model's prediction.

6.3 Predicting the effectiveness of attacks

Next we correlate attributions with efficacy of the AddSent attacks. We analyzed 1000 (question, attack phrase) instances[7] where Yu et al. (2018) model has the correct baseline prediction. Of the 1000 cases, 508 are able to fool the model, while 492 are not. We split the examples into two groups. The first group has examples where a noun or adjective in the question has high attribution, but is missing from the adversarial sentence and the rest are in the second group. Our attribution analysis suggests that we should find more failed examples in the first group. That is indeed the case. The first group has 63% failed examples, while the second has only 40%.

Recall that the attack sentences were constructed by (a) generating a sentence that answers the question, (b) replacing all the adjectives and nouns with antonyms, and named entities by the nearest word in GloVe word vector space (Pennington et al., 2014) and (c) crowdsourcing to check that the new sentence is grammatically correct. This suggests a use of attributions to improve the effectiveness of the attacks, namely ensuring that question words that the model thinks are important are left untouched in step (b) (we note that other changes in should be carried out). In table 4, we show a few examples where an original attack did not fool the model, but preserving a noun with high attribution did.

[7]data sourced from https://worksheets.codalab.org/worksheets/0xc86d3ebe69a3427d91f9aaa63f7d1e7d/

7 Conclusion

We analyzed three question answering models using an attribution technique. Attributions helped us identify weaknesses of these models more effectively than conventional methods (based on validation sets). We believe that a workflow that uses attributions can aid the developer in iterating on model quality more effectively.

While the attacks in this paper may seem unrealistic, they do expose real weaknesses that affect the usage of a QA product. Under-reliance on important question terms is not safe. We also believe that other QA models may share these weaknesses. Our attribution-based methods can be directly used to gauge the extent of such problems. Additionally, our perturbation attacks (sections 4.4 and 5.5) serve as empirical validation of attributions.

Reproducibility

Code to generate attributions and reproduce our results is freely available at https://github.com/pramodkaushik/acl18_results.

Acknowledgments

We thank the anonymous reviewers and Kevin Gimpel for feedback on our work, and David Dohan for helping with the reading comprehension network. We are grateful to Jiří Šimša for helpful comments on drafts of this paper.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*.

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C Lawrence Zitnick, Dhruv Batra, and Devi Parikh. 2015. Vqa: Visual question answering. *arXiv preprint arXiv:1505.00468*.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*, pages 1803–1831.

Hedi Ben-younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. 2017. Mutan: Multimodal tucker fusion for visual question answering. *arXiv preprint arXiv:1705.06676*.

Alexander Binder, Grégoire Montavon, Sebastian Bach, Klaus-Robert Müller, and Wojciech Samek. 2016. Layer-wise relevance propagation for neural networks with local renormalization layers. *CoRR*.

Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3429–3437.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *arXiv preprint arXiv:1612.00837*.

Jia-Hong Huang, Cuong Duc Dao, Modar Alfadly, and Bernard Ghanem. 2017. A novel framework for robustness analysis of visual qa models. *arXiv preprint arXiv:1711.06232*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*.

Kushal Kafle and Christopher Kanan. 2017. An analysis of visual question answering algorithms. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1983–1991. IEEE.

Vahid Kazemi and Ali Elqursh. 2017. Show, ask, attend, and answer: A strong baseline for visual question answering. *arXiv preprint arXiv:1704.03162*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. It was the training data pruning too! *arXiv preprint arXiv:1803.04579*.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *International Conference on Learning Representations ICLR*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. *arXiv preprint arXiv:1606.06900*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. Nothing else matters: model-agnostic explanations by identifying prediction invariance. *arXiv preprint arXiv:1611.05817*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. *CoRR*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for simplicity: The all convolutional net. *CoRR*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3319–3328.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2017. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *arXiv preprint arXiv:1708.02711*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and yang: Balancing and answering binary visual questions. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 5014–5022. IEEE.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004.

Harvesting Paragraph-Level Question-Answer Pairs from Wikipedia

Xinya Du and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
{xdu, cardie}@cs.cornell.edu

Abstract

We study the task of generating from Wikipedia articles question-answer pairs that cover content beyond a single sentence. We propose a neural network approach that incorporates coreference knowledge via a novel gating mechanism. Compared to models that only take into account sentence-level information (Heilman and Smith, 2010; Du et al., 2017; Zhou et al., 2017), we find that the linguistic knowledge introduced by the coreference representation aids question generation significantly, producing models that outperform the current state-of-the-art. We apply our system (composed of an answer span extraction system and the passage-level QG system) to the 10,000 top-ranking Wikipedia articles and create a corpus of over one million question-answer pairs. We also provide a qualitative analysis for this large-scale generated corpus from Wikipedia.

Paragraph:
[(1)] *Tesla* was renowned for *his* achievements and showmanship, eventually earning *him* a reputation in popular culture as an archetypal "mad scientist". [(2)] *His* patents earned *him* a considerable amount of money, much of which was used to finance *his* own projects with varying degrees of success. [(3)] *He* lived most of his life in a series of New York hotels, through *his* retirement. [(4)] *Tesla* died on 7 January 1943. ...

Questions:
– What was Tesla's reputation in popular culture?
mad scientist

– How did Tesla finance his work?
patents

– Where did Tesla live for much of his life?
New York hotels

Figure 1: Example input from the fourth paragraph of a Wikipedia article on *Nikola Tesla*, along with the natural questions and their answers from the SQuAD (Rajpurkar et al., 2016) dataset. We show in italics the set of mentions that refer to Nikola Tesla — *Tesla*, *him*, *his*, *he*, etc.

1 Introduction

Recently, there has been a resurgence of work in NLP on reading comprehension (Hermann et al., 2015; Rajpurkar et al., 2016; Joshi et al., 2017) with the goal of developing systems that can answer questions about the content of a given passage or document. Large-scale QA datasets are indispensable for training expressive statistical models for this task and play a critical role in advancing the field. And there have been a number of efforts in this direction. Miller et al. (2016), for example, develop a dataset for open-domain question answering; Rajpurkar et al. (2016) and Joshi et al. (2017) do so for reading comprehension (RC); and Hill et al. (2015) and Hermann et al. (2015), for the related task of answering cloze questions (Winograd, 1972; Levesque et al., 2011). To create these datasets, either crowdsourcing or *(semi-)synthetic* approaches are used. The (semi-)synthetic datasets (e.g., Hermann et al. (2015)) are large in size and cheap to obtain; however, they do not share the same characteristics as explicit QA/RC questions (Rajpurkar et al., 2016). In comparison, high-quality *crowdsourced* datasets are much smaller in size, and the annotation process is quite expensive because the labeled examples require expertise and careful design (Chen et al., 2016).

Thus, there is a need for methods that can automatically generate high-quality question-answer pairs. Serban et al. (2016) propose the use of recurrent neural networks to generate QA pairs from structured knowledge resources such as Freebase. Their work relies on the existence of automatically acquired KBs, which are known to have errors and suffer from incompleteness. They are also nontrivial to obtain. In addition, the questions in the resulting dataset are limited to queries regarding a single fact (i.e., tuple) in the KB.

Motivated by the need for large scale QA pairs and the limitations of recent work, we investigate methods that can automatically "harvest" (generate) question-answer pairs from raw text/unstructured documents, such as Wikipedia-type articles.

Recent work along these lines (Du et al., 2017; Zhou et al., 2017) (see Section 2) has proposed the use of attention-based recurrent neural models trained on the crowdsourced SQuAD dataset (Rajpurkar et al., 2016) for question generation. While successful, the resulting QA pairs are based on information from a single sentence. As described in Du et al. (2017), however, nearly 30% of the questions in the human-generated questions of SQuAD rely on information beyond a single sentence. For example, in Figure 1, the second and third questions require coreference information (i.e., recognizing that "His" in sentence 2 and "He" in sentence 3 both corefer with "Tesla" in sentence 1) to answer them.

Thus, our research studies methods for incorporating coreference information into the training of a question generation system. In particular, we propose gated **Coref**erence knowledge for **N**eural **Q**uestion **G**eneration (**CorefNQG**), a neural sequence model with a novel gating mechanism that leverages continuous representations of *coreference clusters* — the set of mentions used to refer to each entity — to better encode linguistic knowledge introduced by coreference, for paragraph-level question generation.

In an evaluation using the SQuAD dataset, we find that CorefNQG enables better question generation. It outperforms significantly the baseline neural sequence models that encode information from a single sentence, and a model that encodes *all* preceding context and the input sentence itself. When evaluated on only the portion of SQuAD that requires coreference resolution, the gap between our system and the baseline systems is even larger.

By applying our approach to the 10,000 top-ranking Wikipedia articles, we obtain a question answering/reading comprehension dataset with over one million QA pairs; we provide a qualitative analysis in Section 6. The dataset and the source code for the system are available at `https://github.com/xinyadu/HarvestingQA`.

2 Related Work

2.1 Question Generation

Since the work by Rus et al. (2010), question generation (QG) has attracted interest from both the NLP and NLG communities. Most early work in QG employed rule-based approaches to transform input text into questions, usually requiring the application of a sequence of well-designed general rules or templates (Mitkov and Ha, 2003; Labutov et al., 2015). Heilman and Smith (2010) introduced an overgenerate-and-rank approach: their system generates a set of questions and then ranks them to select the top candidates. Apart from generating questions from raw text, there has also been research on question generation from symbolic representations (Yao et al., 2012; Olney et al., 2012).

With the recent development of deep representation learning and large QA datasets, there has been research on recurrent neural network based approaches for question generation. Serban et al. (2016) used the encoder-decoder framework to generate QA pairs from knowledge base triples; Reddy et al. (2017) generated questions from a knowledge graph; Du et al. (2017) studied how to generate questions from sentences using an attention-based sequence-to-sequence model and investigated the effect of exploiting sentence- vs. paragraph-level information. Du and Cardie (2017) proposed a hierarchical neural sentence-level sequence tagging model for identifying question-worthy sentences in a text passage. Finally, Duan et al. (2017) investigated how to use question generation to help improve question answering systems on the sentence selection subtask.

In comparison to the related methods from above that generate questions from raw text, our method is different in its ability to take into account contextual information beyond the sentence-level by introducing coreference knowledge.

2.2 Question Answering Datasets and Creation

Recently there has been an increasing interest in question answering with the creation of many datasets. Most are built using crowdsourcing; they are generally comprised of fewer than 100,000 QA pairs and are time-consuming to create. WebQuestions (Berant et al., 2013), for example, contains 5,810 questions crawled via the Google Suggest API and is designed for knowledge base QA with answers restricted to Freebase entities. To tackle the size issues associated with WebQuestions, Bordes et al. (2015) introduce SimpleQuestions, a dataset of 108,442 questions authored by English speakers. SQuAD (Rajpurkar et al., 2016) is a dataset for machine comprehension; it is created by showing a Wikipedia paragraph to human annotators and asking them to write questions based on the paragraph. TriviaQA (Joshi et al., 2017) includes 95k question-answer authored by trivia enthusiasts and corresponding evidence documents.

(Semi-)synthetic generated datasets are easier to build to large-scale (Hill et al., 2015; Hermann et al., 2015). They usually come in the form of cloze-style questions. For example, Hermann et al. (2015) created over a million examples by pairing CNN and Daily Mail news articles with their summarized bullet points. Chen et al. (2016) showed that this dataset is quite noisy due to the method of data creation and concluded that performance of QA systems on the dataset is almost saturated.

Closest to our work is that of Serban et al. (2016). They train a neural triple-to-sequence model on SimpleQuestions, and apply their system to Freebase to produce a large collection of human-like question-answer pairs.

3 Task Definition

Our goal is to harvest high quality question-answer pairs from the paragraphs of an article of interest. In our task formulation, this consists of two steps: **candidate answer extraction** and **answer-specific question generation**. Given an input paragraph, we first identify a set of *question-worthy* candidate answers $ans = (ans_1, ans_2, ..., ans_l)$, each a span of text as denoted in color in Figure 1. For each candidate answer ans_i, we then aim to generate a question Q — a sequence of tokens $y_1, ..., y_N$ — based on the sentence S that contains candidate ans_i such that:

- Q asks about an aspect of ans_i that is of potential interest to a human;
- Q might rely on information from sentences that precede S in the paragraph.

Mathematically then,

$$Q = \arg\max_{Q} P\left(Q|S, C\right) \tag{1}$$

where $P(Q|S,C) = \prod_{n=1}^{N} P\left(y_n|y_{<n}, S, C\right)$ where C is the set of sentences that precede S in the paragraph.

4 Methodology

In this section, we introduce our framework for harvesting the question-answer pairs. As described above, it consists of the question generator CorefNQG (Figure 2) and a candidate answer extraction module. During test/generation time, we (1) run the answer extraction module on the input text to obtain answers, and then (2) run the question generation module to obtain the corresponding questions.

4.1 Question Generation

As shown in Figure 2, our generator prepares the feature-rich input embedding — a concatenation of (a) a refined coreference position feature embedding, (b) an answer feature embedding, and (c) a word embedding, each of which is described below. It then encodes the textual input using an LSTM unit (Hochreiter and Schmidhuber, 1997). Finally, an attention-copy equipped decoder is used to decode the question.

More specifically, given the input sentence S (containing an answer span) and the preceding context C, we first run a coreference resolution system to get the coref-clusters for S and C and use them to create a *coreference transformed* input sentence: for each pronoun, we append its most representative non-pronominal coreferent mention. Specifically, we apply the simple feed-forward network based mention-ranking model of Clark and Manning (2016) to the concatenation of C and S to get the coref-clusters for all entities in C and S. The C&M model produces a score/representation s for each mention pair (m_1, m_2),

$$s(m_1, m_2) = \mathbf{W}_m h_m(m_1, m_2) + b_m \tag{2}$$

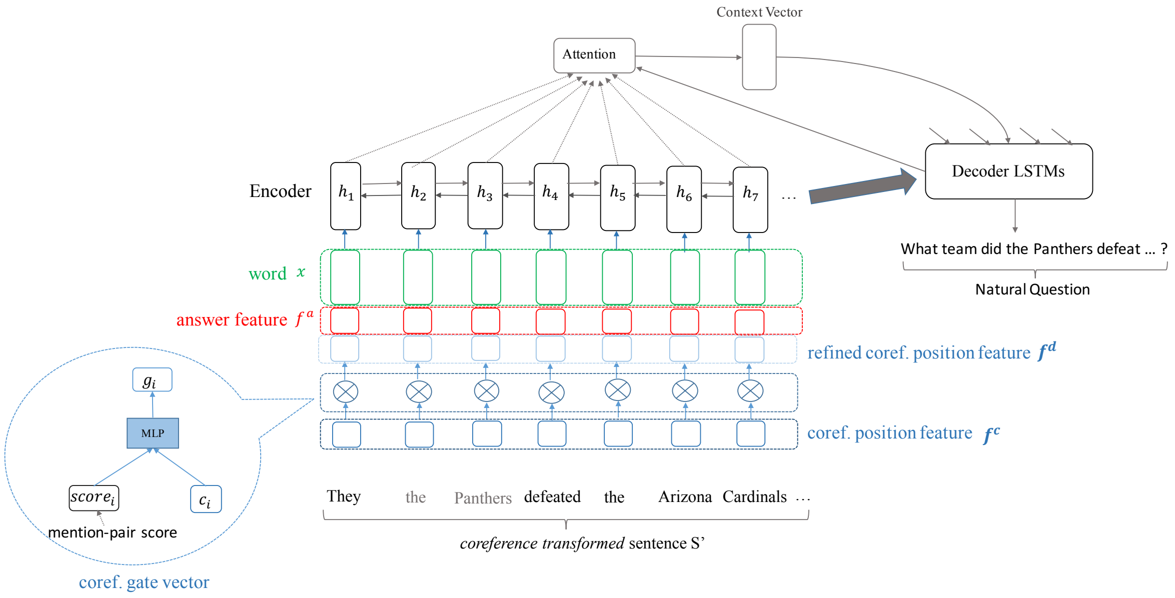

Context Vector
Attention
Encoder
Decoder LSTMs
What team did the Panthers defeat ... ?
Natural Question
word x
answer feature f^a
refined coref. position feature f^d
coref. position feature f^c
g_i
MLP
$score_i$
c_i
mention-pair score
coref. gate vector
They the Panthers defeated the Arizona Cardinals ...
coreference transformed sentence S'

Figure 2: The gated **Coref**erence knowledge for **N**eural **Q**uestion **G**eneration (**CorefNQG**) Model.

word	they	the	panthers	defeated	the	arizona	cardinals	49	–	15	...
ans. feature	O	O	O	O	B_ANS	I_ANS	I_ANS	O	O	O	...
coref. feature	B_PRO	B_ANT	I_ANT	O	O	O	O	O	O	O	...

Table 1: Example input sentence with coreference and answer position features. The corresponding gold question is "What team did the Panthers defeat in the NFC championship game ?"

where $\mathbf{W}_m$ is a $1 \times d$ weight matrix and b is the bias. $h_m(m_1, m_2)$ is representation of the last hidden layer of the three layer feedforward neural network.

For each pronoun in S, we then heuristically identify the most "representative" antecedent from its coref-cluster. (Proper nouns are preferred.) We append the new mention after the pronoun. For example, in Table 1, "the panthers" is the most representative mention in the coref-cluster for "they". The new sentence with the appended coreferent mention is our *coreference transformed* input sentence $S^{'}$ (see Figure 2).

Coreference Position Feature Embedding For each token in $S^{'}$, we also maintain one position feature $\mathbf{f^c} = (c_1, ..., c_n)$, to denote pronouns (e.g., "they") and antecedents (e.g., "the panthers"). We use the BIO tagging scheme to label the associated spans in $S^{'}$. "B_ANT" denotes the start of an antecedent span, tag "I_ANT" continues the antecedent span and tag "O" marks tokens that do not form part of a mention span. Similarly, tags "B_PRO" and "I_PRO" denote the pronoun span. (See Table 1, "coref. feature".)

Refined Coref. Position Feature Embedding Inspired by the success of gating mechanisms for controlling information flow in neural networks (Hochreiter and Schmidhuber, 1997; Dauphin et al., 2017), we propose to use a gating network here to obtain a refined representation of the coreference position feature vectors $\mathbf{f^c} = (c_1, ..., c_n)$. The main idea is to utilize the mention-pair score (see Equation 2) to help the neural network learn the importance of the coreferent phrases. We compute the refined (gated) coreference position feature vector $\mathbf{f^d} = (d_1, ..., d_n)$ as follows,

$$\begin{aligned} g_i &= \text{ReLU}(\mathbf{W}_a c_i + \mathbf{W}_b score_i + b) \\ d_i &= g_i \odot c_i \end{aligned} \tag{3}$$

where $\odot$ denotes an element-wise product between two vectors and ReLU is the rectified linear activation function. $score_i$ denotes the mention-pair score for each antecedent token (e.g., "the" and "panthers") with the pronoun (e.g., "they"); $score_i$ is obtained from the trained model (Equation 2) of the C&M. If token i is not added later as an antecedent token, $score_i$ is set to zero. $\mathbf{W}_a$, $\mathbf{W}_b$ are weight matrices and b is the bias vector.

Answer Feature Embedding We also include an answer position feature embedding to generate answer-specific questions; we denote the answer span with the usual BIO tagging scheme (see, e.g., "the arizona cardinals" in Table 1). During training and testing, the answer span feature (i.e., "`B_ANS`", "`I_ANS`" or "`O`") is mapped to its feature embedding space: $\mathbf{f^a} = (a_1, ..., a_n)$.

Word Embedding To obtain the word embedding for the tokens themselves, we just map the tokens to the word embedding space: $\mathbf{x} = (x_1, ..., x_n)$.

Final Encoder Input As noted above, the final input to the LSTM-based encoder is a concatenation of (1) the refined coreference position feature embedding (light blue units in Figure 2), (2) the answer position feature embedding (red units), and (3) the word embedding for the token (green units),

$$e_i = \text{concat}(d_i, a_i, x_i) \tag{4}$$

Encoder As for the encoder itself, we use bidirectional LSTMs to read the input $\mathbf{e} = (e_1, ..., e_n)$ in both the forward and backward directions. After encoding, we obtain two sequences of hidden vectors, namely, $\overrightarrow{\mathbf{h}} = (\overrightarrow{h_1}, ..., \overrightarrow{h_n})$ and $\overleftarrow{\mathbf{h}} = (\overleftarrow{h_1}, ..., \overleftarrow{h_n})$. The final output state of the encoder is the concatenation of $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$ where

$$h_i = \text{concat}(\overrightarrow{h_i}, \overleftarrow{h_i}) \tag{5}$$

Question Decoder with Attention & Copy On top of the feature-rich encoder, we use LSTMs with attention (Bahdanau et al., 2015) as the decoder for generating the question $y_1, ..., y_m$ one token at a time. To deal with rare/unknown words, the decoder also allows directly copying words from the source sentence via pointing (Vinyals et al., 2015).

At each time step t, the decoder LSTM reads the previous word embedding w_{t-1} and previous hidden state s_{t-1} to compute the new hidden state,

$$s_t = \text{LSTM}(w_{t-1}, s_{t-1}) \tag{6}$$

Then we calculate the *attention distribution* α_t as in Bahdanau et al. (2015),

$$\begin{aligned} e_{t,i} &= h_i^T \mathbf{W}_c s_{t-1} \\ \alpha_t &= \text{softmax}(e_t) \end{aligned} \tag{7}$$

where $\mathbf{W}_c$ is a weight matrix and attention distribution α_t is a probability distribution over the source sentence words. With α_t, we can obtain the context vector h_t^*,

$$h_t^* = \sum_{i=1}^{n} \alpha_t^i h_i \tag{8}$$

Then, using the context vector h_t^* and hidden state s_t, the probability distribution over the target (question) side vocabulary is calculated as,

$$P_{vocab} = \text{softmax}(\mathbf{W}_d \text{concat}(h_t^*, s_t)) \tag{9}$$

Instead of directly using P_{vocab} for training/generating with the fixed target side vocabulary, we also consider *copying* from the source sentence. The copy probability is based on the context vector h_t^* and hidden state s_t,

$$\lambda_t^{copy} = \sigma\left(\mathbf{W}_e h_t^* + \mathbf{W}_f s_t\right) \tag{10}$$

and the probability distribution over the source sentence words is the sum of the attention scores of the corresponding words,

$$P_{copy}(w) = \sum_{i=1}^{n} \alpha_t^i * \mathbb{1}\{w == w_i\} \tag{11}$$

Finally, we obtain the probability distribution over the dynamic vocabulary (i.e., union of original target side and source sentence vocabulary) by summing over P_{copy} and P_{vocab},

$$P(w) = \lambda_t^{copy} P_{copy}(w) + (1 - \lambda_t^{copy}) P_{vocab}(w) \tag{12}$$

where σ is the sigmoid function, and $\mathbf{W}_d$, $\mathbf{W}_e$, $\mathbf{W}_f$ are weight matrices.

4.2 Answer Span Identification

We frame the problem of identifying candidate answer spans from a paragraph as a sequence labeling task and base our model on the BiLSTM-CRF approach for named entity recognition (Huang et al., 2015). Given a paragraph of n tokens, instead of directly feeding the sequence of word vectors $\mathbf{x} = (x_1, ..., x_n)$ to the LSTM units, we first construct the feature-rich embedding $\mathbf{x}'$ for each token, which is the concatenation of the word embedding, an NER feature embedding, and a character-level representation of the word (Lample et al., 2016). We use the concatenated vector as the "final" embedding $\mathbf{x}'$ for the token,

$$x_i' = \text{concat}(x_i, \text{CharRep}_i, \text{NER}_i) \tag{13}$$

where CharRep_i is the concatenation of the last hidden states of a character-based biLSTM. The intuition behind the use of NER features is that SQuAD answer spans contain a large number of named entities, numeric phrases, etc.

Then a multi-layer Bi-directional LSTM is applied to $(x_1', ..., x_n')$ and we obtain the output state

z_t for time step t by concatenation of the hidden states (forward and backward) at time step t from the last layer of the BiLSTM. We apply the softmax to $(z_1, ..., z_n)$ to get the normalized score representation for each token, which is of size $n \times k$, where k is the number of tags.

Instead of using a softmax training objective that minimizes the cross-entropy loss for each individual word, the model is trained with a CRF (Lafferty et al., 2001) objective, which minimizes the negative log-likelihood for the entire correct sequence: $-\log(p_{\mathbf{y}})$,

$$p_{\mathbf{y}} = \frac{\exp(q(\mathbf{x}', \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathbf{Y}'} \exp(q(\mathbf{x}', \mathbf{y}'))} \quad (14)$$

where $q(\mathbf{x}', \mathbf{y}) = \sum_{t=1}^{n} P_{t,y_t} + \sum_{t=0}^{n-1} A_{y_t, y_{t+1}}$, P_{t,y_t} is the score of assigning tag y_t to the t^{th} token, and $A_{y_t,y_{t+1}}$ is the transition score from tag y_t to y_{t+1}, the scoring matrix A is to be learned. $\mathbf{Y}'$ represents all the possible tagging sequences.

5 Experiments

5.1 Dataset

We use the SQuAD dataset (Rajpurkar et al., 2016) to train our models. It is one of the largest general purpose QA datasets derived from Wikipedia with over 100k questions posed by crowdworkers on a set of Wikipedia articles. The answer to each question is a segment of text from the corresponding Wiki passage. The crowdworkers were users of Amazon's Mechanical Turk located in the US or Canada. To obtain high-quality articles, the authors sampled 500 articles from the top 10,000 articles obtained by Nayuki's Wikipedia's internal PageRanks. The question-answer pairs were generated by annotators from a paragraph; and although the dataset is typically used to evaluate reading comprehension, it has also been used in an open domain QA setting (Chen et al., 2017; Wang et al., 2018). For training/testing answer extraction systems, we pair each paragraph in the dataset with the gold answer spans that it contains. For the question generation system, we pair each sentence that contains an answer span with the corresponding gold question as in Du et al. (2017).

To quantify the effect of using predicted (rather than gold standard) answer spans on question generation (e.g., predicted answer span boundaries can be inaccurate), we also train the models on an augmented "Training set w/ noisy examples" (see Table 2). This training set contains all of the original training examples *plus* new examples for predicted answer spans (from the top-performing answer extraction model, bottom row of Table 3) that *overlap* with a gold answer span. We pair the new training sentence (w/ predicted answer span) with the gold question. The added examples comprise 42.21% of the noisy example training set.

For generation of our one million QA pair corpus, we apply our systems to the 10,000 top-ranking articles of Wikipedia.

5.2 Evaluation Metrics

For question generation evaluation, we use BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014).[1] BLEU measures average n-gram precision vs. a set of reference questions and penalizes for overly short sentences. METEOR is a recall-oriented metric that takes into account synonyms, stemming, and paraphrases.

For answer candidate extraction evaluation, we use precision, recall and F-measure vs. the gold standard SQuAD answers. Since answer boundaries are sometimes ambiguous, we compute *Binary Overlap* and *Proportional Overlap* metrics in addition to *Exact Match*. Binary Overlap counts every predicted answer that overlaps with a gold answer span as correct, and Proportional Overlap give partial credit proportional to the amount of overlap (Johansson and Moschitti, 2010; Irsoy and Cardie, 2014).

5.3 Baselines and Ablation Tests

For question generation, we compare to the state-of-the-art baselines and conduct ablation tests as follows: **Du et al. (2017)**'s model is an attention-based RNN sequence-to-sequence neural network (without using the answer location information feature). **Seq2seq + copy$_{\text{w/ answer}}$** is the attention-based sequence-to-sequence model augmented with a copy mechanism, with answer features concatenated with the word embeddings during encoding. **Seq2seq + copy$_{\text{w/ full context + answer}}$** is the same model as the previous one, but we allow access to the full context (i.e., all the preceding sentences and the input sentence itself). We denote it as **ContextNQG** henceforth for simplicity. **CorefNQG** is the coreference-based model proposed in this paper. **CorefNQG–gating** is an

[1]We use the evaluation scripts of Du et al. (2017).

Models	Training set			Training set w/ noisy examples		
	BLEU-3	BLEU-4	METEOR	BLEU-3	BLEU-4	METEOR
Baseline (Du et al., 2017) (w/o answer)	17.50	12.28	16.62	15.81	10.78	15.31
Seq2seq + copy (w/ answer)	20.01	14.31	18.50	19.61	13.96	18.19
ContextNQG: Seq2seq + copy (w/ full context + answer)	20.31	14.58	18.84	19.57	14.05	18.19
CorefNQG	**20.90**	**15.16**	**19.12**	**20.19**	**14.52**	18.59
- gating	20.68	14.84	18.98	20.08	14.40	**18.64**
- mention-pair score	20.56	14.75	18.85	19.73	14.13	18.38

Table 2: Evaluation results for question generation.

Models	Precision			Recall			F-measure		
	Prop.	Bin.	Exact	Prop.	Bin.	Exact	Prop.	Bin.	Exact
NER	24.54	25.94	12.77	**58.20**	**67.66**	**38.52**	34.52	37.50	19.19
BiLSTM	43.54	45.08	22.97	28.43	35.99	18.87	34.40	40.03	20.71
BiLSTM w/ NER	44.35	46.02	25.33	33.30	40.81	23.32	38.04	43.26	24.29
BiLSTM-CRF w/ char	**49.35**	**51.92**	**38.58**	30.53	32.75	24.04	37.72	40.16	29.62
BiLSTM-CRF w/ char w/ NER	45.96	51.61	33.90	41.05	43.98	28.37	**43.37**	**47.49**	**30.89**

Table 3: Evaluation results of answer extraction systems.

ablation test, the gating network is removed and the coreference position embedding is not refined. **CorefNQG–mention-pair score** is also an ablation test where all mention-pair $score_i$ are set to zero.

For answer span extraction, we conduct experiments to compare the performance of an off-the-shelf NER system and BiLSTM based systems.

For **training and implementation details**, please see the Supplementary Material.

6 Results and Analysis

6.1 Automatic Evaluation

Table 2 shows the BLEU-$\{3, 4\}$ and METEOR scores of different models. Our CorefNQG outperforms the seq2seq baseline of Du et al. (2017) by a large margin. This shows that the copy mechanism, answer features and coreference resolution all aid question generation. In addition, CorefNQG outperforms both Seq2seq+Copy models significantly, whether or not they have access to the full context. This demonstrates that the coreference knowledge encoded with the gating network explicitly helps with the training and generation: it is more difficult for the neural sequence model to learn the coreference knowledge in a latent way. (See input 1 in Figure 3 for an example.) Building end-to-end models that take into account coreference knowledge in a latent way is an interesting direction to explore. In the ablation tests, the performance drop of CorefNQG–gating

	BLEU-3	BLEU-4	METEOR
Seq2seq + copy (w/ ans.)	17.81	12.30	17.11
ContextNQG	18.05	12.53	17.33
CorefNQG	**18.46**	**12.96**	**17.58**

Table 4: Evaluation results for question generation on the portion that requires coreference knowledge (36.42% examples of the original test set).

shows that the gating network is playing an important role for getting *refined* coreference position feature embedding, which helps the model learn the importance of an antecedent. The performance drop of CorefNQG–mention-pair score shows the mention-pair score introduced from the external system (Clark and Manning, 2016) helps the neural network better encode coreference knowledge.

To better understand the effect of coreference resolution, we also evaluate our model and the baseline models on just that portion of the test set that requires pronoun resolution (36.42% of the examples) and show the results in Table 4. The gaps of performance between our model and the baseline models are still significant. Besides, we see that all three systems' performance drop on this partial test set, which demonstrates the hardness of generating questions for the cases that require pronoun resolution (passage context).

We also show in Table 2 the results of the QG models trained on the training set augmented with noisy examples with predicted answer spans.

Input 1: The elizabethan navigator, sir francis drake was born in the nearby town of tavistock and was the mayor of plymouth. ... . he died of dysentery in 1596 off the coast of puerto rico.
Human: In what year did Sir Francis Drake die ?
ContextNQG: When did he die ?
CorefNQG: When did sir francis drake die ?

Input 2: american idol is an american singing competition ... . it began airing on fox on june 11 , 2002, as an addition to the idols format based on the british series pop idol and has since become one of the most successful shows in the history of american television.
Human: When did american idol first air on tv ?
ContextNQG: When did fox begin airing ?
CorefNQG: When did american idol begin airing ?

Input 3: ... the a38 dual-carriageway runs from east to west across the north of the city . within the city it is designated as ' the parkway ' and represents the boundary between the urban parts of the city and the generally more recent suburban areas .
Human: What is the a38 called inside the city ?
ContextNQG: What is another name for the city ?
CorefNQG: What is the city designated as ?

Figure 3: Example questions (with answers highlighted) generated by human annotators (ground truth questions), by our system CorefNQG, and by the Seq2seq+Copy model trained with full context (i.e., ContextNQG).

There is a consistent but acceptable drop for each model on this new training set, given the inaccuracy of predicted answer spans. We see that CorefNQG still outperforms the baseline models across all metrics.

Figure 3 provides sample output for input sentences that require contextual coreference knowledge. We see that ContextNQG fails in all cases; our model misses only the third example due to an error introduced by coreference resolution — the "city" and "it" are considered coreferent. We can also see that human-generated questions are more natural and varied in form with better paraphrasing.

In Table 3, we show the evaluation results for different answer extraction models. First we see that all variants of BiLSTM models outperform the off-the-shelf NER system (that proposes all NEs as answer spans), though the NER system has a higher recall. The BiLSTM-CRF that encodes the character-level and NER features for each token performs best in terms of F-measure.

6.2 Human Study

We hired four native speakers of English to rate the systems' outputs. Detailed guidelines for the raters are listed in the supplementary materials.

	Grammaticality	Making Sense	Answerability	Avg. rank
ContextNQG	3.793	3.836	3.892	1.768
CorefNQG	3.804*	3.847**	3.895*	1.762
Human	**3.807**	**3.850**	**3.902**	**1.758**

Table 5: Human evaluation results for question generation. "Grammaticality", "Making Sense" and "Answerability" are rated on a 1–5 scale (5 for the best, see the supplementary materials for a detailed rating scheme), "Average rank" is rated on a 1–3 scale (1 for the most preferred, ties are allowed.) Two-tailed t-test results are shown for our method compared to ContextNQG (stat. significance is indicated with $^{*}(p < 0.05)$, $^{**}(p < 0.01)$.)

The evaluation can also be seen as a measure of the quality of the generated dataset (Section 6.3). We randomly sampled 11 passages/paragraphs from the test set; there are in total around 70 question-answer pairs for evaluation.

We consider three metrics — "grammaticality", "making sense" and "answerability". The evaluators are asked to first rate the grammatical correctness of the generated question (before being shown the associated input sentence or any other textual context). Next, we ask them to rate the degree to which the question "makes sense" given the input sentence (i.e., without considering the correctness of the answer span). Finally, evaluators rate the "answerability" of the question given the full context.

Table 5 shows the results of the human evaluation. Bold indicates top scores. We see that the original human questions are preferred over the two NQG systems' outputs, which is understandable given the examples in Figure 3. The human-generated questions make more sense and correspond better with the provided answers, particularly when they require information in the preceding context. How exactly to capture the preceding context so as to *ask* better and more diverse questions is an interesting future direction for research. In terms of grammaticality, however, the neural models do quite well, achieving very close to human performance. In addition, we see that our method (CorefNQG) performs statistically significantly better across all metrics in comparison to the baseline model (ContextNQG), which has access to the entire preceding context in the passage.

6.3 The Generated Corpus

Our system generates in total 1,259,691 question-answer pairs, nearly 126 questions per article. Figure 5 shows the distribution of different types of

	Exact Match		F-1	
	Dev	Test	Dev	Test
DocReader (Chen et al., 2017)	82.33	81.65	88.20	87.79

Table 6: Performance of the neural machine reading comprehension model (no initialization with pretrained embeddings) on our generated corpus.

The United States of America (USA), commonly referred to as the United States (U.S.) or America, is a federal republic composed of states, a federal district, five major self-governing territories, and various possessions. ... . The territories are scattered about the Pacific Ocean and the Caribbean Sea. Nine time zones are covered. The geography, climate and wildlife of the country are extremely diverse.
Q1: What is another name for the united states of america ?
Q2: How many major territories are in the united states?
Q3: What are the territories scattered about ?

Figure 4: Example question-answer pairs from our generated corpus.

questions in our dataset vs. the SQuAD training set. We see that the distribution for "In what", "When", "How long", "Who", "Where", "What does" and "What do" questions in the two datasets is similar. Our system generates more "What is", "What was" and "What percentage" questions, while the proportions of "What did", "Why" and "Which" questions in SQuAD are larger than ours. One possible reason is that the "Why", "What did" questions are more *complicated* to ask (sometimes involving world knowledge) and the answer spans are longer phrases of various types that are harder to identify. "What is" and "What was" questions, on the other hand, are often *safer* for the neural networks systems to ask.

In Figure 4, we show some examples of the generated question-answer pairs. The answer extractor identifies the answer span boundary well and all three questions correspond to their answers. Q2 is valid but not entirely accurate. For more examples, please refer to our supplementary materials.

Table 6 shows the performance of a top-performing system for the SQuAD dataset (Document Reader (Chen et al., 2017)) when applied to the development and test set portions of our generated dataset. The system was trained on the training set portion of our dataset. We use the SQuAD evaluation scripts, which calculate exact match (EM) and F-1 scores.[2] Performance of the

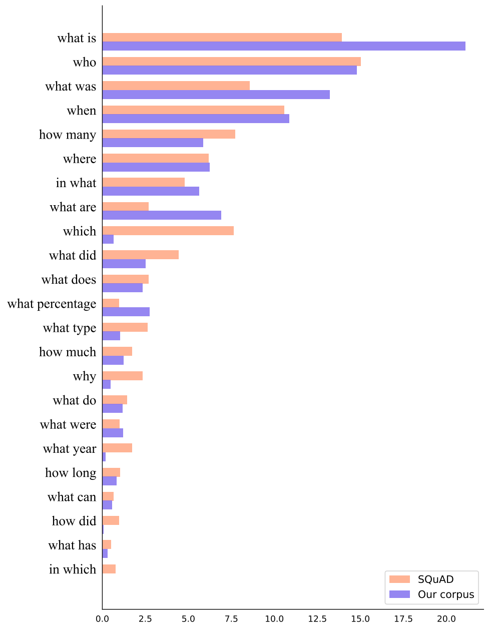


Figure 5: Distribution of question types of our corpus and SQuAD training set. The categories are the ones used in Wang et al. (2016), we add one more category: "what percentage".

neural machine reading model is reasonable. We also train the DocReader on our training set and test the models' performance on the *original* dev set of SQuAD; for this, the performance is around 45.2% on EM and 56.7% on F-1 metric. DocReader trained on the *original* SQuAD training set achieves 69.5% EM, 78.8% F-1 indicating that our dataset is more difficult and/or less natural than the crowd-sourced QA pairs of SQuAD.

7 Conclusion

We propose a new neural network model for better encoding coreference knowledge for paragraph-level question generation. Evaluations with different metrics on the SQuAD machine reading dataset show that our model outperforms state-of-the-art baselines. The ablation study shows the effectiveness of different components in our model. Finally, we apply our question generation framework to produce a corpus of 1.26 million question-answer pairs, which we hope will benefit the QA research community. It would also be interesting to apply our approach to incorporating coreference knowledge to other text generation tasks.

Acknowledgments

We thank the anonymous reviewers and members of Cornell NLP group for helpful comments.

[2]F-1 measures the average overlap between the predicted answer span and ground truth answer (Rajpurkar et al., 2016).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations Workshop (ICLR)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1533–1544. http://www.aclweb.org/anthology/D13-1160.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. http://www.aclweb.org/anthology/P16-1223.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1870–1879. https://doi.org/10.18653/v1/P17-1171.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 643–653. https://doi.org/10.18653/v1/P16-1061.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*. pages 933–941.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 376–380. http://www.aclweb.org/anthology/W14-3348.

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2067–2073. http://aclweb.org/anthology/D17-1219.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1342–1352. https://doi.org/10.18653/v1/P17-1123.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 866–874. http://aclweb.org/anthology/D17-1090.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 609–617. http://www.aclweb.org/anthology/N10-1086.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

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

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

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 720–728. https://doi.org/10.3115/v1/D14-1080.

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 67–76.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1601–1611. https://doi.org/10.18653/v1/P17-1147.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 889–898.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 260–270. https://doi.org/10.18653/v1/N16-1030.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*. volume 46, page 47.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1400–1409. https://doi.org/10.18653/v1/D16-1147.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*. Association for Computational Linguistics, pages 17–22.

Andrew M Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. *Dialogue & Discourse* 3(2):75–99.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 376–385. http://aclweb.org/anthology/E17-1036.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, pages 251–257.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 588–598. http://www.aclweb.org/anthology/P16-1056.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology* 3(1):1–191.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue & Discourse* 3(2):11–42.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .

Multi-Passage Machine Reading Comprehension with Cross-Passage Answer Verification

Yizhong Wang[1]*, Kai Liu[2], Jing Liu[2], Wei He[2],
Yajuan Lyu[2], Hua Wu[2], Sujian Li[1] and Haifeng Wang[2]

[1]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2]Baidu Inc., Beijing, China
{yizhong, lisujian}@pku.edu.cn, {liukai20, liujing46,
hewei06, lvyajuan, wu_hua, wanghaifeng}@baidu.com

Abstract

Machine reading comprehension (MRC) on real web data usually requires the machine to answer a question by analyzing multiple passages retrieved by search engine. Compared with MRC on a single passage, multi-passage MRC is more challenging, since we are likely to get multiple confusing answer candidates from different passages. To address this problem, we propose an end-to-end neural model that enables those answer candidates from different passages to verify each other based on their content representations. Specifically, we jointly train three modules that can predict the final answer based on three factors: the answer boundary, the answer content and the cross-passage answer verification. The experimental results show that our method outperforms the baseline by a large margin and achieves the state-of-the-art performance on the English MS-MARCO dataset and the Chinese DuReader dataset, both of which are designed for MRC in real-world settings.

1 Introduction

Machine reading comprehension (MRC), empowering computers with the ability to acquire knowledge and answer questions from textual data, is believed to be a crucial step in building a general intelligent agent (Chen et al., 2016). Recent years have seen rapid growth in the MRC community. With the release of various datasets, the MRC task has evolved from the early cloze-style test (Hermann et al., 2015; Hill et al., 2015) to answer extraction from a single passage (Rajpurkar et al., 2016) and to the latest more complex question answering on web data (Nguyen et al., 2016; Dunn et al., 2017; He et al., 2017).

*This work was done while the first author was doing internship at Baidu Inc.

Great efforts have also been made to develop models for these MRC tasks , especially for the answer extraction on single passage (Wang and Jiang, 2016; Seo et al., 2016; Pan et al., 2017). A significant milestone is that several MRC models have exceeded the performance of human annotators on the SQuAD dataset[1] (Rajpurkar et al., 2016). However, this success on single Wikipedia passage is still not adequate, considering the ultimate goal of reading the whole web. Therefore, several latest datasets (Nguyen et al., 2016; He et al., 2017; Dunn et al., 2017) attempt to design the MRC tasks in more realistic settings by involving search engines. For each question, they use the search engine to retrieve multiple passages and the MRC models are required to read these passages in order to give the final answer.

One of the intrinsic challenges for such multi-passage MRC is that since all the passages are question-related but usually independently written, it's probable that multiple confusing answer candidates (correct or incorrect) exist. Table 1 shows an example from MS-MARCO. We can see that all the answer candidates have semantic matching with the question while they are literally different and some of them are even incorrect. As is shown by Jia and Liang (2017), these confusing answer candidates could be quite difficult for MRC models to distinguish. Therefore, special consideration is required for such multi-passage MRC problem.

In this paper, we propose to leverage the answer candidates from different passages to verify the final correct answer and rule out the noisy incorrect answers. Our hypothesis is that the cor-

[1]https://rajpurkar.github.io/SQuAD-explorer/

Question: What is the difference between a mixed and pure culture?

Passages:

[1] **A culture is a society's total way of living and a society is a group that live in a defined territory and participate in common culture.** While the answer given is in essence true, societies originally form for the express purpose to enhance . . .

[2] . . . There has been resurgence in the economic system known as capitalism during the past two decades. 4. **The mixed economy is a balance between socialism and capitalism.** As a result, some institutions are owned and maintained by . . .

[3] **A pure culture is one in which only one kind of microbial species is found whereas in mixed culture two or more microbial species formed colonies.** Culture on the other hand, is the lifestyle that the people in the country . . .

[4] Best Answer: **A pure culture comprises a single species or strains. A mixed culture is taken from a source and may contain multiple strains or species.** A contaminated culture contains organisms that derived from some place . . .

[5] . . . It will be at that time when we can truly obtain a pure culture. **A pure culture is a culture consisting of only one strain.** You can obtain a pure culture by picking out a small portion of the mixed culture . . .

[6] **A pure culture is one in which only one kind of microbial species is found whereas in mixed culture two or more microbial species formed colonies.** A pure culture is a culture consisting of only one strain. . . .

. . . . . .

Reference Answer: A pure culture is one in which only one kind of microbial species is found whereas in mixed culture two or more microbial species formed colonies.

Table 1: An example from MS-MARCO. The text in bold is the predicted answer candidate from each passage according to the boundary model. The candidate from [1] is chosen as the final answer by this model, while the correct answer is from [6] and can be verified by the answers from [3], [4], [5].

rect answers could occur more frequently in those passages and usually share some commonalities, while incorrect answers are usually different from one another. The example in Table 1 demonstrates this phenomenon. We can see that the answer candidates extracted from the last four passages are all valid answers to the question and they are semantically similar to each other, while the answer candidates from the other two passages are incorrect and there is no supportive information from other passages. As human beings usually compare the answer candidates from different sources to deduce the final answer, we hope that MRC model can also benefit from the cross-passage answer verification process.

The overall framework of our model is demonstrated in Figure 1 , which consists of three modules. First, we follow the boundary-based MRC models (Seo et al., 2016; Wang and Jiang, 2016) to find an answer candidate for each passage by identifying the start and end position of the answer (Figure 2). Second, we model the meanings of the answer candidates extracted from those passages and use the content scores to measure the quality of the candidates from a second perspective. Third, we conduct the answer verification by enabling each answer candidate to attend to the other candidates based on their representations. We hope that the answer candidates can collect supportive information from each other according to their semantic similarities and further decide whether each candidate is correct or not. Therefore, the final answer is determined by three factors: the boundary, the content and the cross-passage answer verification. The three steps are modeled using different modules, which can be jointly trained in our end-to-end framework.

We conduct extensive experiments on the MS-MARCO (Nguyen et al., 2016) and DuReader (He et al., 2017) datasets. The results show that our answer verification MRC model outperforms the baseline models by a large margin and achieves the state-of-the-art performance on both datasets.

2 Our Approach

Figure 1 gives an overview of our multi-passage MRC model which is mainly composed of three modules including answer boundary prediction, answer content modeling and answer verification. First of all, we need to model the question and passages. Following Seo et al. (2016), we compute the question-aware representation for each passage (Section 2.1). Based on this representation, we employ a Pointer Network (Vinyals et al., 2015) to predict the start and end position of the answer in the module of answer boundary prediction (Section 2.2). At the same time, with the answer content model (Section 2.3), we estimate whether each word should be included in the answer and thus obtain the answer representations. Next, in the answer verification module (Section 2.4), each answer candidate can attend to the other answer candidates to collect supportive information and we compute one score for each candidate

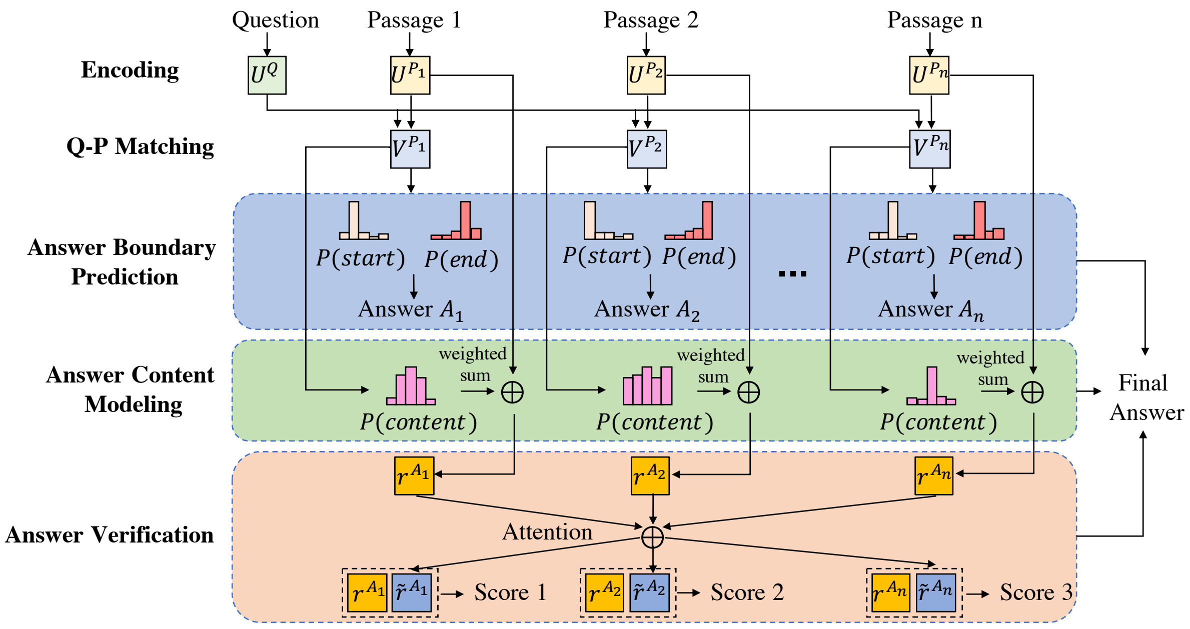


Figure 1: Overview of our method for multi-passage machine reading comprehension

to indicate whether it is correct or not according to the verification. The final answer is determined by not only the boundary but also the answer content and its verification score (Section 2.5).

2.1 Question and Passage Modeling

Given a question $\mathbf{Q}$ and a set of passages $\{\mathbf{P}_i\}$ retrieved by search engines, our task is to find the best concise answer to the question. First, we formally present the details of modeling the question and passages.

Encoding We first map each word into the vector space by concatenating its word embedding and sum of its character embeddings. Then we employ bi-directional LSTMs (BiLSTM) to encode the question $\mathbf{Q}$ and passages $\{\mathbf{P}_i\}$ as follows:

$$\mathbf{u}_t^Q = \text{BiLSTM}_Q(\mathbf{u}_{t-1}^Q, [\mathbf{e}_t^Q, \mathbf{c}_t^Q]) \quad (1)$$

$$\mathbf{u}_t^{P_i} = \text{BiLSTM}_P(\mathbf{u}_{t-1}^{P_i}, [\mathbf{e}_t^{P_i}, \mathbf{c}_t^{P_i}]) \quad (2)$$

where $\mathbf{e}_t^Q$, $\mathbf{c}_t^Q$, $\mathbf{e}_t^{P_i}$, $\mathbf{c}_t^{P_i}$ are the word-level and character-level embeddings of the t^{th} word. $\mathbf{u}_t^Q$ and $\mathbf{u}_t^{P_i}$ are the encoding vectors of the t^{th} words in $\mathbf{Q}$ and $\mathbf{P}_i$ respectively. Unlike previous work (Wang et al., 2017c) that simply concatenates all the passages, we process the passages independently at the encoding and matching steps.

Q-P Matching One essential step in MRC is to match the question with passages so that important information can be highlighted. We use the Attention Flow Layer (Seo et al., 2016) to conduct the Q-P matching in two directions. The similarity matrix $\mathbf{S} \in \mathbb{R}^{|\mathbf{Q}|\times|\mathbf{P}_i|}$ between the question and passage i is changed to a simpler version, where the similarity between the t^{th} word in the question and the k^{th} word in passage i is computed as:

$$\mathbf{S}_{t,k} = \mathbf{u}_t^{Q\top} \cdot \mathbf{u}_k^{P_i} \quad (3)$$

Then the context-to-question attention and question-to-context attention is applied strictly following Seo et al. (2016) to obtain the question-aware passage representation $\{\tilde{\mathbf{u}}_t^{P_i}\}$. We do not give the details here due to space limitation. Next, another BiLSTM is applied in order to fuse the contextual information and get the new representation for each word in the passage, which is regarded as the match output:

$$\mathbf{v}_t^{P_i} = \text{BiLSTM}_M(\mathbf{v}_{t-1}^{P_i}, \tilde{\mathbf{u}}_t^{P_i}) \quad (4)$$

Based on the passage representations, we introduce the three main modules of our model.

2.2 Answer Boundary Prediction

To extract the answer span from passages, mainstream studies try to locate the boundary of the answer, which is called boundary model. Following (Wang and Jiang, 2016), we employ Pointer Network (Vinyals et al., 2015) to compute the probability of each word to be the start or end position of the span:

$$g_k^t = \mathbf{w}_1^{a\intercal} \tanh(\mathbf{W}_2^a[\mathbf{v}_k^P, \mathbf{h}_{t-1}^a]) \quad (5)$$

$$\alpha_k^t = \exp(g_k^t) / \sum\nolimits_{j=1}^{|\mathbf{P}|} \exp(g_j^t) \quad (6)$$

$$\mathbf{c}_t = \sum\nolimits_{k=1}^{|\mathbf{P}|} \alpha_k^t \mathbf{v}_k^P \quad (7)$$

$$\mathbf{h}_t^a = \text{LSTM}(\mathbf{h}_{t-1}^a, \mathbf{c}_t) \quad (8)$$

By utilizing the attention weights, the probability of the k^{th} word in the passage to be the start and end position of the answer is obtained as α_k^1 and α_k^2. It should be noted that the pointer network is applied to the concatenation of all passages, which is denoted as P so that the probabilities are comparable across passages. This boundary model can be trained by minimizing the negative log probabilities of the true start and end indices:

$$\mathcal{L}_{boundary} = -\frac{1}{N}\sum_{i=1}^{N}(\log \alpha_{y_i^1}^1 + \log \alpha_{y_i^2}^2) \quad (9)$$

where N is the number of samples in the dataset and y_i^1, y_i^2 are the gold start and end positions.

2.3 Answer Content Modeling

Previous work employs the boundary model to find the text span with the maximum boundary score as the final answer. However, in our context, besides locating the answer candidates, we also need to model their meanings in order to conduct the verification. An intuitive method is to compute the representation of the answer candidates separately after extracting them, but it could be hard to train such model end-to-end. Here, we propose a novel method that can obtain the representation of the answer candidates based on probabilities.

Specifically, we change the output layer of the classic MRC model. Besides predicting the boundary probabilities for the words in the passages, we also predict whether each word should be included in the content of the answer. The content probability of the k^{th} word is computed as:

$$p_k^c = \text{sigmoid}(\mathbf{w}_1^{c\intercal}\text{ReLU}(\mathbf{W}_2^c \mathbf{v}_k^{P_i})) \quad (10)$$

Training this content model is also quite intuitive. We transform the boundary labels into a continuous segment, which means the words within the answer span will be labeled as 1 and other words will be labeled as 0. In this way, we define the loss function as the averaged cross entropy:

$$\mathcal{L}_{content} = -\frac{1}{N}\frac{1}{|\text{P}|}\sum_{i=1}^{N}\sum_{j=1}^{|P|}[y_k^c \log p_k^c + (1-y_k^c)\log(1-p_k^c)] \quad (11)$$

The content probabilities provide another view to measure the quality of the answer in addition to the boundary. Moreover, with these probabilities, we can represent the answer from passage i as a weighted sum of all the word embeddings in this passage:

$$\mathbf{r}^{A_i} = \frac{1}{|\mathbf{P}_i|}\sum\nolimits_{k=1}^{|\mathbf{P}_i|} p_k^c[\mathbf{e}_k^{P_i}, \mathbf{c}_k^{P_i}] \quad (12)$$

2.4 Cross-Passage Answer Verification

The boundary model and the content model focus on extracting and modeling the answer within a single passage respectively, with little consideration of the cross-passage information. However, as is discussed in Section 1, there could be multiple answer candidates from different passages and some of them may mislead the MRC model to make an incorrect prediction. It's necessary to aggregate the information from different passages and choose the best one from those candidates. Therefore, we propose a method to enable the answer candidates to exchange information and verify each other through the cross-passage answer verification process.

Given the representation of the answer candidates from all passages $\{\mathbf{r}^{A_i}\}$, each answer candidate then attends to other candidates to collect supportive information via attention mechanism:

$$s_{i,j} = \begin{cases} 0, & \text{if } i = j, \\ \mathbf{r}^{A_i\intercal} \cdot \mathbf{r}^{A_j}, & \text{otherwise} \end{cases} \quad (13)$$

$$\alpha_{i,j} = \exp(s_{i,j}) / \sum\nolimits_{k=1}^{n} \exp(s_{i,k}) \quad (14)$$

$$\tilde{\mathbf{r}}^{A_i} = \sum\nolimits_{j=1}^{n} \alpha_{i,j}\mathbf{r}^{A_j} \quad (15)$$

Here $\tilde{\mathbf{r}}^{A_i}$ is the collected verification information from other passages based on the attention weights. Then we pass it together with the original representation $\mathbf{r}^{A_i}$ to a fully connected layer:

$$g_i^v = \mathbf{w}^{v\intercal}[\mathbf{r}^{A_i}, \tilde{\mathbf{r}}^{A_i}, \mathbf{r}^{A_i} \odot \tilde{\mathbf{r}}^{A_i}] \quad (16)$$

We further normalize these scores over all passages to get the verification score for answer candidate A_i:

$$p_i^v = \exp(g_i^v) / \sum\nolimits_{j=1}^{n} \exp(g_j^v) \quad (17)$$

In order to train this verification model, we take the answer from the gold passage as the gold answer. And the loss function can be formulated as the negative log probability of the correct answer:

$$\mathcal{L}_{verify} = -\frac{1}{N}\sum_{i=1}^{N} \log p^v_{y_i^v} \tag{18}$$

where y_i^v is the index of the correct answer in all the answer candidates of the i^{th} instance .

2.5 Joint Training and Prediction

As is described above, we define three objectives for the reading comprehension model over multiple passages: 1. finding the boundary of the answer; 2. predicting whether each word should be included in the content; 3. selecting the best answer via cross-passage answer verification. According to our design, these three tasks can share the same embedding, encoding and matching layers. Therefore, we propose to train them together as multi-task learning (Ruder, 2017). The joint objective function is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{boundary} + \beta_1\mathcal{L}_{content} + \beta_2\mathcal{L}_{verify} \tag{19}$$

where β_1 and β_2 are two hyper-parameters that control the weights of those tasks.

When predicting the final answer, we take the boundary score, content score and verification score into consideration. We first extract the answer candidate A_i that has the maximum boundary score from each passage i. This boundary score is computed as the product of the start and end probability of the answer span. Then for each answer candidate A_i, we average the content probabilities of all its words as the content score of A_i. And we can also predict the verification score for A_i using the verification model. Therefore, the final answer can be selected from all the answer candidates according to the product of these three scores.

3 Experiments

To verify the effectiveness of our model on multi-passage machine reading comprehension, we conduct experiments on the MS-MARCO (Nguyen et al., 2016) and DuReader (He et al., 2017) datasets. Our method achieves the state-of-the-art performance on both datasets.

3.1 Datasets

We choose the MS-MARCO and DuReader datasets to test our method, since both of them are designed from real-world search engines and involve a large number of passages retrieved from the web. One difference of these two datasets is that MS-MARCO mainly focuses on the English web data, while DuReader is designed for Chinese MRC. This diversity is expected to reflect the generality of our method. In terms of the data size, MS-MARCO contains 102023 questions, each of which is paired up with approximately 10 passages for reading comprehension. As for DuReader, it keeps the top-5 search results for each question and there are totally 201574 questions.

	MS-MARCO	DuReader
Multiple Answers	9.93%	67.28%
Multiple Spans	40.00%	56.38%

Table 2: Percentage of questions that have multiple valid answers or answer spans

One prerequisite for answer verification is that there should be multiple correct answers so that they can verify each other. Both the MS-MARCO and DuReader datasets require the human annotators to generate multiple answers if possible. Table 2 shows the proportion of questions that have multiple answers. However, the same answer that occurs many times is treated as one single answer here. Therefore, we also report the proportion of questions that have multiple answer spans to match with the human-generated answers. A span is taken as valid if it can achieve F1 score larger than 0.7 compared with any reference answer. From these statistics, we can see that the phenomenon of multiple answers is quite common for both MS-MARCO and DuReader. These answers will provide strong signals for answer verification if we can leverage them properly.

3.2 Implementation Details

For MS-MARCO, we preprocess the corpus with the reversible tokenizer from Stanford CoreNLP (Manning et al., 2014) and we choose the span that achieves the highest ROUGE-L score with the reference answers as the gold span for training. We employ the 300-D pre-trained Glove embeddings (Pennington et al., 2014) and keep it fixed during training. The character embeddings are randomly initialized with its dimension as 30. For DuReader, we follow the preprocessing described in He et al. (2017).

We tune the hyper-parameters according to the

Model	ROUGE-L	BLEU-1
FastQA_Ext (Weissenborn et al., 2017)	33.67	33.93
Prediction (Wang and Jiang, 2016)	37.33	40.72
ReasoNet (Shen et al., 2017)	38.81	39.86
R-Net (Wang et al., 2017c)	42.89	42.22
S-Net (Tan et al., 2017)	45.23	43.78
Our Model	**46.15**	**44.47**
S-Net (Ensemble)	46.65	44.78
Our Model (Ensemble)	**46.66**	**45.41**
Human	47	46

Table 3: Performance of our method and competing models on the MS-MARCO test set

validation performance on the MS-MARCO development set. The hidden size is set to be 150 and we apply $L2$ regularization with its weight as 0.0003. The task weights β_1, β_2 are both set to be 0.5. To train our model, we employ the Adam algorithm (Kingma and Ba, 2014) with the initial learning rate as 0.0004 and the mini-batch size as 32. Exponential moving average is applied on all trainable variables with a decay rate 0.9999.

Two simple yet effective technologies are employed to improve the final performance on these two datasets respectively. For MS-MARCO, approximately 8% questions have the answers as Yes or No, which usually cannot be solved by extractive approach (Tan et al., 2017). We address this problem by training a simple Yes/No classifier for those questions with certain patterns (e.g., starting with "is"). Concretely, we simply change the output layer of the basic boundary model so that it can predict whether the answer is "Yes" or "No". For DuReader, the retrieved document usually contains a large number of paragraphs that cannot be fed into MRC models directly (He et al., 2017). The original paper employs a simple a simple heuristic strategy to select a representative paragraph for each document, while we train a paragraph ranking model for this. We will demonstrate the effects of these two technologies later.

3.3 Results on MS-MARCO

Table 3 shows the results of our system and other state-of-the-art models on the MS-MARCO test set. We adopt the official evaluation metrics, including ROUGE-L (Lin, 2004) and BLEU-1 (Papineni et al., 2002). As we can see, for both metrics, our single model outperforms all the other competing models with an evident margin, which is extremely hard considering the near-human performance. If we ensemble the models trained with different random seeds and hyper-parameters, the results can be further improved and outperform the ensemble model in Tan et al. (2017), especially in terms of the BLEU-1.

Model	BLEU-4	ROUGE-L
Match-LSTM	31.8	39.0
BiDAF	31.9	39.2
PR + BiDAF	37.55	41.81
Our Model	**40.97**	**44.18**
Human	56.1	57.4

Table 4: Performance on the DuReader test set

Model	ROUGE-L	Δ
Complete Model	**45.65**	**-**
Answer Verification	44.38	-1.27
Content Modeling	44.27	-1.38
Joint Training	44.12	-1.53
YesNo Classification	41.87	-3.78
Boundary Baseline	**38.95**	**-6.70**

Table 5: Ablation study on MS-MARCO development set

3.4 Results on DuReader

The results of our model and several baseline systems on the test set of DuReader are shown in Table 4. The BiDAF and Match-LSTM models are provided as two baseline systems (He et al., 2017). Based on BiDAF, as is described in Section 3.2, we tried a new paragraph selection strategy by employing a paragraph ranking (PR) model. We can see that this paragraph ranking can boost the BiDAF baseline significantly. Finally, we implement our system based on this new strategy, and our system (single model) achieves further improvement by a large margin.

Question: What is the difference between a mixed and pure culture	**Scores**		
Answer Candidates:	**Boundary**	**Content**	**Verification**
[1] A culture is a society's total way of living and a society is a group ...	$\mathbf{1.0 \times 10^{-2}}$	$\mathbf{1.0 \times 10^{-1}}$	1.1×10^{-1}
[2] The mixed economy is a balance between socialism and capitalism.	1.0×10^{-4}	4.0×10^{-2}	3.2×10^{-2}
[3] A pure culture is one in which only one kind of microbial species is ...	5.5×10^{-3}	7.7×10^{-2}	1.2×10^{-1}
[4] A pure culture comprises a single species or strains. A mixed ...	2.7×10^{-3}	8.1×10^{-2}	1.3×10^{-1}
[5] A pure culture is a culture consisting of only one strain.	5.8×10^{-4}	7.9×10^{-2}	5.1×10^{-2}
[6] A pure culture is one in which only one kind of microbial species ...	5.8×10^{-3}	9.1×10^{-2}	$\mathbf{2.7 \times 10^{-1}}$
......		......	

Table 6: Scores predicted by our model for the answer candidates shown in Table 1. Although the candidate [1] gets high boundary and content scores, the correct answer [6] is preferred by the verification model and is chosen as the final answer.

4 Analysis and Discussion

4.1 Ablation Study

To get better insight into our system, we conduct in-depth ablation study on the development set of MS-MARCO, which is shown in Table 5. Following Tan et al. (2017), we mainly focus on the ROUGE-L score that is averaged case by case.

We first evaluate the answer verification by ablating the cross-passage verification model so that the verification loss and verification score will not be used during training and testing. Then we remove the content model in order to test the necessity of modeling the content of the answer. Since we don't have the content scores, we use the boundary probabilities instead to compute the answer representation for verification. Next, to show the benefits of joint training, we train the boundary model separately from the other two models. Finally, we remove the yes/no classification in order to show the real improvement of our end-to-end model compared with the baseline method that predicts the answer with only the boundary model.

From Table 5, we can see that the answer verification makes a great contribution to the overall improvement, which confirms our hypothesis that cross-passage answer verification is useful for the multi-passage MRC. For the ablation of the content model, we analyze that it will not only affect the content score itself, but also violate the verification model since the content probabilities are necessary for the answer representation, which will be further analyzed in Section 4.3. Another discovery is that jointly training the three models can provide great benefits, which shows that the three tasks are actually closely related and can boost each other with shared representations at bottom layers. At last, comparing our method with the baseline, we achieve an improvement of nearly 3 points without the yes/no classification. This significant improvement proves the effectiveness of our approach.

4.2 Case Study

To demonstrate how each module of our model takes effect when predicting the final answer, we conduct a case study in Table 6 with the same example that we discussed in Section 1. For each answer candidate, we list three scores predicted by the boundary model, content model and verification model respectively.

On the one hand, we can see that these three scores generally have some relevance. For example, the second candidate is given lowest scores by all the three models. We analyze that this is because the models share the same encoding and matching layers at bottom level and this relevance guarantees that the content and verification models will not violate the boundary model too much. On the other hand, we also see that the verification score can really make a difference here when the boundary model makes an incorrect decision among the confusing answer candidates ([1], [3], [4], [6]). Besides, as we expected, the verification model tends to give higher scores for those answers that have semantic commonality with each other ([3], [4], [6]), which are all valid answers in this case. By multiplying the three scores, our model finally predicts the answer correctly.

4.3 Necessity of the Content Model

In our model, we compute the answer representation based on the content probabilities predicted by a separate content model instead of directly using the boundary probabilities. We argue that this content model is necessary for our answer verification process. Figure 2 plots the predicted content probabilities as well as the boundary probabilities

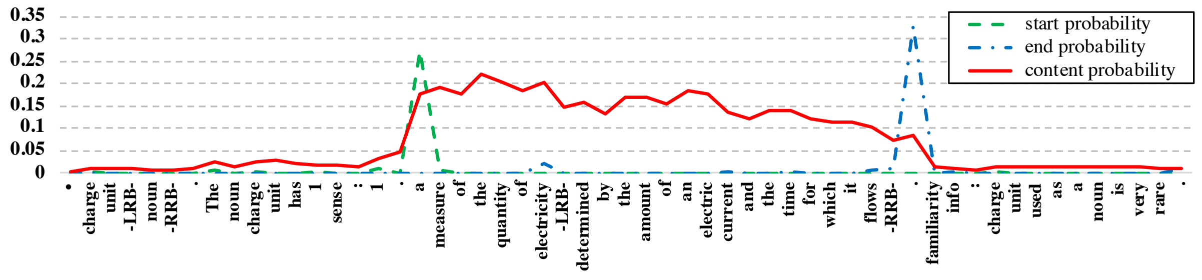


Figure 2: The boundary probabilities and content probabilities for the words in a passage

for a passage. We can see that the boundary and content probabilities capture different aspects of the answer. Since answer candidates usually have similar boundary words, if we compute the answer representation based on the boundary probabilities, it's difficult to model the real difference among different answer candidates. On the contrary, with the content probabilities, we pay more attention to the content part of the answer, which can provide more distinguishable information for verifying the correct answer. Furthermore, the content probabilities can also adjust the weights of the words within the answer span so that unimportant words (e.g. "and" and ".") get lower weights in the final answer representation. We believe that this refined representation is also good for the answer verification process.

5 Related Work

Machine reading comprehension made rapid progress in recent years, especially for single-passage MRC task, such as SQuAD (Rajpurkar et al., 2016). Mainstream studies (Seo et al., 2016; Wang and Jiang, 2016; Xiong et al., 2016) treat reading comprehension as extracting answer span from the given passage, which is usually achieved by predicting the start and end position of the answer. We implement our boundary model similarly by employing the boundary-based pointer network (Wang and Jiang, 2016). Another inspiring work is from Wang et al. (2017c), where the authors propose to match the passage against itself so that the representation can aggregate evidence from the whole passage. Our verification model adopts a similar idea. However, we collect information across passages and our attention is based on the answer representation, which is much more efficient than attention over all passages. For the model training, Xiong et al. (2017) argues that the boundary loss encourages exact answers at the cost of penalizing overlapping answers. Therefore they propose a mixed objective that incorporates rewards derived from word overlap. Our joint training approach has a similar function. By taking the content and verification loss into consideration, our model will give less loss for overlapping answers than those unmatched answers, and our loss function is totally differentiable.

Recently, we also see emerging interests in multi-passage MRC from both the academic (Dunn et al., 2017; Joshi et al., 2017) and industrial community (Nguyen et al., 2016; He et al., 2017). Early studies (Shen et al., 2017; Wang et al., 2017c) usually concat those passages and employ the same models designed for single-passage MRC. However, more and more latest studies start to design specific methods that can read multiple passages more effectively. In the aspect of passage selection, Wang et al. (2017a) introduced a pipelined approach that rank the passages first and then read the selected passages for answering questions. Tan et al. (2017) treats the passage ranking as an auxiliary task that can be trained jointly with the reading comprehension model. Actually, the target of our answer verification is very similar to that of the passage selection, while we pay more attention to the answer content and the answer verification process. Speaking of the answer verification, Wang et al. (2017b) has a similar motivation to ours. They attempt to aggregate the evidence from different passages and choose the final answer from n-best candidates. However, they implement their idea as a separate reranking step after reading comprehension, while our answer verification is a component of the whole model that can be trained end-to-end.

6 Conclusion

In this paper, we propose an end-to-end framework to tackle the multi-passage MRC task . We creatively design three different modules in our model, which can find the answer boundary, model the answer content and conduct cross-passage answer verification respectively. All these three modules can be trained with different forms of the answer labels and training them jointly can provide further improvement. The experimental results demonstrate that our model outperforms the baseline models by a large margin and achieves the state-of-the-art performance on two challenging datasets, both of which are designed for MRC on real web data.

Acknowledgments

This work is supported by the National Basic Research Program of China (973 program, No. 2014CB340505) and Baidu-Peking University Joint Project. We thank the Microsoft MS-MARCO team for evaluating our results on the anonymous test set. We also thank Ying Chen, Xuan Liu and the anonymous reviewers for their constructive criticism of the manuscript.

References

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179* .

Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073* .

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 2021–2031.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 1601–1611.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* .

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*. pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* .

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. pages 1047–1055.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R^3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023* .

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017b. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116* .

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017c. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*. pages 271–280.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. DCN+: mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106* .

Language Generation via DAG Transduction

Yajie Ye, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{yeyajie,ws,wanxiaojun}@pku.edu.cn

Abstract

A DAG automaton is a formal device for manipulating graphs. By augmenting a DAG automaton with transduction rules, a DAG transducer has potential applications in fundamental NLP tasks. In this paper, we propose a novel DAG transducer to perform graph-to-program transformation. The target structure of our transducer is a program licensed by a declarative programming language rather than linguistic structures. By executing such a program, we can easily get a surface string. Our transducer is designed especially for natural language generation (NLG) from type-logical semantic graphs. Taking Elementary Dependency Structures, a format of English Resource Semantics, as input, our NLG system achieves a BLEU-4 score of 68.07. This remarkable result demonstrates the feasibility of applying a DAG transducer to resolve NLG, as well as the effectiveness of our design.

1 Introduction

The recent years have seen an increased interest as well as rapid progress in semantic parsing and surface realization based on graph-structured semantic representations, e.g. Abstract Meaning Representation (AMR; Banarescu et al., 2013), Elementary Dependency Structure (EDS; Oepen and Lønning, 2006) and Depedendency-based Minimal Recursion Semantics (DMRS; Copestake, 2009). Still underexploited is a formal framework for manipulating graphs that parallels automata, tranducers or formal grammars for strings and trees. Two such formalisms have recently been proposed and applied for NLP. One is graph grammar, e.g. Hyperedge Replacement Grammar (HRG; Ehrig et al., 1999). The other is DAG automata, originally studied by Kamimura and Slutzki (1982) and extended by Chiang et al. (2018). In this paper, we study DAG transducers in depth, with the goal of building accurate, efficient yet robust natural language generation (NLG) systems.

The meaning representation studied in this work is what we call type-logical semantic graphs, i.e. semantic graphs grounded under type-logical semantics (Carpenter, 1997), one dominant theoretical framework for modeling natural language semantics. In this framework, adjuncts, such as adjective and adverbal phrases, are analyzed as (higher-order) functors, the function of which is to consume complex arguments (Kratzer and Heim, 1998). In the same spirit, generalized quantifiers, prepositions and function words in many languages other than English are also analyzed as higher-order functions. Accordingly, all the linguistic elements are treated as roots in type-logical semantic graphs, such as EDS and DMRS. This makes the typological structure quite flat rather than hierachical, which is an essential distinction between natural language semantics and syntax.

To the best of our knowledge, the only existing DAG transducer for NLG is the one proposed by Quernheim and Knight (2012). Quernheim and Knight introduced a DAG-to-tree transducer that can be applied to AMR-to-text generation. This transducer is designed to handle hierarchical structures with limited reentrencies, and it is unsuitable for meaning graphs transformed from type-logical semantics. Furthermore, Quernheim and Knight did not describe how to acquire graph recognition and transduction rules from linguistic data, and reported no result of practical generation. It is still unknown to what extent a DAG transducer suits realistic NLG.

The design for string and tree transducers

(Comon et al., 1997) focuses on not only the logic of the computation for a new data structure, but also the corresponding control flow. This is very similar the imperative programming paradigm: implementing algorithms with exact details in explicit steps. This design makes it very difficult to transform a type-logical semantic graph into a string, due to the fact their internal structures are highly diverse. We borrow ideas from declarative programming, another programming paradigm, which describes what a program must accomplish, rather than how to accomplish it. We propose a novel DAG transducer to perform graph-to-program transformation (§3). The input of our transducer is a semantic graph, while the output is a program licensed by a declarative programming language rather than linguistic structures. By executing such a program, we can easily get a surface string. This idea can be extended to other types of linguistic structures, e.g. syntactic trees or semantic representations of another language.

We conduct experiments on richly detailed semantic annotations licensed by English Resource Grammar (ERG; Flickinger, 2000). We introduce a principled method to derive transduction rules from DeepBank (Flickinger et al., 2012). Furthermore, we introduce a fine-to-coarse strategy to ensure that at least one sentence is generated for any input graph. Taking EDS graphs, a variable-free ERS format, as input, our NLG system achieves a BLEU-4 score of 68.07. On average, it produces more than 5 sentences in a second on an x86_64 GNU/Linux platform with two Intel Xeon E5-2620 CPUs. Since the data for experiments is newswire data, i.e. WSJ sentences from PTB (Marcus et al., 1993), the input graphs are quite large on average. The remarkable accuracy, efficiency and robustness demonstrate the feasibility of applying a DAG transducer to resolve NLG, as well as the effectiveness of our transducer design.

2 Previous Work and Challenges

2.1 Preliminaries

A node-labeled **simple** graph over alphabet Σ is a triple $G = (V, E, \ell)$, where V is a finite set of nodes, $E \subseteq V \times V$ is an finite set of edges and $\ell : V \rightarrow \Sigma$ is a labeling function. For a node $v \in V$, sets of its incoming and outgoing edges are denoted by $in(v)$ and $out(v)$ respectively. For an edge $e \in E$, its source node and target node are denoted by $src(e)$ and $tar(e)$ respectively. Generally speaking, a DAG is a **directed acyclic simple graph**. Different from trees, a DAG allows nodes to have multiple incoming edges. In this paper, we only consider DAGs that are **unordered**, **node-labeled**, **multi-rooted**[1] and **connected**.

Conceptual graphs, including AMR and EDS, are both node-labeled and edge-labeled. It seems that without edge labels, a DAG is inadequate, but this problem can be solved easily by using the strategies introduced in (Chiang et al., 2018). Take a labeled edge $\texttt{proper\_q} \xrightarrow{\texttt{BV}} \texttt{named}$ for example[2]. We can represent the same information by replacing it with two unlabeled edges and a new labeled node: $\texttt{proper\_q} \rightarrow \texttt{BV} \rightarrow \texttt{named}$.

2.2 Previous Work

DAG automata are the core engines of graph transducers (Bohnet and Wanner, 2010; Quernheim and Knight, 2012). In this work, we adopt Chiang et al. (2018)'s design and define a weighted DAG automaton as a tuple $M = \langle \Sigma, Q, \delta, \mathbb{K} \rangle$:

- Σ is an alphabet of node labels.
- Q is a finite set of states.
- $(\mathbb{K}, \oplus, \otimes, 0, 1)$ is a semiring of weights.
- $\delta : \Theta \rightarrow \mathbb{K}\backslash\{0\}$ is a weight function that assigns nonzero weights to a finite transition set Θ. Every transition $t \in \Theta$ is of the form

  $$\{q_1, \cdots, q_m\} \xrightarrow{\sigma} \{r_1, \cdots, r_n\}$$

  where q_i and r_j are states in Q. A transition t gets m states on the incoming edges of a node and puts n states on the outgoing edges. A transition that does not belong to Θ recieves a weight of zero.

A *run* of M on a DAG $D = \langle V, E, \ell \rangle$ is an edge labeling function $\rho : E \rightarrow Q$. The weight of a run ρ (denoted as $\delta'(\rho)$) is the product of all weights of local transitions:

$$\delta'(\rho) = \bigotimes_{v \in V} \delta\left(\rho(in(v)) \xrightarrow{\ell(v)} \rho(out(v))\right)$$

Here, for a function f, we use $f(\{a_1, \cdots, a_n\})$ to represent $\{f(a_1), \cdots, f(a_n)\}$. If $\mathbb{K}$ is a boolean semiring, the automata fall backs to an unweighted

[1] A node without incoming edges is called *root* and a node without outgoing edges is called *leaf*.

[2] proper_q and named are node labels, while BV is the edge label.

DAG automata or DAG acceptor. A *accepting run* or *recognition* is a run, the weight of which is 1, meaning *true*.

2.3 Challenges

The DAG automata defined above can only be used for recognition. In order to generate sentences from semantic graphs, we need DAG transducers. A DAG transducer is a DAG automata-augmented computation model for transducing well-formed DAGs to other data structures. Quernheim and Knight (2012) focused on feature structures and introduced a *DAG-to-Tree* transducer to perform graph-to-tree transformation. The input of their transducer is limited to single-rooted DAGs. When the labels of the leaves of an output tree in order are interpreted as words, this transducer can be applied to generate natural language sentences.

When applying Quernheim and Knight's DAG-to-Tree transducer on type-logic semantic graphs, e.g. ERS, there are some significant problems. First, it lacks the ability to *reverse* the direction of edges during transduction because it is difficult to keep acyclicy anymore if edge reversing is allowed. Second, it cannot handle multiple roots. But we have discussed and reached the conclusion that multi-rootedness is a necessary requirement for representing type-logical semantic graphs. It is difficult to decide which node should be the tree root during a 'top-down' transduction and it is also difficult to merge multiple unconnected nodes into one during a 'bottom-up' transduction. At the risk of oversimplifying, we argue that the function of the existing DAG-to-Tree transducer is to transform a hierachical structure into another hierarchical structure. Since the type-local semantic graphs are so flat, it is extremely difficult to adopt Quernheim and Knight's design to handle such graphs. Third, there are unconnected nodes with direct dependencies, meaning that their correpsonding surface expressions appear to be very close. The conceptual nodes `_even_x_deg` and `_steep_a_1` in Figure 4 are an example. It is extremely difficult for the DAG-to-Tree transducer to handle this situation.

3 A New DAG Transducer

3.1 Basic Idea

In this paper, we introduce a design of transducers that can perform structure transformation towards many data structures, including but not limited to trees. The basic idea is to give up the *rewritting* method to directly generate a new data structure piece by piece, while recognizing an input DAG. Instead, our transducer obtains target structures based on side effects of DAG recognition. The output of our transducer is no longer the target data structure itself, e.g. a tree or another DAG, and is now a program, i.e. a bunch of statements licensed by a particular **declarative** programming language. The target structures are constructed by executing such programs.

Since our main concern of this paper is natural language generation, we take strings, namely sequences of words, as our target structures. In this section, we introduce an extremely simple programming language for string concatenation and then details about how to leverage the power of declarative programming to perform DAG-to-string transformation.

3.2 A Declarative Programming Language

The syntax in the BNF format of our declarative programming language, denoted as $\mathcal{L}_c$, for string calculation is:

$$\begin{aligned} \langle program\rangle &::= \langle statement\rangle * \\ \langle statement\rangle &::= \langle variable\rangle = \langle expr\rangle \\ \langle expr\rangle &::= \langle variable\rangle \mid \langle string\rangle \\ &\quad \mid \langle expr\rangle + \langle expr\rangle \end{aligned}$$

Here a *string* is a sequence of characters selected from an alphabet (denoted as Σ_{out}) and can be empty (denoted as ϵ). The semantics of '=' is *value assignment*, while the semantics of '+' is *string concatenation*. The value of variables are strings. For every statement, the left hand side is a variable and the right hand side is a sequence of string literals and variables that are combined through '+'. Equation (1) presents an exmaple program licensed by this language.

$$\begin{aligned} S &= x_{21} + \texttt{want} + x_{11} \\ x_{11} &= \texttt{to} + \texttt{go} \\ x_{21} &= x_{41} + \texttt{John} \\ x_{41} &= \epsilon \end{aligned} \tag{1}$$

After solving these statements, we can query the values of all variables. In particular, we are interested in S, which is related to the desired natural language expression `John want to go`[3].

[3] The expression is a sequence of lemmas rather than inflected words. Refer to §4 for more details.

Using the relation between the variables, we can easily convert the statements in (1) to a rooted tree. The result is shown in Figure 1. This tree is significantly different from the target structures discussed by Quernheim and Knight (2012) or other normal tree transducers (Comon et al., 1997). This tree represents calculation to solve the program. Constructing such *internal* trees is an essential function of the *compiler* of our programming language.

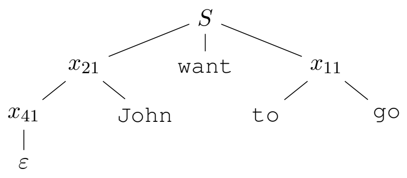


Figure 1: Variable relation tree.

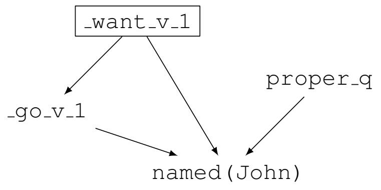


Figure 2: An input graph. The intended reading is *John wants to go*.

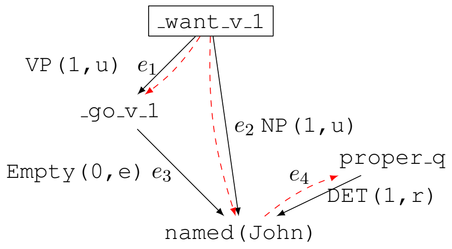


Figure 3: A run of the graph in Figure 2.

3.3 Informal Illustration

We introduce our DAG transducer using a simple example. Figure 2 shows the original input graph $D = (V, E, \ell)$. Without any loss of generality, we remove edge labels. Table 1 lists the rule set—R—for this example. Every row represents an applicable transduction rule that consists of two parts. The left column is the recognition part displayed in the form $I \xrightarrow{\sigma} O$, where I, O and σ decode the state set of incoming edges, the state set of outgoing edges and the node label respectively. The right column is the generation part which consists of (multiple) templates of statements licensed by the programming language defined in the previous section. In practice, two different rules may have a same recognition part but different generation parts.

Every state q is of the form $l(n, d)$ where l is the finite state label, n is the count of possible variables related to q, and d denotes the direction. The value of d can only be `r` (reversed), `u` (unchanged) or `e`(empty). Variable $v_{l(j,d)}$ represents the jth ($1 \le j \le n$) variable related to state q. For example, $v_{\texttt{X(2,r)}}$ means the second variable of state `X(3,r)`. There are two special variables: S, which corresponds to the whole sentence and L, which corresponds to the output string associated to current node label. It is reasonable to assume that there exists a function $\psi : \Sigma \to \Sigma^*_{\text{out}}$ that maps a particular node label, i.e. concept, to a surface string. Therefore L is determined by ψ.

Now we are ready to apply transduction rules to translate D into a string. The transduction consists of two steps:

Recognition The goal of this step is to find an edge labeling function $\rho : E \to Q$ which satisfies that for every node v, $\rho(in(v)) \xrightarrow{\ell(v)} \rho(out(v))$ matches the recognition part of a rule in R. The recognition result is shown in Figure 3. The red dashed edges in Figure 3 make up an *intermediate graph* $T(\rho)$, which is a subgraph of D if edge direction is not taken into account. Sometimes, $T(\rho)$ paralles the syntactic structure of an output sentence. For a labeling function ρ, we can construct *intermediate graph* $T(\rho)$ by checking the direction parameter of every edge state. For an edge $e = (u, v) \in E$, if the direction of $\rho(e)$ is `r`, then (v, u) is in $T(\rho)$. If the direction is `u`, then (u, v) is in $T(\rho)$. If the direction is `e`, neither (u, v) nor (v, u) is included. The recognition process is slightly different from the one in Chiang et al. (2018). Since incoming edges with an `Empty(0,e)` state carry no semantic information, they will be ignored during recognition. For example, in Figure 3, we will only use e_2 and e_4 to match transducation rules for node `named(John)`.

Instantiation We use $rule(v)$ to denotes the rule used on node v. Assume s is the generation part of $rule(v)$. For every edge e_i adjacent to v, assume $\rho(e_i) = l(n, d)$. We replace L with $\psi(\ell(v))$ and replace every occurrence of $v_{l(j,d)}$ in s with a new variable x_{ij} ($1 \le j \le n$). Then we

$Q = \{$`DET(1,r)`,`Empty(0,e)`,`VP(1,u)`,`NP(1,u)`$\}$		
Rule	For Recognition	For Generation
1	$\{\} \xrightarrow{\texttt{proper\_q}} \{\texttt{DET(1,r)}\}$	$v_{\texttt{DET(1,r)}} = \epsilon$
2	$\{\} \xrightarrow{\texttt{\_want\_v\_1}} \{\texttt{VP(1,u),NP(1,u)}\}$	$S = v_{\texttt{NP(1,u)}} + L + v_{\texttt{VP(1,u)}}$
3	$\{\texttt{VP(1,u)}\} \xrightarrow{\texttt{\_go\_v\_1}} \{\texttt{Empty(0,e)}\}$	$v_{\texttt{VP(1,u)}} = \texttt{to} + L$
4	$\{\texttt{NP(1,u),DET(1,r)}\} \xrightarrow{\texttt{named}} \{\}$	$v_{\texttt{NP(1,u)}} = v_{\texttt{DET(1,r)}} + L$

Table 1: Sets of states (Q) and rules (R) that can be used to process the graph in Figure 2.

get a newly generated expression for v. For example, node `_want_v_1` is recognized using Rule 2, so we replace $v_{\texttt{NP(1,u)}}$ with x_{21}, $v_{\texttt{VP(1,u)}}$ with x_{11} and L with `want`. After instantiation, we get all the statements in Equation (1).

Our transducer is suitable for type-logical semantic graphs. Because declarative programming brings in more freedom for graph transduction. We can arrange the variables in almost any order without regard to the edge directions in original graphs. Meanwhile, the multi-rooted problem can be solved easily because the generation is based on side effects. We do not need to decide which node is the tree root.

3.4 Definition

The formal definition of our DAG transducer described above is a tuple $M = (\Sigma, Q, R, w, V, S)$ where:

- Σ is an alphabet of node labels.
- Q is a finite set of edge states. Every state $q \in Q$ is of the form $l(n, d)$ where l is the state label, n is the variable count and d is the direction of state which can be `r`, `u` or `e`.
- R is a finite set of rules. Every rule is of the form $I \xrightarrow{\sigma} \langle O, E\rangle$. E can be any kind of statement in a declarative programming language. It is called the generation part. I, σ and O have the same meanings as they do in the previous section and they are called the recognition part.
- w is a score function. Given a particular run and an anchor node, w assigns a score to measure the preference for a particular rule at this anchor node.
- V is the set of parameterized variables that can be used in every expression.
- $S \in V$ is a distinguished, global variable. It is like the 'goal' of a program.

4 DAG Transduction-based NLG

Different languages exhibit different morpho-syntactic and syntactico-semantic properties. For example, Russian and Arabic are morphologically-rich languages and heavily utilize grammatical markers to indicate grammatical as well as semantic functions. On the contrary, Chinese, as an analytic language, encodes grammatical and semantic information in a highly configurational rather than either inflectional or derivational way. Such differences affects NLG significantly. Considering generating Chinese sentences, it seems sufficient to employ our DAG transducer to obtain a sequence of lemmas, since no morpholical production is needed. But for morphologically-rich languages, we do need to model complex morphological changes.

To unify a general framework for DAG transduction-based NLG, we propose a two-step strategy to achive meaning-to-text transformation.

- In the first phase, we are concerned with syntactico-semantic properties and utilize our DAG transducer to translate a semantic graph into sequential lemmas. Information such as tense, apsects, gender, etc. is attached to anchor lemmas. Actually, our transducer generates "`want.PRES`" rather than "`wants`". Here, "PRES" indicates a particular tense.
- In the second phase, we are concerned with morpho-syntactic properties and utilize a neural sequence-to-sequence model to obtain final surface strings from the outputs of the DAG transducer.

5 Inducing Transduction Rules

We present an empirical study on the feasibility of DAG transduction-based NLG. We focus on

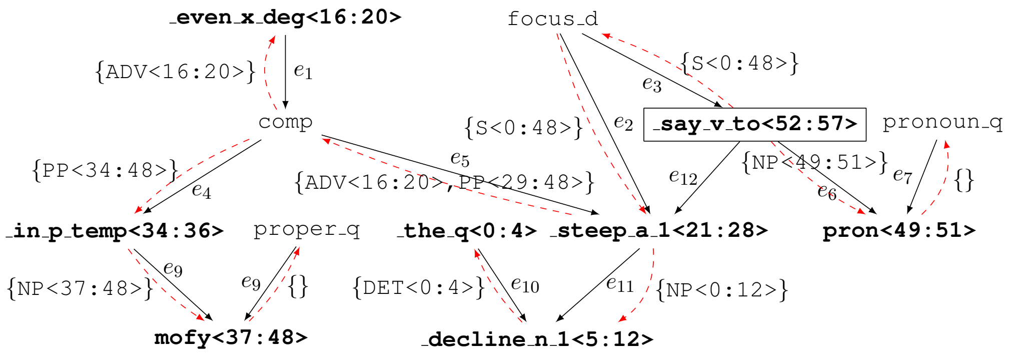


Figure 4: An example graph. The intended reading is *"the decline is even steeper than in September", he said.* Original edge labels are removed for clarity. Every edge is associated with a span list, and spans are written in the form `label<begin:end>`. The red dashed edges belong to the intermediate graph T.

variable-free MRS representations, namely EDS (Oepen and Lønning, 2006). The data set used in this work is DeepBank 1.1 (Flickinger et al., 2012).

5.1 EDS-specific Constraints

In order to generate reasonable strings, three constraints must be kept during transduction. First, for a rule $I \xrightarrow{\sigma} \langle O, E\rangle$, a state with direction `u` in I or a state with direction `r` in O is called *head state* and its variables are called *head variables*. For example, the head state of rule 3 in Table 1 is `VP(1,u)` and the head state of rule 2 is `DET(1,r)`. There is at most one head state in a rule and only head variables or S can be the left sides of statements. If there is no head state, we assign the global S as its head. Otherwise, the number of statements is equal to the number of head variables and each statement has a distinguished left side variable. An empty state does not have any variables. Second, every rule has no-copying, no-deleting statements. In other words, all variables must be used exactly once in a statement. Third, during recognition, a labeling function ρ is valid only if $T(\rho)$ is a rooted tree.

After transduction, we get result ρ^*. The first and second constraints ensure that for all nodes, there is at most one incoming red dashed edge in $T(\rho^*)$ and 'data' carried by variables of the only incoming red dashed edge or S is separated into variables of outgoing red dashed edges. The last constraint ensures that we can solve all statements by a bottom-up process on tree $T(\rho^*)$.

5.2 Fine-to-Coarse Transduction

Almost all NLG systems that heavily utilize a symbolic system to encode deep syntactico-semantic information lack some robustness, meaning that some input graphs may not be successfully processed. There are two reasons: (1) some explicit linguistic constraints are not included; (2) exact decoding is too time-consuming while inexact decoding cannot cover the whole search space. To solve the robustness problem, we introduce a fine-to-coarse strategy to ensure that at least one sentence is generated for any input graph. There are three types of rules in our system, namely *induced rules*, *extended rules* and *dynamic rules*. The most fine-grained rules are applied to bring us precision, while the most coarse-grained rules are for robustness.

In order to extract reasonable rules, we will use both EDS graphs and the corresponding derivation trees provided by ERG. The details will be described step-by-step in the following sections.

5.3 Induced Rules

Figure 4 shows an example for obtaining induced rules. The *induced rules* are directly obtained by following three steps:

Finding intermediate tree T EDS graphs are highly regular semantic graphs. It is not difficult to generate T based on a highly customized 'breadth-first' search. The generation starts from the 'top' node (`_say_v_to` in Figure 4) given by the EDS graph and traverse the whole graph. No more than thirty heuristic rules are used to decide the visiting order of nodes.

Assigning states EDS graphs also provide span information for nodes. We select a group of *lexical* nodes which have corresponding substrings in the original sentence. In Figure 4, these nodes are in bold font and directly followed by a span. Then we merge spans from the bottom of T to the top to assign each red edge a span list. For each node n in T, we collect spans of every outgoing dashed edge of n into a list s. Some additional spans may be inserted into s. These spans do not occur in the EDS graph but they do occur in the sentence, i.e. `than<29:33>`. Then we merge continuous spans in s and assign the remaining spans in s to the incoming red dashed edge of n. We apply a similar method to the derivation tree. As a result, every inner node of the derivation tree is associated with a span. Then we align the edges in T to nodes of the inner derivation tree by comparing their spans. Finally edge labels in Figure 4 are generated.

We use the concatenation of the edge labels in a span list as the state label. The edge labels are joined in order with '_'. `Empty(0,e)` is the state of the edges that do not belong to T (ignoring direction), such as e_{12}. The variable count of a state is equal to the size of the span list and the direction of a state is decided by whether the edge in T related to the state and its corresponding edge in D have different directions. For example, the state of e_5 should be `ADV_PP(2,r)`.

Generating statements After the above two steps, we are ready to generate statements according to how spans are merged. For all nodes, spans of the incoming edges represent the left hand side and the outgoing edges represent the right hand side. For example, the rule for node `comp` will be:

$$\{\texttt{ADV(1,r)}\} \xrightarrow{\texttt{comp}} \{\texttt{PP(1,u)}, \texttt{ADV\_PP(2,r)}\}$$

$$v_{\texttt{ADV\_PP(1,r)}} = v_{\texttt{ADV(1,r)}}$$

$$v_{\texttt{ADV\_PP(2,r)}} = \texttt{than} + v_{\texttt{PP(1,u)}}$$

5.4 Extended Rules

Extended rules are used when no induced rules can cover a given node. In theory, there can be unlimited modifier nodes pointing to a given node, such as `PP` and `ADJ`. We use some manually written rules to slightly change an induced rule (prototype) by addition or deletion to generate a group of extended rules. The motivation here is to deal with the data sparseness problem.

For a group of selected non-head states in I, such as `PP` and `ADJ`. We can produce new rules by removing or duplicating more of them. For example:

$$\{\texttt{NP(1,u)},\texttt{ADJ(1,r)}\} \xrightarrow{\texttt{\_X\_n\_1}} \{\}$$
$$v_{\texttt{NP(1,u)}} = v_{\texttt{ADJ(1,r)}} + L$$

As a result, we get the two rules below:

$$\{\texttt{NP(1,u)}\} \xrightarrow{\texttt{\_X\_n\_1}} \{\} \quad v_{\texttt{NP(1,u)}} = L$$

$$\{\texttt{NP(1,u)},\texttt{ADJ(1,r)}_1, \texttt{ADJ(1,r)}_2\} \xrightarrow{\texttt{\_X\_n\_1}} \{\}$$
$$v_{\texttt{NP(1,u)}} = v_{\texttt{ADJ(1,r)}_1} + v_{\texttt{ADJ(1,r)}_2} + L$$

5.5 Dynamic Rules

During decoding, when neither induced nor extended rule is applicable, we *create* a dynamic rule on-the-fly. Our rule creator builds a new rule following the Markov assumption:

$$P(O|C) = P(q_1|C)\prod_{i=2}^{n} P(q_i|C)P(q_i|q_{i-1}, C)$$

$C = \langle I, D\rangle$ represents the context. $O = \{q_1, \cdots, q_n\}$ denotes the outgoing states and I, D have the same meaning as before. Though they are unordered multisets, we can give them an explicit alphabet order by their edge labels. There is also a group of hard constraints to make sure that the predicted rules are well-formed as the definition in §5 requires. This Markovization strategy is widely utilized by lexicalized and unlexicalized PCFG parsers (Collins, 1997; Klein and Manning, 2003). For a dynamic rule, all variables in this rule will appear in the statement. We use a simple perceptron-based scorer to assign every variable a score and arrange them in an decreasing order.

6 Evaluation and Analysis

6.1 Set-up

We use DeepBank 1.1 (Flickinger et al., 2012), i.e. gold-standard ERS annotations, as our main experimental data set to train a DAG transducer as well as a sequence-to-sequence morpholyzer, and wikiwoods (Flickinger et al., 2010), i.e. automatically-generated ERS annotations by ERG, as additional data set to enhance the sequence-to-sequence morpholyzer. The training, development and test data sets are from DeepBank and split according to DeepBank's recommendation. There are 34,505, 1,758 and 1,444 sentences (all disconnected graphs as well as their associated sentences are removed) in the training, development and test data sets. We use a small portion of wikiwoods data, c.a. 300K sentences, for experiments.

37,537 induced rules are directly extracted from the training data set, and 447,602 extended rules are obtained. For DAG recognition, at one particular position, there may be more than one rule applicable. In this case, we need a disambiguation model as well as a decoder to search for a globally optimal solution. In this work, we train a structured perceptron model (Collins, 2002) for disambiguation and employ a beam decoder. The perceptron model used by our dynamic rule generator are trained with the induced rules. To get a sequence-to-sequence model, we use the open source tool—OpenNMT[4].

6.2 The Decoder

We implement a fine-to-coarse beam search decoder. Given a DAG D, our goal is to find the highest scored labeling function ρ:

$$\rho = \arg\max_{\rho} \prod_{i=1}^{n} \sum_{j} w_j \cdot f_j(rule(v_i), D)$$

$$\text{s.t.} \quad rule(v_i) = \rho(in(v_i)) \xrightarrow{\ell(v_i)} \langle \rho(out(v_i)), E_i \rangle$$

where n is the node count and $f_j(\cdot,\cdot)$ and w_j represent a feature and the corresponding weight, respectively. The features are chosen from the context of the given node v_i. We perform 'top-down' search to translate an input DAG into a morphology-function-enhanced lemma sequence. Each hypothesis consists of the current DAG graph, the partial labeling function, the current hypothesis score and other graph information used to perform rule selection. The decoder will keep the corresponding partial intermediate graph T acyclic when decoding. The algorithm used by our decoder is displayed in Algorithm 1. Function `FindRules`(h, n, R) will use hard constraints to select rules from the rule set R according to the contextual information. It will also perform an acyclic check on T. Function `Insert`(h, r, n, B) will create and score a new hypothesis made from the given context and then insert it into beam B.

[4]https://github.com/OpenNMT/OpenNMT/

After we get the edge labeling function ρ, we use a simple linear equation solver to convert all statements to a sequence of lemmas.

Algorithm 1: Algorithm for our decoder.

```
Input: $D$ is the EDS graph. $R_I$ and $R_E$
       are induced-rules and
       extended-rules respectively.
Result: The edge labeling function $\rho$.
1  $Q \leftarrow$ all the roots in $D$
2  $B1 \leftarrow$ empty beam
3  $E \leftarrow \varnothing$
4  Insert initial hypothesis into $B1$
5  while $Q$ is not empty:
6      $B2 \leftarrow$ empty beam
7      $n \leftarrow$ dequeue a node from $Q$
8      for $h \in B1$:
9          $rules \leftarrow$ FindRules$(h, n, R_I)$
10         if $rules$ is not empty:
11             for $r \in rules$:
12                 Insert$(h, r, n, B2)$
           else:
13             $rules \leftarrow$ FindRules$(h, n, R_E)$
14             for $r \in rules$:
15                 Insert$(h, r, n, B2)$
16     if $B2$ is still empty:
17         for $h \in B1$:
18             $r \leftarrow$ RuleGenerator$(h, n)$
19             Insert$(h, r, n, B2)$
20     $B1 \leftarrow B2$
21     for $e \in out(n)$:
22         $E \leftarrow E \cup \{e\}$
23         if $in(tar(e)) \subseteq E$:
24             $Q \leftarrow Q \cup \{tar(e)\}$
25 Extract $\rho$ from best hypothesis in $B1$
```

6.3 Accuracy

In order to evaluate the effectiveness of our transducer for NLG, we try a group of tests showed in Table 2. All sequence-to-sequence models (either from lemma sequences to lemma sequences or lemma sequences to sentences) are trained on DeepBank and wikiwoods data set and tuned on the development data. The second column shows the BLEU-4 scores between generated lemma sequences and golden sequences of lemmas. The third column shows the BLEU-4 scores between generated sentences and golden sentences. The fourth column shows the fraction of graphs in the test data set that can reach output sentences.

Transducer	Lemmas	Sentences	Coverage
I	89.44	74.94	67%
I+E	88.41	74.03	77%
I+E+D	82.04	68.07	100%
DFS-NN	50.45		100%
AMR-NN		33.8	100%
AMR-NRG		25.62	100%

Table 2: Accuracy (BLEU-4 score) and coverage of different systems. *I* denotes transduction only using induced rules; *I+E* denotes transduction using both induced and extended rules; *I+E+D* denotes transduction using all kinds of rules. *DFS-NN* is a rough implementation of Konstas et al. (2017) but with the EDS data, while *AMR-NN* includes the results originally reported by Konstas et al., which are evaluated on the AMR data. *AMR-NRG* includes the results obtained by a synchronous graph grammar (Song et al., 2017).

The graphs that cannot received any natural language sentences are removed while conducting the BLEU evaluation.

As we can conclude from Table 2, using only induced rules achieves the highest accuracy but the coverage is not satisfactory. Extended rules lead to a slight accuracy drop but with a great improvement of coverage (c.a. 10%). Using dynamic rules, we observe a significant accuracy drop. Nevertheless, we are able to handle all EDS graphs. The full-coverage robustness may benefit many NLP applications. The lemma sequences generated by our transducer are really close to the golden one. This means that our model actually works and most reordering patterns are handled well by induced rules.

Compared to the AMR generation task, our transducer on EDS graphs achieves much higher accuracies. To make clear how much improvement is from the data and how much is from our DAG transducer, we implement a purely neural baseline. The baseline converts a DAG into a concept sequence by a pre-order DFS traversal on the intermediate tree of this DAG. Then we use a sequence-to-sequence model to transform this concept sequence to the lemma sequence for comparison. This is a kind of implementation of Konstas et al.'s model but evaluated on the EDS data. We can see that on this task, our transducer is much better than a pure sequence-to-sequence model on DeepBank data.

Transducer	Average (s)	Maximal (s)
I	0.090	0.40
I+E	0.093	0.46
I+E+D	0.18	3.2

Table 3: Efficiency of our NL generator.

6.4 Efficiency

Table 3 shows the efficiency of the beam search decoder with a beam size of 128. The platform for this experiment is x86_64 GNU/Linux with two Intel Xeon E5-2620 CPUs. The second and third columns represent the average and the maximal time (in seconds) to translate an EDS graph. Using dynamic rules slow down the decoder to a great degree. Since the data for experiments is newswire data, i.e. WSJ sentences from PTB (Marcus et al., 1993), the input graphs are quite large on average. On average, it produces more than 5 sentences per second on CPU. We consider this a promising speed.

7 Conclusion

We extend the work on DAG automata in Chiang et al. (2018) and propose a general method to build flexible DAG transducer. The key idea is to leverage a declarative programming language to minimize the computation burden of a graph transducer. We think may NLP tasks that involve graph manipulation may benefit from this design. To exemplify our design, we develop a practical system for the semantic-graph-to-string task. Our system is accurate (BLEU 68.07), efficient (more than 5 sentences per second on a CPU) and robust (full-coverage). The empirical evaluation confirms the usefulness a DAG transducer to resolve NLG, as well as the effectiveness of our design.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61772036, 61331011) and the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Weiwei Sun is the corresponding author.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Bernd Bohnet and Leo Wanner. 2010. Open soucre graph transducer interpreter and grammar development environment. In *LREC*.

B. Carpenter. 1997. *Type-Logical Semantics*. Bradford books. MIT Press.

David Chiang, Frank Drewes, Daniel Gildea, Adam Lopez, and Giorgio Satta. 2018. Weighted DAG automata for semantic graphs. *Computational Linguistics*. To appear.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics.

Hubert Comon, Max Dauchet, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 1997. Tree automata techniques and applications. Technical report.

Ann Copestake. 2009. ***Invited Talk:*** slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece. Association for Computational Linguistics.

H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. 1999. *Handbook of Graph Grammars and Computing by Graph Transformation: Vol. 3: Concurrency, Parallelism, and Distribution*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Nat. Lang. Eng.*, 6(1):15–28.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestl. 2010. Wikiwoods: Syntacto-semantic annotation for English wikipedia. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pages 85–96.

Tsutomu Kamimura and Giora Slutzki. 1982. Transductions of dags and trees. *Mathematical Systems Theory*, 15(3):225–249.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Angelika Kratzer and Irene Heim. 1998. *Semantics in generative grammar*. Blackwell Oxford.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy. European Language Resources Association (ELRA). ACL Anthology Identifier: L06-1214.

Daniel Quernheim and Kevin Knight. 2012. Towards probabilistic acceptors and transducers for feature structures. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-6 '12, pages 76–85, Stroudsburg, PA, USA. Association for Computational Linguistics.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13, Vancouver, Canada. Association for Computational Linguistics.

A Distributional and Orthographic Aggregation Model for English Derivational Morphology

Daniel Deutsch*, **John Hewitt*** **and Dan Roth**
Department of Computer and Information Science
University of Pennsylvania
{ddeutsch,johnhew,danroth}@seas.upenn.edu

Abstract

Modeling derivational morphology to generate words with particular semantics is useful in many text generation tasks, such as machine translation or abstractive question answering. In this work, we tackle the task of derived word generation. That is, given the word "run," we attempt to generate the word "runner" for "someone who runs." We identify two key problems in generating derived words from root words and transformations: suffix ambiguity and orthographic irregularity. We contribute a novel aggregation model of derived word generation that learns derivational transformations both as orthographic functions using sequence-to-sequence models and as functions in distributional word embedding space. Our best open-vocabulary model, which can generate novel words, and our best closed-vocabulary model, show 22% and 37% relative error reductions over current state-of-the-art systems on the same dataset.

1 Introduction

The explicit modeling of morphology has been shown to improve a number of tasks (Seeker and Çetinoglu, 2015; Luong et al., 2013). In a large number of the world's languages, many words are composed through morphological operations on subword units. Some languages are rich in *inflectional* morphology, characterized by syntactic transformations like pluralization. Similarly, languages like English are rich in *derivational* morphology, where the semantics of words are composed from smaller parts. The AGENT derivational transformation, for example, answers the question, *what is the word for 'someone who runs'?* with the answer, a *runner*.[1] Here, AGENT is spelled out as suffixing *-ner* onto the root verb *run*.

*These authors contributed equally; listed alphabetically.

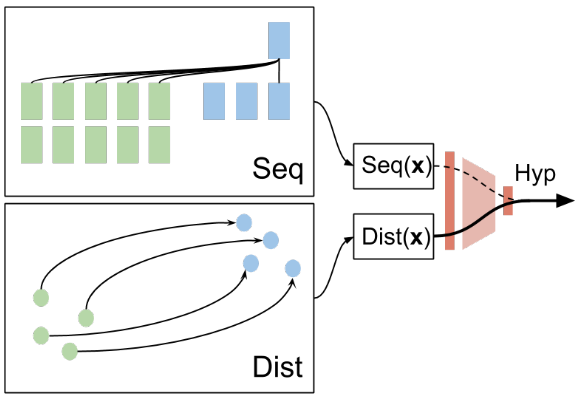


Figure 1: Diagram depicting the flow of our aggregation model. Two models generate a hypothesis according to orthogonal information; then one is chosen as the final model generation. Here, the hypothesis from the distributional model is chosen.

We tackle the task of derived word generation. In this task, a root word $\mathbf{x}$ and a derivational transformation t are given to the learner. The learner's job is to produce the result of the transformation on the root word, called the derived word $\mathbf{y}$. Table 1 gives examples of these transformations.

Previous approaches to derived word generation model the task as a character-level sequence-to-sequence (seq2seq) problem (Cotterell et al., 2017b). The letters from the root word and some encoding of the transformation are given as input to a neural encoder, and the decoder is trained to produce the derived word, one letter at a time. We identify the following problems with these approaches:

First, because these models are unconstrained, they can generate sequences of characters that do

[1]We use the verb *run* as a demonstrative example; the transformation can be applied to most verbs.

$\mathbf{x}$	t		$\mathbf{y}$
wise	ADVERB	→	wise*ly*
simulate	RESULT	→	simula*tion*
approve	RESULT	→	approv*al*
overstate	RESULT	→	overstate*ment*
yodel	AGENT	→	yodel*er*
survive	AGENT	→	surviv*or*
intense	NOMINAL	→	intens*ity*
effective	NOMINAL	→	effective*ness*
pessimistic	NOMINAL	→	pessim*ism*

Table 1: The goal of derived word generation is to produce the derived word, $\mathbf{y}$, given both the root word, $\mathbf{x}$, and the transformation t, as demonstrated here with examples from the dataset.

not form actual words. We argue that requiring the model to generate a known word is a reasonable constraint in the special case of English derivational morphology, and doing so avoids a large number of common errors.

Second, sequence-based models can only generalize string manipulations (such as "add *-ment*") if they appear frequently in the training data. Because of this, they are unable to generate derived words that do not follow typical patterns, such as generating *truth* as the nominative derivation of *true*. We propose to learn a function for each transformation in a low dimensional vector space that corresponds to mapping from representations of the root word to the derived word. This eliminates the reliance on orthographic information, unlike related approaches to distributional semantics, which operate at the suffix level (Gupta et al., 2017).

We contribute an aggregation model of derived word generation that produces hypotheses independently from two separate learned models: one from a seq2seq model with only orthographic information, and one from a feed-forward network using only distributional semantic information in the form of pretrained word vectors. The model learns to choose between the hypotheses according to the relative confidence of each. This system can be interpreted as learning to decide between positing an orthographically regular form or a semantically salient word. See Figure 1 for a diagram of our model.

We show that this model helps with two open problems with current state-of-the-art seq2seq derived word generation systems, suffix ambiguity and orthographic irregularity (Section 2). We also improve the accuracy of seq2seq-only derived word systems by adding external information through constrained decoding and hypothesis rescoring. These methods provide orthogonal gains to our main contribution.

We evaluate models in two categories: open vocabulary models that can generate novel words unattested in a preset vocabulary, and closed-vocabulary models, which cannot. Our best open-vocabulary and closed-vocabulary models demonstrate 22% and 37% relative error reductions over the current state of the art.

2 Background: Derivational Morphology

Derivational transformations generate novel words that are semantically composed from the root word and the transformation. We identify two unsolved problems in derived word transformation, each of which we address in Sections 3 and 4.

First, many plausible choices of suffix for a single pair of root word and transformation. For example, for the verb *ground*, the RESULT transformation could plausibly take as many forms as[2]

$$\begin{aligned}
&(\text{ground}, \textsc{Result}) \to \text{ground}\textit{ing}\\
&(\text{ground}, \textsc{Result}) \to {}^*\text{ground}\textit{ation}\\
&(\text{ground}, \textsc{Result}) \to {}^*\text{ground}\textit{ment}\\
&(\text{ground}, \textsc{Result}) \to {}^*\text{ground}\textit{al}
\end{aligned}$$

However, only one is correct, even though each suffix appears often in the RESULT transformation of other words. We will refer to this problem as "suffix ambiguity."

Second, many derived words seem to lack a generalizable orthographic relationship to their root words. For example, the RESULT of the verb *speak* is *speech*. It is unlikely, given an orthographically similar verb *creak*, that the RESULT be *creech* instead of, say, *creaking*. Seq2seq models must grapple with the problem of derived words that are the result of unlikely or potentially unseen string transformations. We refer to this problem as "orthographic irregularity."

3 Sequence Models and Corpus Knowledge

In this section, we introduce the prior state-of-the-art model, which serves as our baseline system. Then we build on top of this system by incorporating a dictionary constraint and rescoring the

[2]The * indicates a non-word.

model's hypotheses with token frequency information to address the suffix ambiguity problem.

3.1 Baseline Architecture

We begin by formalizing the problem and defining some notation. For source word $\mathbf{x} = x_1, x_2, \ldots x_m$, a derivational transformation t, and target word $\mathbf{y} = y_1, y_2, \ldots y_n$, our goal is to learn some function from the pair $(\mathbf{x}, t)$ to $\mathbf{y}$. Here, x_i and y_j are the ith and jth characters of the input strings $\mathbf{x}$ and $\mathbf{y}$. We will sometimes use $\mathbf{x}_{1:i}$ to denote $x_1, x_2, \ldots x_i$, and similarly for $\mathbf{y}_{1:j}$.

The current state-of-the-art model for derived-form generation approaches this problem by learning a character-level encoder-decoder neural network with an attention mechanism (Cotterell et al., 2017b; Bahdanau et al., 2014).

The input to the bidirectional LSTM encoder (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) is the sequence #, $x_1, x_2, \ldots x_m$, #, t, where # is a special symbol to denote the start and end of a word, and the encoding of the derivational transformation t is concatenated to the input characters. The model is trained to minimize the cross entropy of the training data. We refer to our reimplementation of this model as SEQ.

For a more detailed treatment of neural sequence-to-sequence models with attention, we direct the reader to Luong et al. (2015).

3.2 Dictionary Constraint

The suffix ambiguity problem poses challenges for models which rely exclusively on input characters for information. As previously demonstrated, words derived via the same transformation may take different suffixes, and it is hard to select among them based on character information alone. Here, we describe a process for restricting our inference procedure to only generate known English words, which we call a dictionary constraint. We believe that for English morphology, a large enough corpus will contain the vast majority of derived forms, so while this approach is somewhat restricting, it removes a significant amount of ambiguity from the problem.

To describe how we implemented this dictionary constraint, it is useful first to discuss how decoding in a seq2seq model is equivalent to solving a shortest path problem. The notation is specific to our model, but the argument is applicable to seq2seq models in general.

The goal of decoding is to find the most probable structure $\hat{\mathbf{y}}$ conditioned on some observation $\mathbf{x}$ and transformation t. That is, the problem is to solve

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} \mid \mathbf{x}, t) \tag{1}$$

$$= \arg\min_{\mathbf{y} \in \mathcal{Y}} -\log p(\mathbf{y} \mid \mathbf{x}, t) \tag{2}$$

where $\mathcal{Y}$ is the set of valid structures. Sequential models have a natural ordering $\mathbf{y} = y_1, y_2, \ldots y_n$ over which $-\log p(\mathbf{y} \mid \mathbf{x}, t)$ can be decomposed

$$-\log p(\mathbf{y} \mid \mathbf{x}, t) = \sum_{t=1}^{n} -\log p(y_t \mid \mathbf{y}_{1:t-1}, \mathbf{x}, t) \tag{3}$$

Solving Equation 2 can be viewed as solving a shortest path problem from a special starting state to a special ending state via some path which uniquely represents $\mathbf{y}$. Each vertex in the graph represents some sequence $\mathbf{y}_{1:i}$, and the weight of the edge from $\mathbf{y}_{1:i}$ to $\mathbf{y}_{1:i+1}$ is given by

$$-\log p(y_{i+1} \mid \mathbf{y}_{1:i-1}, \mathbf{x}, t) \tag{4}$$

The weight of the path from the start state to the end state via the unique path that describes $\mathbf{y}$ is exactly equal to Equation 3. When the vocabulary size is too large, the exact shortest path is intractable, and approximate search methods, such as beam search, are used instead.

In derived word generation, $\mathcal{Y}$ is an infinite set of strings. Since $\mathcal{Y}$ is unrestricted, almost all of the strings in $\mathcal{Y}$ are not valid words. Given a dictionary $\mathcal{Y}_{\mathcal{D}}$, the search space is restricted to only those words in the dictionary by searching over the trie induced from $\mathcal{Y}_{\mathcal{D}}$, which is a subgraph of the unrestricted graph. By limiting the search space to $\mathcal{Y}_{\mathcal{D}}$, the decoder is guaranteed to generate some known word. Models which use this dictionary-constrained inference procedure will be labeled with +DICT. Algorithm 1 has the pseudocode for our decoding procedure.

We discuss specific details of the search procedure and interesting observations of the search space in Section 6. Section 5.2 describes how we obtained the dictionary of valid words.

3.3 Word Frequency Knowledge through Rescoring

We also consider the inclusion of explicit word frequency information to help solve suffix ambiguity, using the intuition that "real" derived words are likely to be frequently attested. This permits a high-recall, potentially noisy dictionary.

We are motivated by very high top-10 accuracy compared to top-1 accuracy, even among dictionary-constrained models. By rescoring the hypotheses of a model using word frequency (a word-global signal) as a feature, attempt to recover a portion of this top-10 accuracy.

When a model has been trained, we query it for its top-10 most likely hypotheses. The union of all hypotheses for a subset of the training observations forms the training set for a classifier that learns to predict whether a hypothesis generated by the model is correct. Each hypothesis is labelled with its correctness, a value in $\{\pm 1\}$. We train a simple combination of two scores: the seq2seq model score for the hypothesis, and the log of the word frequency of the hypothesis.

To permit a nonlinear combination of word frequency and model score, we train a small multilayer perceptron with the model score and the frequency of a derived word hypothesis as features.

At testing time, the 10 hypotheses generated by a single seq2seq model for a single observation are rescored. The new model top-1 hypothesis, then, is the argmax over the 10 hypotheses according to the rescorer. In this way, we are able to incorporate word-global information, e.g. word frequency, that is ill-suited for incorporation at each character prediction step of the seq2seq model. We label models that are rescored in this way +FREQ.

4 Distributional Models

So far, we have presented models that learn derivational transformations as orthographic operations. Such models struggle by construction with the orthographic irregularity problem, as they are trained to generalize orthographic information. However, the semantic relationships between root words and derived words are the same even when the orthography is dissimilar. It is salient, for example, that irregular word *speech* is related to its root *speak* in about the same way as how *exploration* is related to the word *explore*.

We model distributional transformations as functions in dense distributional word embedding spaces, crucially learning a function per derivational transformation, not per suffix pair. In this way, we aim to explicitly model the semantic transformation, not the othographic information.

4.1 Feed-forward derivational transformations

For all source words $\mathbf{x}$ and all target words $\mathbf{y}$, we look up static distributional embeddings $v_{\mathbf{x}}, v_{\mathbf{y}} \in \mathbb{R}^d$. For each derivational transformation t, we learn a function $f_t : \mathbb{R}^d \to \mathbb{R}^d$ that maps $v_{\mathbf{x}}$ to $v_{\mathbf{y}}$. f_t is parametrized as two-layer perceptron, trained using a squared loss,

$$\mathcal{L} = \mathbf{b}^{\mathrm{T}}\mathbf{b} \tag{5}$$

$$\mathbf{b} = f_t(v_{\mathbf{x}}) - v_{\mathbf{y}} \tag{6}$$

We perform inference by nearest neighbor search in the embedding space. This inference strategy requires a subset of strings for our embedding dictionary, $\mathcal{Y}_V$.

Upon receiving $(\mathbf{x}, t)$ at test time, we compute $f_t(v_{\mathbf{x}})$ and find the most similar embeddings in $\mathcal{Y}_V$. Specifically, we find the top-k most similar embeddings, and take the most similar derived word that starts with the same 4 letters as the root word, and is not identical to it. This heuristic filters out highly implausible hypotheses.

We use the single-word subset of the Google News vectors (Mikolov et al., 2013) as $\mathcal{Y}_V$, so the size of the vocabulary is 929k words.

4.2 SEQ and DIST Aggregation

The seq2seq and distributional models we have presented learn with disjoint information to solve separate problems. We leverage this intuition to build a model that chooses, for each observation, whether to generate according to orthographic information via the SEQ model, or produce a potentially irregular form via the DIST model.

To train this model, we use a held-out portion of the training set, and filter it to only observations for which exactly one of the two models produces the correct derived form. Finally, we make the strong assumption that the probability of a derived form being generated correctly according to 1 model as opposed to the other is dependent only on the unnormalized model score from each. We model this as a logistic regression (t is omitted for clarity):

$$P(\cdot|\mathbf{y_D}, \mathbf{y_S}, \mathbf{x}) = \\ \mathrm{softmax}(\mathbf{W}_e \left[\textsc{Dist}(\mathbf{y_D}|\mathbf{x}); \textsc{Seq}(\mathbf{y_S}|\mathbf{x})\right] + \mathbf{b}_e)$$

where $\mathbf{W_e}$ and $\mathbf{b_e}$ are learned parameters, $\mathbf{y_D}$ and $\mathbf{y_S}$ are the hypotheses of the distributional and seq2seq models, and $\textsc{Dist}(\cdot)$ and $\textsc{Seq}(\cdot)$ are the models' likelihood functions. We denote this aggregate AGGR in our results.

5 Datasets

In this section we describe the derivational morphology dataset used in our experiments and how we collected the dictionary and token frequencies used in the dictionary constraint and rescorer.

5.1 Derivational Morphology

In our experiments, we use the derived word generation derivational morphology dataset released in Cotterell et al. (2017b). The dataset, derived from NomBank (Meyers et al., 2004) , consists of 4,222 training, 905 validation, and 905 test triples of the form $(\mathbf{x}, t, \mathbf{y})$. The transformations are from the following categories: ADVERB (ADJ $\rightarrow$ ADV), RESULT (V $\rightarrow$ N), AGENT (V $\rightarrow$ N), and NOMINAL (ADJ $\rightarrow$ N). Examples from the dataset can be found in Table 1.

5.2 Dictionary and Token Frequency Statistics

The dictionary and token frequency statistics used in the dictionary constraint and frequency reranking come from the Google Books NGram corpus (Michel et al., 2011). The unigram frequency counts were aggregated across years, and any tokens which appear fewer than approximately 2,000 times, do not end in a known possible suffix, or contain a character outside of our vocabulary were removed.

The frequency threshold was determined using development data, optimizing for high recall. We collect a set of known suffixes from the training data by removing the longest common prefix between the source and target words from the target word. The result is a dictionary with frequency information for around 360k words, which covers 98% of the target words in the training data.[3]

6 Inference Procedure Discussion

In many sequence models where the vocabulary size is large, exact inference by finding the true shortest path in the graph discussed in Section 3.2 is intractable. As a result, approximate inference techniques such as beam search are often used, or the size of the search space is reduced, for example, by using a Markov assumption. We, however, observed that exact inference via a shortest path algorithm is not only tractable in our model, but only slightly more expensive than greedy search and significantly less expensive than beam search.

[3] The remaining 2% is mostly words with hyphens or mistakes in the dataset.

Method	Accuracy	Avg. #States
GREEDY	75.9	11.8
BEAM	76.2	101.2
SHORTEST	76.2	11.8
DICT+GREEDY	77.2	11.7
DICT+BEAM	82.6	91.2
DICT+SHORTEST	82.6	12.4

Table 2: The average accuracies and number of states explored in the search graph of 30 runs of the SEQ model with various search procedures. The BEAM models use a beam size of 10.

To quantify this claim, we measured the accuracy and number of states explored by greedy search, beam search, and shortest path with and without a dictionary constraint on the development data. Table 2 shows the results averaged over 30 runs. As expected, beam search and shortest path have higher accuracies than greedy search and explore more of the search space. Surprisingly, beam search and shortest path have nearly identical accuracies, but shortest path explores significantly fewer hypotheses.

At least two factors contribute to the tractability of exact search in our model. First, our character-level sequence model has a vocabulary size of 63, which is significantly smaller than token-level models, in which a vocabulary of 50k words is not uncommon. The search space of sequence models is dependent upon the size of the vocabulary, so the model's search space is dramatically smaller than for a token-level model.

Second, the inherent structure of the task makes it easy to eliminate large subgraphs of the search space. The first several characters of the input word and output word are almost always the same, so the model assigns very low probability to any sequence with different starting characters than the input. Then, the rest of the search procedure is dedicated to deciding between suffixes. Any suffix which does not appear frequently in the training data receives a low score, leaving the search to decide between a handful of possible options. The result is that the learned probability distribution is very spiked; it puts very high probability on just a few output sequences. It is empirically true that the top few most probable sequences have significantly higher scores than the next most probable sequences, which supports this hypothesis.

In our subsequent experiments, we decode using

Algorithm 1 The decoding procedure uses a shortest-path algorithm to find the most probable output sequence. The dictionary constraint is (optionally) implemented on line 9 by only considering prefixes that are contained in some trie T.

```
1: procedure DECODE($\mathbf{x}$, $t$, $V$, $T$)
2:     $H \leftarrow$ Heap()
3:     $H$.insert(0, #)
4:     while $H$ is not empty do
5:         $\mathbf{y} \leftarrow H$.remove()
6:         if $\mathbf{y}$ is a complete word then return $\mathbf{y}$
7:         for $y \in V$ do
8:             $\mathbf{y}' \leftarrow \mathbf{y} + y$
9:             if $\mathbf{y}' \in T$ then
10:                $s \leftarrow$ FORWARD($\mathbf{x}, t, \mathbf{y}'$)
11:                $H$.insert($s$, $\mathbf{y}'$)
```

exact inference by running a shortest path algorithm (see Algorithm 1). For reranking models, instead of typically using a beam of size k, we use the top k most probable sequences.

7 Results

In all of our experiments, we use the training, development, and testing splits provided by Cotterell et al. (2017b) and average over 30 random restarts. Table 3 displays the accuracies and average edit distances on the test set of each of the systems presented in this work and the state-of-the-art model from Cotterell et al. (2017b).

First, we observed that SEQ outperforms the results reported in Cotterell et al. (2017b) by a large margin, despite the fact that the model architectures are the same. We attribute this difference to better hyperparameter settings and improved learning rate annealing.

Then, it is clear that the accuracy of the distributional model, DIST, is significantly lower than any seq2seq model. We believe the orthography-informed models perform better because most observations in the dataset are orthographically regular, providing low-hanging fruit.

Open-vocabulary models Our open-vocabulary aggregation model AGGR improves performance by 3.8 points accuracy over SEQ, indicating that the sequence models and the distributional model are contributing complementary signals. AGGR is an open-vocabulary model like Cotterell et al. (2017b) and improves upon it by 6.3 points, making it our best comparable model. We provide an in-depth analysis of the strengths of SEQ and DIST in Section 7.1.

Model	Accuracy	Edit
Cotterell et al. (2017b)	71.7	0.97
DIST	54.9	3.23
SEQ	74.2	0.88
AGGR	78.0	0.83
SEQ+FREQ	79.3	0.71
DUAL+FREQ	**82.0**	**0.64**
SEQ+DICT	80.4	0.72
AGGR+DICT	81.0	0.78
SEQ+FREQ+DICT	81.2	0.71
AGGR+FREQ+DICT	**82.4**	**0.67**

Table 3: The accuracies and edit distances of the models presented in this paper and prior work. For edit distance, lower is better. The dictionary-constrained models are on the lower half of the table.

Closed-vocabulary models We now consider closed-vocabulary models that improve upon the seq2seq model in AGGR. First, we see that restricting the decoder to only generate known words is extremely useful, with SEQ+DICT improving over SEQ by 6.2 points. Qualitatively, we note that this constraint helps solve the suffix ambiguity problem, since orthographically plausible incorrect hypotheses are pruned as non-words. See Table 6 for examples of this phenomenon. Additionally, we observe that the dictionary-constrained model outperforms the unconstrained model according to top-10 accuracy (see Table 5).

Rescoring (+FREQ) provides further improvement of 0.8 points, showing that the decoding dictionary constraint provides a higher-quality beam that still has room for top-1 improvement. All together, AGGR+FREQ+DICT provides a 4.4 point improvement over the best open-vocabulary model, AGGR. This shows the disambiguating power of assuming a closed vocabulary.

Edit Distance One interesting side effect of the dictionary constraint appears when comparing AGGR+FREQ with and without the dictionary constraint. Although the accuracy of the dictionary-constrained model is better, the average edit distance is worse. The unconstrained model is free to put invalid words which are orthographically similar to the target word in its top-k, however the constrained model can only choose valid words. This means it is easier for the unconstrained model to generate words which have a low edit distance to the ground truth, whereas the constrained model

	Cotterell et al. (2017b)		AGGR		AGGR+FREQ +DICT	
	acc	edit	acc	edit	acc	edit
NOMINAL	35.1	2.67	**68.0**	**1.32**	62.1	1.40
RESULT	52.9	1.86	59.1	1.83	**69.7**	**1.29**
AGENT	65.6	0.78	73.5	0.65	**79.1**	**0.57**
ADVERB	93.3	**0.18**	94.0	**0.18**	**95.0**	0.22

Table 4: The accuracies and edit distances of our best open-vocabulary and closed-vocabulary models, AGGR and AGGR+FREQ+DICT compared to prior work, evaluated at the transformation level. For edit distance, lower is better.

	Cotterell et al. (2017b)	SEQ	SEQ+DICT
	top-10-acc	top-10-acc	top-10-acc
NOMINAL	70.2	73.7	**87.5**
RESULT	72.6	79.9	**90.4**
AGENT	82.2	88.4	**91.6**
ADVERB	96.5	**96.9**	**96.9**

Table 5: The accuracies of the top-10 best outputs for the SEQ, SEQ+DICT, and prior work demonstrate that the dictionary constraint significantly improves the overall candidate quality.

can only do that if such a word exists. The result is a more accurate, yet more orthographically diverse, set of hypotheses.

Results by Transformation Next, we compare our best open vocabulary and closed vocabulary models to previous work across each derivational transformation. These results are in Table 4.

The largest improvement over the baseline system is for NOMINAL transformations, in which the AGGR has a 49% reduction in error. We attribute most of this gain to the difficulty of this particular transformation. NOMINAL is challenging because there are several plausible endings (e.g. *-ity*, *-ness*, *-ence*) which occur at roughly the same rate. Additionally, NOMINAL examples are the least frequent transformation in the dataset, so it is challenging for a sequential model to learn to generalize. The distributional model, which does not rely on suffix information, does not have this same weakness, so the aggregation AGGR model has better results.

The performance of AGGR+FREQ+DICT is worse than AGGR, however. This is surprising because, in all other transformations, adding dictionary information improves the accuracies. We believe this is due to the ambiguity of the ground truth: Many root words have seemingly multiple plausible nominal transformations, such as *rigid* $\rightarrow$ {*rigidness*, *rigidity*} and *equivalent* $\rightarrow$ {*equivalence*, *equivalency*}. The dictionary constraint produces a better set of hypotheses to rescore, as demonstrated in Table 5. Therefore, the dictionary-constrained model is likely to have more of these ambiguous cases, which makes the task more difficult.

7.1 Strengths of SEQ and DIST

In this subsection we explore why AGGR improves consistently over SEQ even though it maintains an open vocabulary. We have argued that DIST is able to correctly produce derived words that are orthographically irregular or infrequent in the training data. Figure 2 quantifies this phenomenon, analyzing the difference in accuracy between the two models, and plotting this in relationship to the frequency of the suffix in the training data. The plot shows that SEQ excels at generating derived words ending in *-ly, -ion*, and other suffixes that appeared frequently in the training data. DIST's improvements over SEQ are generally much less frequent in the training data, or as in the case of *-ment*, are less frequent than other suffixes for the same transformation (like *-ion*.) By producing derived words whose suffixes show up rarely in the training data, DIST helps solve the orthographic irregularity problem.

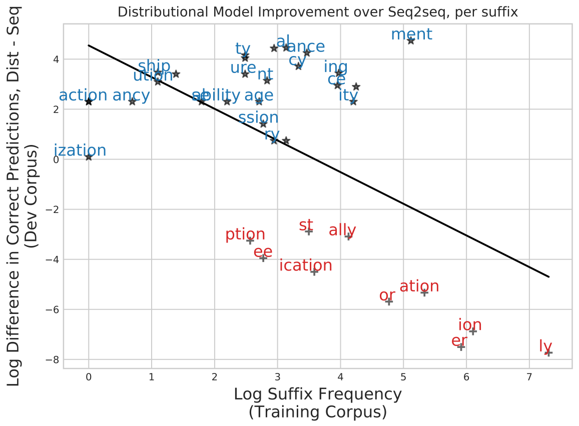


Figure 2: Aggregating across 30 random restarts, we tallied when SEQ and DIST correctly produced derived forms of each suffix. The y-axis shows the logarithm of the difference, per suffix, between the tally for DIST and the tally for SEQ. On the x-axis is the logarithm of the frequency of derived words with each suffix in the training data. A linear regression line is plotted to show the relationship between log suffix frequency and log difference in correct predictions. Suffixes that differ only by the first letter, as with *-ger* and *-er*, have been merged and represented by the more frequent of the two.

8 Prior Work

There has been much work on the related task of inflected word generation (Durrett and DeNero,

x	*t*	DIST	SEQ	AGGR	AGGR+DICT
approve	RESULT	**approval**	approvation	**approval**	**approval**
bankrupt	NOMINAL	**bankruptcy**	bankruption	**bankruptcy**	**bankruptcy**
irretrievable	ADVERB	irreparably	**irretrievably**	**irretrievably**	**irretrievably**
connect	RESULT	connectivity	**connection**	**connection**	**connection**
stroll	AGENT	strolls	**stroller**	**stroller**	**stroller**
emigrate	SUBJECT	emigre	emigrator	emigrator	**emigrant**
ubiquitous	NOMINAL	**ubiquity**	ubiquit	ubiquit	**ubiquity**
hinder	AGENT	hinderer	hinderer	hinderer	hinderer
vacant	NOMINAL	vacance	vacance	vacance	vacance

Table 6: Sample output from a selection of models. The words in bold mark the correct derivations. "Hindrance" and "vacancy" are the expected derived words for the last two rows.

2013; Rastogi et al., 2016; Hulden et al., 2014). It is a structurally similar task to ours, but does not have the same difficulty of challenges (Cotterell et al., 2017a,b), which we have addressed in our work. The paradigm completion for derivational morphology dataset we use in this work was introduced in Cotterell et al. (2017b). They apply the model that won the 2016 SIGMORPHON shared task on inflectional morphology to derivational morphology (Kann and Schütze, 2016; Cotterell et al., 2016). We use this as our baseline.

Our implementation of the dictionary constraint is an example of a special constraint which can be directly incorporated into the inference algorithm at little additional cost. Roth and Yih (2004, 2007) propose a general inference procedure that naturally incorporates constraints through recasting inference as solving an integer linear program.

Beam or hypothesis rescoring to incorporate an expensive or non-decomposable signal into search has a history in machine translation (Huang and Chiang, 2007). In inflectional morphology, Nicolai et al. (2015) use this idea to rerank hypotheses using orthographic features and Faruqui et al. (2016) use a character-level language model. Our approach is similar to Faruqui et al. (2016) in that we use statistics from a raw corpus, but at the token level.

There have been several attempts to use distributional information in morphological generation and analysis. Soricut and Och (2015) collect pairs of words related by any morphological change in an unsupervised manner, then select a vector offset which best explains their observations. There has been subsequent work exploring the vector offset method, finding it unsuccessful in capturing derivational transformations (Gladkova et al., 2016). However, we use more expressive, nonlinear functions to model derivational transformations and report positive results. Gupta et al. (2017) then learn a linear transformation per orthographic rule to solve a word analogy task. Our distributional model learns a function per derivational transformation, not per orthographic rule, which allows it to generalize to unseen orthography.

9 Implementation Details

Our models are implemented in Python using the DyNet deep learning library (Neubig et al., 2017). The code is freely available for download.[4]

Sequence Model The sequence-to-sequence model uses character embeddings of size 20, which are shared across the encoder and decoder, with a vocabulary size of 63. The hidden states of the LSTMs are of size 40.

For training, we use Adam with an initial learning rate of 0.005, a batch size of 5, and train for a maximum of 30 epochs. If after one epoch of the training data, the loss on the validation set does not decrease, we anneal the learning rate by half and revert to the previous best model.

During decoding, we find the top 1 most probable sequence as discussed in Section 6 unless rescoring is used, in which we use the top 10.

Rescorer The rescorer is a 1-hidden-layer perceptron with a `tanh` nonlinearity and 4 hidden units. It is trained for a maximum of 5 epochs.

Distributional Model The DIST model is a 1-hidden-layer perceptron with a `tanh` nonlinearity

[4] https://github.com/danieldeutsch/acl2018

and 100 hidden units. It is trained for a maximum of 25 epochs.

10 Conclusion

In this work, we present a novel aggregation model for derived word generation. This model learns to choose between the predictions of orthographically- and distributionally-informed models. This ameliorates suffix ambiguity and orthographic irregularity, the salient problems of the generation task. Concurrently, we show that derivational transformations can be usefully modeled as nonlinear functions on distributional word embeddings. The distributional and orthographic models aggregated contribute orthogonal information to the aggregate, as shown by substantial improvements over state-of-the-art results, and qualitative analysis. Two ways of incorporating corpus knowledge – constrained decoding and rescoring – demonstrate further improvements to our main contribution.

Acknowledgements

We would like to thank Shyam Upadhyay, Jordan Kodner, and Ryan Cotterell for insightful discussions about derivational morphology. We would also like to thank our anonymous reviewers for helpful feedback on clarity and presentation.

This work was supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 1–30. http://www.aclweb.org/anthology/K17-2001.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The sigmorphon 2016 shared taskmorphological reinflection. *ACL 2016* page 10.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017b. Paradigm completion for derivational morphology. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 714–720. https://www.aclweb.org/anthology/D17-1074.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1185–1195. http://www.aclweb.org/anthology/N13-1138.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 634–643. http://www.aclweb.org/anthology/N16-1077.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, San Diego, California, pages 8–15. http://www.aclweb.org/anthology/N16-2002.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Arihant Gupta, Syed Sarfaraz Akhtar, Avijit Vajpayee, Arjit Srivastava, Madan Gopal Jhanwar, and Manish Shrivastava. 2017. Exploiting morphological regularities in distributional word representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 292–297. https://www.aclweb.org/anthology/D17-1028.

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

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational*

Linguistics. Association for Computational Linguistics, Prague, Czech Republic, pages 144–151. http://www.aclweb.org/anthology/P07-1019.

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 569–578. http://www.aclweb.org/anthology/E14-1060.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 555–560. http://anthology.aclweb.org/P16-2090.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 104–113. http://www.aclweb.org/anthology/W13-3512.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 24–31.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331 6014:176–82.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop Papers*. Scottsdale, Arizona. https://arxiv.org/pdf/1301.3781.pdf.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 922–931. http://www.aclweb.org/anthology/N15-1093.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 623–633. http://www.aclweb.org/anthology/N16-1076.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, pages 1–8. http://cogcomp.org/papers/RothYi04.pdf.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation http://cogcomp.org/papers/RothYi07.pdf.

Wolfgang Seeker and Özlem Çetinoglu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics* 3:359–373.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1627–1637. http://www.aclweb.org/anthology/N15-1186.

Deep-speare: A joint neural model of poetic language, meter and rhyme

Jey Han Lau[1,2] **Trevor Cohn**[2] **Timothy Baldwin**[2]
Julian Brooke[3] **Adam Hammond**[4]

[1] IBM Research Australia
[2] School of Computing and Information Systems, The University of Melbourne
[3] Thomson Reuters
[4] Department of English, University of Toronto

jeyhan.lau@gmail.com, t.cohn@unimelb.edu.au, tb@ldwin.net,
julian.brooke@gmail.com, adam.hammond@utoronto.ca

Abstract

In this paper, we propose a joint architecture that captures language, rhyme and meter for sonnet modelling. We assess the quality of generated poems using crowd and expert judgements. The stress and rhyme models perform very well, as generated poems are largely indistinguishable from human-written poems. Expert evaluation, however, reveals that a vanilla language model captures meter implicitly, and that machine-generated poems still underperform in terms of readability and emotion. Our research shows the importance expert evaluation for poetry generation, and that future research should look beyond rhyme/meter and focus on poetic language.

1 Introduction

With the recent surge of interest in deep learning, one question that is being asked across a number of fronts is: can deep learning techniques be harnessed for creative purposes? Creative applications where such research exists include the composition of music (Humphrey et al., 2013; Sturm et al., 2016; Choi et al., 2016), the design of sculptures (Lehman et al., 2016), and automatic choreography (Crnkovic-Friis and Crnkovic-Friis, 2016). In this paper, we focus on a creative textual task: automatic poetry composition.

A distinguishing feature of poetry is its *aesthetic forms*, e.g. rhyme and rhythm/meter.[1] In this work, we treat the task of poem generation as a constrained language modelling task, such that lines of a given poem rhyme, and each line follows a canonical meter and has a fixed number of stresses. Specifically, we focus on sonnets and generate quatrains in iambic pentameter (e.g. see Figure 1), based on an unsupervised model of language, rhyme and meter trained on a novel corpus of sonnets.

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date:

Figure 1: 1st quatrain of Shakespeare's *Sonnet 18*.

Our findings are as follows:

- our proposed stress and rhyme models work very well, generating sonnet quatrains with stress and rhyme patterns that are indistinguishable from human-written poems and rated highly by an expert;
- a vanilla language model trained over our sonnet corpus, surprisingly, captures meter implicitly at human-level performance;
- while crowd workers rate the poems generated by our best model as nearly indistinguishable from published poems by humans, an expert annotator found the machine-generated poems to lack readability and emotion, and our best model to be only comparable to a vanilla language model on these dimensions;
- most work on poetry generation focuses on meter (Greene et al., 2010; Ghazvininejad et al., 2016; Hopkins and Kiela, 2017); our results suggest that future research should look beyond meter and focus on improving readability.

In this, we develop a new annotation framework for the evaluation of machine-generated poems, and release both a novel data of sonnets and the full source code associated with this research.[2]

[1]Noting that there are many notable divergences from this in the work of particular poets (e.g. Walt Whitman) and poetry types (such as free verse or haiku).

[2]`https://github.com/jhlau/deepspeare`

2 Related Work

Early poetry generation systems were generally rule-based, and based on rhyming/TTS dictionaries and syllable counting (Gervás, 2000; Wu et al., 2009; Netzer et al., 2009; Colton et al., 2012; Toivanen et al., 2013). The earliest attempt at using statistical modelling for poetry generation was Greene et al. (2010), based on a language model paired with a stress model.

Neural networks have dominated recent research. Zhang and Lapata (2014) use a combination of convolutional and recurrent networks for modelling Chinese poetry, which Wang et al. (2016) later simplified by incorporating an attention mechanism and training at the character level. For English poetry, Ghazvininejad et al. (2016) introduced a finite-state acceptor to explicitly model rhythm in conjunction with a recurrent neural language model for generation. Hopkins and Kiela (2017) improve rhythm modelling with a cascade of weighted state transducers, and demonstrate the use of character-level language model for English poetry. A critical difference over our work is that we jointly model both poetry content and forms, and unlike previous work which use dictionaries (Ghazvininejad et al., 2016) or heuristics (Greene et al., 2010) for rhyme, we learn it automatically.

3 Sonnet Structure and Dataset

The sonnet is a poem type popularised by Shakespeare, made up of 14 lines structured as 3 quatrains (4 lines) and a couplet (2 lines);[3] an example quatrain is presented in Figure 1. It follows a number of *aesthetic forms*, of which two are particularly salient: stress and rhyme.

A sonnet line obeys an alternating stress pattern, called the iambic pentameter, e.g.:

$$\begin{array}{cccccccccc} S^- & S^+ & S^- & S^+ & S^- & S^+ & S^- & S^+ & S^- & S^+ \\ \textit{Shall} & \textit{I} & \textit{com} & \textit{pare} & \textit{thee} & \textit{to} & \textit{a} & \textit{sum} & \textit{mer's} & \textit{day?} \end{array}$$

where S^- and S^+ denote unstressed and stressed syllables, respectively.

A sonnet also rhymes, with a typical rhyming scheme being *ABAB CDCD EFEF GG*. There are a number of variants, however, mostly seen in the quatrains; e.g. *AABB* or *ABBA* are also common.

We build our sonnet dataset from the latest image of Project Gutenberg.[4] We first create a (generic) poetry document collection using the GutenTag tool (Brooke et al., 2015), based on its inbuilt poetry classifier and rule-based structural tagging of individual poems.

Given the poems, we use word and character statistics derived from Shakespeare's 154 sonnets to filter out all non-sonnet poems (to form the "BACKGROUND" dataset), leaving the sonnet corpus ("SONNET").[5] Based on a small-scale manual analysis of SONNET, we find that the approach is sufficient for extracting sonnets with high precision. BACKGROUND serves as a large corpus (34M words) for pre-training word embeddings, and SONNET is further partitioned into training, development and testing sets. Statistics of SONNET are given in Table 1.[6]

Partition	#Sonnets	#Words
Train	2685	367K
Dev	335	46K
Test	335	46K

Table 1: SONNET dataset statistics.

4 Architecture

We propose modelling both content and forms jointly with a neural architecture, composed of 3 components: (1) a language model; (2) a pentameter model for capturing iambic pentameter; and (3) a rhyme model for learning rhyming words.

Given a sonnet line, the language model uses standard categorical cross-entropy to predict the next word, and the pentameter model is similarly trained to learn the alternating iambic stress patterns.[7] The rhyme model, on the other hand, uses a margin-based loss to separate rhyming word pairs from non-rhyming word pairs in a quatrain. For generation we use the language model to generate one word at a time, while applying the pentame-

[3]There are other forms of sonnets, but the Shakespearean sonnet is the dominant one. Hereinafter "sonnet" is used to specifically mean Shakespearean sonnets.

[4]`https://www.gutenberg.org/`.

[5]The following constraints were used to select sonnets: $8.0 \leqslant$ mean words per line $\leqslant 11.5$; $40 \leqslant$ mean characters per line $\leqslant 51.0$; min/max number of words per line of 6/15; min/max number of characters per line of 32/60; and min letter ratio per line $\geqslant 0.59$.

[6]The sonnets in our collection are largely in Modern English, with possibly a small number of poetry in Early Modern English. The potentially mixed-language dialect data might add noise to our system, and given more data it would be worthwhile to include time period as a factor in the model.

[7]There are a number of variations in addition to the standard pattern (Greene et al., 2010), but our model uses only the standard pattern as it is the dominant one.

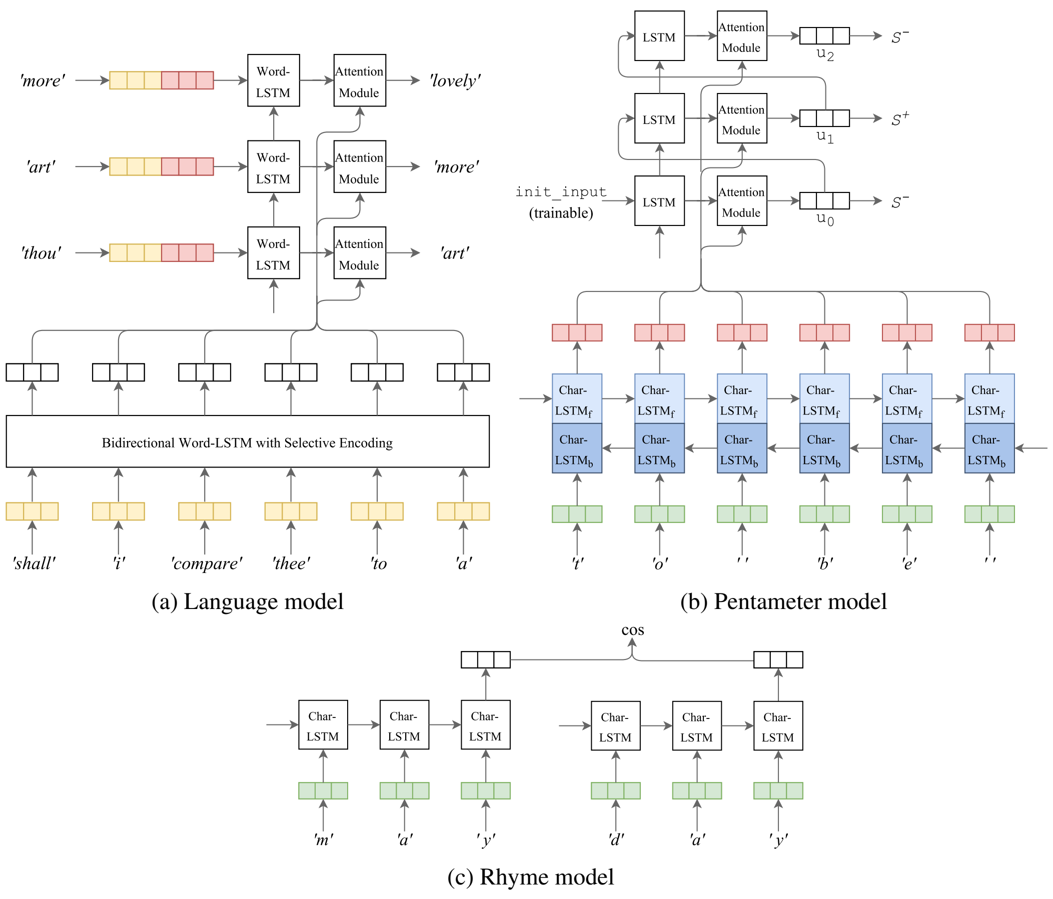


Figure 2: Architecture of the language, pentameter and rhyme models. Colours denote shared weights.

ter model to sample meter-conforming sentences and the rhyme model to enforce rhyme. The architecture of the joint model is illustrated in Figure 2. We train all the components together by treating each component as a sub-task in a multi-task learning setting.[8]

4.1 Language Model

The language model is a variant of an LSTM encoder–decoder model with attention (Bahdanau et al., 2015), where the encoder encodes the preceding context (i.e. all sonnet lines before the current line) and the decoder decodes one word at a time for the current line, while attending to the preceding context.

In the encoder, we embed context words z_i using embedding matrix $\mathbf{W}_{wrd}$ to yield $\mathbf{w}_i$, and feed them to a biLSTM[9] to produce a sequence of encoder hidden states $\mathbf{h}_i = [\vec{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$. Next we apply a selective mechanism (Zhou et al., 2017) to each $\mathbf{h}_i$. By defining the representation of the whole context $\overline{\mathbf{h}} = [\vec{\mathbf{h}}_C; \overleftarrow{\mathbf{h}}_1]$ (where C is the number of words in the context), the selective mechanism filters the hidden states $\mathbf{h}_i$ using $\overline{\mathbf{h}}$ as follows:

$$\mathbf{h}'_i = \mathbf{h}_i \odot \sigma(\mathbf{W}_a \mathbf{h}_i + \mathbf{U}_a \overline{\mathbf{h}} + \mathbf{b}_a)$$

where $\odot$ denotes element-wise product. Hereinafter $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are used to refer to model parameters. The intuition behind this procedure is to selectively filter less useful elements from the context words.

In the decoder, we embed words x_t in the current line using the encoder-shared embedding matrix ($\mathbf{W}_{wrd}$) to produce $\mathbf{w}_t$. In addition to the word embeddings, we also embed the characters of a word using embedding matrix $\mathbf{W}_{chr}$ to produce $\mathbf{c}_{t,i}$, and feed them to a bidirectional (character-level) LSTM:

$$\begin{aligned} \vec{\mathbf{u}}_{t,i} &= \mathrm{LSTM}_f(\mathbf{c}_{t,i}, \vec{\mathbf{u}}_{t,i-1}) \\ \overleftarrow{\mathbf{u}}_{t,i} &= \mathrm{LSTM}_b(\mathbf{c}_{t,i}, \overleftarrow{\mathbf{u}}_{t,i+1}) \end{aligned} \qquad (1)$$

We represent the character encoding of a word by concatenating the last forward and first back-

[8]We stress that although the components appear to be disjointed, the shared parameters allow the components to mutually influence each other during joint training. To exemplify this, we found that the pentameter model performs very poorly when we train each component separately.

[9]We use a single layer for all LSTMs.

ward hidden states $\overline{\mathbf{u}}_t = [\vec{\mathbf{u}}_{t,L}; \overleftarrow{\mathbf{u}}_{t,1}]$, where L is the length of the word. We incorporate character encodings because they provide orthographic information, improve representations of unknown words, and are shared with the pentameter model (Section 4.2).[10] The rationale for sharing the parameters is that we see word stress and language model information as complementary.

Given the word embedding $\mathbf{w}_t$ and character encoding $\overline{\mathbf{u}}_t$, we concatenate them together and feed them to a unidirectional (word-level) LSTM to produce the decoding states:

$$\mathbf{s}_t = \text{LSTM}([\mathbf{w}_t; \overline{\mathbf{u}}_t], \mathbf{s}_{t-1}) \tag{2}$$

We attend $\mathbf{s}_t$ to encoder hidden states $\mathbf{h}'_i$ and compute the weighted sum of $\mathbf{h}'_i$ as follows:

$$\begin{aligned} e^t_i &= \mathbf{v}_b^\intercal \tanh(\mathbf{W}_b\mathbf{h}'_i + \mathbf{U}_b\mathbf{s}_t + \mathbf{b}_b) \\ \mathbf{a}^t &= \text{softmax}(\mathbf{e}^t) \\ \mathbf{h}^*_t &= \sum_i a^t_i \mathbf{h}'_i \end{aligned}$$

To combine $\mathbf{s}_t$ and $\mathbf{h}^*_t$, we use a gating unit similar to a GRU (Cho et al., 2014; Chung et al., 2014): $\mathbf{s}'_t = \text{GRU}(\mathbf{s}_t, \mathbf{h}^*_t)$. We then feed $\mathbf{s}'_t$ to a linear layer with softmax activation to produce the vocabulary distribution (i.e. softmax$(\mathbf{W}_{out}\mathbf{s}'_t + \mathbf{b}_{out})$, and optimise the model with standard categorical cross-entropy loss. We use dropout as regularisation (Srivastava et al., 2014), and apply it to the encoder/decoder LSTM outputs and word embedding lookup. The same regularisation method is used for the pentameter and rhyme models.

As our sonnet data is relatively small for training a neural language model (367K words; see Table 1), we pre-train word embeddings and reduce parameters further by introducing weight-sharing between output matrix $\mathbf{W}_{out}$ and embedding matrix $\mathbf{W}_{wrd}$ via a projection matrix $\mathbf{W}_{prj}$ (Inan et al., 2016; Paulus et al., 2017; Press and Wolf, 2017):

$$\mathbf{W}_{out} = \tanh(\mathbf{W}_{wrd}\mathbf{W}_{prj})$$

4.2 Pentameter Model

This component is designed to capture the alternating iambic stress pattern. Given a sonnet line, the pentameter model learns to attend to the appropriate characters to predict the 10 binary stress symbols sequentially.[11] As punctuation is not pronounced, we preprocess each sonnet line to remove all punctuation, leaving only spaces and letters. Like the language model, the pentameter model is fashioned as an encoder–decoder network.

In the encoder, we embed the characters using the shared embedding matrix $\mathbf{W}_{chr}$ and feed them to the shared bidirectional character-level LSTM (Equation (1)) to produce the character encodings for the sentence: $\mathbf{u}_j = [\vec{\mathbf{u}}_j; \overleftarrow{\mathbf{u}}_j]$.

In the decoder, it attends to the characters to predict the stresses sequentially with an LSTM:

$$\mathbf{g}_t = \text{LSTM}(\mathbf{u}^*_{t-1}, \mathbf{g}_{t-1})$$

where $\mathbf{u}^*_{t-1}$ is the weighted sum of character encodings from the previous time step, produced by an attention network which we describe next,[12] and $\mathbf{g}_t$ is fed to a linear layer with softmax activation to compute the stress distribution.

The attention network is designed to focus on stress-producing characters, whose positions are monotonically increasing (as stress is predicted sequentially). We first compute μ_t, the mean position of focus:

$$\begin{aligned} \mu'_t &= \sigma(\mathbf{v}_c^\intercal \tanh(\mathbf{W}_c\mathbf{g}_t + \mathbf{U}_c\mu_{t-1} + \mathbf{b}_c)) \\ \mu_t &= M \times \min(\mu'_t + \mu_{t-1}, 1.0) \end{aligned}$$

where M is the number of characters in the sonnet line. Given μ_t, we can compute the (unnormalised) probability for each character position:

$$p^t_j = \exp\left(\frac{-(j-\mu_t)^2}{2T^2}\right)$$

where standard deviation T is a hyper-parameter. We incorporate this position information when computing $\mathbf{u}^*_t$:[13]

$$\begin{aligned} \mathbf{u}'_j &= p^t_j \mathbf{u}_j \\ d^t_j &= \mathbf{v}_d^\intercal \tanh(\mathbf{W}_d\mathbf{u}'_j + \mathbf{U}_d\mathbf{g}_t + \mathbf{b}_d) \\ \mathbf{f}^t &= \text{softmax}(\mathbf{d}^t + \log \mathbf{p}^t) \\ \mathbf{u}^*_t &= \sum_j b^t_j \mathbf{u}_j \end{aligned}$$

[10] We initially shared the character encodings with the rhyme model as well, but found sub-par performance for the rhyme model. This is perhaps unsurprising, as rhyme and stress are qualitatively very different aspects of forms.

[11] That is, given the input line *Shall I compare thee to a summer's day?* the model is required to output S^- S^+ S^- S^+ S^- S^+ S^- S^+ S^- S^+, based on the syllable boundaries from Section 3.

[12] Initial input ($\mathbf{u}^*_0$) and state ($\mathbf{g}_0$) is a trainable vector and zero vector respectively.

[13] Spaces are masked out, so they always yield zero attention weights.

Intuitively, the attention network incorporates the position information at two points, when computing: (1) d_j^t by weighting the character encodings; and (2) $\mathbf{f}^t$ by adding the position log probabilities. This may appear excessive, but preliminary experiments found that this formulation produces the best performance.

In a typical encoder–decoder model, the attended encoder vector $\mathbf{u}_t^*$ would be combined with the decoder state $\mathbf{g}_t$ to compute the output probability distribution. Doing so, however, would result in a zero-loss model as it will quickly learn that it can simply ignore $\mathbf{u}_t^*$ to predict the alternating stresses based on $\mathbf{g}_t$. For this reason we use only $\mathbf{u}_t^*$ to compute the stress probability:

$$P(S^-) = \sigma(\mathbf{W}_e\mathbf{u}_t^* + b_e)$$

which gives the loss $\mathcal{L}_{ent} = \sum_t -\log P(S_t^\star)$ for the whole sequence, where $S_t^\star$ is the target stress at time step t.

We find the decoder still has the tendency to attend to the same characters, despite the incorporation of position information. To regularise the model further, we introduce two loss penalties: repeat and coverage loss.

The repeat loss penalises the model when it attends to previously attended characters (See et al., 2017), and is computed as follows:

$$\mathcal{L}_{rep} = \sum_t \sum_j \min(f_j^t, \sum_{t=1}^{t-1} f_j^t)$$

By keeping a sum of attention weights over all previous time steps, we penalise the model when it focuses on characters that have non-zero history weights.

The repeat loss discourages the model from focussing on the same characters, but does not assure that the appropriate characters receive attention. Observing that stresses are aligned with the vowels of a syllable, we therefore penalise the model when vowels are ignored:

$$\mathcal{L}_{cov} = \sum_{j \in V} \text{ReLU}(C - \sum_{t=1}^{10} f_j^t)$$

where V is a set of positions containing vowel characters, and C is a hyper-parameter that defines the minimum attention threshold that avoids penalty.

To summarise, the pentameter model is optimised with the following loss:

$$\mathcal{L}_{pm} = \mathcal{L}_{ent} + \alpha\mathcal{L}_{rep} + \beta\mathcal{L}_{cov} \tag{3}$$

where α and β are hyper-parameters for weighting the additional loss terms.

4.3 Rhyme Model

Two reasons motivate us to learn rhyme in an unsupervised manner: (1) we intend to extend the current model to poetry in other languages (which may not have pronunciation dictionaries); and (2) the language in our SONNET data is not Modern English, and so contemporary dictionaries may not accurately reflect the rhyme of the data.

Exploiting the fact that rhyme exists in a quatrain, we feed sentence-ending word pairs of a quatrain as input to the rhyme model and train it to learn how to separate rhyming word pairs from non-rhyming ones. Note that the model does not assume any particular rhyming scheme — it works as long as quatrains have rhyme.

A training example consists of a number of word pairs, generated by pairing one target word with 3 other reference words in the quatrain, i.e. $\{(x_t, x_r), (x_t, x_{r+1}), (x_t, x_{r+2})\}$, where x_t is the target word and x_{r+i} are the reference words.[14] We assume that in these 3 pairs there should be one rhyming and 2 non-rhyming pairs. From preliminary experiments we found that we can improve the model by introducing additional non-rhyming or negative reference words. Negative reference words are sampled uniform randomly from the vocabulary, and the number of additional negative words is a hyper-parameter.

For each word x in the word pairs we embed the characters using the shared embedding matrix $\mathbf{W}_{chr}$ and feed them to an LSTM to produce the character states $\mathbf{u}_j$.[15] Unlike the language and pentameter models, we use a unidirectional forward LSTM here (as rhyme is largely determined by the final characters), and the LSTM parameters are not shared. We represent the encoding of the whole word by taking the last state $\overline{\mathbf{u}} = \mathbf{u}_L$, where L is the character length of the word.

Given the character encodings, we use a

[14]E.g. for the quatrain in Figure 1, a training example is $\{$(*day*, *temperate*), (*day*, *may*), (*day*, *date*)$\}$.

[15]The character embeddings are the only shared parameters in this model.

margin-based loss to optimise the model:

$$Q = \{\cos(\overline{\mathbf{u}}_t, \overline{\mathbf{u}}_r), \cos(\overline{\mathbf{u}}_t, \overline{\mathbf{u}}_{r+1}), ...\}$$
$$\mathcal{L}_{rm} = \max(0, \delta - \text{top}(Q, 1) + \text{top}(Q, 2))$$

where $\text{top}(Q, k)$ returns the k-th largest element in Q, and δ is a margin hyper-parameter.

Intuitively, the model is trained to learn a sufficient margin (defined by δ) that separates the best pair with *all others*, with the second-best being used to quantify *all others*. This is the justification used in the multi-class SVM literature for a similar objective (Wang and Xue, 2014).

With this network we can estimate whether two words rhyme by computing the cosine similarity score during generation, and resample words as necessary to enforce rhyme.

4.4 Generation Procedure

We focus on quatrain generation in this work, and so the aim is to generate 4 lines of poetry. During generation we feed the hidden state from the previous time step to the language model's decoder to compute the vocabulary distribution for the current time step. Words are sampled using a temperature between 0.6 and 0.8, and they are resampled if the following set of words is generated: (1) UNK token; (2) non-stopwords that were generated before;[16] (3) any generated words with a frequency $\geqslant 2$; (4) the preceding 3 words; and (5) a number of symbols including parentheses, single and double quotes.[17] The first sonnet line is generated without using any preceding context.

We next describe how to incorporate the pentameter model for generation. Given a sonnet line, the pentameter model computes a loss $\mathcal{L}_{pm}$ (Equation (3)) that indicates how well the line conforms to the iambic pentameter. We first generate 10 candidate lines (all initialised with the same hidden state), and then sample one line from the candidate lines based on the pentameter loss values ($\mathcal{L}_{pm}$). We convert the losses into probabilities by taking the softmax, and a sentence is sampled with temperature $= 0.1$.

To enforce rhyme, we randomly select one of the rhyming schemes (*AABB*, *ABAB* or *ABBA*) and resample sentence-ending words as necessary. Given a pair of words, the rhyme model produces a cosine similarity score that estimates how well the two words rhyme. We resample the second word of a rhyming pair (e.g. when generating the second *A* in *AABB*) until it produces a cosine similarity $\geqslant$ 0.9. We also resample the second word of a non-rhyming pair (e.g. when generating the first *B* in *AABB*) by requiring a cosine similarity $\leqslant$ 0.7.[18]

When generating in the forward direction we can never be sure that any particular word is the last word of a line, which creates a problem for resampling to produce good rhymes. This problem is resolved in our model by reversing the direction of the language model, i.e. generating the last word of each line first. We apply this inversion trick at the word level (character order of a word is not modified) and only to the language model; the pentameter model receives the original word order as input.

5 Experiments

We assess our sonnet model in two ways: (1) component evaluation of the language, pentameter and rhyme models; and (2) poetry generation evaluation, by crowd workers and an English literature expert. A sample of machine-generated sonnets are included in the supplementary material.

We tune the hyper-parameters of the model over the development data (optimal configuration in the supplementary material). Word embeddings are initialised with pre-trained skip-gram embeddings (Mikolov et al., 2013a,b) on the BACKGROUND dataset, and are updated during training. For optimisers, we use Adagrad (Duchi et al., 2011) for the language model, and Adam (Kingma and Ba, 2014) for the pentameter and rhyme models. We truncate backpropagation through time after 2 sonnet lines, and train using 30 epochs, resetting the network weights to the weights from the previous epoch whenever development loss worsens.

5.1 Component Evaluation

5.1.1 Language Model

We use standard perplexity for evaluating the language model. In terms of model variants, we have:[19]

- **`LM`:** Vanilla LSTM language model;
- **`LM`*:** LSTM language model that incorporates character encodings (Equation (2));

[16]We use the NLTK stopword list (Bird et al., 2009).

[17]We add these constraints to prevent the model from being too repetitive, in generating the same words.

[18]Maximum number of resampling steps is capped at 1000. If the threshold is exceeded the model is reset to generate from scratch again.

[19]All models use the same (applicable) hyper-parameter configurations.

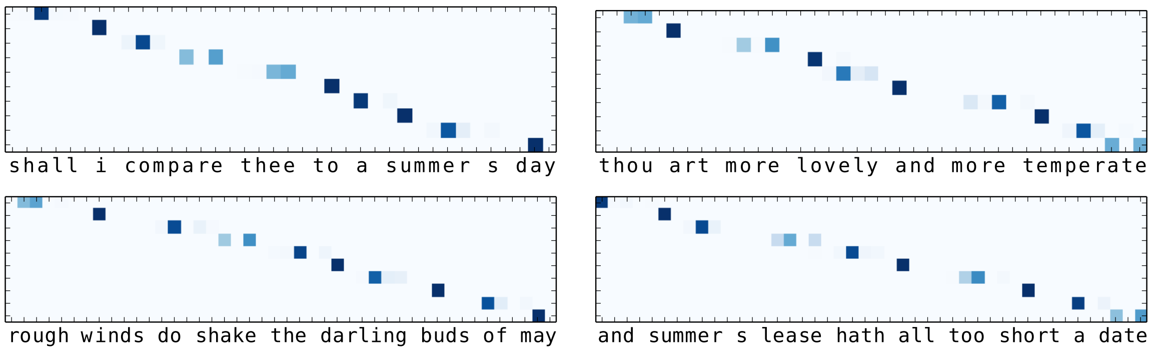


Figure 3: Character attention weights for the first quatrain of Shakespeare's *Sonnet 18*.

Model	Ppl	Stress Acc	Rhyme F1
`LM`	90.13	–	–
`LM`*	84.23	–	–
`LM`**	80.41	–	–
`LM`**`-C`	83.68	–	–
`LM`**`+PM+RM`	80.22	0.74	0.91
`Stress-BL`	–	0.80	–
`Rhyme-BL`	–	–	0.74
`Rhyme-EM`	–	–	0.71

Table 2: Component evaluation for the language model ("Ppl" = perplexity), pentameter model ("Stress Acc"), and rhyme model ("Rhyme F1"). Each number is an average across 10 runs.

- **`LM`**:** LSTM language model that incorporates both character encodings and preceding context;
- **`LM`**`-C`:** Similar to `LM`**, but preceding context is encoded using convolutional networks, inspired by the poetry model of Zhang and Lapata (2014);[20]
- **`LM`**`+PM+RM`:** the full model, with joint training of the language, pentameter and rhyme models.

Perplexity on the test partition is detailed in Table 2. Encouragingly, we see that the incorporation of character encodings and preceding context improves performance substantially, reducing perplexity by almost 10 points from `LM` to `LM`**. The inferior performance of `LM`**`-C` compared to `LM`** demonstrates that our approach of processing context with recurrent networks with selective encoding is more effective than convolutional networks. The full model `LM`**`+PM+RM`, which learns stress and rhyme patterns simultaneously, also appears to improve the language model slightly.

5.1.2 Pentameter Model

To assess the pentameter model, we use the attention weights to predict stress patterns for words in the test data, and compare them against stress patterns in the CMU pronunciation dictionary.[21] Words that have no coverage or have non-alternating patterns given by the dictionary are discarded. We use accuracy as the metric, and a predicted stress pattern is judged to be correct if it matches any of the dictionary stress patterns.

To extract a stress pattern for a word from the model, we iterate through the pentameter (10 time steps), and append the appropriate stress (e.g. 1st time step $= S^-$) to the word if any of its characters receives an attention $\geqslant 0.20$.

For the baseline (`Stress-BL`) we use the pre-trained weighted finite state transducer (WFST) provided by Hopkins and Kiela (2017).[22] The WFST maps a sequence word to a sequence of stresses by assuming each word has 1–5 stresses and the full word sequence produces iambic pentameter. It is trained using the EM algorithm on a sonnet corpus developed by the authors.

We present stress accuracy in Table 2. `LM`**`+PM+RM` performs competitively, and informal inspection reveals that a number of mistakes are due to dictionary errors. To understand the predicted stresses qualitatively, we display attention heatmaps for the the first quatrain of Shakespeare's *Sonnet 18* in Figure 3. The y-axis represents the ten stresses of the iambic pentameter, and

[20]In Zhang and Lapata (2014), the authors use a series of convolutional networks with a width of 2 words to convert 5/7 poetry lines into a fixed size vector; here we use a standard convolutional network with max-pooling operation (Kim, 2014) to process the context.

[21]`http://www.speech.cs.cmu.edu/cgi-bin/cmudict`. Note that the dictionary provides 3 levels of stresses: 0, 1 and 2; we collapse 1 and 2 to S^+.

[22]`https://github.com/JackHopkins/ACLPoetry`

CMU Rhyming Pairs		**CMU Non-Rhyming Pairs**	
Word Pair	Cos	Word Pair	Cos
(*endeavour*, *never*)	0.028	(*blood*, *stood*)	1.000
(*nowhere*, *compare*)	0.098	(*mood*, *stood*)	1.000
(*supply*, *sigh*)	0.164	(*overgrown*, *frown*)	1.000
(*sky*, *high*)	0.164	(*understood*, *food*)	1.000
(*me*, *maybe*)	0.165	(*brood*, *wood*)	1.000
(*cursed*, *burst*)	0.172	(*rove*, *love*)	0.999
(*weigh*, *way*)	0.200	(*sire*, *ire*)	0.999
(*royally*, *we*)	0.217	(*moves*, *shoves*)	0.998
(*use*, *juice*)	0.402	(*afraid*, *said*)	0.998
(*dim*, *limb*)	0.497	(*queen*, *been*)	0.996

Table 3: Rhyming errors produced by the model. Examples on the left (right) side are rhyming (non-rhyming) word pairs — determined using the CMU dictionary — that have low (high) cosine similarity. “Cos” denote the system predicted cosine similarity for the word pair.

x-axis the characters of the sonnet line (punctuation removed). The attention network appears to perform very well, without any noticeable errors. The only minor exception is *lovely* in the second line, where it predicts 2 stresses but the second stress focuses incorrectly on the character *e* rather than *y*. Additional heatmaps for the full sonnet are provided in the supplementary material.

5.1.3 Rhyme Model

We follow a similar approach to evaluate the rhyme model against the CMU dictionary, but score based on F1 score. Word pairs that are not included in the dictionary are discarded. Rhyme is determined by extracting the final stressed phoneme for the paired words, and testing if their phoneme patterns match.

We predict rhyme for a word pair by feeding them to the rhyme model and computing cosine similarity; if a word pair is assigned a score $\geqslant 0.8$,[23] it is considered to rhyme. As a baseline (`Rhyme-BL`), we first extract for each word the last vowel and all following consonants, and predict a word pair as rhyming if their extracted sequences match. The extracted sequence can be interpreted as a proxy for the last syllable of a word.

Reddy and Knight (2011) propose an unsupervised model for learning rhyme schemes in poems via EM. There are two latent variables: ϕ specifies the distribution of rhyme schemes, and θ defines the pairwise rhyme strength between two words. The model's objective is to maximise poem likelihood over all possible rhyme scheme assignments under the latent variables ϕ and θ. We train this model (`Rhyme-EM`) on our data[24] and use the learnt θ to decide whether two words rhyme.[25]

Table 2 details the rhyming results. The rhyme model performs very strongly at F1 > 0.90, well above both baselines. `Rhyme-EM` performs poorly because it operates at the word level (i.e. it ignores character/orthographic information) and hence does not generalise well to unseen words and word pairs.[26]

To better understand the errors qualitatively, we present a list of word pairs with their predicted cosine similarity in Table 3. Examples on the left side are rhyming word pairs as determined by the CMU dictionary; right are non-rhyming pairs. Looking at the rhyming word pairs (left), it appears that these words tend not to share any word-ending characters. For the non-rhyming pairs, we spot several CMU errors: (*sire*, *ire*) and (*queen*, *been*) clearly rhyme.

5.2 Generation Evaluation

5.2.1 Crowdworker Evaluation

Following Hopkins and Kiela (2017), we present a pair of quatrains (one machine-generated and one human-written, in random order) to crowd workers on CrowdFlower, and ask them to guess which is the human-written poem. Generation quality is estimated by computing the accuracy of workers at correctly identifying the human-written poem (with lower values indicate better results for the model).

We generate 50 quatrains each for `LM`, `LM`** and `LM`**`+PM+RM` (150 in total), and as a control, generate 30 quatrains with `LM` trained for one epoch. An equal number of human-written quatrains was sampled from the training partition. A HIT contained 5 pairs of poems (of which one is a control), and workers were paid \$0.05 for each HIT. Workers who failed to identify the human-written poem in the control pair reliably (minimum accuracy $= 70\%$) were removed by CrowdFlower automati-

[23]0.8 is empirically found to be the best threshold based on development data.

[24]We use the original authors' implementation: `https://github.com/jvamvas/rhymediscovery`.

[25]A word pair is judged to rhyme if $\theta_{w_1,w_2} \geqslant 0.02$; the threshold (0.02) is selected based on development performance.

[26]Word pairs that did not co-occur in a poem in the training data have rhyme strength of zero.

Model	Accuracy
LM	0.742
LM**	0.672
LM**+PM+RM	0.532
LM**+RM	0.532

Table 4: Crowdworker accuracy performance.

Model	Meter	Rhyme	Read.	Emotion
LM	4.00±0.73	1.57±0.67	2.77±0.67	2.73±0.51
LM**	4.07±1.03	1.53±0.88	3.10±1.04	2.93±0.93
LM**+PM+RM	4.10±0.91	4.43±0.56	2.70±0.69	2.90±0.79
Human	3.87±1.12	4.10±1.35	4.80±0.48	4.37±0.71

Table 5: Expert mean and standard deviation ratings on several aspects of the generated quatrains.

cally, and they were restricted to do a maximum of 3 HITs. To dissuade workers from using search engines to identify real poems, we presented the quatrains as images.

Accuracy is presented in Table 4. We see a steady decrease in accuracy (= improvement in model quality) from LM to LM** to LM**+PM+RM, indicating that each model generates quatrains that are less distinguishable from human-written ones. Based on the suspicion that workers were using rhyme to judge the poems, we tested a second model, LM**+RM, which is the full model without the pentameter component. We found identical accuracy (0.532), confirming our suspicion that crowd workers depend on only rhyme in their judgements. These observations demonstrate that meter is largely ignored by lay persons in poetry evaluation.

5.2.2 Expert Judgement

To better understand the qualitative aspects of our generated quatrains, we asked an English literature expert (a Professor of English literature at a major English-speaking university; the last author of this paper) to directly rate 4 aspects: meter, rhyme, readability and emotion (i.e. amount of emotion the poem evokes). All are rated on an ordinal scale between 1 to 5 (1 = worst; 5 = best). In total, 120 quatrains were annotated, 30 each for LM, LM**, LM**+PM+RM, and human-written poems (Human). The expert was blind to the source of each poem. The mean and standard deviation of the ratings are presented in Table 5.

We found that our full model has the highest ratings for both rhyme and meter, even higher than human poets. This might seem surprising, but in fact it is well established that real poets regularly break rules of form to create other effects (Adams, 1997). Despite excellent form, the output of our model can easily be distinguished from human-written poetry due to its lower emotional impact and readability. In particular, there is evidence here that our focus on form actually hurts the readability of the resulting poems, relative even to the simpler language models. Another surprise is how well simple language models do in terms of their grasp of meter: in this expert evaluation, we see only marginal benefit as we increase the sophistication of the model. Taken as a whole, this evaluation suggests that future research should look beyond forms, towards the substance of good poetry.

6 Conclusion

We propose a joint model of language, meter and rhyme that captures language and form for modelling sonnets. We provide quantitative analyses for each component, and assess the quality of generated poems using judgements from crowdworkers and a literature expert. Our research reveals that vanilla LSTM language model captures meter implicitly, and our proposed rhyme model performs exceptionally well. Machine-generated generated poems, however, still underperform in terms of readability and emotion.

References

Stephen Adams. 1997. *Poetic designs: An introduction to meters, verse forms, and figures of speech.* Broadview Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, San Diego, USA.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Sebastopol, USA.

Julian Brooke, Adam Hammond, and Graeme Hirst. 2015. GutenTag: An NLP-driven tool for digital humanities research in the Project Gutenberg corpus. In *Proceedings of the 4nd Workshop on Computational Literature for Literature (CLFL '15).*

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar.

Keunwoo Choi, George Fazekas, and Mark Sandler. 2016. Text-based LSTM networks for automatic music composition. In *Proceedings of the 1st Conference on Computer Simulation of Musical Creativity*, Huddersfield, UK.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*, pages 103–111, Montreal, Canada.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, pages 95–102.

Luka Crnkovic-Friis and Louise Crnkovic-Friis. 2016. Generative choreography using deep learning. In *Proceedings of the 7th International Conference on Computational Creativity*, Paris, France.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Pablo Gervás. 2000. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects of AI*, pages 93–100.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. pages 1183–1191, Austin, Texas.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 524–533, Massachusetts, USA.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 168–178, Vancouver, Canada.

Eric J. Humphrey, Juan P. Bello, and Yann LeCun. 2013. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Y. Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751, Doha, Qatar.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Joel Lehman, Sebastian Risi, and Jeff Clune. 2016. Creative generation of 3D objects with deep learning and innovation engines. In *Proceedings of the 7th International Conference on Computational Creativity*, Paris, France.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations, 2013*, Scottsdale, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the EACL (EACL 2017)*, pages 157–163, Valencia, Spain.

Sravana Reddy and Kevin Knight. 2011. Unsupervised discovery of rhyme schemes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT 2011)*, pages 77–82, Portland, Oregon, USA.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1073–1083, Vancouver, Canada.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Bob L. Sturm, Jo ao Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. 2016. Music transcription modelling and composition using deep learning. In *Proceedings of the 1st Conference on Computer Simulation of Musical Creativity*, Huddersfield, UK.

Jukka M. Toivanen, Matti Järvisalo, and Hannu Toivonen. 2013. Harnessing constraint programming for poetry composition. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 160–160.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016. Chinese song iambics generation with neural attention-based model. In *Proceedings of the 25nd International Joint Conference on Artificial Intelligence (IJCAI-2016)*, pages 2943–2949, New York, USA.

Zhe Wang and Xiangyang Xue. 2014. In *Support Vector Machines Applications*, pages 23–48. Springer.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. Newhitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. *Entertainment Computing-ICEC 2009*, pages 191–196.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 670–680, Doha, Qatar.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104, Vancouver, Canada.

NeuralREG: An end-to-end approach to referring expression generation

Thiago Castro Ferreira[1] **Diego Moussallem**[2,3] **Ákos Kádár**[1] **Sander Wubben**[1] **Emiel Krahmer**[1]
[1]Tilburg center for Cognition and Communication (TiCC), Tilburg University, The Netherlands
[2]AKSW Research Group, University of Leipzig, Germany
[3]Data Science Group, University of Paderborn, Germany
{tcastrof,a.kadar,s.wubben,e.j.krahmer}@tilburguniversity.edu
moussallem@informatik.uni-leipzig.de

Abstract

Traditionally, Referring Expression Generation (REG) models first decide on the form and then on the content of references to discourse entities in text, typically relying on features such as salience and grammatical function. In this paper, we present a new approach (NeuralREG), relying on deep neural networks, which makes decisions about form and content in one go without explicit feature extraction. Using a delexicalized version of the WebNLG corpus, we show that the neural model substantially improves over two strong baselines. Data and models are publicly available[1].

1 Introduction

Natural Language Generation (NLG) is the task of automatically converting non-linguistic data into coherent natural language text (Reiter and Dale, 2000; Gatt and Krahmer, 2018). Since the input data will often consist of entities and the relations between them, generating references for these entities is a core task in many NLG systems (Dale and Reiter, 1995; Krahmer and van Deemter, 2012). Referring Expression Generation (REG), the task responsible for generating these references, is typically presented as a two-step procedure. First, the referential form needs to be decided, asking whether a reference at a given point in the text should assume the form of, for example, a proper name ("Frida Kahlo"), a pronoun ("she") or description ("the Mexican painter"). In addition, the REG model must account for the different ways in which a particular referential form can be realized. For example, both "Frida" and "Kahlo" are name-variants that may occur in a text, and she can alternatively also be described as, say, "the famous female painter".

Most of the earlier REG approaches focus either on selecting referential form (Orita et al., 2015; Castro Ferreira et al., 2016), or on selecting referential content, typically zooming in on one specific kind of reference such as a pronoun (e.g., Henschel et al., 2000; Callaway and Lester, 2002), definite description (e.g., Dale and Haddock, 1991; Dale and Reiter, 1995) or proper name generation (e.g., Siddharthan et al., 2011; van Deemter, 2016; Castro Ferreira et al., 2017b). Instead, in this paper, we propose NeuralREG: an end-to-end approach addressing the full REG task, which given a number of entities in a text, produces corresponding referring expressions, simultaneously selecting both form and content. Our approach is based on neural networks which generate referring expressions to discourse entities relying on the surrounding linguistic context, without the use of any feature extraction technique.

Besides its use in traditional pipeline NLG systems (Reiter and Dale, 2000), REG has also become relevant in modern "end-to-end" NLG approaches, which perform the task in a more integrated manner (see e.g. Konstas et al., 2017; Gardent et al., 2017b). Some of these approaches have recently focused on inputs which references to entities are delexicalized to general tags (e.g., ENTITY-1, ENTITY-2) in order to decrease data sparsity. Based on the delexicalized input, the model generates outputs which may be likened to templates in which references to the discourse entities are not realized (as in "The ground of ENTITY-1 is located in ENTITY-2.").

While our approach, dubbed as NeuralREG, is compatible with different applications of REG models, in this paper, we concentrate on the last one, relying on a specifically constructed set of

[1]https://github.com/ThiagoCF05/NeuralREG

78,901 referring expressions to 1,501 entities in the context of the semantic web, derived from a (delexicalized) version of the WebNLG corpus (Gardent et al., 2017a,b). Both this data set and the model will be made publicly available. We compare NeuralREG against two baselines in an automatic and human evaluation, showing that the integrated neural model is a marked improvement.

2 Related work

In recent years, we have seen a surge of interest in using (deep) neural networks for a wide range of NLG-related tasks, as the generation of (first sentences of) Wikipedia entries (Lebret et al., 2016), poetry (Zhang and Lapata, 2014), and texts from abstract meaning representations (e.g., Konstas et al., 2017; Castro Ferreira et al., 2017a). However, the usage of deep neural networks for REG has remained limited and we are not aware of any other integrated, end-to-end model for generating referring expressions in discourse.

There is, however, a lot of earlier work on selecting the form and content of referring expressions, both in psycholinguistics and in computational linguistics. In psycholinguistic models of reference, various linguistic factors have been proposed as influencing the form of referential expressions, including cognitive status (Gundel et al., 1993), centering (Grosz et al., 1995) and information density (Jaeger, 2010). In models such as these, notions like *salience* play a central role, where it is assumed that entities which are salient in the discourse are more likely to be referred to using shorter referring expressions (like a pronoun) than less salient entities, which are typically referred to using longer expressions (like full proper names).

Building on these ideas, many REG models for generating references in texts also strongly rely on the concept of salience and factors contributing to it. Reiter and Dale (2000) for instance, discussed a straightforward rule-based method based on this notion, stating that full proper names can be used for initial references, typically less salient than subsequent references, which, according to the study, can be realized by a pronoun in case there is no mention to any other entity of same person, gender and number between the reference and its antecedents. More recently, Castro Ferreira et al. (2016) proposed a data-driven, non-deterministic model for generating referential forms, taking into account salience features extracted from the discourse such as grammatical position, givenness and recency of the reference. Importantly, these models do not specify which contents a particular reference, be it a proper name or description, should have. To this end, separate models are typically used, including, for example, Dale and Reiter (1995) for generating descriptions, and Siddharthan et al. (2011); van Deemter (2016) for proper names.

Of course, when texts are generated in practical settings, both form and content need to be chosen. This was the case, for instance, in the GREC shared task (Belz et al., 2010), which aimed to evaluate models for automatically generated referring expressions grounded in discourse. The input for the models were texts in which the referring expressions to the topic of the relevant Wikipedia entry were removed and appropriate references throughout the text needed to be generated (by selecting, for each gap, from a list of candidate referring expressions of different forms and with different contents). Some participating systems approached this with traditional pipelines for selecting referential form, followed by referential content, while others proposed more integrated methods. More details about the models can be seen on Belz et al. (2010).

In sum, existing REG models for text generation strongly rely on abstract features such as the salience of a referent for deciding on the form or content of a referent. Typically, these features are extracted automatically from the context, and engineering relevant ones can be complex. Moreover, many of these models only address part of the problem, either concentrating on the choice of referential form or on deciding on the contents of, for example, proper names or definite descriptions. In contrast, we introduce NeuralREG, an end-to-end approach based on neural networks which generates referring expressions to discourse entities directly from a delexicalized/wikified text fragment, without the use of any feature extraction technique. Below we describe our model in more detail, as well as the data on which we develop and evaluate it.

3 Data and processing

3.1 WebNLG corpus

Our data is based on the WebNLG corpus (Gardent et al., 2017a), which is a parallel resource ini-

Subject	**Predicate**	**Object**
108_St_Georges_Terrace	location	Perth
Perth	country	Australia
108_St_Georges_Terrace	completionDate	1988@*year*
108_St_Georges_Terrace	cost	120 million (Australian dollars)@*USD*
108_St_Georges_Terrace	floorCount	50@*Integer*

↓

108 St Georges Terrace was completed in 1988 in Perth, Australia. It has a total of 50 floors and cost 120m Australian dollars.

Figure 1: Example of a set of triples (top) and corresponding text (bottom).

tially released for the eponymous NLG challenge. In this challenge, participants had to automatically convert non-linguistic data from the Semantic Web into a textual format (Gardent et al., 2017b). The source side of the corpus are sets of *Resource Description Framework* (RDF) triples. Each RDF triple is formed by a Subject, Predicate and Object, where the Subject and Object are constants or Wikipedia entities, and predicates represent a relation between these two elements in the triple. The target side contains English texts, obtained by *crowdsourcing*, which describe the source triples. Figure 1 depicts an example of a set of 5 RDF triples and the corresponding text.

The corpus consists of 25,298 texts describing 9,674 sets of up to 7 RDF triples (an average of 2.62 texts per set) in 15 domains (Gardent et al., 2017b). In order to be able to train and evaluate our models for referring expression generation (the topic of this study), we produced a delexicalized version of the original corpus.

3.2 Delexicalized WebNLG

We delexicalized the training and development parts of the WebNLG corpus by first automatically mapping each entity in the source representation to a general tag. All entities that appear on the left and right side of the triples were mapped to AGENTs and PATIENTs, respectively. Entities which appear on both sides in the relations of a set were represented as BRIDGEs. To distinguish different AGENTs, PATIENTs and BRIDGEs in a set, an ID was given to each entity of each kind (PATIENT-1, PATIENT-2, etc.). Once all entities in the text were mapped to different roles, the first two authors of this study manually replaced the referring expressions in the original target texts by their respective tags. Figure 2 shows the entity mapping and the delexicalized template for the example in Figure 1 in its versions representing the references with general tags and Wikipedia IDs.

We delexicalized 20,198 distinct texts describing 7,812 distinct sets of RDF triples, resulting in 16,628 distinct templates. While this dataset (which we make available) has various uses, we used it to extract a collection of referring expressions to Wikipedia entities in order to evaluate how well our REG model can produce references to entities throughout a (small) text.

3.3 Referring expression collection

Using the delexicalized version of the WebNLG corpus, we automatically extracted all referring expressions by tokenizing the original and delexicalized versions of the texts and then finding the non overlapping items. For instance, by processing the text in Figure 1 and its delexicalized template in Figure 2, we would extract referring expressions like "108 St Georges Terrace" and "It" to ⟨ *AGENT-1, 108_St_Georges_Terrace* ⟩, "Perth" to ⟨ *BRIDGE-1, Perth* ⟩, "Australia" to ⟨ *PATIENT-1, Australia* ⟩ and so on.

Once all texts were processed and the referring expressions extracted, we filtered only the ones referring to Wikipedia entities, removing references to constants like dates and numbers, for which no references are generated by the model. In total, the final version of our dataset contains 78,901 referring expressions to 1,501 Wikipedia entities, in which 71.4% (56,321) are proper names, 5.6% (4,467) pronouns, 22.6% (17,795) descriptions and 0.4% (318) demonstrative referring expressions. We split this collection in training, developing and test sets, totaling 63,061, 7,097 and 8,743 referring expressions in each one of them.

Each instance of the final dataset consists of a truecased tokenized referring expression, the target entity (distinguished by its Wikipedia ID), and the discourse context preceding and following the relevant reference (we refer to these as the pre- and pos-context). Pre- and pos-contexts are the lowercased, tokenized and delexicalized

Tag	**Entity**
AGENT-1	108_St_Georges_Terrace
BRIDGE-1	Perth
PATIENT-1	Australia
PATIENT-2	1988@*year*
PATIENT-3	"120 million (Australian dollars)"@*USD*
PATIENT-4	50@*Integer*

AGENT-1 was completed in **PATIENT-2** in **BRIDGE-1** , **PATIENT-1** . **AGENT-1** has a total of **PATIENT-4** floors and cost **PATIENT-3** .

$\downarrow_{Wiki}$

108_St_Georges_Terrace was completed in **1988** in **Perth** , **Australia** . **108_St_Georges_Terrace** has a total of **50** floors and cost **20_million_(Australian_dollars)** .

Figure 2: Mapping between tags and entities for the related delexicalized/wikified templates.

pieces of text before and after the target reference. References to other discourse entities in the pre- and pos-contexts are represented by their Wikipedia ID, whereas constants (numbers, dates) are represented by a one-word ID removing quotes and replacing white spaces with underscores (e.g., *120_million_(Australian_dollars)* for "120 million (Australian dollars)" in Figure 2).

Although the references to discourse entities are represented by general tags in a delexicalized template produced in the generation process (AGENT-1, BRIDGE-1, etc.), for the purpose of disambiguation, NeuralREG's inputs have the references represented by the Wikipedia ID of their entities. In this context, it is important to observe that the conversion of the general tags to the Wikipedia IDs can be done in constant time during the generation process, since their mapping, like the first representation in Figure 2, is the first step of the process. In the next section, we show in detail how NeuralREG models the problem of generating a referring expression to a discourse entity.

4 NeuralREG

NeuralREG aims to generate a referring expression $y = \{y_1, y_2, ..., y_T\}$ with T tokens to refer to a target entity token $x^{(wiki)}$ given a discourse pre-context $X^{(pre)} = \{x_1^{(pre)}, x_2^{(pre)}, ..., x_m^{(pre)}\}$ and pos-context $X^{(pos)} = \{x_1^{(pos)}, x_2^{(pos)}, ..., x_l^{(pos)}\}$ with m and l tokens, respectively. The model is implemented as a multi-encoder, attention-decoder network with bidirectional (Schuster and Paliwal, 1997) Long-Short Term Memory Layers (LSTM) (Hochreiter and Schmidhuber, 1997) sharing the same input word-embedding matrix V, as explained further.

4.1 Context encoders

Our model starts by encoding the pre- and pos-contexts with two separate bidirectional LSTM encoders (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997). These modules learn feature representations of the text surrounding the target entity $x^{(wiki)}$, which are used for the referring expression generation. The pre-context $X^{(pre)} = \{x_1^{(pre)}, x_2^{(pre)}, ..., x_m^{(pre)}\}$ is represented by forward and backward hidden-state vectors $(\overrightarrow{h}_1^{(pre)}, \cdots, \overrightarrow{h}_m^{(pre)})$ and $(\overleftarrow{h}_1^{(pre)}, \cdots, \overleftarrow{h}_m^{(pre)})$. The final annotation vector for each encoding timestep t is obtained by the concatenation of the forward and backward representations $h_t^{(pre)} = [\overrightarrow{h}_t^{(pre)}, \overleftarrow{h}_t^{(pre)}]$. The same process is repeated for the pos-context resulting in representations $(\overrightarrow{h}_1^{(pos)}, \cdots, \overrightarrow{h}_l^{(pos)})$ and $(\overleftarrow{h}_1^{(pos)}, \cdots, \overleftarrow{h}_l^{(pos)})$ and annotation vectors $h_t^{(pos)} = [\overrightarrow{h}_t^{(pos)}, \overleftarrow{h}_t^{(pos)}]$. Finally, the encoding of target entity $x^{(wiki)}$ is simply its entry in the shared input word-embedding matrix V_{wiki}.

4.2 Decoder

The referring expression generation module is an LSTM decoder implemented in 3 different versions: `Seq2Seq`, `CAtt` and `HierAtt`. All decoders at each timestep i of the generation process take as input features their previous state s_{i-1}, the target entity-embedding V_{wiki}, the embedding of the previous word of the referring expression $V_{y_{i-1}}$ and finally the summary vector of the pre- and pos-contexts c_i. The difference between the decoder variations is the method to compute c_i.

`Seq2Seq` models the context vector c_i at each timestep i concatenating the pre- and pos-context annotation vectors averaged over time:

$$\hat{h}^{(pre)} = \frac{1}{N}\sum_{i}^{N} h_i^{(pre)} \tag{1}$$

$$\hat{h}^{(pos)} = \frac{1}{N}\sum_{i}^{N} h_i^{(pos)} \tag{2}$$

$$c_i = [\hat{h}^{(pre)}, \hat{h}^{(pos)}] \tag{3}$$

CAtt is an LSTM decoder augmented with an attention mechanism (Bahdanau et al., 2015) over the pre- and pos-context encodings, which is used to compute c_i at each timestep. We compute energies $e_{ij}^{(pre)}$ and $e_{ij}^{(pos)}$ between encoder states $h_i^{(pre)}$ and $h_i^{(post)}$ and decoder state s_{i-1}. These scores are normalized through the application of the softmax function to obtain the final attention probability $\alpha_{ij}^{(pre)}$ and $\alpha_{ij}^{(post)}$. Equations 4 and 5 summarize the process with k ranging over the two encoders ($k \in [pre, pos]$), being the projection matrices $W_a^{(k)}$ and $U_a^{(k)}$ and attention vectors $v_a^{(k)}$ trained parameters.

$$e_{ij}^{(k)} = v_a^{(k)T} \tanh(W_a^{(k)} s_{i-1} + U_a^{(k)} h_j^{(k)}) \tag{4}$$

$$\alpha_{ij}^{(k)} = \frac{\exp(e_{ij}^{(k)})}{\sum_{n=1}^{N} \exp(e_{in}^{(k)})} \tag{5}$$

In general, the attention probability $\alpha_{ij}^{(k)}$ determines the amount of contribution of the jth token of k-context in the generation of the ith token of the referring expression. In each decoding step i, a final summary-vector for each context $c_i^{(k)}$ is computed by summing the encoder states $h_j^{(k)}$ weighted by the attention probabilities $\alpha_i^{(k)}$:

$$c_i^{(k)} = \sum_{j=1}^{N} \alpha_{ij}^{(k)} h_j^{(k)} \tag{6}$$

To combine $c_i^{(pre)}$ and $c_i^{(pos)}$ into a single representation, this model simply concatenate the pre- and pos-context summary vectors $c_i = [c_i^{(pre)}, c_i^{(pos)}]$.

HierAtt implements a second attention mechanism inspired by Libovický and Helcl (2017) in order to generate attention weights for the pre- and pos-context summary-vectors $c_i^{(pre)}$ and $c_i^{(pos)}$ instead of concatenate them. Equations 7, 8 and 9 depict the process, being the projection matrices $W_b^{(k)}$ and $U_b^{(k)}$ as well as attention vectors $v_b^{(k)}$ trained parameters ($k \in [pre, pos]$).

$$e_i^{(k)} = v_b^{(k)T} \tanh(W_b^{(k)} s_{i-1} + U_b^{(k)} c_i^{(k)}) \tag{7}$$

$$\beta_i^{(k)} = \frac{\exp(e_i^{(k)})}{\sum_n \exp(e_i^{(n)})} \tag{8}$$

$$c_i = \sum_k \beta_i^{(k)} U_b^{(k)} c_i^{(k)} \tag{9}$$

Decoding Given the summary-vector c_i, the embedding of the previous referring expression token $V_{y_{i-1}}$, the previous decoder state s_{i-1} and the entity-embedding V_{wiki}, the decoders predict their next state which later is used to compute a probability distribution over the tokens in the output vocabulary for the next timestep as Equations 10 and 11 show.

$$s_i = \Phi_{\text{dec}}(s_{i-1}, [c_i, V_{y_{i-1}}, V_{wiki}]) \tag{10}$$

$$p(y_i | y_{<i}, X^{(pre)}, x^{(wiki)}, X^{(pos)}) = \text{softmax}(W_c s_i + b) \tag{11}$$

In Equation 10, s_0 and c_0 are zero-initialized vectors. In order to find the referring expression y that maximizes the likelihood in Equation 11, we apply a beam search with length normalization with $\alpha = 0.6$ (Wu et al., 2016):

$$lp(y) = \frac{(5 + |y|)^\alpha}{(5+1)^\alpha} \tag{12}$$

The decoder is trained to minimize the negative log likelihood of the next token in the target referring expression:

$$J(\theta) = -\sum_i \log \text{p}(y_i | y_{<i}, X^{(pre)}, x^{(wiki)}, X^{(pos)}) \tag{13}$$

5 Models for Comparison

We compared the performance of NeuralREG against two baselines: *OnlyNames* and a model based on the choice of referential form method of Castro Ferreira et al. (2016), dubbed *Ferreira*.

OnlyNames is motivated by the similarity among the Wikipedia ID of an element and a proper name reference to it. This method refers to each entity by their Wikipedia ID, replacing each underscore in the ID for whitespaces (e.g., *Appleton_International_Airport* to *"Appleton International Airport"*).

Ferreira works by first choosing whether a reference should be a proper name, pronoun, description or demonstrative. The choice is made by a Naive Bayes method as Equation 14 depicts.

$$P(f \mid X) \propto \frac{P(f) \prod\limits_{x \in X} P(x \mid f)}{\sum\limits_{f' \in F} P(f') \prod\limits_{x \in X} P(x \mid f')} \tag{14}$$

The method calculates the likelihood of each referential form f given a set of features X, consisting of grammatical position and information status (new or given in the text and sentence). Once the choice of referential form is made, the most frequent variant is chosen in the training corpus given the referent, syntactic position and information status. In case a referring expression for a wiki target is not found in this way, a backoff method is applied by removing one factor at a time in the following order: sentence information status, text information status and grammatical position. Finally, if a referring expression is not found in the training set for a given entity, the same method as *OnlyNames* is used. Regarding the features, syntactic position distinguishes whether a reference is the subject, object or subject determiner (genitive) in a sentence. Text and sentence information statuses mark whether a reference is a initial or a subsequent mention to an entity in the text and the sentence, respectively. All features were extracted automatically from the texts using the sentence tokenizer and dependency parser of Stanford CoreNLP (Manning et al., 2014).

6 Automatic evaluation

Data We evaluated our models on the training, development and test referring expression sets described in Section 3.3.

Metrics We compared the referring expressions produced by the evaluated models with the gold-standards ones using accuracy and String Edit Distance (Levenshtein, 1966). Since pronouns are highlighted as the most likely referential form to be used when a referent is salient in the discourse, as argued in the introduction, we also computed pronoun accuracy, precision, recall and F1-score in order to evaluate the performance of the models for capturing discourse salience. Finally, we lexicalized the original templates with the referring expressions produced by the models and compared them with the original texts in the corpus using accuracy and BLEU score (Papineni et al., 2002) as a measure of fluency. Since our model does not handle referring expressions for constants (dates and numbers), we just copied their source version into the template.

Post-hoc McNemar's and Wilcoxon signed ranked tests adjusted by the Bonferroni method were used to test the statistical significance of the models in terms of accuracy and string edit distance, respectively. To test the statistical significance of the BLEU scores of the models, we used a bootstrap resampling together with an approximate randomization method (Clark et al., 2011)[2].

Settings NeuralREG was implemented using Dynet (Neubig et al., 2017). Source and target word embeddings were 300D each and trained jointly with the model, whereas hidden units were 512D for each direction, totaling 1024D in the bidirection layers. All non-recurrent matrices were initialized following the method of Glorot and Bengio (2010). Models were trained using stochastic gradient descent with Adadelta (Zeiler, 2012) and mini-batches of size 40. We ran each model for 60 epochs, applying early stopping for model selection based on accuracy on the development set with patience of 20 epochs. For each decoding version (`Seq2Seq`, `CAtt` and `HierAtt`), we searched for the best combination of drop-out probability of 0.2 or 0.3 in both the encoding and decoding layers, using beam search with a size of 1 or 5 with predictions up to 30 tokens or until 2 ending tokens were predicted (*EOS*). The results described in the next section were obtained on the test set by the NeuralREG version with the highest accuracy on the development set over the epochs.

Results Table 1 summarizes the results for all models on all metrics on the test set and Table 2 depicts a text example lexicalized by each model. The first thing to note in the results of the first table is that the baselines in the top two rows performed quite strong on this task, generating more than half of the referring expressions exactly as in the gold-standard. The method based on Castro Ferreira et al. (2016) performed statistically better than *OnlyNames* on all metrics due to its capability, albeit to a limited extent, to predict pronominal references (which *OnlyNames* obviously cannot).

We reported results on the test set for NeuralREG+`Seq2Seq` and NeuralREG+`CAtt` using

[2]`https://github.com/jhclark/multeval`

	All References		Pronouns				Text	
	Acc.	SED	Acc.	Prec.	Rec.	F-Score	Acc.	BLEU
OnlyNames	0.53^{D}	4.05^{D}	-	-	-	-	0.15^{D}	69.03^{D}
Ferreira	0.61^{C}	3.18^{C}	0.43^{B}	0.57	0.54	0.55	0.19^{C}	72.78^{C}
NeuralREG+`Seq2Seq`	$0.74^{A,B}$	$2.32^{A,B}$	0.75^{A}	0.77	0.78	0.78	0.28^{B}	$79.27^{A,B}$
NeuralREG+`CAtt`	0.74^{A}	2.25^{A}	0.75^{A}	0.73	0.78	0.75	0.30^{A}	79.39^{A}
NeuralREG+`HierAtt`	0.73^{B}	2.36^{B}	0.73^{A}	0.74	0.77	0.75	$0.28^{A,B}$	79.01^{B}

Table 1: (1) Accuracy (Acc.) and String Edit Distance (SED) results in the prediction of all referring expressions; (2) Accuracy (Acc.), Precision (Prec.), Recall (Rec.) and F-Score results in the prediction of pronominal forms; and (3) Accuracy (Acc.) and BLEU score results of the texts with the generated referring expressions. Rankings were determined by statistical significance.

dropout probability 0.3 and beam size 5, and NeuralREG+`HierAtt` with dropout probability of 0.3 and beam size of 1 selected based on the highest accuracy on the development set. Importantly, the three NeuralREG variant models statistically outperformed the two baseline systems. They achieved BLEU scores, text and referential accuracies as well as string edit distances in the range of 79.01-79.39, 28%-30%, 73%-74% and 2.25-2.36, respectively. This means that NeuralREG predicted 3 out of 4 references completely correct, whereas the incorrect ones needed an average of 2 post-edition operations in character level to be equal to the gold-standard. When considering the texts lexicalized with the referring expressions produced by NeuralREG, at least 28% of them are similar to the original texts. Especially noteworthy was the score on pronoun accuracy, indicating that the model was well capable of predicting when to generate a pronominal reference in our dataset.

The results for the different decoding methods for NeuralREG were similar, with the NeuralREG+`CAtt` performing slightly better in terms of the BLEU score, text accuracy and String Edit Distance. The more complex NeuralREG+`HierAtt` yielded the lowest results, even though the differences with the other two models were small and not even statistically significant in many of the cases.

7 Human Evaluation

Complementary to the automatic evaluation, we performed an evaluation with human judges, comparing the quality judgments of the original texts to the versions generated by our various models.

Material We quasi-randomly selected 24 instances from the delexicalized version of the WebNLG corpus related to the test part of the referring expression collection. For each of the selected instances, we took into account its source triple set and its 6 target texts: one original (randomly chosen) and its versions with the referring expressions generated by each of the 5 models introduced in this study (two baselines, three neural models). Instances were chosen following 2 criteria: the number of triples in the source set (ranging from 2 to 7) and the differences between the target texts.

For each size group, we randomly selected 4 instances (of varying degrees of variation between the generated texts) giving rise to 144 trials (= 6 triple set sizes $*$ 4 instances $*$ 6 text versions), each consisting of a set of triples and a target text describing it with the lexicalized referring expressions highlighted in yellow.

Method The experiment had a latin-square design, distributing the 144 trials over 6 different lists such that each participant rated 24 trials, one for each of the 24 corpus instances, making sure that participants saw equal numbers of triple set sizes and generated versions. Once introduced to a trial, the participants were asked to rate the fluency ("does the text flow in a natural, easy to read manner?"), grammaticality ("is the text grammatical (no spelling or grammatical errors)?") and clarity ("does the text clearly express the data?") of each target text on a 7-Likert scale, focussing on the highlighted referring expressions. The experiment is available on the website of the author[3].

Participants We recruited 60 participants, 10 per list, via Mechanical Turk. Their average age was 36 years and 27 of them were females. The majority declared themselves native speakers of

[3]https://ilk.uvt.nl/~tcastrof/acl2018/evaluation/

Model	Text
OnlyNames	**alan shepard** was born in **new hampshire** on **1923-11-18** . before **alan shepard** death in **california** **alan shepard** had been awarded **distinguished service medal (united states navy)** an award higher than **department of commerce gold medal** .
Ferreira	**alan shepard** was born in **new hampshire** on **1923-11-18** . before **alan shepard** death in **california** **him** had been awarded **distinguished service medal** an award higher than **department of commerce gold medal** .
`Seq2Seq`	**alan shepard** was born in **new hampshire** on **1923-11-18** . before **his** death in **california him** had been awarded **the distinguished service medal by the united states navy** an award higher than **the department of commerce gold medal** .
`CAtt`	**alan shepard** was born in **new hampshire** on **1923-11-18** . before **his** death in **california he** had been awarded **the distinguished service medal by the us navy** an award higher than **the department of commerce gold medal** .
`HierAtt`	**alan shephard** was born in **new hampshire** on **1923-11-18** . before **his** death in **california he** had been awarded **the distinguished service medal** an award higher than **the department of commerce gold medal** .
Original	**alan shepard** was born in **new hampshire** on **18 november 1923** . before **his** death in **california he** had been awarded **the distinguished service medal by the us navy** an award higher than **the department of commerce gold medal** .

Table 2: Example of text with references lexicalized by each model.

	Fluency	Grammar	Clarity
OnlyNames	4.74^{C}	4.68^{B}	4.90^{B}
Ferreira	4.74^{C}	4.58^{B}	4.93^{B}
NeuralREG+`Seq2Seq`	$4.95^{B,C}$	$4.82^{A,B}$	4.97^{B}
NeuralREG+`CAtt`	$5.23^{A,B}$	$4.95^{A,B}$	$5.26^{A,B}$
NeuralREG+`HierAtt`	$5.07^{B,C}$	$4.90^{A,B}$	$5.13^{A,B}$
Original	5.41^{A}	5.17^{A}	5.42^{A}

Table 3: Fluency, Grammaticality and Clarity results obtained in the human evaluation. Rankings were determined by statistical significance.

English (44), while 14 and 2 self-reported as fluent or having a basic proficiency, respectively.

Results Table 3 summarizes the results. Inspection of the Table reveals a clear pattern: all three neural models scored higher than the baselines on all metrics, with especially NeuralREG+`CAtt` approaching the ratings for the original sentences, although – again – differences between the neural models were small. Concerning the size of the triple sets, we did not find any clear pattern.

To test the statistical significance of the pairwise comparisons, we used the Wilcoxon signed-rank test corrected for multiple comparisons using the Bonferroni method. Different from the automatic evaluation, the results of both baselines were not statistically significant for the three metrics. In comparison with the neural models, NeuralREG+`CAtt` significantly outperformed the baselines in terms of fluency, whereas the other comparisons between baselines and neural models were not statistically significant. The results for the 3 different decoding methods of NeuralREG also did not reveal a significant difference. Finally, the original texts were rated significantly higher than both baselines in terms of the three metrics, also than NeuralREG+`Seq2Seq` and NeuralREG+`HierAtt` in terms of fluency, and than NeuralREG+`Seq2Seq` in terms of clarity.

8 Discussion

This study introduced NeuralREG, an end-to-end approach based on neural networks which tackles the full Referring Expression Generation process. It generates referring expressions for discourse entities by simultaneously selecting form and content without any need of feature extraction techniques. The model was implemented using an encoder-decoder approach where a target referent and its surrounding linguistic contexts were first encoded and combined into a single vector representation which subsequently was decoded into a referring expression to the target, suitable for the specific discourse context. In an automatic evaluation on a collection of 78,901 referring expressions to 1,501 Wikipedia entities, the different versions of the model all yielded better results than the two (competitive) baselines. Later in a complementary human evaluation, the texts with referring expressions generated by a variant of our novel model were considered statistically more fluent than the texts lexicalized by the two baselines.

Data The collection of referring expressions used in our experiments was extracted from a novel, delexicalized and publicly available version of the WebNLG corpus (Gardent et al., 2017a,b), where the discourse entities were replaced with general tags for decreasing the data sparsity. Besides the REG task, these data can be useful for many other tasks related to, for instance, the NLG process (Reiter and Dale, 2000; Gatt and Krahmer, 2018) and Wikification (Moussallem et al., 2017).

Baselines We introduced two strong baselines which generated roughly half of the referring expressions identical to the gold standard in an automatic evaluation. These baselines performed relatively well because they frequently generated full names, which occur often for our wikified references. However, they performed poorly when it came to pronominalization, which is an important ingredient for fluent, coherent text. *OnlyNames*, as the name already reveals, does not manage to generate any pronouns. However, the approach of Castro Ferreira et al. (2016) also did not perform well in the generation of pronouns, revealing a poor capacity to detect highly salient entities in a text.

NeuralREG was implemented with 3 different decoding architectures: `Seq2Seq`, `CAtt` and `HierAtt`. Although all the versions performed relatively similar, the concatenative-attention (`CAtt`) version generated the closest referring expressions from the gold-standard ones and presented the highest textual accuracy in the automatic evaluation. The texts lexicalized by this variant were also considered statistically more fluent than the ones generated by the two proposed baselines in the human evaluation.

Surprisingly, the most complex variant (`HierAtt`) with a hierarchical-attention mechanism gave lower results than `CAtt`, producing lexicalized texts which were rated as less fluent than the original ones and not significantly more fluent from the ones generated by the baselines. This result appears to be not consistent with the findings of Libovický and Helcl (2017), who reported better results on multi-modal machine translation with hierarchical-attention as opposed to the flat variants (Specia et al., 2016).

Finally, our NeuralREG variant with the lowest results were our 'vanilla' sequence-to-sequence (`Seq2Seq`), whose the lexicalized texts were significantly less fluent and clear than the original ones. This shows the importance of the attention mechanism in the decoding step of NeuralREG in order to generate fine-grained referring expressions in discourse.

Conclusion We introduced a deep learning model for the generation of referring expressions in discourse texts. NeuralREG decides both on referential form and on referential content in an integrated, end-to-end approach, without using explicit features. Using a new delexicalized version of the WebNLG corpus (made publicly available), we showed that the neural model substantially improves over two strong baselines in terms of accuracy of the referring expressions and fluency of the lexicalized texts.

Acknowledgments

This work has been supported by the National Council of Scientific and Technological Development from Brazil (CNPq) under the grants 203065/2014-0 and 206971/2014-1.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate.

Anja Belz, Eric Kow, Jette Viethen, and Albert Gatt. 2010. Generating referring expressions in context: The GREC task evaluation challenges. In Emiel Krahmer and Mariët Theune, editors, *Empirical Methods in Natural Language Generation*, pages 294–327. Springer-Verlag, Berlin, Heidelberg.

Charles B. Callaway and James C. Lester. 2002. Pronominalization in generated discourse and dialogue. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL'02, pages 88–95, Philadelphia, Pennsylvania. Association for Computational Linguistics.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017a. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, INLG'17, pages 1–10, Santiago de Compostela, Spain. Association for Computational Linguistics.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2016. Towards more variation in text generation: Developing and evaluating variation models for choice of referential form. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL'16, pages 568—-577, Berlin, Germany. Association for Computational Linguistics.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2017b. Generating flexible proper name references in text: Data, models and evaluation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, EACL'17, pages 655–664, Valencia, Spain. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, ACL'11, pages 176–181, Portland, Oregon.

Robert Dale and Nicholas Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of the fifth conference on European chapter of the Association for Computational Linguistics*, EACL'91, pages 161–166, Berlin, Germany. Association for Computational Linguistics.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263.

Kees van Deemter. 2016. Designing algorithms for referring with proper names. In *Proceedings of the 9th International Natural Language Generation conference*, INLG'16, pages 31–35, Edinburgh, UK. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL'17, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, INLG'17, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Jeanette K Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, pages 274–307.

Renate Henschel, Hua Cheng, and Massimo Poesio. 2000. Pronominalization revisited. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING'00, pages 306–312, Saarbrücken, Germany. Association for Computational Linguistics.

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

T Florian Jaeger. 2010. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive psychology*, 61(1):23–62.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL'17, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP'16, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Jindřich Libovický and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL'17, pages 196–202, Vancouver, Canada. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

D. Moussallem, R. Usbeck, M. Röder, and A.-C. Ngonga Ngomo. 2017. MAG: A Multilingual, Knowledge-base Agnostic and Deterministic Entity Linking Approach. *ArXiv e-prints*.

G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, K. Duh, M. Faruqui, C. Gan, D. Garrette, Y. Ji, L. Kong, A. Kuncoro, G. Kumar, C. Malaviya, P. Michel, Y. Oda, M. Richardson, N. Saphra, S. Swayamdipta, and P. Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *ArXiv e-prints*.

Naho Orita, Eliana Vornov, Naomi Feldman, and Hal Daumé III. 2015. Why discourse affects speakers' choice of referring expressions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, ACL'15, pages 1639–1649, Beijing, China. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, ACL'02, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 543–553, Berlin, Germany. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

Stock Movement Prediction from Tweets and Historical Prices

Yumo Xu and **Shay B. Cohen**

School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
yumo.xu@ed.ac.uk, scohen@inf.ed.ac.uk

Abstract

Stock movement prediction is a challenging problem: the market is highly *stochastic*, and we make *temporally-dependent* predictions from *chaotic* data. We treat these three complexities and present a novel deep generative model jointly exploiting text and price signals for this task. Unlike the case with discriminative or topic modeling, our model introduces recurrent, continuous latent variables for a better treatment of stochasticity, and uses neural variational inference to address the intractable posterior inference. We also provide a hybrid objective with temporal auxiliary to flexibly capture predictive dependencies. We demonstrate the state-of-the-art performance of our proposed model on a new stock movement prediction dataset which we collected.[1]

1 Introduction

Stock movement prediction has long attracted both investors and researchers (Frankel, 1995; Edwards et al., 2007; Bollen et al., 2011; Hu et al., 2018). We present a model to predict stock price movement from tweets and historical stock prices.

In natural language processing (NLP), public news and social media are two primary content resources for stock market prediction, and the models that use these sources are often discriminative. Among them, classic research relies heavily on feature engineering (Schumaker and Chen, 2009; Oliveira et al., 2013). With the prevalence of deep neural networks (Le and Mikolov, 2014), event-driven approaches were studied with structured event representations (Ding et al., 2014, 2015). More recently, Hu et al. (2018) propose to mine news sequence directly from text with hierarchical attention mechanisms for stock trend prediction.

However, stock movement prediction is widely considered difficult due to the high stochasticity of the market: stock prices are largely driven by new information, resulting in a random-walk pattern (Malkiel, 1999). Instead of using only deterministic features, generative topic models were extended to jointly learn topics and sentiments for the task (Si et al., 2013; Nguyen and Shirai, 2015). Compared to discriminative models, generative models have the natural advantage in depicting the generative process from market information to stock signals and introducing randomness. However, these models underrepresent chaotic social texts with bag-of-words and employ simple discrete latent variables.

In essence, stock movement prediction is a time series problem. The significance of the temporal dependency between movement predictions is not addressed in existing NLP research. For instance, when a company suffers from a major scandal on a trading day d_1, generally, its stock price will have a downtrend in the coming trading days until day d_2, i.e. $[d_1, d_2]$.[2] If a stock predictor can recognize this decline pattern, it is likely to benefit all the predictions of the movements during $[d_1, d_2]$. Otherwise, the accuracy in this interval might be harmed. This predictive dependency is a result of the fact that public information, e.g. a company scandal, needs time to be absorbed into movements over time (Luss and d'Aspremont, 2015), and thus is largely shared across temporally-close predictions.

Aiming to tackle the above-mentioned outstanding research gaps in terms of modeling *high market stochasticity*, *chaotic market information* and *temporally-dependent prediction*, we propose

[1] https://github.com/yumoxu/stocknet-dataset

[2] We use the notation $[a, b]$ to denote the interval of integer numbers between a and b.

StockNet, a deep generative model for stock movement prediction.

To better incorporate stochastic factors, we generate stock movements from *latent driven factors* modeled with recurrent, continuous latent variables. Motivated by Variational Auto-Encoders (VAEs; Kingma and Welling, 2013; Rezende et al., 2014), we propose a novel decoder with a variational architecture and derive a recurrent variational lower bound for end-to-end training (Section 5.2). To the best of our knowledge, StockNet is the first deep generative model for stock movement prediction.

To fully exploit market information, StockNet directly learns from data without pre-extracting structured events. We build market sources by referring to both fundamental information, e.g. tweets, and technical features, e.g. historical stock prices (Section 5.1).[3] To accurately depict predictive dependencies, we assume that the movement prediction for a stock can benefit from learning to predict its historical movements in a lag window. We propose trading-day alignment as the framework basis (Section 4), and further provide a novel multi-task learning objective (Section 5.3).

We evaluate StockNet on a stock movement prediction task with a new dataset that we collected. Compared with strong baselines, our experiments show that StockNet achieves state-of-the-art performance by incorporating both data from Twitter and historical stock price listings.

2 Problem Formulation

We aim at predicting the movement of a target stock s in a pre-selected stock collection $\mathcal{S}$ on a target trading day d. Formally, we use the market information comprising of relevant social media corpora $\mathcal{M}$, i.e. tweets, and historical prices, in the lag $[d - \Delta d,\ d - 1]$ where Δd is a *fixed* lag size. We estimate the binary movement where 1 denotes rise and 0 denotes fall,

$$y = \mathbb{1}\left(p_d^c > p_{d-1}^c\right) \tag{1}$$

where p_d^c denotes the adjusted closing price adjusted for corporate actions affecting stock prices, e.g. dividends and splits.[4] The adjusted closing price is widely used for predicting stock price movement (Xie et al., 2013) or financial volatility (Rekabsaz et al., 2017).

3 Data Collection

In finance, stocks are categorized into 9 industries: *Basic Materials*, *Consumer Goods*, *Healthcare*, *Services*, *Utilities*, *Conglomerates*, *Financial*, *Industrial Goods* and *Technology*.[5] Since high-trade-volume-stocks tend to be discussed more on Twitter, we select the two-year price movements from 01/01/2014 to 01/01/2016 of 88 stocks to target, coming from all the 8 stocks in *Conglomerates* and the top 10 stocks in capital size in each of the other 8 industries (see supplementary material).

We observe that there are a number of targets with exceptionally minor movement ratios. In a three-way stock trend prediction task, a common practice is to categorize these movements to another "preserve" class by setting upper and lower thresholds on the stock price change (Hu et al., 2018). Since we aim at the binary classification of stock changes identifiable from social media, we set two particular thresholds, -0.5% and 0.55% and simply remove 38.72% of the selected targets with the movement percents between the two thresholds. Samples with the movement percents $\leq$-0.5% and $>$0.55% are labeled with 0 and 1, respectively. The two thresholds are selected to balance the two classes, resulting in 26,614 prediction targets in the whole dataset with 49.78% and 50.22% of them in the two classes. We split them temporally and 20,339 movements between 01/01/2014 and 01/08/2015 are for training, 2,555 movements from 01/08/2015 to 01/10/2015 are for development, and 3,720 movements from 01/10/2015 to 01/01/2016 are for test.

There are two main components in our dataset:[6] a Twitter dataset and a historical price dataset. We access Twitter data under the official license of Twitter, then retrieve stock-specific tweets by querying regexes made up of NASDAQ ticker symbols, e.g. "\$GOOG\b" for *Google Inc.*. We preprocess tweet texts using the NLTK package (Bird et al., 2009) with the particular Twitter

[3]To a fundamentalist, stocks have their intrinsic values that can be derived from the behavior and performance of their company. On the contrary, technical analysis considers only the trends and patterns of the stock price.

[4] Technically, $d - 1$ may not be an eligible trading day and thus has no available price information. In the rest of this paper, the problem is solved by keeping the notational consistency with our recurrent model and using its time step t to index trading days. Details will be provided in Section 4. We use d here to make the formulation easier to follow.

[5]https://finance.yahoo.com/industries

[6]Our dataset is available at `https://github.com/yumoxu/stocknet-dataset`.

mode, including for tokenization and treatment of hyperlinks, hashtags and the "@" identifier. To alleviate sparsity, we further filter samples by ensuring there is at least one tweet for each corpus in the lag. We extract historical prices for the 88 selected stocks to build the historical price dataset from Yahoo Finance.[7]

4 Model Overview

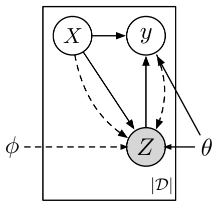


Figure 1: Illustration of the generative process from observed market information to stock movements. We use solid lines to denote the generation process and dashed lines to denote the variational approximation to the intractable posterior.

We provide an overview of data alignment, model factorization and model components.

As explained in Section 1, we assume that predicting the movement on trading day d can benefit from predicting the movements on its former trading days. However, due to the general principle of sample independence, building connections directly across samples with temporally-close target dates is problematic for model training.

As an alternative, we notice that within a sample with a target trading day d there are likely to be other trading days than d in its lag that can simulate the prediction targets close to d. Motivated by this observation and multi-task learning (Caruana, 1998), we make movement predictions not only for d, but also other trading days existing in the lag. For instance, as shown in Figure 2, for a sample targeting 07/08/2012 and a 5-day lag, 03/08/2012 and 06/08/2012 are eligible trading days in the lag and we also make predictions for them using the market information in this sample. The relations between these predictions can thus be captured within the scope of a sample.

As shown in the instance above, not every single date in a lag is an eligible trading day, e.g. weekends and holidays. To better organize and use the input, we regard the trading day, instead of the calendar day used in existing research, as the basic unit for building samples. To this end, we first find all the T eligible trading days referred in a sample, in other words, existing in the time interval $[d - \Delta d + 1,\ d]$. For clarity, in the scope of one sample, we index these trading days with $t \in [1, T]$,[8] and each of them maps to an actual (absolute) trading day d_t. We then propose *trading-day alignment*: we reorganize our inputs, including the tweet corpora and historical prices, by aligning them to these T trading days. Specifically, on the tth trading day, we recognize market signals from the corpus $\mathcal{M}_t$ in $[d_{t-1}, d_t)$ and the historical prices p_t on d_{t-1}, for predicting the movement y_t on d_t. We provide an aligned sample for illustration in Figure 2. As a result, every single unit in a sample is a trading day, and we can predict a sequence of movements $y = [y_1, \ldots, y_T]$. The *main target* is y_T while the remainder $y^* = [y_1, \ldots, y_{T-1}]$ serves as the *temporal auxiliary target*. We use these in addition to the main target to improve prediction accuracy (Section 5.3).

We model the generative process shown in Figure 1. We encode observed market information as a random variable $X = [x_1; \ldots; x_T]$, from which we generate the *latent driven factor* $Z = [z_1; \ldots; z_T]$ for our prediction task. For the aforementioned multi-task learning purpose, we aim at modeling the conditional probability distribution $p_\theta(y|X) = \int_Z p_\theta(y, Z|X)$ instead of $p_\theta(y_T|X)$. We write the following factorization for generation,

$$p_\theta(y, Z|X) = p_\theta(y_T|X, Z)\, p_\theta(z_T|z_{<T}, X) \prod_{t=1}^{T-1} p_\theta(y_t|x_{\le t}, z_t)\, p_\theta(z_t|z_{<t}, x_{\le t}, y_t) \quad (2)$$

where for a given indexed matrix of T vectors $[v_1; \ldots; v_T]$, we denote by $v_{<t}$ and $v_{\le t}$ the submatrix $[v_1; \ldots; v_{t-1}]$ and the submatrix $[v_1; \ldots; v_t]$, respectively. Since y^* is known in generation, we use the posterior $p_\theta(z_t|z_{<t}, x_{\le t}, y_t)$, $t < T$ to incorporate market signals more accurately and only use the prior $p_\theta(z_T|z_{<T}, X)$ when generating z_T. Besides, when $t < T$, y_t is independent of $z_{<t}$ while our main prediction target, y_T is made dependent on $z_{<T}$ through a temporal attention mechanism (Section 5.3).

We show StockNet modeling the above generative process in Figure 2. In a nutshell, StockNet

[7]http://finance.yahoo.com

[8]It holds that $T \ge 1$ since d is undoubtedly a trading day.

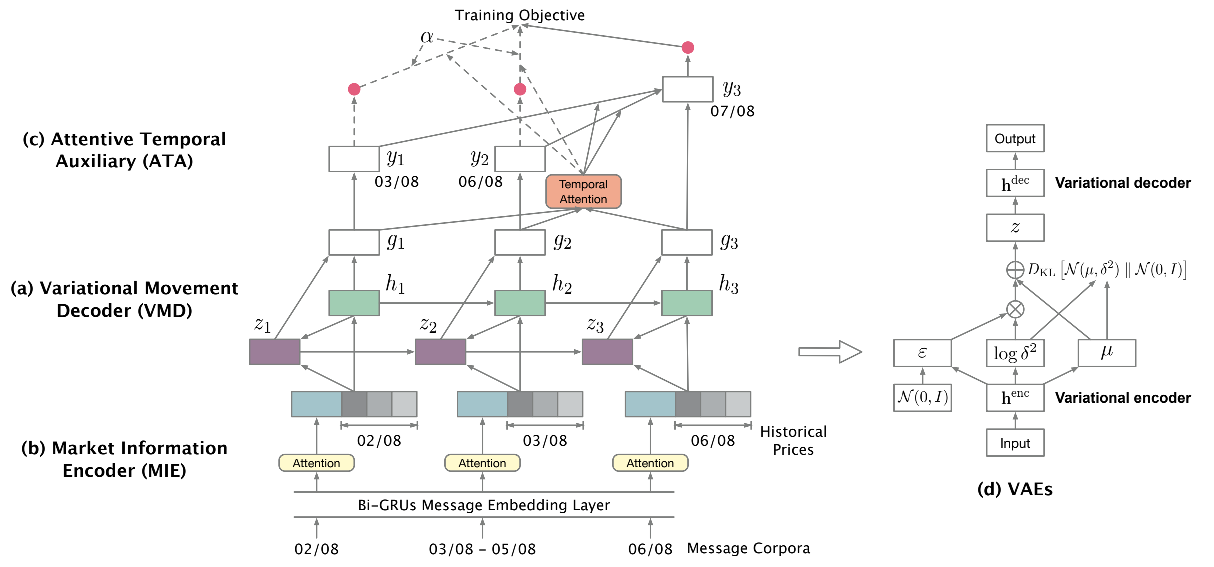


Figure 2: The architecture of StockNet. We use the main target of 07/08/2012 and the lag size of 5 for illustration. Since 04/08/2012 and 05/08/2012 are not trading days (a weekend), trading-day alignment helps StockNet to organize message corpora and historical prices for the other three trading days in the lag. We use dashed lines to denote auxiliary components. Red points denoting temporal objectives are integrated with a temporal attention mechanism to acquire the final training objective.

comprises three primary components following a bottom-up fashion,

1. Market Information Encoder (MIE) that encodes tweets and prices to X;

2. Variational Movement Decoder (VMD) that infers Z with X, y and decodes stock movements y from X, Z;

3. Attentive Temporal Auxiliary (ATA) that integrates temporal loss through an attention mechanism for model training.

5 Model Components

We detail next the components of our model (MIE, VMD, ATA) and the way we estimate our model parameters.

5.1 Market Information Encoder

MIE encodes information from social media and stock prices to enhance market information quality, and outputs the market information input X for VMD. Each temporal input is defined as

$$x_t = [c_t, p_t] \tag{3}$$

where c_t and p_t are the corpus embedding and the historical price vector, respectively.

The basic strategy of acquiring c_t is to first feed messages into the Message Embedding Layer for their low-dimensional representations, then selectively gather them according to their quality. To handle the circumstance that multiple stocks are discussed in one single message, in addition to text information, we incorporate the position information of stock symbols mentioned in messages as well. Specifically, the layer consists of a forward GRU and a backward GRU for the preceding and following contexts of a stock symbol, s, respectively. Formally, in the message corpus of the tth trading day, we denote the word sequence of the kth message, $k \in [1, K]$, as $\mathcal{W}$ where $\mathcal{W}_{\ell^\star} = s, \ell^\star \in [1, L]$, and its word embedding matrix as $E = [e_1; e_2; \ldots; e_L]$. We run the two GRUs as follows,

$$\overrightarrow{h}_f = \overrightarrow{\text{GRU}}(e_f, \overrightarrow{h}_{f-1}) \tag{4}$$

$$\overleftarrow{h}_b = \overleftarrow{\text{GRU}}(e_b, \overleftarrow{h}_{b+1}) \tag{5}$$

$$m = (\overrightarrow{h}_{\ell^\star} + \overleftarrow{h}_{\ell^\star})/2 \tag{6}$$

where $f \in [1, \ldots, \ell^\star]$, $b \in [\ell^\star, \ldots, L]$. The stock symbol is regarded as the last unit in both the preceding and the following contexts where the hidden values, $\overrightarrow{h}_{l^\star}$, $\overleftarrow{h}_{l^\star}$, are averaged to acquire the message embedding m. Gathering all message embeddings for the tth trading day, we have a message embedding matrix $M_t \in \mathbb{R}^{d_m \times K}$. In practice, the layer takes as inputs a five-rank tensor for a mini-batch, and yields all M_t in the batch with shared parameters.

Tweet quality varies drastically. Inspired by the news-level attention (Hu et al., 2018), we weight messages with their respective salience in collective intelligence measurement. Specifically, we first project M_t non-linearly to u_t, the normalized attention weight over the corpus,

$$u_t = \zeta(w_u^\intercal \tanh(W_{m,u} M_t)) \tag{7}$$

where $\zeta(\cdot)$ is the softmax function and $W_{m,u} \in \mathbb{R}^{d_m \times d_m}, w_u \in \mathbb{R}^{d_m \times 1}$ are model parameters. Then we compose messages accordingly to acquire the corpus embedding,

$$c_t = M_t u_t^\intercal. \tag{8}$$

Since it is the price change that determines the stock movement rather than the absolute price value, instead of directly feeding the raw price vector $\tilde{p}_t = \left[\tilde{p}_t^c, \tilde{p}_t^h, \tilde{p}_t^l\right]$ comprising of the adjusted closing, highest and lowest price on a trading day t, into the networks, we normalize it with its last adjusted closing price, $p_t = \tilde{p}_t / \tilde{p}_{t-1}^c - 1$. We then concatenate c_t with p_t to form the final market information input x_t for the decoder.

5.2 Variational Movement Decoder

The purpose of VMD is to recurrently infer and decode the latent driven factor Z and the movement y from the encoded market information X.

Inference

While latent driven factors help to depict the market status leading to stock movements, the posterior inference in the generative model shown in Eq. (2) is intractable. Following the spirit of the VAE, we use deep neural networks to fit latent distributions, i.e. the prior $p_\theta\left(z_t|z_{<t}, x_{\le t}\right)$ and the posterior $p_\theta\left(z_t|z_{<t}, x_{\le t}, y_t\right)$, and sidestep the intractability through *neural approximation* and *reparameterization* (Kingma and Welling, 2013; Rezende et al., 2014). We first employ a variational approximator $q_\phi\left(z_t|z_{<t}, x_{\le t}, y_t\right)$ for the intractable posterior. We observe the following factorization,

$$q_\phi\left(Z|X, y\right) = \prod_{t=1}^{T} q_\phi\left(z_t|z_{<t}, x_{\le t}, y_t\right). \tag{9}$$

Neural approximation aims at minimizing the Kullback-Leibler divergence between the $q_\phi\left(Z|X, y\right)$ and $p_\theta\left(Z|X, y\right)$. Instead of optimizing it directly, we observe that the following equation naturally holds,

$$\begin{aligned} &\log p_\theta\left(y|X\right) \\ =&D_{\text{KL}}\left[q_\phi\left(Z|X, y\right) \parallel p_\theta\left(Z|X, y\right)\right] \\ +&\mathbb{E}_{q_\phi(Z|X,y)}\left[\log p_\theta\left(y|X, Z\right)\right] \\ -&D_{\text{KL}}\left[q_\phi\left(Z|X, y\right) \parallel p_\theta\left(Z|X\right)\right] \end{aligned} \tag{10}$$

where $D_{\text{KL}}\left[q \parallel p\right]$ is the Kullback-Leibler divergence between the distributions q and p. Therefore, we equivalently maximize the following variational recurrent lower bound by plugging Eq. (2, 9) into Eq. (10),

$$\begin{aligned} &\mathcal{L}\left(\theta, \phi; X, y\right) \\ =&\sum_{t=1}^{T} \mathbb{E}_{q_\phi\left(z_t|z_{<t}, x_{\le t}, y_t\right)} \Big\{ \log p_\theta\left(y_t|x_{\le t}, z_{\le t}\right) - \\ &D_{\text{KL}}\left[q_\phi\left(z_t|z_{<t}, x_{\le t}, y_t\right) \parallel p_\theta\left(z_t|z_{<t}, x_{\le t}\right)\right] \Big\} \\ \le&\log p_\theta\left(y|X\right) \end{aligned} \tag{11}$$

where the likelihood term

$$p_\theta\left(y_t|x_{\le t}, z_{\le t}\right) = \begin{cases} p_\theta\left(y_t|x_{\le t}, z_t\right), & \text{if } t < T \\ p_\theta\left(y_T|X, Z\right), & \text{if } t = T. \end{cases} \tag{12}$$

Li et al. (2017) also provide a lower bound for inferring directly-connected recurrent latent variables in text summarization. In their work, priors are modeled with $p_\theta\left(z_t\right) \sim \mathcal{N}(0, I)$, which, in fact, turns the KL term into a static regularization term encouraging sparsity. In Eq. (11), we provide a more theoretically rigorous lower bound where the KL term with $p_\theta\left(z_t|z_{<t}, x_{\le t}\right)$ plays a dynamic role in inferring dependent latent variables for every different model input and latent history.

Decoding

As per time series, VMD adopts an RNN with a GRU cell to extract features and decode stock signals recurrently,

$$h_t^s = \text{GRU}(x_t, h_{t-1}^s). \tag{13}$$

We let the approximator $q_\phi\left(z_t|z_{<t}, x_{\le t}, y_t\right)$ subject to a standard multivariate Gaussian distribution $\mathcal{N}(\mu, \delta^2 I)$. We calculate μ and δ as

$$\mu_t = W_{z,\mu}^\phi h_t^z + b_\mu^\phi \tag{14}$$

$$\log \delta_t^2 = W_{z,\delta}^\phi h_t^z + b_\delta^\phi \tag{15}$$

and the shared hidden representation h_t^z as

$$h_t^z = \tanh(W_z^\phi[z_{t-1}, x_t, h_t^s, y_t] + b_z^\phi) \quad (16)$$

where $W_{z,\mu}^\phi, W_{z,\delta}^\phi, W_z^\phi$ are weight matrices and $b_\mu^\phi, b_\delta^\phi, b_z^\phi$ are biases.

Since Gaussian distribution belongs to the "location-scale" distribution family, we can further *reparameterize* z_t as

$$z_t = \mu_t + \delta_t \odot \epsilon \quad (17)$$

where $\odot$ denotes an element-wise product. The noise term $\epsilon \sim \mathcal{N}(0, I)$ naturally involves stochastic signals in our model.

Similarly, We let the prior $p_\theta\left(z_t | z_{<t}, x_{\le t}\right) \sim \mathcal{N}(\mu', {\delta'}^2 I)$. Its calculation is the same as that of the posterior except the absence of y_t and independent model parameters,

$$\mu'_t = W_{o,\mu}^\theta h_t^{z\prime} + b_\mu^\theta \quad (18)$$

$$\log {\delta'}_t^2 = W_{o,\delta}^\theta h_t^{z\prime} + b_\delta^\theta \quad (19)$$

where

$$h_t^{z\prime} = \tanh(W_z^\theta[z_{t-1}, x_t, h_t^s] + b_z^\theta). \quad (20)$$

Following Zhang et al. (2016), differently from the posterior, we set the prior $z_t = \mu'_t$ during decoding. Finally, we integrate deterministic features and the final prediction hypothesis is given as

$$g_t = \tanh(W_g[x_t, h_t^s, z_t] + b_g) \quad (21)$$

$$\tilde{y}_t = \zeta(W_y g_t + b_y), t < T \quad (22)$$

where W_g, W_y are weight matrices and b_g, b_y are biases. The softmax function $\zeta(\cdot)$ outputs the confidence distribution over up and down. As introduced in Section 4, the decoding of the main target y_T depends on $z_{<T}$ and thus lies at the interface between VMD and ATA. We will elaborate on it in the next section.

5.3 Attentive Temporal Auxiliary

With the acquisition of a sequence of auxiliary predictions $\tilde{Y}^* = [\tilde{y}_1; \ldots; \tilde{y}_{T-1}]$, we incorporate two-folded auxiliary effects into the main prediction and the training objective flexibly by first introducing a shared *temporal attention* mechanism.

Since each hypothesis of a temporal auxiliary contributes unequally to the main prediction and model training, as shown in Figure 3, temporal attention calculates their weights in these two contributions by employing two scoring components: an

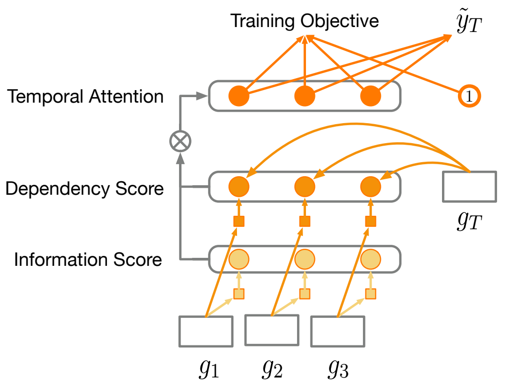


Figure 3: The temporal attention in our model. Squares are the non-linear projections of g_t and points are scores or normalized weights.

information score and a *dependency score*. Specifically,

$$v'_i = w_i^\intercal \tanh(W_{g,i} G^*) \quad (23)$$

$$v'_d = g_T^\intercal \tanh(W_{g,d} G^*) \quad (24)$$

$$v^* = \zeta(v'_i \odot v'_d) \quad (25)$$

where $W_{g,i}, W_{g,d} \in \mathbb{R}^{d_g \times d_g}, w_i \in \mathbb{R}^{d_g \times 1}$ are model parameters. The integrated representations $G^* = [g_1; \ldots; g_{T-1}]$ and g_T are reused as the final representations of temporal market information. The information score v'_i evaluates historical trading days as per their own information quality, while the dependency score v'_d captures their dependencies with our main target. We integrate the two and acquire the final normalized attention weight $v^* \in \mathbb{R}^{1 \times (T-1)}$ by feeding their element-wise product into the softmax function.

As a result, the main prediction can benefit from temporally-close hypotheses have been made and we decode our main hypothesis $\tilde{y}_T$ as

$$\tilde{y}_T = \zeta(W_T[\tilde{Y}^* v^{*\intercal}, g_T] + b_T) \quad (26)$$

where W_T is a weight matrix and b_T is a bias.

As to the model objective, we use the Monte Carlo method to approximate the expectation term in Eq. (11) and typically only one sample is used for gradient computation. To incorporate varied temporal importance at the objective level, we first break down the approximated $\mathcal{L}$ into a series of temporal objectives $f \in \mathbb{R}^{T \times 1}$ where f_t comprises a likelihood term and a KL term for a trading day t,

$$f_t = \log p_\theta\left(y_t | x_{\le t}, z_{\le t}\right) - \lambda D_{\mathrm{KL}}\left[q_\phi\left(z_t | z_{<t}, x_{\le t}, y_t\right) \parallel p_\theta\left(z_t | z_{<t}, x_{\le t}\right)\right] \quad (27)$$

where we adopt the KL term annealing trick (Bowman et al., 2016; Semeniuta et al., 2017) and add a linearly-increasing KL term weight $\lambda \in (0, 1]$ to gradually release the KL regularization effect in the training procedure. Then we reuse v^* to build the final temporal weight vector $v \in \mathbb{R}^{1 \times T}$,

$$v = [\alpha v^*, 1] \tag{28}$$

where 1 is for the main prediction and we adopt the auxiliary weight $\alpha \in [0, 1]$ to control the overall auxiliary effects on the model training. α is tuned on the development set and its effects will be discussed at length in Section 6.5. Finally, we write the training objective $\mathcal{F}$ by recomposition,

$$\mathcal{F}(\theta, \phi; X, y) = \frac{1}{N} \sum_{n}^{N} v^{(n)} f^{(n)} \tag{29}$$

where our model can learn to generalize with the selective attendance of temporal auxiliary. We take the derivative of $\mathcal{F}$ with respect to all the model parameters $\{\theta, \phi\}$ through backpropagation for the update.

6 Experiments

In this section, we detail our experimental setup and results.

6.1 Training Setup

We use a 5-day lag window for sample construction and 32 shuffled samples in a batch.[9] The maximal token number contained in a message and the maximal message number on a trading day are empirically set to 30 and 40, respectively, with the excess clipped. Since all tweets in the batched samples are simultaneously fed into the model, we set the word embedding size to 50 instead of larger sizes to control memory costs and make model training feasible on one single GPU (11GB memory). We set the hidden size of Message Embedding Layer to 100 and that of VMD to 150. All weight matrices in the model are initialized with the fan-in trick and biases are initialized with zero. We train the model with an Adam optimizer (Kingma and Ba, 2014) with the initial learning rate of 0.001. Following Bowman et al. (2016), we use the input dropout rate of 0.3 to regularize latent variables. Tensorflow (Abadi et al., 2016) is used to construct the computational graph of StockNet and hyper-parameters are tweaked on the development set.

6.2 Evaluation Metrics

Following previous work for stock prediction (Xie et al., 2013; Ding et al., 2015), we adopt the standard measure of accuracy and Matthews Correlation Coefficient (MCC) as evaluation metrics. MCC avoids bias due to data skew. Given the confusion matrix $\left(\begin{smallmatrix} \mathrm{tp} & \mathrm{fn} \\ \mathrm{fp} & \mathrm{tn} \end{smallmatrix}\right)$ containing the number of samples classified as true positive, false positive, true negative and false negative, MCC is calculated as

$$\mathrm{MCC} = \frac{\mathrm{tp} \times \mathrm{tn} - \mathrm{fp} \times \mathrm{fn}}{\sqrt{(\mathrm{tp} + \mathrm{fp})(\mathrm{tp} + \mathrm{fn})(\mathrm{tn} + \mathrm{fp})(\mathrm{tn} + \mathrm{fn})}}. \tag{30}$$

6.3 Baselines and Proposed Models

We construct the following five baselines in different genres,[10]

- RAND: a naive predictor making random guess in up or down.
- ARIMA: Autoregressive Integrated Moving Average, an advanced technical analysis method using only price signals (Brown, 2004) .
- RANDFOREST: a discriminative Random Forest classifier using Word2vec text representations (Pagolu et al., 2016).
- TSLDA: a generative topic model jointly learning topics and sentiments (Nguyen and Shirai, 2015).
- HAN: a state-of-the-art discriminative deep neural network with hierarchical attention (Hu et al., 2018).

To make a detailed analysis of all the primary components in StockNet, in addition to HEDGEFUNDANALYST, the fully-equipped StockNet, we also construct the following four variations,

- TECHNICALANALYST: the generative StockNet using only historical prices.
- FUNDAMENTALANALYST: the generative StockNet using only tweet information.
- INDEPENDENTANALYST: the generative StockNet without temporal auxiliary targets.

[9]Typically the lag size is set between 3 and 10. As introduced in Section 4, trading days are treated as basic units in StockNet and 3 calendar days are thus too short to guarantee the existence of more than one trading day in a lag, e.g. the prediction for the movement of Monday. We also experiment with 7 and 10 but they do not yield better results than 5.

[10]We do not treat event-driven models as comparable methods since our model uses no event pre-extraction tool.

Baseline models	**Acc.**	**MCC**	**StockNet variations**	**Acc.**	**MCC**
RAND	50.89	-0.002266	TECHNICALANALYST	54.96	0.016456
ARIMA (Brown, 2004)	51.39	-0.020588	FUNDAMENTALANALYST	58.23	0.071704
RANDFOREST (Pagolu et al., 2016)	53.08	0.012929	INDEPENDENTANALYST	57.54	0.036610
TSLDA (Nguyen and Shirai, 2015)	54.07	**0.065382**	DISCRIMINATIVEANALYST	56.15	0.056493
HAN (Hu et al., 2018)	**57.64**	0.051800	HEDGEFUNDANALYST	**58.23**	**0.080796**

Table 1: Performance of baselines and StockNet variations in accuracy and MCC.

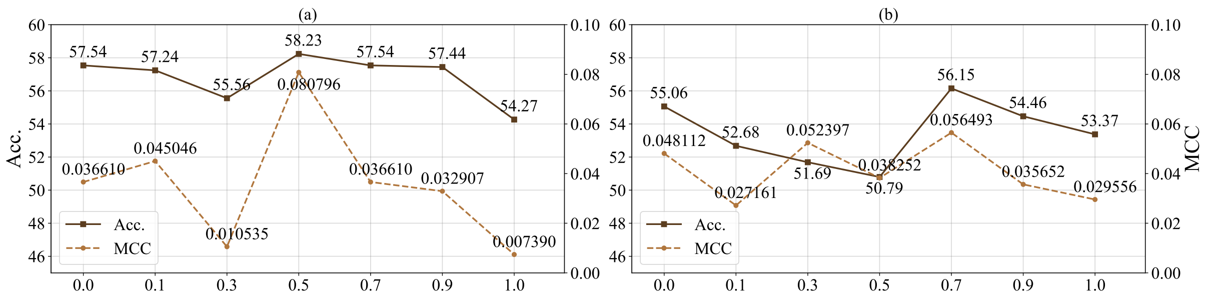


Figure 4: (a) Performance of HEDGEFUNDANALYST with varied α, see Eq. (28). (b) Performance of DISCRIMINATIVEANALYST with varied α.

- DISCRIMINATIVEANALYST: the discriminative StockNet directly optimizing the likelihood objective. Following Zhang et al. (2016), we set $z_t = \mu'_t$ to take out the effects of the KL term.

6.4 Results

Since stock prediction is a challenging task and a minor improvement usually leads to large potential profits, the accuracy of 56% is generally reported as a satisfying result for binary stock movement prediction (Nguyen and Shirai, 2015). We show the performance of the baselines and our proposed models in Table 1. TLSDA is the best baseline in MCC while HAN is the best baseline in accuracy. Our model, HEDGEFUNDANALYST achieves the best performance of 58.23 in accuracy and 0.080796 in MCC, outperforming TLSDA and HAN with 4.16, 0.59 in accuracy, and 0.015414, 0.028996 in MCC, respectively.

Though slightly better than random guess, classic technical analysis, e.g. ARIMA, does not yield satisfying results. Similar in using only historical prices, TECHNICALANALYST shows an obvious advantage in this task compared ARIMA. We believe there are two major reasons: (1) TECHNICALANALYST learns from training data and incorporates more flexible non-linearity; (2) our test set contains a large number of stocks while ARIMA is more sensitive to peculiar sequence stationarity. It is worth noting that FUNDAMENTALANALYST gains exceptionally competitive results with only 0.009092 less in MCC than HEDGEFUNDANALYST. The performance of FUNDAMENTALANALYST and TECHNICALANALYST confirm the positive effects from tweets and historical prices in stock movement prediction, respectively. As an effective ensemble of the two market information, HEDGEFUNDANALYST gains even better performance.

Compared with DISCRIMINATIVEANALYST, the performance improvements of HEDGEFUNDANALYST are not from enlarging the networks, demonstrating that modeling underlying market status explicitly with latent driven factors indeed benefits stock movement prediction. The comparison with INDEPENDENTANALYST also shows the effectiveness of capturing temporal dependencies between predictions with the temporal auxiliary. However, the effects of the temporal auxiliary are more complex and will be analyzed further in the next section.

6.5 Effects of Temporal Auxiliary

We provide a detailed discuss of how the temporal auxiliary affects model performance. As introduced in Eq. (28), the temporal auxiliary weight α controls the overall effects of the objective-level temporal auxiliary to our model. Figure 4 presents how the performance of HEDGEFUNDANALYST and DISCRIMINATIVEANALYST fluctuates with α.

As shown in Figure 4, enhanced by the temporal auxiliary, HEDGEFUNDANALYST approaches the best performance at 0.5, and DISCRIMINATIVEANALYST achieves its maximum at 0.7. In fact, objective-level auxiliary can be regarded as a denoising regularizer: for a sample with a specific movement as the main target, the market source in the lag can be heterogeneous, e.g. affected by bad news, tweets on earlier days are negative but turn to positive due to timely crises management. Without temporal auxiliary tasks, the model tries to identify positive signals on earlier days only for the main target of rise movement, which is likely to result in pure noise. In such cases, temporal auxiliary tasks help to filter market sources in the lag as per their respective aligned auxiliary movements. Besides, from the perspective of training variational models, the temporal auxiliary helps HEDGEFUNDANALYST to encode more useful information into the latent driven factor Z, which is consistent with recent research in VAEs (Semeniuta et al., 2017). Compared with HEDGEFUNDANALYST that contains a KL term performing dynamic regularization, DISCRIMINATIVEANALYST requires stronger regularization effects coming with a bigger α to achieve its best performance.

Since y^* also involves in generating y_T through the temporal attention, tweaking α acts as a trade-off between focusing on the main target and generalizing by denoising. Therefore, as shown in Figure 4, our models do not linearly benefit from incorporating temporal auxiliary. In fact, the two models follow a similar pattern in terms of performance change: the curves first drop down with the increase of α, except the MCC curve for DISCRIMINATIVEANALYST rising up temporarily at 0.3. After that, the curves ascend abruptly to their maximums, then keep descending till $\alpha = 1$. Though the start phase of increasing α even leads to worse performance, when auxiliary effects are properly introduced, the two models finally gain better results than those with no involvement of auxiliary effects, e.g. INDEPENDENTANALYST.

7 Conclusion

We demonstrated the effectiveness of deep generative approaches for stock movement prediction from social media data by introducing StockNet, a neural network architecture for this task. We tested our model on a new comprehensive dataset and showed it performs better than strong baselines, including implementation of previous work. Our comprehensive dataset is publicly available at `https://github.com/yumoxu/stocknet-dataset`.

Acknowledgments

The authors would like to thank the three anonymous reviewers and Miles Osborne for their helpful comments. This research was supported by a grant from Bloomberg and by the H2020 project SUMMA, under grant agreement 688139.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of computational science* 2(1):1–8.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany, pages 10–21.

Robert Goodell Brown. 2004. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1415–1425.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*. Buenos Aires, Argentina, pages 2327–2333.

Robert D Edwards, WHC Bassetti, and John Magee. 2007. *Technical analysis of stock trends*. CRC press.

Jeffrey A Frankel. 1995. *Financial markets and monetary policy*. MIT Press.

Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, Los Angeles, California, USA, pages 261–269.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*. JMLR. org, Beijing, China, pages 1188–1196.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 2081–2090.

Ronny Luss and Alexandre d'Aspremont. 2015. Predicting abnormal returns from news using text classification. *Quantitative Finance* 15(6):999–1012.

Burton Gordon Malkiel. 1999. *A random walk down Wall Street: including a life-cycle guide to personal investing*. WW Norton & Company.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Beijing, China, volume 1, pages 1354–1364.

Nuno Oliveira, Paulo Cortez, and Nelson Areal. 2013. Some experiments on modeling stock market behavior using investor sentiment analysis and posting volume from twitter. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*. ACM, Madrid, Spain, page 31.

Venkata Sasank Pagolu, Kamal Nayan Reddy, Ganapati Panda, and Babita Majhi. 2016. Sentiment analysis of twitter data for predicting stock market movements. In *Proceedings of 2016 International Conference on Signal Processing, Communication, Power and Embedded System*. IEEE, Rajaseetapuram, India, pages 1345–1350.

Navid Rekabsaz, Mihai Lupu, Artem Baklanov, Alexander Dür, Linda Andersson, and Allan Hanbury. 2017. Volatility prediction using financial disclosures sentiments with word embedding-based ir models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, volume 1, pages 1712–1721.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning*. Beijing, China, pages 1278–1286.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems* 27(2):12.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 627–637.

Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. 2013. Exploiting topic based twitter sentiment for stock prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria, volume 2, pages 24–29.

Boyi Xie, Rebecca J Passonneau, Leon Wu, and Germán G Creamer. 2013. Semantic frames to predict stock price movement. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, volume 1, pages 873–883.

Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, USA, pages 521–530.

Rumor Detection on Twitter with Tree-structured Recursive Neural Networks

Jing Ma[1], **Wei Gao**[2], **Kam-Fai Wong**[1,3]
[1]The Chinese University of Hong Kong, Hong Kong SAR
[2]Victoria University of Wellington, New Zealand
[3]MoE Key Laboratory of High Confidence Software Technologies, China
[1]{majing,kfwong}@se.cuhk.edu.hk, [2]wei.gao@vuw.ac.nz

Abstract

Automatic rumor detection is technically very challenging. In this work, we try to learn discriminative features from tweets content by following their non-sequential propagation structure and generate more powerful representations for identifying different type of rumors. We propose two *recursive* neural models based on a *bottom-up* and a *top-down* tree-structured neural networks for rumor representation learning and classification, which naturally conform to the propagation layout of tweets. Results on two public Twitter datasets demonstrate that our recursive neural models 1) achieve much better performance than state-of-the-art approaches; 2) demonstrate superior capacity on detecting rumors at very early stage.

1 Introduction

Rumors have always been a social disease. In recent years, it has become unprecedentedly convenient for the "evil-doers" to create and disseminate rumors in massive scale with low cost thanks to the popularity of social media outlets on Twitter, Facebook, etc. The worst effect of false rumors could be devastating to individual and/or society.

Research pertaining rumors spans multiple disciplines, such as philosophy and humanities (DiFonzo and Bordia, 2007; Donovan, 2007), social psychology (Allport and Postman, 1965; Jaeger et al., 1980; Rosnow and Foster, 2005), political studies (Allport and Postman, 1946; Berinsky, 2017), management science (DiFonzo et al., 1994; Kimmel, 2004) and recently computer science and artificial intelligence (Qazvinian et al., 2011; Ratkiewicz et al., 2011; Castillo et al., 2011; Hannak et al., 2014; Zhao et al., 2015; Ma et al., 2015). Rumor is commonly defined as information that emerge and spread among people whose truth value is unverified or intentionally false (DiFonzo and Bordia, 2007; Qazvinian et al., 2011). Analysis shows that people tend to stop spreading a rumor if it is known as false (Zubiaga et al., 2016b). However, identifying such misinformation is non-trivial and needs investigative journalism to fact check the suspected claim, which is labor-intensive and time-consuming. The proliferation of social media makes it worse due to the ever-increasing information load and dynamics. Therefore, it is necessary to develop automatic and assistant approaches to facilitate real-time rumor tracking and debunking.

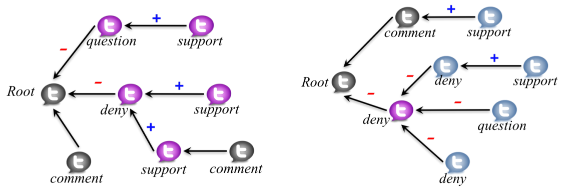
For automating rumor detection, most of the previous studies focused on text mining from sequential microblog streams using supervised models based on feature engineering (Castillo et al., 2011; Kwon et al., 2013; Liu et al., 2015; Ma et al., 2015), and more recently deep neural models (Ma et al., 2016; Chen et al., 2017; Ruchansky et al., 2017). These methods largely ignore or oversimplify the structural information associated with message propagation which however has been shown conducive to provide useful clues for identifying rumors. Kernel-based method (Wu et al., 2015; Ma et al., 2017) was thus proposed to model the structure as propagation trees in order to differentiate rumorous and non-rumorous claims by comparing their tree-based similarities. But such kind of approach cannot directly classify a tree without pairwise comparison with all other trees imposing unnecessary overhead, and it also cannot automatically learn any high-level feature representations out of the noisy surface features.

In this paper, we present a neural rumor detection approach based on recursive neural networks (RvNN) to bridge the content semantics and propagation clues. RvNN and its variants were originally used to compose phrase or sentence representation for syntactic and semantic parsing (Socher et al., 2011, 2012). Unlike parsing, the input into our model is a propagation tree rooted from a source post rather than the parse tree of an individual sentence, and each tree node is a responsive post instead of an individual words. The content semantics of posts and the responsive relationship among them can be jointly captured via the recursive feature learning process along the tree structure.

So, why can such neural model do better for the task? Analysis has generally found that Twitter could "self-correct" some inaccurate information as users share opinions, conjectures and evidences (Zubiaga et al., 2017). To illustrate our intuition, Figure 1 exemplifies the propagation trees of two rumors in our dataset, one being false and the other being true[1]. Structure-insensitive methods basically relying on the relative ratio of different stances in the text cannot do well when such clue is unclear like this example. However, it can be seen that when a post denies the false rumor, it tends to spark supportive or affirmative replies confirming the denial; in contrast, denial to a true rumor tends to trigger question or denial in its replies. This observation may suggest a more general hypothesis that the repliers tend to disagree with (or question) who support a false rumor or deny a true rumor, and also they tend to agree with who deny a false rumor or support a true rumor. Meanwhile, a reply, rather than directly responding to the source tweet (i.e., the root), is usually responsive to its immediate ancestor (Lukasik et al., 2016; Zubiaga et al., 2016a), suggesting obvious local characteristic of the interaction. The recursive network naturally models such structures for learning to capture the rumor indicative signals and enhance the representation by recursively aggregating the signals from different branches.

To this end, we extend the standard RvNN into two variants, i.e., a *bottom-up* (BU) model and a *top-down* (TD) model, which represent the propagation tree structure from different angles, in order to visit the nodes and combine their representations following distinct directions. The important merit of such architecture is that the node features can be selectively refined by the recursion given the connection and direction of all paths of the

[1]False (true) rumor means the veracity of the rumorous claim is false (true).



(a) False rumor (b) True rumor

Figure 1: Propagation trees of two rumorous source tweets. Nodes may express stances on their parent as commenting, supporting, questioning or denying. The edge arrow indicates the direction from a response to its responded node, and the polarity is marked as '+' ('-') for support (denial). The same node color indicates the same stance on the veracity of root node (i.e., source tweet).

tree. As a result, it can be expected that the discriminative signals are better embedded into the learned representations.

We evaluate our proposed approach based on two public Twitter datasets. The results show that our method outperforms strong rumor detection baselines with large margin and also demonstrate much higher effectiveness for detection at early stage of propagation, which is promising for real-time intervention and debunking. Our contributions are summarized as follows in three folds:

- This is the first study that deeply integrates both structure and content semantics based on tree-structured recursive neural networks for detecting rumors from microblog posts.
- We propose two variants of RvNN models based on bottom-up and top-down tree structures to generate better integrated representations for a claim by capturing both structural and textural properties signaling rumors.
- Our experiments based on real-world Twitter datasets achieve superior improvements over state-of-the-art baselines on both rumor classification and early detection tasks. We make the source codes in our experiments publicly accessible [2].

2 Related Work

Most previous automatic approaches for rumor detection (Castillo et al., 2011; Yang et al., 2012; Liu

[2]https://github.com/majingCUHK/Rumor_RvNN

et al., 2015) intended to learn a supervised classifier by utilizing a wide range of features crafted from post contents, user profiles and propagation patterns. Subsequent studies were then conducted to engineer new features such as those representing rumor diffusion and cascades (Friggeri et al., 2014; Hannak et al., 2014) characterized by comments with links to debunking websites. Kwon et al. (2013) introduced a time-series-fitting model based on the volume of tweets over time. Ma et al. (2015) extended their model with more chronological social context features. These approaches typically require heavy preprocessing and feature engineering.

Zhao et al. (2015) alleviated the engineering effort by using a set of regular expressions (such as "really?", "not true", etc) to find questing and denying tweets, but the approach was oversimplified and suffered from very low recall. Ma et al. (2016) used recurrent neural networks (RNN) to learn automatically the representations from tweets content based on time series. Recently, they studied to mutually reinforce stance detection and rumor classification in a neural multi-task learning framework (Ma et al., 2018). However, the approaches cannot embed features reflecting how the posts are propagated and requires careful data segmentation to prepare for time sequence.

Some kernel-based methods were exploited to model the propagation structure. Wu et al. (2015) proposed a hybrid SVM classifier which combines a RBF kernel and a random-walk-based graph kernel to capture both flat and propagation patterns for detecting rumors on Sina Weibo. Ma et al. (2017) used tree kernel to capture the similarity of propagation trees by counting their similar substructures in order to identify different types of rumors on Twitter. Compared to their studies, our model can learn the useful features via a more natural and general approach, i.e., the tree-structured neural network, to jointly generate representations from both structure and content.

RvNN has demonstrated state-of-the-art performances in a variety of tasks, e.g., images segmentation (Socher et al., 2011), phrase representation from word vectors (Socher et al., 2012), and sentiment classification in sentences (Socher et al., 2013). More recently, a deep RvNN was proposed to model the compositionality in natural language for fine-grained sentiment classification by stacking multiple recursive layers (Irsoy and Cardie, 2014). In order to avoid gradient vanishing, some studies integrated Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to RvNN (Zhu et al., 2015; Tai et al., 2015). Mou et al. (2015) used a convolutional network over tree structures for syntactic tree parsing of natural language sentences.

3 Problem Statement

We define a Twitter rumor detection dataset as a set of claims $\mathcal{C} = \{C_1, C_2, \cdots, C_{|\mathcal{C}|}\}$, where each claim C_i corresponds to a source tweet r_i which consists of ideally all its relevant responsive tweets in chronological order, i.e., $C_i = \{r_i, x_{i1}, x_{i2}, \cdots, x_{im}\}$ where each x_{i*} is a responsive tweet of the root r_i. Note that although the tweets are notated sequentially, there are connections among them based on their reply or repost relationships, which can form a propagation tree structure (Wu et al., 2015; Ma et al., 2017) with r_i being the root node.

We formulate this task as a supervised classification problem, which learns a classifier f from labeled claims, that is $f : C_i \to Y_i$, where Y_i takes one of the four finer-grained classes: *non-rumor*, *false rumor*, *true rumor*, and *unverified rumor* that are introduced in the literature (Ma et al., 2017; Zubiaga et al., 2016b).

An important issue of the tree structure is concerned about the direction of edges, which can result in two different architectures of the model: 1) a bottom-up tree; 2) a top-down tree, which are defined as follows:

- **Bottom-up tree** takes the similar shape as shown in Figure 1, where responsive nodes always point to their responded nodes and leaf nodes not having any response are laid out at the furthest level. We represent a tree as $\mathcal{T}_i = \langle V_i, E_i \rangle$, where $V_i = C_i$ which consists of all relevant posts as nodes, and E_i denotes a set of all directed links, where for any $u, v \in V_i$, $u \leftarrow v$ exists if v *responses to* u. This structure is similar to a citation network where a response mimics a reference.

- **Top-down tree** naturally conforms to the direction of information propagation, in which a link $u \to v$ means the information flows from u to v and v sees it and provides a response to u. This structure reverses bottom-up tree and simulates how information cas-

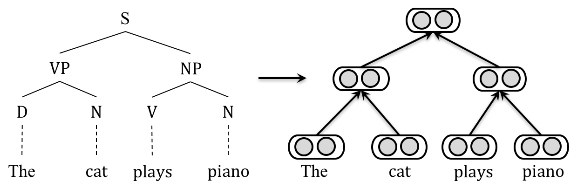


Figure 2: A binarized sentence parse tree (left) and its corresponding RvNN architecture (right).

cades from a source tweet, i.e., the root, to all its receivers, i.e., the decedents, which is similar as (Wu et al., 2015; Ma et al., 2017).

4 RvNN-based Rumor Detection

The core idea of our method is to strengthen the high-level representation of tree nodes by the recursion following the propagation structure over different branches in the tree. For instance, the responsive nodes confirming or supporting a node (e.g., "I agree", "be right", etc) can further reinforce the stance of that node while denial or questioning responses (e.g., "disagree, "really?!) otherwise weaken its stance. Compared to the kernel-based method using propagation tree (Wu et al., 2015; Ma et al., 2017), our method does not need pairwise comparison among large number of subtrees, and can learn much stronger representation of content following the response structure.

In this section, we will describe our extension to the standard RvNN for modeling rumor detection based on the bottom-up and top-down architectures presented in Section 3.

4.1 Standard Recursive Neural Networks

RvNN is a type of tree-structured neural networks. The original version of RvNN utilized binarized sentence parse trees (Socher et al., 2012), in which the representation associated with each node of a parse tree is computed from its direct children. The overall structure of the standard RvNN is illustrated as the right side of Figure 2, corresponding to the input parse tree at the left side.

Leaf nodes are the words in an input sentence, each represented by a low-dimensional word embedding. Non-leaf nodes are sentence constituents, computed by recursion based on the presentations of child nodes. Let p be the feature vector of a parent node whose children are c_1 and c_2, the representation of the parent is computed by $p = f(W \cdot [c_1; c_2] + b)$, where $f(\cdot)$ is the activation function with W and b as parameters. This computation is done recursively over all tree nodes; the learned hidden vectors of the nodes can then be used for various classification tasks.

4.2 Bottom-up RvNN

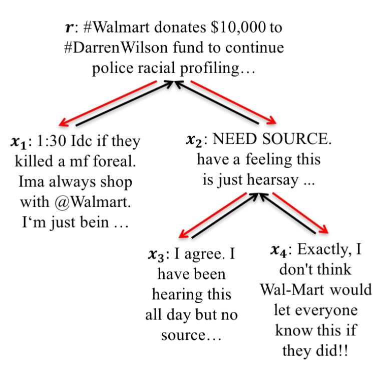
The core idea of bottom-up model is to generate a feature vector for each subtree by recursively visiting every node from the leaves at the bottom to the root at the top. In this way, the subtrees with similar contexts, such as those subtrees having a denial parent and a set of supportive children, will be projected into the proximity in the representation space. And thus such local rumor indicative features are aggregated along different branches into some global representation of the whole tree.

For this purpose, we make a natural extension to the original RvNN. The overall structure of our proposed bottom-up model is illustrated in Figure 3(b), taking a bottom-up tree (see Figure 3(a)) as input. Different from the standard RvNN, the input of each node in the bottom-up model is a post represented as a vector of words in the vocabulary in terms of $tfidf$ values. Here, every node has an input vector, and the number of children of nodes varies significantly[3].

In rumor detection, long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU) (Cho et al., 2014) were used to learn textual representation, which adopts memory units to store information over long time steps (Ma et al., 2016). In this paper, we choose to extend GRU as hidden unit to model long-distance interactions over the tree nodes because it is more efficient due to fewer parameters. Let $\mathcal{S}(j)$ denote the set of direct children of the node j. The transition equations of node j in the bottom-up model are formulated as follows:

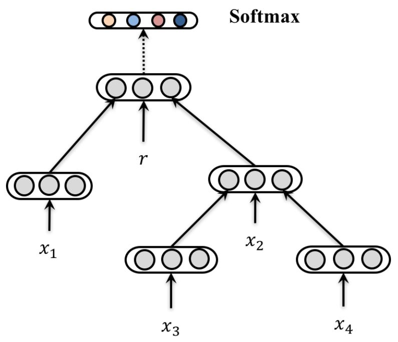
$$\begin{aligned}
\tilde{x}_j &= x_j E \\
h_{\mathcal{S}} &= \sum_{s \in \mathcal{S}(j)} h_s \\
r_j &= \sigma\left(W_r \tilde{x}_j + U_r h_{\mathcal{S}}\right) \\
z_j &= \sigma\left(W_z \tilde{x}_j + U_z h_{\mathcal{S}}\right) \\
\tilde{h}_j &= tanh\left(W_h \tilde{x}_j + U_h (h_{\mathcal{S}} \odot r_j)\right) \\
h_j &= (1 - z_j) \odot h_{\mathcal{S}} + z_j \odot \tilde{h}_j
\end{aligned} \tag{1}$$

[3]In standard RvNN, since an input instance is the parse tree of a sentence, only leaf nodes have input vector, each node representing a word of the input sentence, and the non-leaf nodes are constituents of the sentence, and thus the number of children of a node is limited.

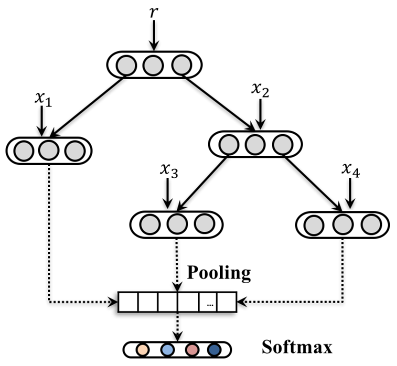


r: #Walmart donates $10,000 to #DarrenWilson fund to continue police racial profiling…
x1: 1:30 Idc if they killed a mf foreal. Ima always shop with @Walmart. I'm just bein …
x2: NEED SOURCE. have a feeling this is just hearsay ...
x3: I agree. I have been hearing this all day but no source…
x4: Exactly, I don't think Wal-Mart would let everyone know this if they did!!

(a) Bottom-up/Top-down tree



(b) Bottom-up RvNN model



(c) Top-down RvNN model

Figure 3: A bottom-up/top-down propagation tree and the corresponding RvNN-based models. The black-color and red-color edges differentiate the bottom-up and top-down tree in Figure 3(a).

where x_j is the original input vector of node j, E denotes the parameter matrix for transforming this input post, $\tilde{x}_j$ is the transformed representation of j, $[W_*, U_*]$ are the weight connections inside GRU, and h_j and h_s refer to the hidden state of j and its s-th child. Thus $h_{\mathcal{S}}$ denotes the sum of the hidden state of all the children of j assuming that all children are equally important to j. As with the standard GRU, $\odot$ denotes element-wise multiplication; a reset gate r_j determines how to combine the current input $\tilde{x}_j$ with the memory of children, and an update gate z_j defines how much memory from the children is cascaded into the current node; and $\tilde{h}_j$ denotes the candidate activation of the hidden state of the current node. Different from the standard GRU unit, the gating vectors in our variant of GRU are dependent on the states of many child units, allowing our model to incorporate representations from different children.

After recursive aggregation from bottom to up, the state of root node (i.e., source tweet) can be regard as the representation of the whole tree which is used for supervised classification. So, an output layer is connected to the root node for predicting the class of the tree using a softmax function:

$$\hat{y} = Softmax(Vh_0 + b) \tag{2}$$

where h_0 is the learned hidden vector of root node; V and b are the weights and bias in output layer.

4.3 Top-down RvNN

This model is designed to leverage the structure of top-down tree to capture complex propagation patterns for classifying rumorous claims, which is shown in Figure 3(c). It models how the information flows from source post to the current node. The idea of this top-down approach is to generate a strengthened feature vector for each post considering its propagation path, where rumor-indicative features are aggregated along the propagation history in the path. For example, if current post agree with its parent's stance which denies the source post, the denial stance from the root node down to the current node on this path should be reinforced. Due to different branches of any non-leaf node, the top-down visit to its subtree nodes is also recursive. However, the nature of top-down tree lends this model different from the bottom-up one. The representation of each node is computed by combining its own input and its parent node instead of its children nodes. This process proceeds recursively from the root node to its children until all leaf nodes are reached.

Suppose that the hidden state of a non-leaf node can be passed synchronously to all its child nodes without loss. Then the hidden state h_j of a node j can be computed by combining the hidden state $h_{\mathcal{P}(j)}$ of its parent node $\mathcal{P}(j)$ and its own input vector x_j. Therefore, the transition equations of node j can be formulated as a standard GRU:

$$\begin{aligned}
\tilde{x}_j &= x_j E \\
r_j &= \sigma\left(W_r \tilde{x}_j + U_r h_{\mathcal{P}(j)}\right) \\
z_j &= \sigma\left(W_z \tilde{x}_j + U_z h_{\mathcal{P}(j)}\right) \\
\tilde{h}_j &= tanh\left(W_h \tilde{x}_j + U_h (h_{\mathcal{P}(j)} \odot r_j)\right) \\
h_j &= (1 - z_j) \odot h_{\mathcal{P}(j)} + z_j \odot \tilde{h}_j
\end{aligned} \tag{3}$$

Through the top-down recursion, the learned representations are eventually embedded into the hidden vector of all the leaf nodes. Since the number of leaf nodes varies, the resulting vectors cannot be directly fed into a fixed-size neural layer for output. Therefore, we add a max-pooling layer to take the maximum value of each dimension of the vectors over all the leaf nodes. This can also help capture the most appealing indicative features from all the propagation paths.

Based on the pooling result, we finally use a softmax function in the output layer to predict the label of the tree:

$$\hat{y} = Softmax(Vh_\infty + b) \tag{4}$$

where h_∞ is the pooling vector over all leaf nodes, V and b are parameters in the output layer.

Although both of the two RvNN models aim to capture the structural properties by recursively visiting all nodes, we can conjecture that the top-down model would be better. The hypothesis is that in the bottom-up case the final output relies on the representation of single root, and its information loss can be larger than the top-down one since in the top-down case the representations embedded into all leaf nodes along different propagation paths can be incorporated via pooling holistically.

4.4 Model Training

The model is trained to minimize the squared error between the probability distributions of the predictions and the ground truth:

$$L(y, \hat{y}) = \sum_{n=1}^{N} \sum_{c=1}^{C} (y_c - \hat{y}_c)^2 + \lambda ||\theta||_2^2 \tag{5}$$

where y_c is the ground truth and $\hat{y}_c$ is the prediction probability of a class, N is the number of training claims, C is the number of classes, $||.||_2$ is the L_2 regularization term over all model parameters θ, and λ is the trade-off coefficient.

During training, all the model parameters are updated using efficient back-propagation through structure (Goller and Kuchler, 1996; Socher et al., 2013), and the optimization is gradient-based following the Ada-grad update rule (Duchi et al., 2011) to speed up the convergence. We empirically initialize the model parameters with uniform distribution and set the vocabulary size as 5,000, the size of embedding and hidden units as 100. We iterate over all the training examples in each epoch and continue until the loss value converges or the maximum epoch number is met.

5 Experiments and Results

5.1 Datasets

For experimental evaluation, we use two publicly available Twitter datasets released by Ma et al. (2017), namely Twitter15 and Twitter16[4], which respectively contains 1,381 and 1,181 propagation trees (see (Ma et al., 2017) for detailed statistics). In each dataset, a group of wide spread source tweets along with their propagation threads, i.e., replies and retweets, are provided in the form of tree structure. Each tree is annotated with one of the four class labels, i.e., non-rumor, false rumor, true rumor and unverified rumor. We remove the retweets from the trees since they do not provide any extra information or evidence content-wise. We build two versions for each tree, one for the bottom-up tree and the other for the top-down tree, by flipping the edges' direction.

5.2 Experimental Setup

We make comprehensive comparisons between our models and some state-of-the-art baselines on rumor classification and early detection tasks.

- DTR: Zhao et al. (2015) proposed a Decision-Tree-based Ranking model to identify trending rumors by searching for inquiry phrases.

- DTC: The information credibility model using a Decision-Tree Classifier (Castillo et al., 2011) based on manually engineering various statistical features of the tweets.

- RFC: The Random Forest Classier using 3 fitting parameters as temporal properties and a set of handcrafted features on user, linguistic and structural properties (Kwon et al., 2013).

- SVM-TS: A linear SVM classifier that uses time-series to model the variation of handcrafted social context features (Ma et al., 2015).

- SVM-BOW: A naive baseline we built by representing text content using bag-of-words and using linear SVM for rumor classification.

- SVM-TK and **SVM-HK**: SVM classifier uses a Tree Kernel (Ma et al., 2017) and that uses a Hybrid Kernel (Wu et al., 2015), respectively, both of which model propagation structures with kernels.

- GRU-RNN: A detection model based on recurrent neural networks (Ma et al., 2016) with GRU units for learning rumor representations by modeling sequential structure of relevant posts.

[4]`https://www.dropbox.com/s/7ewzdrbelpmrnxu/rumdetect2017.zip?dl=0`

(a) Twitter15 dataset

Method	Acc.	NR F_1	FR F_1	TR F_1	UR F_1
DTR	0.409	0.501	0.311	0.364	0.473
DTC	0.454	0.733	0.355	0.317	0.415
RFC	0.565	**0.810**	0.422	0.401	0.543
SVM-TS	0.544	0.796	0.472	0.404	0.483
SVM-BOW	0.548	0.564	0.524	0.582	0.512
SVM-HK	0.493	0.650	0.439	0.342	0.336
SVM-TK	0.667	0.619	0.669	0.772	0.645
GRU-RNN	0.641	0.684	0.634	0.688	0.571
BU-RvNN	0.708	0.695	0.728	0.759	0.653
TD-RvNN	**0.723**	0.682	**0.758**	**0.821**	**0.654**

(b) Twitter16 dataset

Method	Acc.	NR F_1	FR F_1	TR F_1	UR F_1
DTR	0.414	0.394	0.273	0.630	0.344
DTC	0.465	0.643	0.393	0.419	0.403
RFC	0.585	0.752	0.415	0.547	0.563
SVM-TS	0.574	**0.755**	0.420	0.571	0.526
SVM-BOW	0.585	0.553	0.556	0.655	0.578
SVM-HK	0.511	0.648	0.434	0.473	0.451
SVM-TK	0.662	0.643	0.623	0.783	0.655
GRU-RNN	0.633	0.617	0.715	0.577	0.527
BU-RvNN	0.718	0.723	0.712	0.779	0.659
TD-RvNN	**0.737**	0.662	**0.743**	**0.835**	**0.708**

Table 1: Results of rumor detection. (NR: non-rumor; FR: false rumor; TR: true rumor; UR: unverified rumor)

- **BU-RvNN** and **TD-RvNN**: Our bottom-up and top-down RvNN models, respectively.

We implement **DTC** and **RFC** using Weka[5], SVM-based models using LibSVM[6] and all neural-network-based models with Theano[7]. We conduct 5-fold cross-validation on the datasets and use accuracy over all the four categories and F1 measure on each class to evaluate the performance of models.

5.3 Rumor Classification Performance

As shown in Table 1, our proposed models basically yield much better performance than other methods on both datasets via the modeling of interaction structures of posts in the propagation.

It is observed that the performance of the 4 baselines in the first group based on handcrafted features is obviously poor, varying between 0.409 and 0.585 in accuracy, indicating that they fail to generalize due to the lack of capacity capturing helpful features. Among these baselines, SVM-TS and RFC perform relatively better because they use additional temporal traits, but they are still clearly worse than the models not relying on feature engineering. DTR uses a set of regular expressions indicative of stances. However, only 19.6% and 22.2% tweets in the two datasets contain strings covered by these regular expressions, rendering unsatisfactory result.

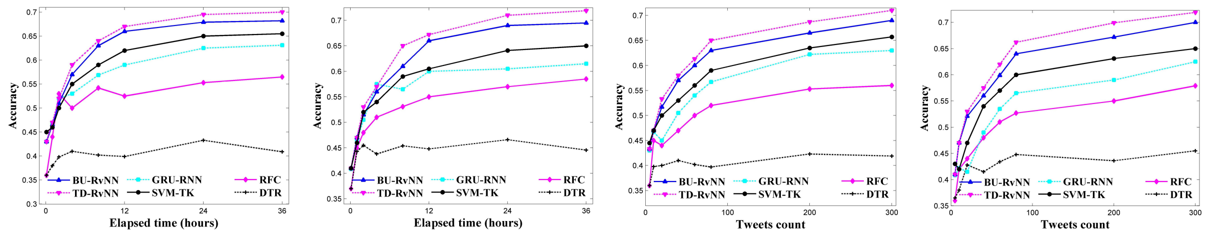
Among the two kernel methods that are based on comparing propagation structures, we observe that SVM-TK is much more effective than SVM-HK. There are two reasons: 1) SVM-HK was originally proposed and experimented on Sina Weibo (Wu et al., 2015), which may not be generalize well on Twitter. 2) SVM-HK loosely couples two separate kernels: a RBF kernel based on handcrafted features, plus a random walk-based kernel which relies on a set of pre-defined keywords for jumping over the nodes probabilistically. This under utilizes the propagation information due to such oversimplified treatment of tree structure. In contrast, SVM-TK is an integrated kernel and can fully utilize the structure by comparing the trees based on both textual and structural similarities.

It appears that using bag-of-words is already a decent model evidenced as the fairly good performance of SVM-BOW which is even better than SVM-HK. This is because the features of SVM-HK are handcrafted for binary classification (i.e., non-rumor vs rumor), ignoring the importance of indicative words or units that benefit finer-grained classification which can be captured more effectively by SVM-BOW.

The sequential neural model GRU-RNN performs slightly worse than SVM-TK, but much worse than our recursive models. This is because it is a special case of the recursive model where each non-leaf node has only one child. It has to rely on a linear chain as input, which missed out valuable structural information. However, it does learn high-level features from the post content via hidden units of the neural model while SVM-TK cannot which can only evaluates similarities based on the overlapping words among subtrees. Our recursive models are inherently tree-structured and take advantages of representation learning following the propagation structure, thus beats SVM-TK.

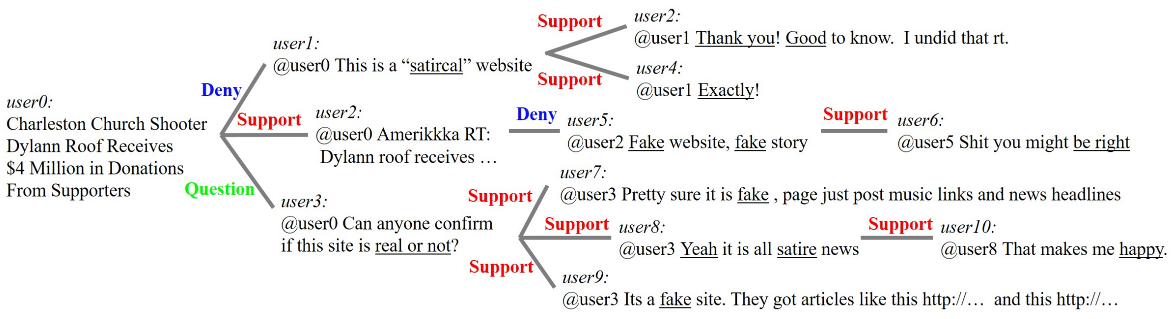
In the two recursive models, TD-RvNN outperforms BU-RvNN, which indicates that the bottom-up model may suffer from larger information loss than the top-down one. This verifies the hypothesis we made in Section 4.3 that the pooling layer

[5]www.cs.waikato.ac.nz/ml/weka
[6]www.csie.ntu.edu.tw/~cjlin/libsvm
[7]deeplearning.net/software/theano



(a) Twitter15 (elapsed time) (b) Twitter16 (elapsed time) (c) Twitter15 (tweets count) (d) Twitter16 (tweets count)

Figure 4: Early rumor detection accuracy at different checkpoints in terms of elapsed time (tweets count).



user0: Charleston Church Shooter Dylann Roof Receives $4 Million in Donations From Supporters

user1: @user0 This is a "satircal" website

user2: @user1 Thank you! Good to know. I undid that rt.

user4: @user1 Exactly!

user2: @user0 Amerikkka RT: Dylann roof receives …

user5: @user2 Fake website, fake story

user6: @user5 Shit you might be right

user3: @user0 Can anyone confirm if this site is real or not?

user7: @user3 Pretty sure it is fake , page just post music links and news headlines

user8: @user3 Yeah it is all satire news

user10: @user8 That makes me happy.

user9: @user3 Its a fake site. They got articles like this http://… and this http://…

Figure 5: A correctly detected false rumor at early stage by both of our models, where propagation paths are marked with relevant stances. Note that edge direction is not shown as it applies to either case.

in the top-down model can effectively select important features embedded into the leaf nodes.

For only the non-rumor class, it seems that our method does not perform so well as some feature-engineering baselines. This can be explained by the fact that these baselines are trained with additional features such as user information (e.g., profile, verification status, etc) which may contain clues for differentiating non-rumors from rumors. Also, the responses to non-rumors are usually much more diverse with little informative indication, making identification of non-rumors more difficult based on content even with the structure.

5.4 Early Rumor Detection Performance

Detecting rumors at early state of propagation is important so that interventions can be made in a timely manner. We compared different methods in term of different time delays measured by either tweet count received or time elapsed since the source tweet is posted. The performance is evaluated by the accuracy obtained when we incrementally add test data up to the check point given the targeted time delay or tweets volume.

Figure 4 shows that the performance of our recursive models climbs more rapidly and starts to supersede the other models at the early stage. Although all the methods are getting to their best performance in the end, TD-RvNN and BU-RvNN only need around 8 hours or about 90 tweets to achieve the comparable performance of the best baseline model, i.e., SVM-TK, which needs about 36 hours or around 300 posts, indicating superior early detection performance of our method.

Figure 5 shows a sample tree at the early stage of propagation that has been correctly classified as a false rumor by both recursive models. We can see that this false rumor demonstrates typical patterns in subtrees and propagation paths indicative of the falsehood, where a set of responses supporting the parent posts that deny or question the source post are captured by our bottom-up model. Similarly, some patterns of propagation from the root to leaf nodes like "support$\rightarrow$deny$\rightarrow$support" are also seized by our top-down model. In comparison, sequential models may be confused because the supportive key terms such as "be right", "yeah", "exactly!" dominate the responses, and the SVM-TK may miss similar subtrees by just comparing the surface words.

6 Conclusions and Future Work

We propose a bottom-up and a top-down tree-structured model based on recursive neural networks for rumor detection on Twitter. The inherent nature of recursive models allows them using propagation tree to guide the learning of representations from tweets content, such as embedding various indicative signals hidden in the structure, for better identifying rumors. Results on two public Twitter datasets show that our method improves rumor detection performance in very large margins as compared to state-of-the-art baselines.

In our future work, we plan to integrate other types of information such as user properties into the structured neural models to further enhance representation learning and detect rumor spreaders at the same time. We also plan to use unsupervised models for the task by exploiting structural information.

Acknowledgment

This work is partly supported by Innovation and Technology Fund (ITF) Project No. 6904333, and General Research Fund (GRF) Project No. 14232816 (12183516). We would like to thank anonymous reviewers for the insightful comments.

References

Gordon W Allport and Leo Postman. 1946. An analysis of rumor. *Public Opinion Quarterly* 10(4):501–517.

G.W. Allport and L.J. Postman. 1965. *The psychology of rumor*. Russell & Russell.

Adam J. Berinsky. 2017. Rumors and health care reform: Experiments in political misinformation. *British Journal of Political Science* 47(2):241262.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of WWW*. pages 675–684.

Tong Chen, Lin Wu, Xue Li, Jun Zhang, Hongzhi Yin, and Yang Wang. 2017. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. *arXiv preprint arXiv:1704.05973* .

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Nicholas DiFonzo and Prashant Bordia. 2007. Rumor, gossip and urban legends. *Diogenes* 54(1):19–35.

Nicholas DiFonzo, Prashant Bordia, and Ralph L Rosnow. 1994. Reining in rumors. *Organizational Dynamics* 23(1):47–62.

Pamela Donovan. 2007. How idle is idle talk? one hundred years of rumor research. *Diogenes* 54(1):59–82.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Adrien Friggeri, Lada A Adamic, Dean Eckles, and Justin Cheng. 2014. Rumor cascades. In *Proceedings of ICWSM*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*. IEEE, volume 1, pages 347–352.

Aniko Hannak, Drew Margolin, Brian Keegan, and Ingmar Weber. 2014. Get back! you don't know me like that: The social mediation of fact checking interventions in twitter conversations. In *Proceedings of ICWSM*.

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

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14, pages 2096–2104.

Marianne E Jaeger, Susan Anthony, and Ralph L Rosnow. 1980. Who hears what from whom and with what effect: A study of rumor. *Personality and Social Psychology Bulletin* 6(3):473–478.

Allan J Kimmel. 2004. *Rumors and rumor control: A manager's guide to understanding and combatting rumors*. Routledge.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *Proceedings of ICDM*. pages 1103–1108.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15, pages 1867–1870.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 393–398.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15, pages 1751–1754.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 708–717.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Detect rumor and stance jointly by neural multi-task learning. In *Companion Proceedings of the The Web Conference 2018*. WWW '18, pages 585–593.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106* .

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11, pages 1589–1599.

Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. 2011. Truthy: mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th International Conference Companion on World Wide Web*. WWW '11, pages 249–252.

Ralph L Rosnow and Eric K Foster. 2005. Rumor and gossip research. *Psychological Science Agenda* 19(4).

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17, pages 797–806.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL '12, pages 1201–1211.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 129–136.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, pages 651–662.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. MDS '12, pages 13:1–13:7.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*. WWW '15, pages 1395–1405.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 1604–1612.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2017. Detection and resolution of rumours in social media: A survey. *arXiv preprint arXiv:1704.00656* .

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016a. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 2438–2448.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016b. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one* 11(3):e0150989.

Visual Attention Model for Name Tagging in Multimodal Social Media

Di Lu[*1], **Leonardo Neves**[2], **Vitor Carvalho**[3], **Ning Zhang**[2], **Heng Ji**[1]
[1]Computer Science, Rensselaer Polytechnic Institute
{lud2,jih}@rpi.edu
[2]Snap Research
{lneves, ning.zhang}@snap.com
[3]Intuit
vitor_carvalho@intuit.com

Abstract

Everyday billions of multimodal posts containing both images and text are shared in social media sites such as Snapchat, Twitter or Instagram. This combination of image and text in a single message allows for more creative and expressive forms of communication, and has become increasingly common in such sites. This new paradigm brings new challenges for natural language understanding, as the textual component tends to be shorter, more informal, and often is only understood if combined with the visual context. In this paper, we explore the task of name tagging in multimodal social media posts. We start by creating two new multimodal datasets: one based on Twitter posts[1] and the other based on Snapchat captions (exclusively submitted to public and crowd-sourced stories). We then propose a novel model based on Visual Attention that not only provides deeper visual understanding on the decisions of the model, but also significantly outperforms other state-of-the-art baseline methods for this task. [2]

1 Introduction

Social platforms, like Snapchat, Twitter, Instagram and Pinterest, have become part of our lives and play an important role in making communication easier and accessible. Once text-centric, social media platforms are becoming increasingly multimodal, with users combining images, videos, audios, and texts for better expressiveness. As social media posts become more multimodal, the natural language understanding of the textual components of these messages becomes increasingly challenging. In fact, it is often the case that the textual component can only be understood in combination with the visual context of the message.

In this context, here we study the task of Name Tagging for social media containing both image and textual contents. Name tagging is a key task for language understanding, and provides input to several other tasks such as Question Answering, Summarization, Searching and Recommendation. Despite its importance, most of the research in name tagging has focused on news articles and longer text documents, and not as much in multimodal social media data (Baldwin et al., 2015).

However, multimodality is not the only challenge to perform name tagging on such data. The textual components of these messages are often very short, which limits context around names. Moreover, there linguistic variations, slangs, typos and colloquial language are extremely common, such as using 'looooove' for 'love', 'LosAngeles' for 'Los Angeles', and '#Chicago #Bull' for 'Chicago Bulls'. These characteristics of social media data clearly illustrate the higher difficulty of this task, if compared to traditional newswire name tagging.

In this work, we modify and extend the current state-of-the-art model (Lample et al., 2016; Ma and Hovy, 2016) in name tagging to incorporate the visual information of social media posts using an Attention mechanism. Although the usually short textual components of social media posts provide limited contextual information, the accompanying images often provide rich information that can be useful for name tagging. For ex-

[*]This work was mostly done during the first author's internship at Snap Research.

[1]The Twitter data and associated images presented in this paper were downloaded from `https://archive.org/details/twitterstream`

[2]We will make the annotations on Twitter data available for research purpose upon request.

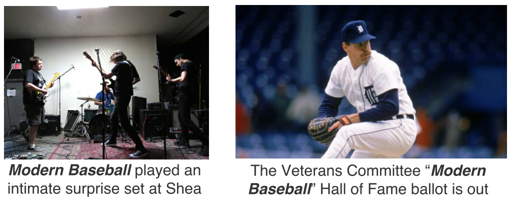

Modern Baseball played an intimate surprise set at Shea
The Veterans Committee "***Modern Baseball***" Hall of Fame ballot is out

Figure 1: Examples of *Modern Baseball* associated with different images.

ample, as shown in Figure 1, both captions include the phrase *'Modern Baseball'*. It is not easy to tell if each *Modern Baseball* refers to a name or not from the textual evidence only. However using the associated images as reference, we can easily infer that *Modern Baseball* in the first sentence should be the name of a band because of the implicit features from the objects like instruments and stage, and the *Modern Baseball* in the second sentence refers to the sport of baseball because of the pitcher in the image.

In this paper, given an image-sentence pair as input, we explore a new approach to leverage visual context for name tagging in text. First, we propose an attention-based model to extract visual features from the regions in the image that are most related to the text. It can ignore irrelevant visual information. Secondly, we propose to use a gate to combine textual features extracted by a Bidirectional Long Short Term Memory (BLSTM) and extracted visual features, before feed them into a Conditional Random Fields(CRF) layer for tag predication. The proposed gate architecture plays the role to modulate word-level multimodal features.

We evaluate our model on two labeled datasets collected from Snapchat and Twitter respectively. Our experimental results show that the proposed model outperforms state-for-the-art name tagger in multimodal social media.

The main contributions of this work are as follows:

- We create two new datasets for name tagging in multimedia data, one using Twitter and the other using crowd-sourced Snapchat posts. These new datasets effectively constitute new benchmarks for the task.

- We propose a visual attention model specifically for name tagging in multimodal social media data. The proposed end-to-end model only uses image-sentence pairs as input without any human designed features, and a Visual Attention component that helps understand the decision making of the model.

2 Model

Figure 2 shows the overall architecture of our model. We describe three main components of our model in this section: BLSTM-CRF sequence labeling model (Section 2.1), Visual Attention Model (Section 2.3) and Modulation Gate (Section 2.4).

Given a pair of sentence and image as input, the Visual Attention Model extracts regional visual features from the image and computes the weighted sum of the regional visual features as the visual context vector, based on their relatedness with the sentence. The BLSTM-CRF sequence labeling model predicts the label for each word in the sentence based on both the visual context vector and the textual information of the words. The modulation gate controls the combination of the visual context vector and the word representations for each word before the CRF layer.

2.1 BLSTM-CRF Sequence Labeling

We model name tagging as a sequence labeling problem. Given a sequence of words: $S = \{s_1, s_2, ..., s_n\}$, we aim to predict a sequence of labels: $L = \{l_1, l_2, ..., l_n\}$, where $l_i \in \mathfrak{L}$ and $\mathfrak{L}$ is a pre-defined label set.

Bidirectional LSTM. Long Short-term Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are variants of Recurrent Neural Networks (RNNs) designed to capture long-range dependencies of input. The equations of a LSTM cell are as follows:

$$\begin{gathered}
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
\tilde{c}_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
h_t = o_t \odot tanh(c_t)
\end{gathered}$$

where x_t, c_t and h_t are the input, memory and hidden state at time t respectively. W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xc}, W_{hc}, W_{xo}, and W_{ho} are weight matrices. $\odot$ is the element-wise product function and σ is the element-wise sigmoid function.

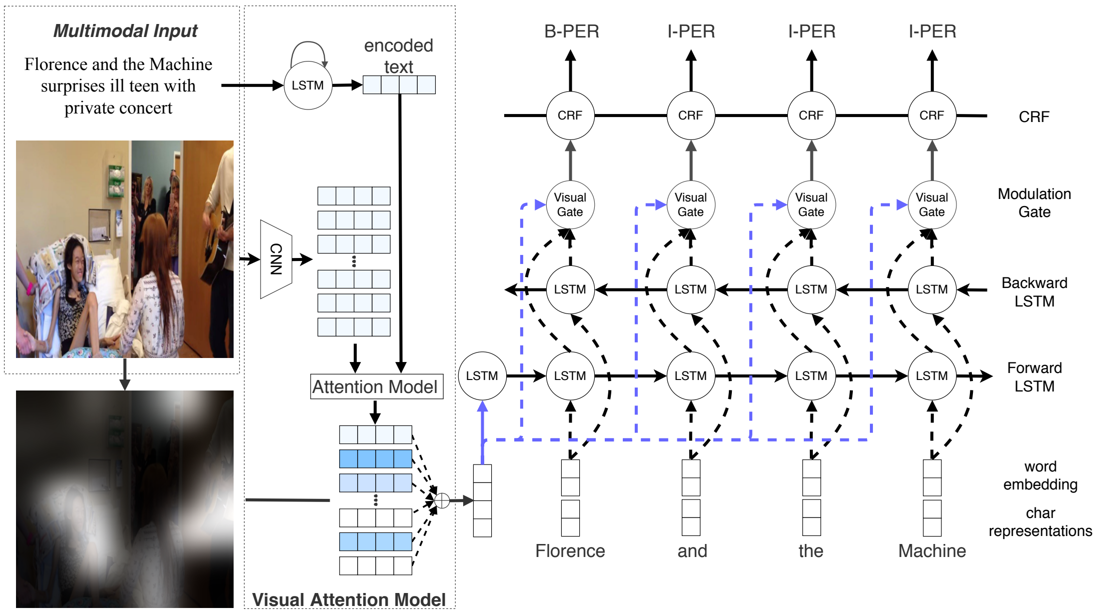


Figure 2: Overall Architecture of the Visual Attention Name Tagging Model.

Name Tagging benefits from both of the past (left) and the future (right) contexts, thus we implement the Bidirectional LSTM (Graves et al., 2013; Dyer et al., 2015) by concatenating the left and right context representations, $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$, for each word.

Character-level Representation. Following (Lample et al., 2016), we generate the character-level representation for each word using another BLSTM. It receives character embeddings as input and generates representations combining implicit prefix, suffix and spelling information. The final word representation x_i is the concatenation of word embedding e_i and character-level representation c_i.

$$c_i = BLSTM_{char}(s_i) \quad s_i \in S$$
$$x_i = [e_i, c_i]$$

Conditional random fields (CRFs). For name tagging, it is important to consider the constraints of the labels in neighborhood (e.g., I-LOC must follow B-LOC). CRFs (Lafferty et al., 2001) are effective to learn those constraints and jointly predict the best chain of labels. We follow the implementation of CRFs in (Ma and Hovy, 2016).

2.2 Visual Feature Representation

We use Convolutional Neural Networks (CNNs) (LeCun et al., 1989) to obtain the representations of images. Particularly, we use Residual Net (ResNet) (He et al., 2016), which

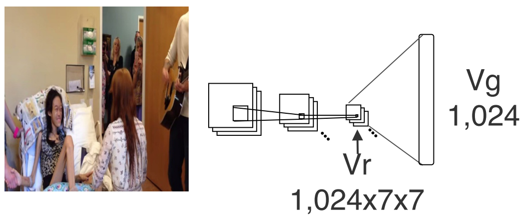


Figure 3: CNN for visual features extraction.

achieves state-of-the-art on ImageNet (Russakovsky et al., 2015) detection, ImageNet localization, COCO (Lin et al., 2014) detection, and COCO segmentation tasks. Given an input pair (S, I), where S represents the word sequence and I represents the image rescaled to 224x224 pixels, we use ResNet to extract visual features for regional areas as well as for the whole image (Fig 3):

$$V_g = ResNet_g(I)$$
$$V_r = ResNet_r(I)$$

where the global visual vector V_g, which represents the whole image, is the output before the last fully connected layer[3]. The dimension of V_g is 1,024. V_r are the visual representations for regional areas and they are extracted from the last convolutional layer of ResNet, and the dimension is 1,024x7x7 as shown in Figure 3. 7x7 is the number of regions in the image and 1,024 is the

[3]the last fully connect layer outputs the probabilities over 1,000 classes of objects.

dimension of the feature vector. Thus each feature vector of V_r corresponds to a 32x32 pixel region of the rescaled input image.

2.3 Visual Attention Model

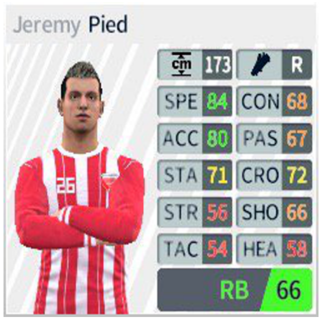


Figure 4: Example of partially related image and sentence. (‘*I have just bought Jeremy Pied.*’)

The global visual representation is a reasonable representation of the whole input image, but not the best. Sometimes only parts of the image are related to the associated sentence. For example, the visual features from the right part of the image in Figure 4 cannot contribute to inferring the information in the associated sentence ‘*I have just bought Jeremy Pied.*’ In this work we utilize visual attention mechanism to combat the problem, which has been proven effective for vision-language related tasks such as Image Captioning (Xu et al., 2015) and Visual Question Answering (Yang et al., 2016b; Lu et al., 2016), by enforcing the model to focus on the regions in images that are mostly related to context textual information while ignoring irrelevant regions. Also the visualization of attention can also help us to understand the decision making of the model. Attention mechanism is mapping a query and a set of key-value pairs to an output. The output is a weighted sum of the values and the assigned weight for each value is computed by a function of the query and corresponding key. We encode the sentence into a query vector using an LSTM, and use regional visual representations V_r as both keys and values.

Text Query Vector. We use an LSTM to encode the sentence into a query vector, in which the inputs of the LSTM are the concatenations of word embeddings and character-level word representations. Different from the LSTM model used for sequence labeling in Section 2.1, the LSTM here aims to get the semantic information of the sentence and it is unidirectional:

$$Q = LSTM_{query}(S) \quad (1)$$

Attention Implementation. There are many implementations of visual attention mechanism such as Multi-layer Perceptron (Bahdanau et al., 2014), Bilinear (Luong et al., 2015), dot product (Luong et al., 2015), Scaled Dot Product (Vaswani et al., 2017), and linear projection after summation (Yang et al., 2016b). Based on our experimental results, dot product implementations usually result in more concentrated attentions and linear projection after summation results in more dispersed attentions. In the context of name tagging, we choose the implementation of linear projection after summation because it is beneficial for the model to utilize as many related visual features as possible, and concentrated attentions may make the model bias. For implementation, we first project the text query vector Q and regional visual features V_r into the same dimensions:

$$P_t = tanh(W_t Q)$$
$$P_v = tanh(W_v V_r)$$

then we sum up the projected query vector with each projected regional visual vector respectively:

$$A = P_t \oplus P_v$$

the weights of the regional visual vectors:

$$E = softmax(W_a A + b_a)$$

where W_a is weights matrix. The weighted sum of the regional visual features is:

$$v_c = \sum \alpha_i v_i \quad \alpha_i \in E, v_i \in V_r$$

We use v_c as the visual context vector to initialize the BLSTM sequence labeling model in Section 2.1. We compare the performances of the models using global visual vector V_g and attention based visual context vector V_c for initialization in Section 4.

2.4 Visual Modulation Gate

The BLSTM-CRF sequence labeling model benefits from using the visual context vector to initialize the LSTM cell. However, the better way to utilize visual features for sequence labeling is to incorporate the features at word level individually. However visual features contribute quite differently when they are used to infer the tags of different words. For example, we can easily find matched visual patterns from associated images for verbs such as '*sing*', '*run*', and '*play*'. Words/Phrases such as names of basketball players, artists, and buildings are often well-aligned with objects in images. However it is difficult to align function words such as '*the*', '*of*' and '*well*' with visual features. Fortunately, most of the challenging cases in name tagging involve nouns and verbs, the disambiguation of which can benefit more from visual features.

We propose to use a visual modulation gate, similar to (Miyamoto and Cho, 2016; Yang et al., 2016a), to dynamically control the combination of visual features and word representation generated by BLSTM at word-level, before feed them into the CRF layer for tag prediction. The equations for the implementation of modulation gate are as follows:

$$\beta_v = \sigma(W_v h_i + U_v v_c + b_v)$$
$$\beta_w = \sigma(W_w h_i + U_w v_c + b_w)$$
$$m = tanh(W_m h_i + U_m v_c + b_m)$$
$$w_m = \beta_w \cdot h_i + \beta_v \cdot m$$

where h_i is the word representation generated by BLSTM, v_c is the computed visual context vector, W_v, W_w, W_m, U_v, U_w and U_m are weight matrices, σ is the element-wise sigmoid function, and w_m is the modulated word representations fed into the CRF layer in Section 2.1. We conduct experiments to evaluate the impact of modulation gate in Section 4.

3 Datasets

We evaluate our model on two multimodal datasets, which are collected from Twitter and Snapchat respectively. Table 1 summarizes the data statistics. Both datasets contain four types of named entities: `Location`, `Person`, `Organization` and `Miscellaneous`. Each data instance contains a pair of sentence and image, and the names in sentences are manually tagged by three expert labelers.

Twitter name tagging. The Twitter name tagging dataset contains pairs of tweets and their associated images extracted from May 2016, January 2017 and June 2017. We use sports and social event related key words, such as *concert, festival, soccer, basketball*, as queries. We don't take into consideration messages without images for this experiment. If a tweet has more than one image associated to it, we randomly select one of the images.

Snap name tagging. The Snap name tagging dataset consists of caption and image pairs exclusively extracted from snaps submitted to public and live stories. They were collected between May and July of 2017. The data contains captions submitted to multiple community curated stories like the Electric Daisy Carnival (EDC) music festival and the Golden State Warrior's NBA parade.

Both Twitter and Snapchat are social media with plenty of multimodal posts, but they have obvious differences with sentence length and image styles. In Twitter, text plays a more important role, and the sentences in the Twitter dataset are much longer than those in the Snap dataset (16.0 tokens vs 8.1 tokens). The image is often more related to the content of the text and added with the purpose of illustrating or giving more context. On the other hand, as users of Snapchat use cameras to communicate, the roles of text and image are switched. Captions are often added to complement what is being portrayed by the snap. On our experiment section we will show that our proposed model outperforms baseline on both datasets.

We believe the Twitter dataset can be an important step towards more research in multimodal name tagging and we plan to provide it as a benchmark upon request.

4 Experiment

4.1 Training

Tokenization. To tokenize the sentences, we use the same rules as (Owoputi et al., 2013), except we separate the hashtag '#' with the words after.

Labeling Schema. We use the standard BIO schema (Sang and Veenstra, 1999), because we see little difference when we switch to BIOES schema (Ratinov and Roth, 2009).

Word embeddings. We use the 100-dimensional GloVe[4] (Pennington et al., 2014) embeddings trained on 2 billions tweets to initialize the lookup table and do fine-tuning during training.

Character embeddings. As in (Lample et al., 2016), we randomly initialize the character embeddings with uniform samples. Based on experimental results, the size of the character embeddings affects little, and we set it as 50.

[4]https://nlp.stanford.edu/projects/glove/

		Training	**Development**	**Testing**
Snapchat	Sentences	4,817	1,032	1,033
	Tokens	39,035	8,334	8,110
Twitter	Sentences	4,290	1,432	1,459
	Tokens	68,655	22,872	23,051

Table 1: Sizes of the datasets in numbers of sentence and token.

Pretrained CNNs. We use the pretrained ResNet-152 (He et al., 2016) from Pytorch.

Early Stopping. We use early stopping (Caruana et al., 2001; Graves et al., 2013) with a patience of 15 to prevent the model from over-fitting.

Fine Tuning. The models are optimized with fine-tuning on both the word-embeddings and the pre-trained ResNet.

Optimization. The models achieve the best performance by using mini-batch stochastic gradient descent (SGD) with batch size 20 and momentum 0.9 on both datasets. We set an initial learning rate of $\eta_0 = 0.03$ with decay rate of $\rho = 0.01$. We use a gradient clipping of 5.0 to reduce the effects of gradient exploding.

Hyper-parameters. We summarize the hyper-parameters in Table 2.

Hyper-parameter	**Value**
LSTM hidden state size	300
Char LSTM hidden state size	50
visual vector size	100
dropout rate	0.5

Table 2: Hyper-parameters of the networks.

4.2 Results

Table 3 shows the performance of the baseline, which is BLSTM-CRF with sentences as input only, and our proposed models on both datasets.

BLSTM-CRF + Global Image Vector: use global image vector to initialize the BLSTM-CRF.

BLSTM-CRF + Visual attention: use attention based visual context vector to initialize the BLSTM-CRF.

BLSTM-CRF + Visual attention + Gate: modulate word representations with visual vector.

Our final model **BLSTM-CRF + VISUAL ATTENTION + GATE**, which has visual attention component and modulation gate, obtains the best F1 scores on both datasets. Visual features successfully play a role of validating entity types. For example, when there is a person in the image, it is more likely to include a person name in the associated sentence, but when there is a soccer field in the image, it is more likely to include a sports team name.

All the models get better scores on Twitter dataset than on Snap dataset, because the average length of the sentences in Snap dataset (8.1 tokens) is much smaller than that of Twitter dataset (16.0 tokens), which means there is much less contextual information in Snap dataset.

Also comparing the gains from visual features on different datasets, we find that the model benefits more from visual features on Twitter dataset, considering the much higher baseline scores on Twitter dataset. Based on our observation, users of Snapchat often post selfies with captions, which means some of the images are not strongly related to their associated captions. In contrast, users of Twitter prefer to post images to illustrate texts

4.3 Attention Visualization

Figure 5 shows some good examples of the attention visualization and their corresponding name tagging results. The model can successfully focus on appropriate regions when the images are well aligned with the associated sentences. Based on our observation, the multimodal contexts in posts related to sports, concerts or festival are usually better aligned with each other, therefore the visual features easily contribute to these cases. For example, the ball and shoot action in example (a) in Figure 5 indicates that the context should be related to basketball, thus the '*Warriors*' should be the name of a sports team. A singing person with a microphone in example (b) indicates that the name of an artist or a band ('*Radiohead*') may appear in the sentence.

The second and the third rows in Figure 5 show some more challenging cases whose tagging results benefit from visual features. In example (d), the model pays attention to the big Apple logo, thus tags the '*Apple*' in the sentence as an Organization name. In example (e) and (i), a small

Model	Snap Captions			Twitter		
	Precision	Recall	F1	Precision	Recall	F1
BLSTM-CRF	57.71	**58.65**	58.18	78.88	77.47	78.17
BLSTM-CRF + Global Image Vector	61.49	57.84	59.61	79.75	77.32	78.51
BLSTM-CRF + Visual attention	65.53	57.03	60.98	80.81	77.36	79.05
BLSTM-CRF + Visual attention + Gate	**66.67**	57.84	**61.94**	**81.62**	**79.90**	**80.75**

Table 3: Results of our models on noisy social media data.

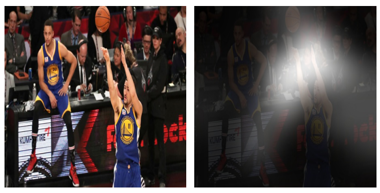
group of people indicates that it is likely to include names of bands (‘*Florence and the Machine*’ and ‘*BTS*’). And a crowd can indicate an organization (‘*Warriorette*’ in example (i)). A jersey shirt on the table indicates a sports team. (‘*Leicester*’ in example (h) can refer to both a city and a soccer club based in it.)

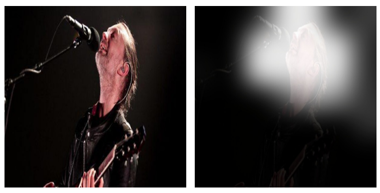
4.4 Error Analysis

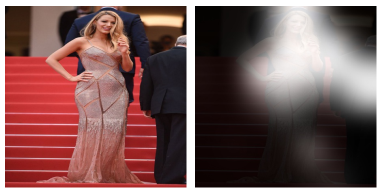
Figure 6 shows some failed examples that are categorized into three types: (1) bad alignments between visual and textual information; (2) blur images; (3) wrong attention made by the model.

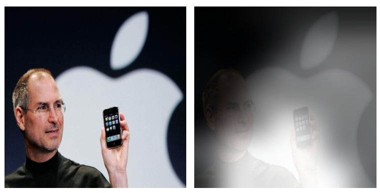
Name tagging greatly benefits from visual features when the sentences are well aligned with the associated image as we show in Section 4.3. But it is not always the case in social media. The example (a) in Figure 6 shows a failed example resulted from poor alignment between sentences and images. In this image, there are two bins standing in front of a wall, but the sentence talks about basketball players. The unrelated visual information makes the model tag ‘*Cleveland*’ as a Location, however it refers to the basketball team ‘*Cleveland Cavaliers*’.

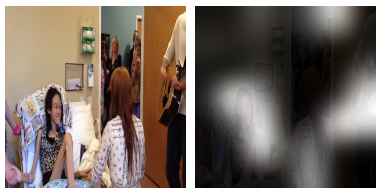
The image in example (b) is blur, so the extracted visual information extracted actually introduces noise instead of additional information. The

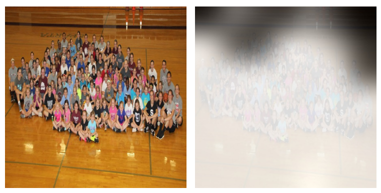

(a). [PER Klay Thompson] **[ORG Warriors]** overwhelm [ORG TrailBlazers] 110-99 go up 2-0 in series

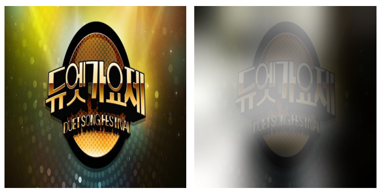

(b). **[PER Radiohead]** offers old and new at first concert in four years...

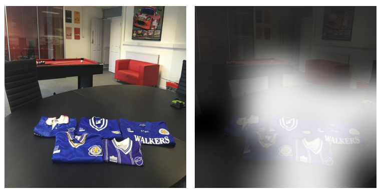

(c). [MISC Cannes] just became the **[PER Blake Lively]** show #Cannes2016 @**[PER blakelively]**


(d). #iPhoneAt10: How [PER Steve Jobs] and **[ORG Apple]** changed modern society


(e). **[PER Florence and the Machine]** surprises ill teen with private concert


(f). **[ORG Warriorette]** Basketball Campers ready for Day 2


(g). Is defending champ [PER Sandeul] able to win for the third time on '**[MISC Duet Song Festival]**'?


(h). Shirts at the ready for our hometown game today #**[ORG Leicester]** #pgautomotive #[ORG premierleague]

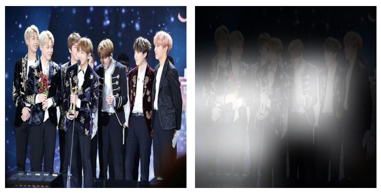
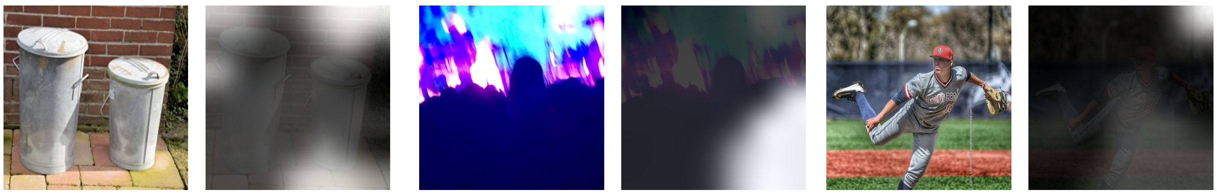

(i). ARMY put up a huge ad in [LOC Times Square] for **[PER BTS]** '4th anniversary!

Figure 5: Examples of visual attentions and NER outputs.



(a). Nice image of [PER Kevin Love] and [PER Kyle Korver] during 1st half #NBAFinals #Cavsin9 #[LOC Cleveland]

(b). Very drunk in a #magnum concert

(c). Looking forward to editing some SBU baseball shots from Saturday.

Figure 6: Examples of Failed Visual Attention.

image in example (c) is about a baseball pitcher, but our model pays attention to the top right corner of the image. The visual context feature computed by our model is not related to the sentence, and results in missed tagging of '*SBU*', which is an organization name.

5 Related Work

In this section, we summarize relevant background on previous work on name tagging and visual attention.

Name Tagging. In recent years, (Chiu and Nichols, 2015; Lample et al., 2016; Ma and Hovy, 2016) proposed several neural network architectures for named tagging that outperform traditional explicit features based methods (Chieu and Ng, 2002; Florian et al., 2003; Ando and Zhang, 2005; Ratinov and Roth, 2009; Lin and Wu, 2009; Passos et al., 2014; Luo et al., 2015). They all use Bidirectional LSTM (BLSTM) to extract features from a sequence of words. For character-level representations, (Lample et al., 2016) proposed to use another BLSTM to capture prefix and suffix information of words, and (Chiu and Nichols, 2015; Ma and Hovy, 2016) used CNN to extract position-independent character features. On top of BLSTM, (Chiu and Nichols, 2015) used a softmax layer to predict the label for each word, and (Lample et al., 2016; Ma and Hovy, 2016) used a CRF layer for joint prediction. Compared with traditional approaches, neural networks based approaches do not require hand-crafted features and achieved state-of-the-art performance on name tagging (Ma and Hovy, 2016). However, these methods were mainly developed for newswire and paid little attention to social media. For name tagging in social media, (Ritter et al., 2011) leveraged a large amount of unlabeled data and many dictionaries into a pipeline model. (Limsopatham and Collier, 2016) adapted the BLSTM-CRF model with additional word shape information, and (Aguilar et al., 2017) utilized an effective multi-task approach. Among these methods, our model is most similar to (Lample et al., 2016), but we designed a new visual attention component and a modulation control gate.

Visual Attention. Since the attention mechanism was proposed by (Bahdanau et al., 2014), it has been widely adopted to language and vision related tasks, such as Image Captioning and Visual Question Answering (VQA), by retrieving the visual features most related to text context (Zhu et al., 2016; Anderson et al., 2017; Xu and Saenko, 2016; Chen et al., 2015). (Xu et al., 2015) proposed to predict a word based on the visual patch that is most related to the last predicted word for image captioning. (Yang et al., 2016b; Lu et al., 2016) applied attention mechanism for VQA, to find the regions in images that are most related to the questions. (Yu et al., 2016) applied the visual attention mechanism on video captioning. Our attention implementation approach in this work is similar to those used for VQA. The model finds the regions in images that are most related to the accompanying sentences, and then feed the visual features into an BLSTM-CRF sequence labeling model. The differences are: (1) we add visual context feature at each step of sequence labeling; and (2) we propose to use a gate to control the combination of the visual information and textual information based on their relatedness. 2

6 Conclusions and Future Work

We propose a gated Visual Attention for name tagging in multimodal social media. We construct two multimodal datasets from Twitter and Snapchat. Experiments show an absolute 3%-4% F-score gain. We hope this work will encourage more research on multimodal social media in the future and we plan on making our benchmark available upon request.

Name Tagging for more fine-grained types (e.g.

soccer team, basketball team, politician, artist) can benefit more from visual features. For example, an image including a pitcher indicates that the '*Giants*' in context should refer to the baseball team '*San Francisco Giants*'. We plan to expand our model to tasks such as fine-grained Name Tagging or Entity Liking in the future.

Acknowledgments

This work was partially supported by the U.S. DARPA AIDA Program No. FA8750-18-2-0014 and U.S. ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2017. Bottom-up and top-down attention for image captioning and vqa. *arXiv preprint arXiv:1707.07998*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*.

Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proceedings of the 2001 Advances in Neural Information Processing Systems*.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational Linguistics*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association of Computational Linguistics*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE international conference on acoustics, speech and signal processing*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

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

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*.

Nut Limsopatham and Nigel Henry Collier. 2016. Bidirectional lstm for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of the 2014 European Conference on Computer Vision*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Proceedings of the 2016 Advances In Neural Information Processing Systems*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*.

Erik F Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 2017 Advances in Neural Information Processing Systems*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *Proceedings of the 2016 European Conference on Computer Vision*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 2015 International Conference on Machine Learning*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2016a. Words or characters? fine-grained gating for reading comprehension. In *Proceedings of the 2016 International Conference on Learning Representations*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016b. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

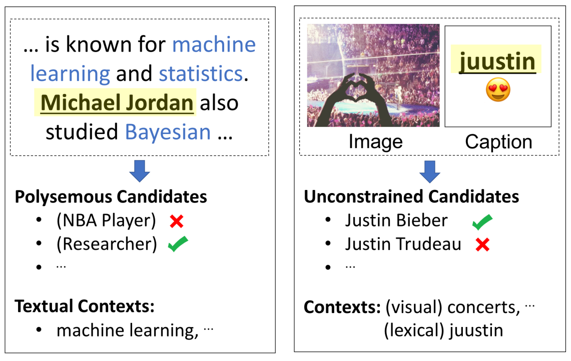
Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zeroshot Multimodal Named Entity Disambiguation for Noisy Social Media Posts

Seungwhan Moon[1,2], Leonardo Neves[2], Vitor Carvalho[3]
[1] Language Technologies Institute, Carnegie Mellon University
[2] Snap Research
[3] Intuit
`seungwhm@cs.cmu.edu, lneves@snap.com, vitor_carvalho@intuit.com`

Abstract

We introduce the new Multimodal Named Entity Disambiguation (MNED) task for multimodal social media posts such as Snapchat or Instagram captions, which are composed of short captions with accompanying images. Social media posts bring significant challenges for disambiguation tasks because 1) ambiguity not only comes from polysemous entities, but also from inconsistent or incomplete notations, 2) very limited context is provided with surrounding words, and 3) there are many emerging entities often unseen during training. To this end, we build a new dataset called *SnapCaptionsKB*, a collection of Snapchat image captions submitted to public and crowd-sourced stories, with named entity mentions fully annotated and linked to entities in an external knowledge base. We then build a deep zeroshot multimodal network for MNED that 1) extracts contexts from both text and image, and 2) predicts correct entity in the knowledge graph embeddings space, allowing for zeroshot disambiguation of entities unseen in training set as well. The proposed model significantly outperforms the state-of-the-art text-only NED models, showing efficacy and potentials of the MNED task.



(a) Traditional NED (b) Multimodal NED

Figure 1: Examples of (a) a traditional NED task, focused on disambiguating polysemous entities based on surrounding textual contexts, and (b) the proposed Multimodal NED task for short media posts, which leverages both visual and textual contexts to disambiguate an entity. Note that mentions are often lexically inconsistent or incomplete, and thus a fixed candidates generation method (based on exact mention-entity statistics) is not viable.

1 Introduction

Online communications are increasingly becoming fast-paced and frequent, and hidden in these abundant user-generated social media posts are insights for understanding users and their preferences. However, these social media posts often come in unstructured text or images, making massive-scale opinion mining extremely challenging. Named entity disambiguation (NED), the task of linking ambiguous entities from free-form text *mention* to specific *entities* in a pre-defined knowledge base (KB), is thus a critical step for extracting structured information which leads to its application for recommendations, advertisement, personalized assistance, etc.

While many previous approaches on NED been successful for well-formed text in disambiguating polysemous entities via context resolution, several additional challenges remain for disambiguating entities from extremely short and coarse text found in social media posts (*e.g.* "*juustin* 😍" as opposed to "*I love Justin Bieber/Justin Trudeau/etc.*"). In many of these cases it is simply impossible to disambiguate entities from text alone, due to enormous number of surface forms arising from incomplete and inconsistent notations. In addition, social media posts often include mentions of newly emerging entities unseen in training sets, making traditional context-based entity linking often not viable.

However, as popular social media platforms are increasingly incorporating a mix of text and images (*e.g.* Snapchat, Instargram, Pinterest, etc.), we can advance the disambiguation task to incorporate additional visual context for understanding posts. For example, the mention of 'juuustin' is completely ambiguous in its textual form, but an accompanying snap image of a concert scene may help disambiguate or re-rank among several lexical candidates (*e.g.* Justin Bieber (a pop singer) versus Justin Trudeau (a politician) in Figure 1).

To this end, we introduce a new task called Multimodal Named Entity Disambiguation (MNED) that handles unique challenges for social media posts composed of extremely short text and images, aimed at disambiguationg entities by leveraging both textual and visual contexts.

We then propose a novel zeroshot MNED model, which obtains visual context vectors from images with a CNN (LeCun et al., 1989), and combines with textual context extracted from a bidirectional LSTM (Dyer et al., 2015) (Section 2.2). In addition, we obtain embeddings representation of 1M entities from a knowledge graph, and train the MNED network to predict label embeddings of entities in the same space as corresponding knowledge graph embeddings (Section 2.4). This approach effectively allows for zeroshot prediction of unseen entities, which is critical for scarce-label scenario due to extensive human annotation efforts required. Lastly, we develop a lexical embeddings model that determines lexical similarity between a mention and potential entities, to aid in prediction of a correct entity (Section 2.3). Section 2.5 details the model combining the components above.

Note that our method takes different perspectives from the previous work on NED (He et al., 2013; Yamada et al., 2016; Eshel et al., 2017) in the following important ways. First, while most of the previous methods generate fixed "candidates" for disambiguation given a mention from mention-entity pair statistics (thus disambiguation is limited for entities with exact surface form matches), we do not fixate candidate generation, due to intractable variety of surface forms for each named entity and unforeseen mentions of emerging entities. Instead, we have a lexical model incorporated into the discriminative score function that serves as soft normalization of various surface forms. Second, we extract auxiliary visual contexts for detected entities from user-generated images accompanied with textual posts, which is crucial because captions in our dataset are substantially shorter than text documents in most other NED datasets. To the best of our knowledge, our work is the first in using visual contexts for the named entity disambiguation task. See Section 4 for the detailed literature review.

Our contributions are as follows: for the new MNED task we introduce, we propose a deep zeroshot multimodal network with (1) a CNN-LSTM hybrid module that extracts contexts from both image and text, (2) a zeroshot learning layer which via embeddings projection allows for entity linking with 1M knowledge graph entities even for entities unseen from captions in training set, and (3) a lexical language model called *Deep Levenshtein* to compute lexical similarities between mentions and entities, relaxing the need for fixed candidates generation. We show that the proposed approaches successfully disambiguate incomplete mentions as well as polysemous entities, outperforming the state-of-the-art models on our newly crawled *SnapCaptionsKB* dataset, composed of 12K image-caption pairs with named entities annotated and linked with an external KB.

2 Proposed Methods

Figure 2 illustrates the proposed model, which maps each multimodal social media post data to one of the corresopnding entities in the KB. Given a multimodal input that contains a mention of an ambiguous entity, we first extract textual and visual features contexts with RCNNs and Bi-LSTMs, respectively (Section 2.2). We also obtain lexical character-level representation of a mention to compare with lexical representation of KB entities, using a proposed model called *Deep Levenshtein* (Section 2.3). We then get high-dimensional label embeddings of KB entities constructed from a knowledge graph, where similar entities are mapped as neighbors in the same space (Section 2.4). Finally, we aggregate all the contextual information extracted from surrounding text, image, and lexical notation of a mention, and predict the best matching KB entity based on knowledge graph label representation and lexical notation of KB entity candidates (Section 2.5).

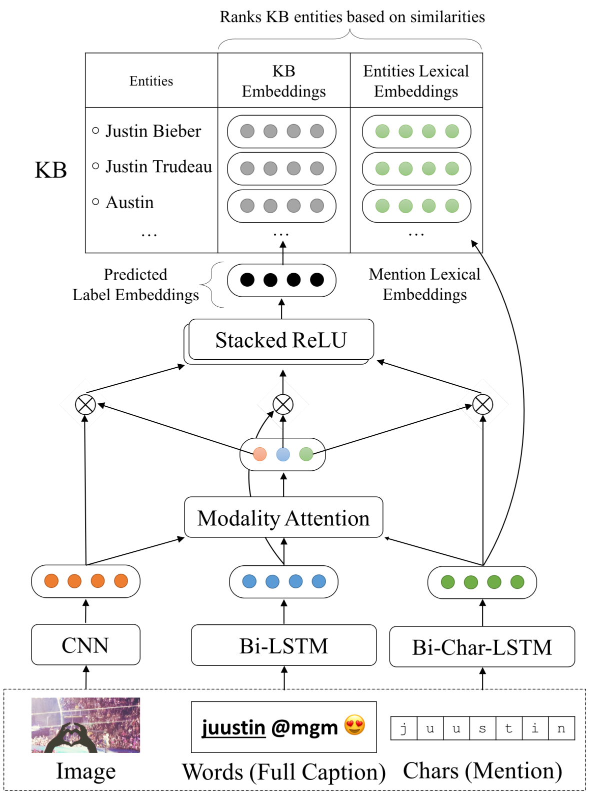


Figure 2: The main architecture of our Multimodal NED network. We extract contextual information from an image, surrounding words, and lexical embeddings of a mention. The modality attention module determines weights for modalities, the weighted projections of which produce label embeddings in the same space as knowledge-base (KB) entity embeddings. We predict a final candidate by ranking based on similarities with KB entity knowledge graph embeddings as well as with lexical embeddings.

2.1 Notations

Let $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ a set of N input social media posts samples for disambiguation, with corresponding ground truth named entities $\mathbf{Y} = \{\mathbf{y}^{(i)}\}_{i=1}^{N}$ for $\mathbf{y} \in \mathbf{Y}_{\text{KB}}$, where $\mathbf{Y}_{\text{KB}}$ is a set of entities in KB. Each input sample is composed of three modalities: $\mathbf{x} = \{\mathbf{x}_w; \mathbf{x}_v; \mathbf{x}_c\}$, where $\mathbf{x}_w = \{\mathbf{x}_{w,t}\}_{t=1}^{L_w}$ is a sequence of words with length L_w surrounding a mention in a post, $\mathbf{x}_v$ is an image associated with a post (Section 2.2), and $\mathbf{x}_c = \{\mathbf{x}_{c,t}\}_{t=1}^{L_c}$ is a sequence of characters comprising a mention (Section 2.3), respectively. We denote high-dimensinal feature extractor functions for each modality as: $\mathbf{w}(\mathbf{x}_w)$, $\mathbf{c}(\mathbf{x}_c)$, $\mathbf{v}(\mathbf{x}_v)$. We represent each output label in two modalities: $\mathbf{y} = \{\mathbf{y}_{KB}; \mathbf{y}_c\}$, where $\mathbf{y}_{\text{KB}}$ is a knowledge base label embeddings representation (Section 2.4), and and $\mathbf{y}_c$ is a character embeddings representation of KB entities (Section 2.3: Deep Levenshtein).

We formulate our zeroshot multimodal NED task as follows:

$$\mathbf{y} = \underset{\mathbf{y}' \in \mathbf{Y}_{\text{KB}}}{\text{argmax}} \ \text{sim}\big(\mathbf{f}_{\mathbf{x}\to\mathbf{y}}(\mathbf{x}), \mathbf{y}'\big)$$

where $\mathbf{f}_{\mathbf{x}\to\mathbf{y}}$ is a function with learnable parameters that project multimodal input samples ($\mathbf{x}$) into the same space as label representations ($\mathbf{y}$), and $\text{sim}(\cdot)$ produces a similarity score between prediction and ground truth KB entities.

2.2 Textual and Visual Contexts Features

Textual features: we represent textual context of surrounding words of a mention with a Bi-LSTM language model (Dyer et al., 2015) with distributed word semantics embeddings. We use the following implementation for the LSTM.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1}) \\
\mathbf{c}_t &= (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} \\
&\quad + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_{w,t} + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_{w,t} + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \\
\mathbf{w}(\mathbf{x}_w) &= [\overrightarrow{\mathbf{h}_{L_w}}; \overleftarrow{\mathbf{h}_{L_w}}] \qquad (1)
\end{aligned}$$

where $\mathbf{h}_t$ is an LSTM hidden layer output at decoding step t, and $\mathbf{w}(\mathbf{x}_w)$ is an output textual representation of bi-directional LSTM concatenating left and right context at the last decoding step $t = L_w$. Biase terms for gates are omitted for simplicity of formulation.

For the Bi-LSTM sentence encoder, we use pretrained word embeddings obtained from an unsupervised language model aimed at learning co-occurrence statistics of words from a large external corpus. Word embeddings are thus represented as distributional semantics of words. In our experiments, we use pre-trained embeddings from Stanford GloVE model (Pennington et al., 2014).

Visaul features: we take the final activation of a modified version of the recurrent convolutional network model called Inception (GoogLeNet) (Szegedy et al., 2015) trained on the ImageNet dataset (Russakovsky et al., 2015) to classify multiple objects in the scene. The final layer representation ($\mathbf{v}(\mathbf{x}_v)$) thus encodes discriminative information describing what objects are shown in an image, providing cues for disambiguation.

2.3 Lexical Embeddings: Deep Levenshtein

While traditional NED tasks assume perfect lexical match between mentions and their corresponding entities, in our task it is important to account for various surface forms of mentions (nicknames, mis-spellings, inconsistent notations, etc.) corresponding to each entity. Towards this goal, we train a separate deep neural network to compute approximate Levenshtein distance which we call Deep Levenshtein (Figure 3), composed of a shared bi-directional character LSTM, shared character embedding matrix, fully connected layers, and a dot product merge operation layer. The optimization is as follows:

$$\min_{\mathbf{c}} \left\| \frac{1}{2}\left(\frac{\mathbf{c}(\mathbf{x}_c)\cdot\mathbf{c}(\mathbf{x'}_c)^\top}{\|\mathbf{c}(\mathbf{x}_c)\|\|\mathbf{c}(\mathbf{x'}_c)\|} + 1 \right) - \mathrm{sim}(\mathbf{x}_c, \mathbf{x'}_c) \right\|^2 \quad (2)$$

$$\text{where } \mathbf{c}(\mathbf{x}_c) = [\overrightarrow{\mathbf{h}_{c,L_c}}; \overleftarrow{\mathbf{h}_{c,L_c}}]$$

where $\mathbf{c}(\cdot)$ is a bi-directional LSTM output vector for a character sequence defined similar as in Eq.1, $\mathrm{sim}(\cdot)$ is an output of the Deep Levenshtein network, producing a normalized similarity score with a range [0,1] based on Levenshtein edit distance, and $(\mathbf{x}_c, \mathbf{x'}_c)$ is any pair of two strings. We generate millions of these pairs as training data by artificially corrupting seed strings by varying degrees (addition, deletion, replacement).

Once trained, it can produce a purely lexical embedding of a string without semantic allusion (via $\mathbf{c}(\cdot)$), and predict lexical similarity between two strings based on their distance in the embedding space. On an intuitive level, this component effectively bypasses normalization steps, and instead incorporates lexical similarities between input mentions and output KB entities into the overall optimization of the disambiguation network.

We use by-product $\mathbf{c}(\cdot)$ network to extract lexical embedings of mentions and KB entities, and freeze $\mathbf{c}$ in training of the disambiguation network. We observe that this approach significantly outperforms alternative ways to obtain character embeddings (*e.g.* having a character Bi-LSTM as a part of the disambiguation network training, which unnecessarily learns semantic allusions that are prone to errors when notations are inconsistent.)

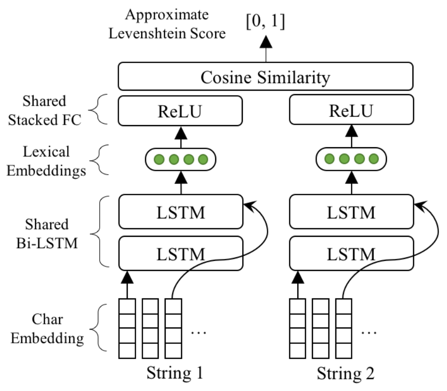


Figure 3: Deep Levenshtein, which predicts approximate Levenshtein scores between two strings. As a byproduct of this model, the shared Bi-LSTM can produce lexical embeddings purely based on lexical property of character sequences.

2.4 Label Embeddings from Knowledge Graph

Due to the overwhelming variety of (newly trending) entities mentioned over social media posts, at test phases we frequently encounter new named entities that are unseen in the training data. In order to address this issue, we propose a zeroshot learning approach (Frome et al., 2013) by inducing embeddings obtained from knowledge graphs on KB entities. Knowledge graph label embeddings are learned from known relations among entities within a graph (*e.g.* 'IS-A', 'LOCATED-AT', etc.), the resulting embeddings of which can group similar entities closer in the same space (*e.g.* 'pop stars' are in a small cluster, 'people' and 'organizations' clusters are far apart, etc.) (Bordes et al., 2013; Wang et al., 2014; Nickel et al., 2016). Once high-level mapping from contextual information to label embeddings is learned, the knowledge-graph based zeroshot approach can improve the entity linking performance given ambiguous entities unseen in training data. In brief formulation, the model for obtaining embeddings from a knowledge graph (composed of subject-relation-object (s, r, o) triplets) is as follows:

$$P(\mathbb{I}_r(s,o) = 1|\mathbf{e}, \mathbf{e}_r, \theta) = \mathrm{score}_\theta\big(\mathbf{e}(s), \mathbf{e}_r(r), \mathbf{e}(o)\big) \quad (3)$$

where $\mathbb{I}_r$ is an indicator function of a known relation r for two entities $(s$,$o)$ (1: valid relation, 0: unknown relation), $\mathbf{e}$ is a function that extracts embeddings for entities, $\mathbf{e}_r$ extracts embeddings for relations, and $\mathrm{score}_\theta(\cdot)$ is a deep neural network that produces a likelihood of a valid triplet.

In our experiments, we use the 1M subset of the Freebase knowledge graph (Bast et al., 2014) to obtain label embeddings with the Holographic KB implementation by (Nickel et al., 2016).

2.5 Deep Zeroshot MNED Network (DZMNED)

Using the contextual information extracted from surrounding text and an accompanying image (Section 2.2) and lexical embeddings of a mention (Section 2.3), we build a Deep Zeroshot MNED network (DZMNED) which predicts a corresponding KB entity based on its knowledge graph embeddings (Section 2.4) and lexical similarity (Section 2.3) with the following objective:

$$\min_{\mathbf{W}} \mathcal{L}_{\text{KB}}(\mathbf{x}, \mathbf{y}_{\text{KB}}; \mathbf{W_w}, \mathbf{W_v}, \mathbf{W_f}) + \mathcal{L}_c(\mathbf{x}_c, \mathbf{y}_c; \mathbf{W_c})$$

where

$$\mathcal{L}_{\text{KB}}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\sum_{\tilde{\mathbf{y}} \neq \mathbf{y}_{\text{KB}}^{(i)}} \max[0, \tilde{\mathbf{y}} \cdot \mathbf{y}_{\text{KB}}^{(i)} - \mathbf{f}(\overline{\mathbf{x}}^{(i)}) \cdot (\mathbf{y}_{\text{KB}}^{(i)} - \tilde{\mathbf{y}})^\top]$$

$$\mathcal{L}_c(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\sum_{\tilde{\mathbf{y}} \neq \mathbf{y}_c^{(i)}} \max[0, \tilde{\mathbf{y}} \cdot \mathbf{y}_c^{(i)} - \mathbf{c}(\mathbf{x}_c^{(i)}) \cdot (\mathbf{y}_c^{(i)} - \tilde{\mathbf{y}})^\top]$$

$\mathcal{R}(\mathbf{W})$: regularization

where $\mathcal{L}_{\text{KB}}(\cdot)$ is the supervised hinge rank loss for knowledge graph embeddings prediction, $\mathcal{L}_c(\cdot)$ is the loss for lexical mapping between mentions and KB entities, $\overline{\mathbf{x}}$ is a weighted average of three modalities $\mathbf{x} = \{\mathbf{x}_w; \mathbf{x}_v; \mathbf{x}_c\}$ via the modality attention module. $\mathbf{f}(\cdot)$ is a transformation function with stacked layers that projects weighted input to the KB embeddings space, $\tilde{\mathbf{y}}$ refers to the embeddings of negative samples randomly sampled from KB entities except the ground truth label of the instance, $\mathbf{W} = \{\mathbf{W_f}, \mathbf{W_c}, \mathbf{W_w}, \mathbf{W_v}\}$ are the learnable parameters for $\mathbf{f}$, $\mathbf{c}$, $\mathbf{w}$, and $\mathbf{v}$ respectively, and $\mathcal{R}(\mathbf{W})$ is a weight decay regularization term.

Similarly to (Moon et al., 2018), we formulate the **modality attention** module for our MNED network as follows, which selectively attenuates or amplifies modalities:

$$[\mathbf{a}_w; \mathbf{a}_c; \mathbf{a}_v] = \sigma\big(\mathbf{W}_m \cdot [\mathbf{x}w; \mathbf{x}c; \mathbf{x}_v] + \mathbf{b}_m\big) \quad (4)$$

$$\alpha_m = \frac{\exp(\mathbf{a}_m)}{\sum_{m' \in \{w,c,v\}} \exp(\mathbf{a}_{m'})} \quad \forall m \in \{w, c, v\}$$

$$\overline{\mathbf{x}} = \sum_{m \in \{w,c,v\}} \alpha_m \mathbf{x}_m \quad (5)$$

where $\alpha = [\alpha_w; \alpha_c; \alpha_v] \in \mathbb{R}^3$ is an attention vector, and $\overline{\mathbf{x}}$ is a final context vector that maximizes information gain.

Intuitively, the model is trained to produce a higher dot product similarity between the projected embeddings with its correct label than with an incorrect negative label in both the knowledge graph label embeddings and the lexical embeddings spaces, where the margin is defined as the similarity between a ground truth sample and a negative sample.

At test time, the following label-producing nearest neighbor (1-NN) classifier is used for the target task (we cache all the label embeddings to avoid repetitive projections):

$$\text{1-NN}(\mathbf{x}) = \underset{(\mathbf{y}_{\text{KB}}, \mathbf{y}_c) \in \mathbf{Y}_{\text{KB}}}{\text{argmax}} \ \mathbf{f}(\overline{\mathbf{x}}) \cdot \mathbf{y}_{\text{KB}}{}^\top + \mathbf{g}(\mathbf{x}_c) \cdot \mathbf{y_c}{}^\top \quad (6)$$

In summary, the model produces (1) projection of input modalities (mention, surrounding text, image) into the knowledge graph embeddings space, and (2) lexical embeddings representation of mention, which then calculates a combined score of contextual (knowledge graph) and string similarities with each entity in $\mathbf{Y}_{\text{KB}}$.

3 Empirical Evaluation

Task: Given a caption and an accompanying image (if available), the goal is to disambiguate and link a target mention in a caption to a corresponding entity from the knowledge base (1M subset of the Freebase knowledge graph (Bast et al., 2014)).

3.1 Datasets

Our **SnapCaptionsKB** dataset is composed of 12K user-generated image and textual caption pairs where named entities in captions and their links to KB entities are manually labeled by expert human annotators. These captions are collected exclusively from snaps submitted to public and crowd-sourced stories (aka *Live Stories* or *Our Stories*). Examples of such stories are “New York Story” or “Thanksgiving Story”, which are aggregated collections of snaps for various public venues, events, etc. Our data do not contain raw images, and we only provide textual captions and obfuscated visual descriptor features extracted from the pre-trained InceptionNet. We split the dataset randomly into train (70%), validation (15%), and test sets (15%). The captions data have average length of 29.5 characters (5.57 words) with vocabulary size 16,553, where 6,803 are considered unknown tokens from Stanford GloVE embeddings (Pennington et al., 2014).

Named entities annotated in the dataset include many of new and emerging entities found in various surface forms. To the best of our knowledge, our *SnapCaptionsKB* is the only dataset that contains image-caption pairs with human-annotated named entities and their links to KB entities.

3.2 Baselines

We report performance of the following state-of-the-art NED models as baselines, with several candidate generation methods and variations of our proposed approach to examine contributions of each component (W: word, C: char, V: visual).

Candidates generation: Note that our zeroshot approach allows for entity disambiguation without a fixed candidates generation process. In fact, we observe that the conventional method for fixed candidates generation harms the performance for noisy social media posts with many emerging entities. This is because the difficulty of entity linking at test time rises not only from multiple entities (e) linking to a single mention (m), but also from each entity found in multiple surface forms of mentions (often unseen at train time). To show the efficacy of our approach that does not require candidates generation, we compare with the following candidates generation methods:

- $m \rightarrow e$ hash list: This method retrieves KB entity (e) candidates per mention (m) based on exact (m, e) pair occurrence statistics from a training corpora. This is the most predominantly used candidates generation method (He et al., 2013; Yamada et al., 2016; Eshel et al., 2017). Note that this approach is especially vulnerable at test time to noisy mentions or emerging entities with no or a few matching candidate entities from training set.

- k-NN: We also consider using lexical neighbors of mentions from KB entities as candidates. This approach can be seen as soft normalization to relax the issue of having to match a variety of surface forms of a mention to KB entities. We use our Deep Levenshtein (Section 2.3) to compute lexical embeddings of KB entities and mentions, and retrieves Euclidean neighbors (and their polysemous entities) as candidates.

NED models: We choose as baselines the following state-of-the-art NED models for noisy text, as well as several configurations of our proposed approach to examine contributions of each component (W: word, C: char, V: visual).

- sDA-NED (W only) (He et al., 2013): uses a deep neural network with stacked denoising autoencoders (sDA) to encode bag-of-words representation of textual contexts and to directly compare mentions and entities.

- ARNN (W only) (Eshel et al., 2017): uses an Attention RNN model that computes similarity between word and entity embeddings to disambiguate among fixed candidates.

- Deep Zeroshot (W only): uses the deep zeroshot architecture similar to Figure 2, but uses word contexts (caption) only.

- (**proposed**) DZMNED + Deep Levenshtein + InceptionNet with modality attention (W+C+V): is the proposed approach as described in Figure 2.

- (**proposed**) DZMNED + Deep Levenshtein + InceptionNet w/o modality attention (W+C+V): concatenates all the modality vectors instead.

- (**proposed**) DZMNED + Deep Levenshtein (W+C): only uses textual context.

- (**proposed**) DZMNED + Deep Levenshtein w/o modality attention (W+C): does not use the modality attention module, and instead concatenates word and lexical embeddings.

3.3 Results

Parameters: We tune the parameters of each model with the following search space (bold indicate the choice for our final model): character embeddings dimension: {25, 50, **100**, 150, 200, 300}, word embeddings size: {25, 50, **100**, 150, 200, 300}, knowledge graph embeddings size: {**100**, 200, 300}, LSTM hidden states: {50, **100**, 150, 200, 300}, and $\overline{x}$ dimension: {25, 50, **100**, 150, 200, 300}. We optimize the parameters with Adagrad (Duchi et al., 2011) with batch size 10, learning rate 0.01, epsilon 10^{-8}, and decay 0.1.

Main Results: Table 1 shows the Top-1, 3, 5, 10, and 50 candidates retrieval accuracy results on the *Snap Captions* dataset. We see that the proposed approach significantly outperforms the baselines which use fixed candidates generation

Modalities	Model	Candidates Generation	Accuracy (%)				
			Top-1	Top-3	Top-5	Top-10	Top-50
W	ARNN (Eshel et al., 2017)	$m{\rightarrow}e$ list	51.2	60.4	66.5	66.9	66.9
W	ARNN	5-NN (lexical)	35.2	43.3	45.0	-	-
W	ARNN	10-NN (lexical)	31.9	40.1	44.5	50.7	-
W	sDA-NED (He et al., 2013)	$m{\rightarrow}e$ list	48.7	57.3	66.3	66.9	66.9
W	Zeroshot	N/A	43.6	63.8	67.1	70.5	77.2
W + C	DZMNED	N/A	67.0	72.7	74.8	76.8	85.0
W + C	DZMNED + Modality Attention	N/A	67.8	73.5	74.8	76.2	84.6
W + C + V	DZMNED	N/A	67.2	74.6	77.7	80.5	**88.1**
W + C + V	DZMNED + Modality Attention	N/A	**68.1**	**75.5**	**78.2**	**80.9**	87.9

Table 1: NED performance on the *SnapCaptionsKB* dataset at Top-1, 3, 5, 10, 50 accuracies. The classification is over 1M entities. Candidates generation methods: N/A, or over a fixed number of candidates generated with methods: $m{\rightarrow}e$ hash list and kNN (lexical neighbors).

KB Embeddings	Top-1	Top-5	Top-10
Trained with 1M entities	**68.1**	**78.2**	**80.9**
Trained with 10K entities	60.3	72.5	75.9
Random embeddings	41.4	45.8	48.0

Table 2: MNED performance (Top-1, 5, 10 accuracies) on SnapCaptionsKB with varying qualities of KB embeddings. Model: DZMNED (W+C+V)

method. Note that $m \rightarrow e$ hash list-based methods, which retrieve as candidates the KB entities that appear in the training set of captions only, has upper performance limit at 66.9%, showing the limitance of fixed candidates generation method for unseen entities in social media posts. k-NN methods which retrieve lexical neighbors of mention (in an attempt to perform soft normalization on mentions) also do not perform well. Our proposed zeroshot approaches, however, do not fixate candidate generation, and instead compares combined contextual and lexical similarities among all 1M KB entities, achieving higher upper performance limit (Top-50 retrieval accuracy reaches 88.1%). This result indicates that the proposed zeroshot model is capable of predicting for unseen entities as well. The lexical sub-model can also be interpreted as functioning as soft neural mapping of mention to potential candidates, rather than heuristic matching to fixed candidates.

In addition, when visual context is available (W+C+V), the performance generally improves over the textual models (W+C), showing that visual information can provide additional contexts for disambiguation. The modality attention module also adds performance gain by re-weighting the modalities based on their informativeness.

Error Analysis: Table 3 shows example cases where incorporation of visual contexts affects disambiguation of mentions in textual captions. For example, polysemous entities such as 'Jordan' in the caption "*Taking the new Jordan for a walk*" or 'CID' as in "*LETS GO CID*" are hard to disambiguate due to the limited textual contexts provided, while visual information (*e.g.* visual tags 'footwear' for Jordan, 'DJ' for CID) provides similarities to each mention's distributional semantics from other training examples. Mentions unseen at train time ('STEPHHHH', 'murica') often resort to lexical neighbors by (W+C), whereas visual contexts can help disambiguate better. A few cases where visual contexts are not helpful include visual tags that are not related to mentions, or do not complement already ambiguous contexts.

Sensitivity to KB Embeddings Quality: The proposed approach relies its prediction on entity matching in the KB embeddings space, and hence the quality of KB embeddings is crucial for successful disambiguation. To characterize this aspect, we provide Table 2 which shows MNED performance with varying quality of embeddings as follows: KB embeddings learned from 1M knowledge graph entities (same as in the main experiments), from 10K subset of entities (less triplets to train with in Eq.3, hence lower quality), and random embeddings (poorest) - while all the other parameters are kept the same. It can be seen that the performance notably drops with lower quality of KB embeddings. When KB embeddings are replaced by random embeddings, the network effectively prevents the contextual zeroshot matching to KB entities and relies only on lexical similarities, achieving the poorest performance.

	Caption (target)	Visual Tags	GT	Top-1 Prediction	
				(W+C+V)	(W+C)
+	*"YA BOI STEPHHHH"*	sports equip, ball, parade, ...	Stephen Curry	(=GT)	Stephenville
	"Taking the new Jordan for a walk"	footwear, shoe, sock, ...	Air Jordan	(=GT)	Michael Jordan
	"out for murica's bday 😎"	parade, flag, people, ...	U.S.A.	(=GT)	Murcia (Spain)
	"Come on now, Dre"	club, DJ, night, ...	Dr. Dre	(=GT)	Dre Kirkpatrick
	"LETS GO CID"	drum, DJ, drummer, ...	CID (DJ)	(=GT)	CID (ORG)
-	*"kick back hmu for addy."*	weather, fog, tile, ...	Adderall	GoDaddy	(=GT)
	"@Sox to see 3️⃣4️⃣ get retired!⚾🍺"	sunglasses, stadium, ...	Red Sox	White Sox	White Sox

Table 3: Error analysis: **when do images help NED**? Ground-truth (GT) and predictions of our model with vision input (W+C+V) and the one without (W+C) for the underlined mention are shown. For interpretability, visual tags (label output of InceptionNet) are presented instead of actual feature vectors.

4 Related Work

NED task: Most of the previous NED models leverage local textual information (He et al., 2013; Eshel et al., 2017) and/or document-wise global contexts (Hoffart et al., 2011; Chisholm and Hachey, 2015; Pershina et al., 2015; Globerson et al., 2016), in addition to other auxiliary contexts or priors for disambiguating a mention. Note that most of the NED datasets (*e.g.* TAC KBP (Ji et al., 2010), ACE (Bentivogli et al., 2010), CoNLL-YAGO (Hoffart et al., 2011), etc.) are extracted from standardized documents with web links such as Wikipedia (with relatively ample textual contexts), and that named entitiy disambiguation specifically for short and noisy social media posts are rarely discussed. Note also that most of the previous literature assume the availability of "candidates" or web links for disambiguation via mention-entity pair counts from training set, which is vulnerable to inconsistent surface forms of entities predominant in social media posts.

Our model improves upon the state-of-the-art NED models in three very critical ways: (1) incorporation of visual contexts, (2) addition of the zeroshot learning layer, which allows for disambiguation of unseen entities during training, and (3) addition of the lexical model that computes lexical similarity entities to correctly recognize inconsistent surface forms of entities.

Multimodal learning studies learning of a joint model that leverages contextual information from multiple modalities in parallel. Some of the relevant multimodal learning task to our MNED system include the multimodal named entity recognition task (Moon et al., 2018), which leverages both text and image to classify each token in a sentence to named entity or not. In their work, they employ an entity LSTM that takes as input each modality, and a softmax layer that outputs an entity label at each decoding step. Contrast to their work, our MNED addresses unique challenges characterized by zeroshot ranking of 1M knowledge-base entities (vs. categorical entity types prediction), incorporation of an external knowledge graph, lexical embeddings, etc. Another is the multimodal machine translation task (Elliott et al., 2015; Specia et al., 2016), which takes as input text in source language as well as an accompanying image to output a translated text in target language. These models usually employ a sequence-to-sequence architecture (*e.g.* target language decoder takes as input both encoded source language and images) often with traditional attention modules widely used in other image captioning systems (Xu et al., 2015; Sukhbaatar et al., 2015). To the best of our knowledge, our approach is the first multimodal learning work at incorporating visual contexts for the NED task.

5 Conclusions

We introduce a new task called Multimodal Named Entity Disambiguation (MNED), which is applied on short user-generated social media posts that are composed of text and accompanying images. Our proposed MNED model improves upon the state-of-the-art models by 1) extracting visual contexts complementary to textual contexts, 2) by leveraging lexical embeddings into entity matching which accounts for various surface forms of entities, removing the need for fixed candidates generation process, and 3) by performing entity matching in the distributed knowledge graph embeddings space, allowing for matching of unseen mentions and entities by context resolutions.

References

Hannah Bast, Florian Baurle, Bjorn Buchhold, and Elmar Haussmann. 2014. Easy access to the freebase dataset. In *WWW*.

Luisa Bentivogli, Pamela Forner, Claudio Giuliano, Alessandro Marchetti, Emanuele Pianta, and Kateryna Tymoshenko. 2010. Extending english ace 2005 corpus annotation with ground-truth links to wikipedia. In *Proceedings of the 2nd Workshop on The Peoples Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 19–27.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association of Computational Linguistics*, 3(1):145–156.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *ACL*.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-language image description with neural sequence models. *CoRR, abs/1510.04709*.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuda Yamada, and Omer Levy. 2017. Named entity disambiguation for noisy text. *CoNLL*.

Andrea Frome, Greg Corrado, Jon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 621–631.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*, volume 3, pages 3–13.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*.

Seungwhan Moon, Leonard Neves, and Vitor Carvalho. 2018. Multimodal named entity recognition for short social media posts. *NAACL*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. *AAAI*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV*.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *WMT*, pages 543–553.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*, pages 2440–2448.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going deeper with convolutions. *CVPR*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *CoNLL*.

Semi-supervised User Geolocation via Graph Convolutional Networks

Afshin Rahimi **Trevor Cohn** **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne
arahimi@student.unimelb.edu.au
{t.cohn,tbaldwin}@unimelb.edu.au

Abstract

Social media user geolocation is vital to many applications such as event detection. In this paper, we propose GCN, a multiview geolocation model based on Graph Convolutional Networks, that uses both text and network context. We compare GCN to the state-of-the-art, and to two baselines we propose, and show that our model achieves or is competitive with the state-of-the-art over three benchmark geolocation datasets when sufficient supervision is available. We also evaluate GCN under a minimal supervision scenario, and show it outperforms baselines. We find that highway network gates are essential for controlling the amount of useful neighbourhood expansion in GCN.

1 Introduction

User geolocation, the task of identifying the "home" location of a user, is an integral component of many applications ranging from public health monitoring (Paul and Dredze, 2011; Chon et al., 2015; Yepes et al., 2015) and regional studies of sentiment, to real-time emergency awareness systems (De Longueville et al., 2009; Sakaki et al., 2010), which use social media as an implicit information resource about people.

Social media services such as Twitter rely on IP addresses, WiFi footprints, and GPS data to geolocate users. Third-party service providers don't have easy access to such information, and have to rely on public sources of geolocation information such as the profile location field, which is noisy and difficult to map to a location (Hecht et al., 2011), or geotagged tweets, which are publicly available for only 1% of tweets (Cheng et al., 2010; Morstatter et al., 2013). The scarcity of publicly available location information motivates predictive user geolocation from information such as tweet text and social interaction data.

Most previous work on user geolocation takes the form of either supervised text-based approaches (Wing and Baldridge, 2011; Han et al., 2012) relying on the geographical variation of language use, or graph-based semi-supervised label propagation relying on location homophily in user–user interactions (Davis Jr et al., 2011; Jurgens, 2013).

Both text and network views are critical in geolocating users. Some users post a lot of local content, but their social network is lacking or is not representative of their location; for them, text is the dominant view for geolocation. Other users have many local social interactions, and mostly use social media to read other people's comments, and for interacting with friends. Single-view learning would fail to accurately geolocate these users if the more information-rich view is not present. There has been some work that uses both the text and network views, but it either completely ignores unlabelled data (Li et al., 2012a; Miura et al., 2017), or just uses unlabelled data in the network view (Rahimi et al., 2015b; Do et al., 2017). Given that the 1% of geotagged tweets is often used for supervision, it is crucial for geolocation models to be able to leverage unlabelled data, and to perform well under a minimal supervision scenario.

In this paper, we propose GCN, an end-to-end user geolocation model based on Graph Convolutional Networks (Kipf and Welling, 2017) that jointly learns from text and network information to classify a user timeline into a location. Our contributions are: (1) we evaluate our model under a minimal supervision scenario which is close to real world applications and show that GCN outperforms two strong baselines; (2) given sufficient supervision, we show that GCN is competitive, although the much simpler MLP-TXT+NET outperforms state-of-the-art models; and (3) we show that highway gates play a significant role in controlling the amount of useful neighbourhood smoothing in `GCN`.[1]

2 Model

We propose a transductive multiview geolocation model, `GCN`, using Graph Convolutional Networks ("GCN": Kipf and Welling (2017)). We also introduce two multiview baselines: `MLP-TXT+NET` based on concatenation of text and network, and `DCCA` based on Deep Canonical Correlation Analysis (Andrew et al., 2013).

2.1 Multivew Geolocation

Let $X \in \mathbb{R}^{|U| \times |V|}$ be the text view, consisting of the bag of words for each user in U using vocabulary V, and $A \in \mathbb{1}^{|U| \times |U|}$ be the network view, encoding user–user interactions. We partition $U = U_S \cup U_H$ into a supervised and heldout (unlabelled) set, U_S and U_H, respectively. The goal is to infer the location of unlabelled samples Y_U, given the location of labelled samples Y_S, where each location is encoded as a one-hot classification label, $y_i \in \mathbb{1}^c$ with c being the number of target regions.

2.2 `GCN`

GCN defines a neural network model $f(X, A)$ with each layer:

$$\begin{aligned} \hat{A} &= \tilde{D}^{-\frac{1}{2}}(A + \lambda I)\tilde{D}^{-\frac{1}{2}} \\ H^{(l+1)} &= \sigma\Big(\hat{A}H^{(l)}W^{(l)} + b\Big) , \end{aligned} \quad (1)$$

where $\tilde{D}$ is the degree matrix of $A + \lambda I$; hyperparameter λ controls the weight of a node against its neighbourhood, which is set to 1 in the original model (Kipf and Welling, 2017); $H^0 = X$ and the $d_{\text{in}} \times d_{\text{out}}$ matrix $W^{(l)}$ and $d_{\text{out}} \times 1$ matrix b are trainable layer parameters; and σ is an arbitrary nonlinearity. The first layer takes an average of each sample and its immediate neighbours (labelled and unlabelled) using weights in $\hat{A}$, and performs a linear transformation using W and b followed by a nonlinear activation function (σ). In other words, for user u_i, the output of layer l is computed by:

$$\vec{h}_i^{l+1} = \sigma\Big(\sum\nolimits_{j \in \text{nhood}(i)} \hat{A}_{ij}\vec{h}_j^l W^l + b^l\Big) , \quad (2)$$

[1]Code and data available at `https://github.com/afshinrahimi/geographconv`

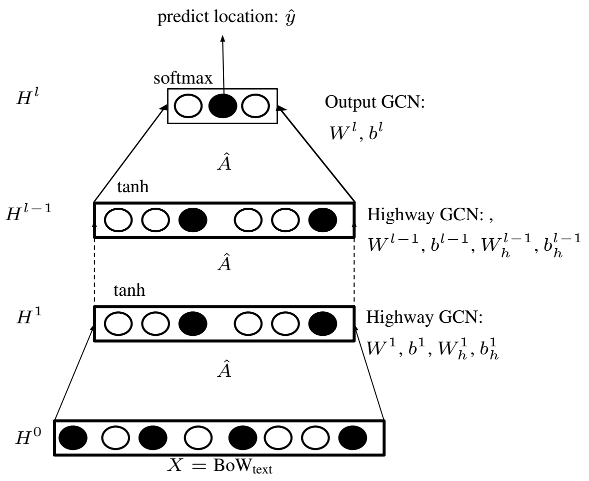


Figure 1: The architecture of `GCN` geolocation model with layer-wise highway gates (W_h^i, b_h^i). `GCN` is applied to a BoW model of user content over the @-mention graph to predict user location.

where W^l and b^l are learnable layer parameters, and $\text{nhood}(i)$ indicates the neighbours of user u_i. Each extra layer in GCN extends the neighbourhood over which a sample is smoothed. For example a GCN with 3 layers smooths each sample with its neighbours up to 3 hops away, which is beneficial if location homophily extends to a neighbourhood of this size.

2.2.1 Highway GCN

Expanding the neighbourhood for label propagation by adding multiple GCN layers can improve geolocation by accessing information from friends that are multiple hops away, but it might also lead to propagation of noisy information to users from an exponentially increasing number of expanded neighbourhood members. To control the required balance of how much neighbourhood information should be passed to a node, we use layer-wise gates similar to highway networks. In highway networks (Srivastava et al., 2015), the output of a layer is summed with its input with gating weights $T(\vec{h}^l)$:

$$\begin{aligned} T(\vec{h}^l) &= \sigma\Big(W_t^l \vec{h}^l + b_t^l\Big) \\ \vec{h}^{l+1} &= \vec{h}^{l+1} \circ T(\vec{h}^l) + \vec{h}^l \circ (1 - T(\vec{h}^l)) , \end{aligned} \quad (3)$$

where $\vec{h}^l$ is the incoming input to layer $l + 1$, (W_t^l, b_t^l) are gating weights and bias variables, $\circ$ is elementwise multiplication, and σ is the Sigmoid function.

2.3 DCCA

Given two views X and $\hat{A}$ (from Equation 1) of data samples, CCA (Hotelling, 1936), and its deep version (DCCA) (Andrew et al., 2013) learn functions $f_1(X)$ and $f_2(\hat{A})$ such that the correlation between the output of the two functions is maximised:

$$\rho = \text{corr}(f_1(X), f_2(\hat{A}))\,. \qquad (4)$$

The resulting representations of $f_1(X)$ and $f_2(\hat{A})$ are the compressed representations of the two views where the uncorrelated noise between them is reduced. The new representations ideally represent user communities for the network view, and the language model of that community for the text view, and their concatenation is a multiview representation of data, which can be used as input for other tasks.

In DCCA, the two views are first projected to a lower dimensionality using a separate multilayer perceptron for each view (the f_1 and f_2 functions of Equation 4), the output of which is used to estimate the CCA cost:

$$\begin{aligned} \text{maximise:} \quad & \text{tr}(W_1^T \Sigma_{12} W_2) \\ \text{subject to:} \quad & W_1^T \Sigma_{11} W_1 = W_2^T \Sigma_{22} W_2 = I \end{aligned} \qquad (5)$$

where Σ_{11} and Σ_{22} are the covariances of the two outputs, and Σ_{12} is the cross-covariance. The weights W_1 and W_2 are the linear projections of the MLP outputs, which are used in estimating the CCA cost. The optimisation problem is solved by SVD, and the error is backpropagated to train the parameters of the two MLPs and the final linear projections. After training, the two networks are used to predict new projections for unseen data. The two projections of unseen data — the outputs of the two networks — are then concatenated to form a multiview sample representation, as shown in Figure 2.

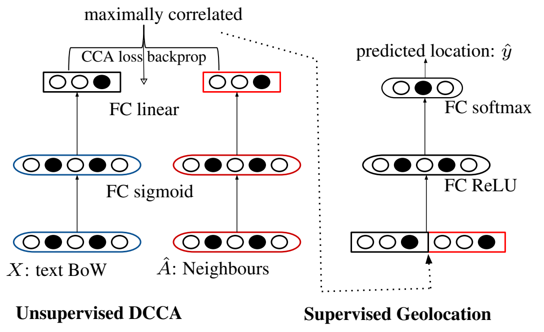


Figure 2: The DCCA model architecture: First the two text and network views X and $\hat{A}$ are fed into two neural networks (left), which are unsupervisedly trained to maximise the correlation of their outputs; next the outputs of the networks are concatenated, and fed as input to another neural network (right), which is trained supervisedly to predict locations.

3 Experiments

3.1 Data

We use three existing Twitter user geolocation datasets: (1) GeoText (Eisenstein et al., 2010), (2) Twitter-US (Roller et al., 2012), and (3) Twitter-World (Han et al., 2012). These datasets have been used widely for training and evaluation of geolocation models. They are all pre-partitioned into training, development and test sets. Each user is represented by the concatenation of their tweets, and labelled with the latitude/longitude of the first collected geotagged tweet in the case of GeoText and Twitter-US, and the centre of the closest city in the case of Twitter-World. GeoText and Twitter-US cover the continental US, and Twitter-World covers the whole world, with 9k, 449k and 1.3m users, respectively. The labels are the discretised geographical coordinates of the training points using a k-d tree following Roller et al. (2012), with the number of labels equal to 129, 256, and 930 for GeoText, Twitter-US, and Twitter-World, respectively.

3.2 Constructing the Views

We build matrix $\hat{A}$ as in Equation 1 using the collapsed @-mention graph between users, where two users are connected ($A_{ij} = 1$) if one mentions the other, or they co-mention another user. The text view is a BoW model of user content with binary term frequency, inverse document frequency, and l_2 normalisation of samples.

3.3 Model Selection

For GCN, we use highway layers to control the amount of neighbourhood information passed to a node. We use 3 layers in GCN with size 300, 600, 900 for GeoText, Twitter-US and Twitter-World respectively. Note that the final softmax layer is also graph convolutional, which sets the radius of the averaging neighbourhood to 4. The

k-d tree bucket size hyperparameter which controls the maximum number of users in each cluster is set to 50, 2400, and 2400 for the respective datasets, based on tuning over the validation set. The architecture of GCN−LP is similar, with the difference that the text view is set to zero. In DCCA, for the unsupervised networks we use a single sigmoid hidden layer with size 1000 and a linear output layer with size 500 for the three datasets. The loss function is CCA loss, which maximises the output correlations. The supervised multilayer perceptron has one hidden layer with size 300, 600, 1000 for GEOTEXT, TWITTER-US, and TWITTER-WORLD, respectively, which we set by tuning over the development sets. We evaluate the models using Median error, Mean error, and Acc@161, accuracy of predicting a user within 161km or 100 miles from the known location.

3.4 Baselines

We also compare DCCA and GCN with two baselines:

GCN−LP is based on GCN, but for input, instead of text-based features , we use one-hot encoding of a user's neighbours, which are then convolved with their k-hop neighbours using the GCN. This approach is similar to label propagation in smoothing the label distribution of a user with that of its neighbours, but uses graph convolutional networks which have extra layer parameters, and also a gating mechanism to control the smoothing neighbourhood radius. Note that for unlabelled samples, the predicted labels are used for input after training accuracy reaches 0.2.

MLP−TXT+NET is a simple transductive supervised model based on a single layer multilayer perceptron where the input to the network is the concatenation of the text view X, the user content's bag-of-words and $\hat{A}$ (Equation 1), which represents the network view as a vector input. For the hidden layer we use a ReLU nonlinearity, and sizes 300, 600, and 600 for GEOTEXT, TWITTER-US, and TWITTER-WORLD, respectively.

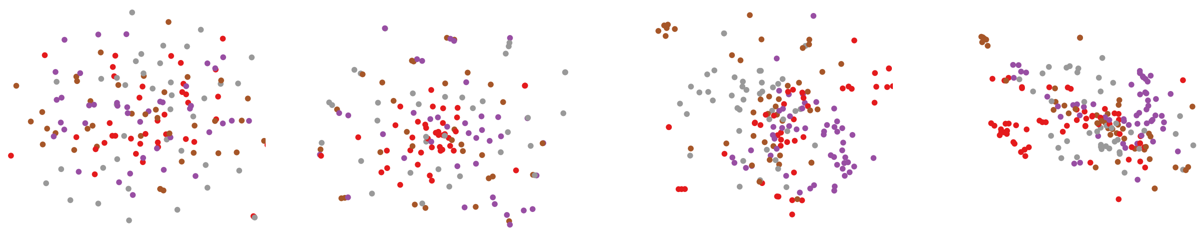
4 Results and Analysis

4.1 Representation

Deep CCA and GCN are able to provide an unsupervised data representation in different ways. Deep CCA takes the two text-based and network-based views, and finds deep non-linear transformations that result in maximum correlation between the two views (Andrew et al., 2013). The representations can be visualised using t-SNE, where we hope that samples with the same label are clustered together. GCN, on the other hand, uses graph convolution. The representations of 50 samples from each of 4 randomly chosen labels of GEOTEXT are shown in Figure 3. As shown, Deep CCA seems to slightly improve the representations from pure concatenation of the two views. GCN, on the other hand, substantially improves the representations. Further application of GCN results in more samples clumping together, which might be desirable when there is strong homophily.

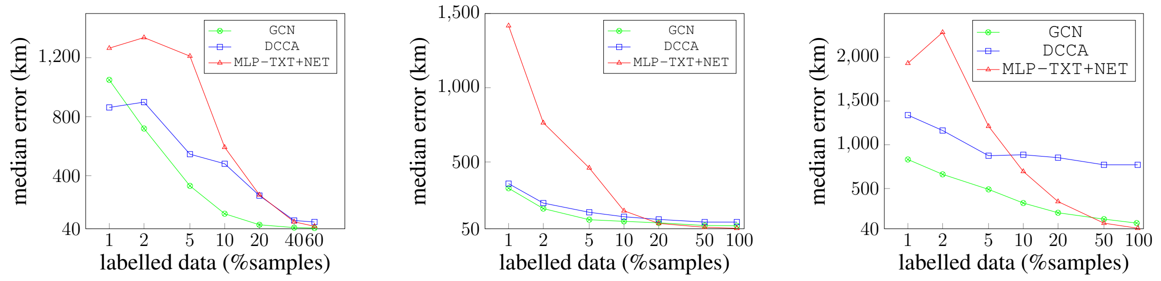
4.2 Labelled Data Size

To achieve good performance in supervised tasks, often large amounts of labelled data are required, which is a big challenge for Twitter geolocation, where only a small fraction of the data is geotagged (about 1%). The scarcity of supervision indicates the importance of semi-supervised learning where unlabelled (e.g. non-geotagged) tweets are used for training. The three models we propose (MLP−TXT+NET, DCCA, and GCN) are all transductive semi-supervised models that use unlabelled data, however, they are different in terms of how much labelled data they require to achieve acceptable performance. Given that in a real-world scenario, only a small fraction of data is geotagged, we conduct an experiment to analyse the effect of labelled samples on the performance of the three geolocation models. We provided the three models with different fractions of samples that are labelled (in terms of % of dataset samples) while using the remainder as unlabelled data, and analysed their Median error performance over the development set of GEOTEXT, TWITTER-US, and TWITTER-WORLD. Note that the text and network view, and the development set, remain fixed for all the experiments. As shown in Figure 4, when the fraction of labelled samples is less than 10% of all the samples, GCN and DCCA outperform MLP−TXT+NET, as a result of having fewer parameters, and therefore, lower supervision requirement to optimise them. When enough training data is available (e.g. more than 20% of all the samples), GCN and MLP−TXT+NET clearly outperform DCCA, possibly as a result of directly modelling the



(a) MLP−TXT+NET (b) DCCA (c) 1 GCN $\hat{A} \cdot X$ (d) 2 GCN $\hat{A} \cdot \hat{A} \cdot X$

Figure 3: Comparing t-SNE visualisations of 50 training samples from each of 4 randomly chosen regions of GEOTEXT using various data representations: (a) concatenation of $\hat{A}$ (Equation 1); (b) concatenation of DCCA transformation of text-based and network-based views X and $\hat{A}$; (c) applying graph convolution $\hat{A} \cdot X$; and (d) applying graph convolution twice $\hat{A} \cdot \hat{A} \cdot X$



(a) GEOTEXT (b) TWITTER-US (c) TWITTER-WORLD

Figure 4: The effect of the amount of labelled data available as a fraction of all samples for GEOTEXT, TWITTER-US, and TWITTER-WORLD on the development performance of GCN, DCCA, and MLP−TXT+NET models in terms of Median error. The dataset sizes are 9k, 440k, and 1.4m for the three datasets, respectively.

interactions between network and text views. When all the training samples of the two larger datasets (95% and 98% for TWITTER-US and TWITTER-WORLD, respectively) are available to the models, MLP−TXT+NET outperforms GCN. Note that the number of parameters increases from DCCA to GCN and to MLP−TXT+NET. In 1% for GEOTEXT, DCCA outperforms GCN as a result of having fewer parameters and just a few labelled samples, insufficient to train the parameters of GCN.

4.3 Highway Gates

Adding more layers to GCN expands the graph neighbourhood within which the user features are averaged, and so might introduce noise, and consequently decrease accuracy as shown in Figure 5 when no gates are used. We see that by adding highway network gates, the performance of GCN slightly improves until three layers are added, but then by adding more layers the performance doesn't change that much as gates are allowing the layer inputs to pass through the network without much change. The performance peaks at 4 layers which is compatible with the distribution of shortest path lengths shown in Figure 6.

4.4 Performance

The performance of the three proposed models (MLP−TXT+NET, DCCA and GCN) is shown in Table 1. The models are also compared with supervised text-based methods (Wing and Baldridge, 2014; Cha et al., 2015; Rahimi et al., 2017b), a network-based method (Rahimi et al., 2015a) and GCN−LP, and also joint text and network models (Rahimi et al., 2017b; Do et al., 2017; Miura et al., 2017). MLP−TXT+NET and GCN outperform all the text- or network-only models, and also the hybrid model of Rahimi et al. (2017b), indicating that joint modelling of text and network features is important. MLP−TXT+NET is competitive with Do et al. (2017), outperforming it on larger datasets, and underperforming on GEO-

	GEOTEXT			TWITTER-US			TWITTER-WORLD		
	Acc@161↑	Mean↓	Median↓	Acc@161↑	Mean↓	Median↓	Acc@161↑	Mean↓	Median↓
Text (inductive)									
Rahimi et al. (2017b)	38	844	389	54	554	120	34	1456	415
Wing and Baldridge (2014)	—	—	—	48	686	191	31	1669	509
Cha et al. (2015)	—	581	425	—	—	—	—	—	—
Network (transductive)									
Rahimi et al. (2015a)	58	586	60	54	705	116	45	2525	279
GCN-LP	58	576	56	53	653	126	45	2357	279
Text+Network (transductive)									
Do et al. (2017)	**62**	**532**	**32**	**66**	433	**45**	53	1044	118
Miura et al. (2017)	—	—	—	61	481	65	—	—	—
Rahimi et al. (2017b)	59	578	61	61	515	77	53	1280	104
MLP-TXT+NET	58	554	58	**66**	**420**	56	**58**	**1030**	**53**
DCCA	56	627	79	58	516	90	21	2095	913
GCN	60	546	45	62	485	71	54	1130	108
Text+Network (transductive)									
MLP-TXT+NET 1%	**8**	1521	1295	14	1436	1411	8	3865	2041
DCCA 1%	7	1425	979	38	869	348	14	3014	1367
GCN 1%	6	**1103**	**609**	**41**	**788**	**311**	**21**	**2071**	**853**

Table 1: Geolocation results over the three Twitter datasets for the proposed models: joint text+network MLP-TXT+NET, DCCA, and GCN and network-based GCN-LP. The models are compared with text-only and network-only methods. The performance of the three joint models is also reported for minimal supervision scenario where only 1% of the total samples are labelled. "—" signifies that no results were reported for the given metric or dataset. Note that Do et al. (2017) use timezone, and Miura et al. (2017) use the description and location fields in addition to text and network.

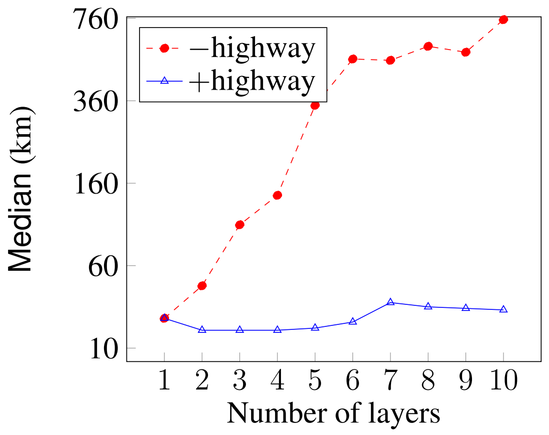


Figure 5: The effect of adding more GCN layers (neighbourhood expansion) to GCN in terms of median error over the development set of GEOTEXT with and without the highway gates, and averaged over 5 runs.

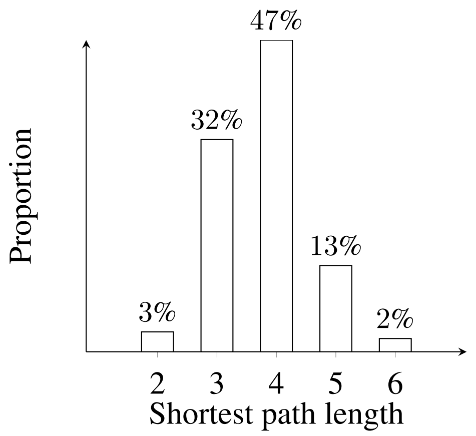


Figure 6: The distribution of shortest path lengths between all the nodes of the largest connected component of GEOTEXT's graph that constitute more than 1% of total.

TEXT. However, it's difficult to make a fair comparison as they use timezone data in their feature set. MLP-TXT+NET outperforms GCN over TWITTER-US and TWITTER-WORLD, which are very large, and have large amounts of labelled data. In a scenario with little supervision (1% of the total samples are labelled) DCCA and GCN clearly outperform MLP-TXT+NET, as they have fewer parameters. Except for Acc@161 over GEOTEXT where the number of labelled samples in the minimal supervision scenario is very low, GCN outperforms DCCA by a large margin, indicating that for a medium dataset where only 1% of samples are labelled (as happens in random samples of Twitter) GCN is superior to MLP-TXT+NET and DCCA, consistent with Section 4.2. Both MLP-TXT+NET and GCN achieve state of the art results compared to network-only, text-only, and hybrid models. The network-based GCN-LP model, which does label propagation using Graph Convolutional Networks, outperforms Rahimi et al. (2015a), which is based on location propagation using Modified Adsorption (Talukdar and Crammer, 2009), possibly because the label propagation in GCN is parametrised.

4.5 Error Analysis

Although the performance of MLP-TXT+NET is better than GCN and DCCA when a large amount of labelled data is available (Table 1), under a scenario where little labelled data is available (1% of data), DCCA and GCN outperform MLP-TXT+NET, mainly because the number of parameters in MLP-TXT+NET grows with the number of samples, and is much larger than GCN and DCCA. GCN outperforms DCCA and MLP-TXT+NET using 1% of data, however, the distribution of errors in the development set of TWITTER-US indicates higher error for smaller states such as Rhode Island (RI), Iowa (IA), North Dakota (ND), and Idaho (ID), which is simply because the number of labelled samples in those states is insufficient.

Although we evaluate geolocation models with Median, Mean, and Acc@161, it doesn't mean that the distribution of errors is uniform over all locations. Big cities often attract more local online discussions, making the geolocation of users in those areas simpler. For example users in LA are more likely to talk about LA-related issues such as their sport teams, Hollywood or local events than users in the state of Rhode Island (RI), which lacks large sport teams or major events. It is also possible that people in less densely populated areas are further apart from each other, and therefore, as a result of discretisation fall in different clusters. The non-uniformity in local discussions results in lower geolocation performance in less densely populated areas like Midwest U.S., and higher performance in densely populated areas such as NYC and LA as shown in Figure 7. The geographical distribution of error for GCN, DCCA and MLP-TXT+NET under the minimal supervision scenario is shown in the supplementary material.

To get a better picture of misclassification between states, we built a confusion matrix based on known state and predicted state for development users of TWITTER-US using GCN using only 1% of labelled data. There is a tendency for users to be wrongly predicted to be in CA, NY, TX, and surprisingly OH. Particularly users from states such as TX, AZ, CO, and NV, which are located close to CA, are wrongly predicted to be in CA, and users from NJ, PA, and MA are misclassified as being in NY. The same goes for OH and TX where users from neighbouring smaller states are misclassified to be there. Users from CA and NY are also misclassified between the two states, which might be the result of business and entertainment connections that exist between NYC and LA/SF. Interestingly, there are a number of misclassifications to FL for users from CA, NY, and TX, which might be the effect of users vacationing or retiring to FL. The full confusion matrix between the U.S. states is provided in the supplementary material.

4.6 Local Terms

In Table 2, local terms of a few regions detected by GCN under minimal supervision are shown. The terms that were present in the labelled data are excluded to show how graph convolutions over the social graph have extended the vocabulary. For example, in case of Seattle, *#goseahawks* is an important term not present in the 1% labelled data but present in the unlabelled data. The convolution over the social graph is able to utilise such terms that don't exist in the labelled data.

5 Related Work

Previous work on user geolocation can be broadly divided into text-based, network-based and multi-view approaches.

Text-based geolocation uses the geographical bias in language use to infer the location of users. There are three main text-based approaches to geolocation: (1) gazetteer-based models which map geographical references in text to location, but ignore non-geographical references and vernacular uses of language (Rauch et al., 2003; Amitay et al., 2004; Lieberman et al., 2010); (2) geographical topic models that learn region-specific topics, but don't scale to the magnitude of social media (Eisenstein et al., 2010; Hong et al., 2012; Ahmed et al., 2013); and (3) supervised models which are often framed as text classification (Serdyukov et al., 2009; Wing and Baldridge, 2011; Roller et al., 2012; Han et al., 2014) or text regression (Iso et al., 2017; Rahimi et al., 2017a). Supervised models scale well and can achieve good performance with sufficient supervision, which is not available in a real world scenario.

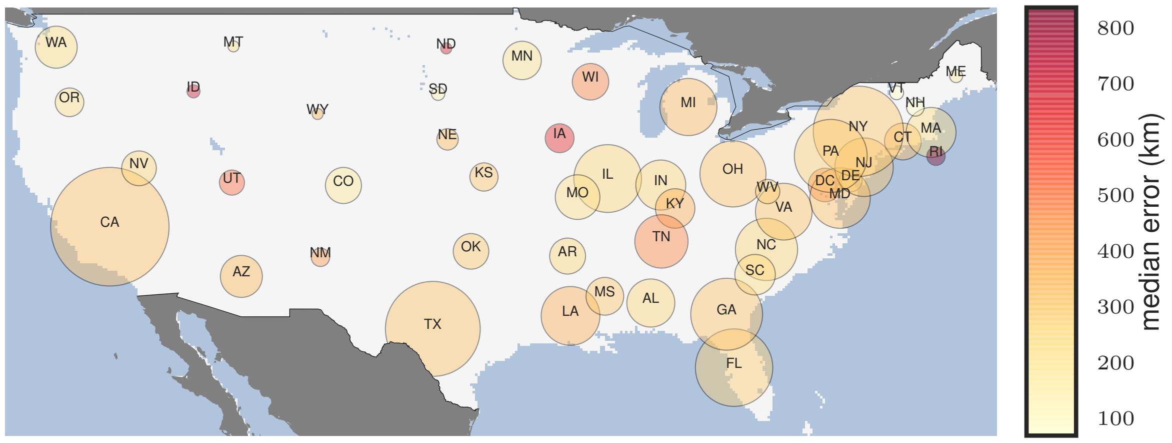


Figure 7: The geographical distribution of Median error of GCN using 1% of labelled data in each state over the development set of TWITTER-US. The colour indicates error and the size indicates the number of development users within the state.

Seattle, WA	Austin, TX	Jacksonville, FL	Columbus, OH	Charlotte, NC	Phoenix, AZ	New Orleans, LA	Baltimore, MD
#goseahawks	stubb	unf	laffayette	#asheville	clutterbuck	mcneese	bhop
smock	gsd	ribault	#weareohio	#depinga	waffels	keela	#dsu
traffuck	#meatsweats	wahoowa	#arcgis	batesburg	bahumbug	pentecostals	chestertown
ferran	lanterna	wjct	#slammin	stewey	iedereen	lutcher	aduh
promissory	pupper	fscj	#ouhc	#bojangles	rockharbor	grogan	umbc
chowdown	effaced	floridian	#cow	#occupyraleigh	redtail	suela	lmt
ckrib	#austin	#jacksonville	mommyhood	gville	gewoon	cajuns	assistly
#uwhuskies	lmfbo	#mer	beering	sweezy	jms	bmu	slurpies

Table 2: Top terms for selected regions detected by GCN using only 1% of TWITTER-US for supervision. We present the terms that were present only in unlabelled data. The terms include city names, hashtags, food names and internet abbreviations.

Network-based methods leverage the *location homophily* assumption: nearby users are more likely to befriend and interact with each other. There are four main network-based geolocation approaches: distance-based, supervised classification, graph-based label propagation, and node embedding methods. Distance-based methods model the probability of friendship given the distance (Backstrom et al., 2010; McGee et al., 2013; Gu et al., 2012; Kong et al., 2014), supervised models use neighbourhood features to classify a user into a location (Rout et al., 2013; Malmi et al., 2015), and graph-based label-propagation models propagate the location information through the user–user graph to estimate unknown labels (Davis Jr et al., 2011; Jurgens, 2013; Compton et al., 2014). Node embedding methods build heterogeneous graphs between user–user, user–location and location–location, and learn an embedding space to minimise the distance of connected nodes, and maximise the distance of disconnected nodes. The embeddings are then used in supervised models for geolocation (Wang et al., 2017). Network-based models fail to geolocate disconnected users: Jurgens et al. (2015) couldn't geolocation 37% of users as a result of disconnectedness.

Previous work on hybrid text and network methods can be broadly categorised into three main approaches: (1) incorporating text-based information such as toponyms or locations predicted from a text-based model as auxiliary nodes into the user–user graph, which is then used in network-based models (Li et al., 2012a,b; Rahimi et al., 2015b,a); (2) ensembling separately trained text- and network-based models (Gu et al., 2012; Ren et al., 2012; Jayasinghe et al., 2016; Ribeiro and Pappa, 2017); and (3) jointly learning geolocation from several information sources such as text and network information (Miura et al., 2017; Do et al., 2017), which can capture the complementary information in text and network views, and also model the interactions between the two. None of the previous multiview approaches — with the exception of Li et al. (2012a) and Li et al. (2012b) that only use toponyms — effectively uses unlabelled data in the text view, and use only the unlabelled information of the network view via the user–user graph.

There are three main shortcomings in the previous work on user geolocation that we address in this paper: (1) with the exception of few recent works (Miura et al., 2017; Do et al., 2017), previous models don't jointly exploit both text and network information, and therefore the interaction between text and network views is not modelled; (2) the unlabelled data in both text and network views is not effectively exploited, which is crucial given the small amounts of available supervision; and (3) previous models are rarely evaluated under a minimal supervision scenario, a scenario which reflects real world conditions.

6 Conclusion

We proposed `GCN`, `DCCA` and `MLP-TXT+NET`, three multiview, transductive, semi-supervised geolocation models, which use text and network information to infer user location in a joint setting. We showed that joint modelling of text and network information outperforms network-only, text-only, and hybrid geolocation models as a result of modelling the interaction between text and network information. We also showed that `GCN` and `DCCA` are able to perform well under a minimal supervision scenario similar to real world applications by effectively using unlabelled data. We ignored the context in which users interact with each other, and assumed all the connections to hold location homophily. In future work, we are interested in modelling the extent to which a social interaction is caused by geographical proximity (e.g. using user–user gates).

References

Amr Ahmed, Liangjie Hong, and Alexander J. Smola. 2013. Hierarchical geographical modeling of user locations from social media posts. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, pages 25–36, Rio de Janeiro, Brazil.

Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer. 2004. Web-a-where: geotagging web content. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 273–280, Sheffield, UK.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, Atlanta, USA.

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*, pages 61–70, Raleigh, USA.

Miriam Cha, Youngjune Gwon, and H.T. Kung. 2015. Twitter geolocation and regional classification via sparse coding. In *Proceedings of the 9th International Conference on Weblogs and Social Media (ICWSM 2015)*, pages 582–585, Oxford, UK.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating Twitter users. In *Proceedings of the 19th ACM International Conference Information and Knowledge Management (CIKM 2010)*, pages 759–768, Toronto, Canada.

Jaime Chon, Ross Raymond, Haiyan Wang, and Feng Wang. 2015. Modeling flu trends with real-time geo-tagged twitter data streams. In *Proceedings of the 10th International Conference on Wireless Algorithms, Systems, and Applications (WASA 2015)*, pages 60–69, Qufu, China.

Ryan Compton, David Jurgens, and David Allen. 2014. Geotagging one hundred million twitter accounts with total variation minimization. In *Proceedings of the IEEE International Conference on Big Data (IEEE BigData 2014)*, pages 393–401, Washington DC, USA.

Clodoveu A Davis Jr, Gisele L Pappa, Diogo Rennó Rocha de Oliveira, and Filipe de L Arcanjo. 2011. Inferring the location of twitter messages based on user relationships. *Transactions in GIS*, 15(6):735–751.

Bertrand De Longueville, Robin S. Smith, and Gianluca Luraschi. 2009. "omg, from here, i can see the flames!": A use case of mining location based social networks to acquire spatio-temporal data on forest fires. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, pages 73–80, New York, USA.

Tien Huu Do, Duc Minh Nguyen, Evaggelia Tsiligianni, Bruno Cornelis, and Nikos Deligiannis. 2017. Multiview deep learning for predicting twitter users' location. *arXiv preprint arXiv:1712.08091.*

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 1277–1287, Boston, USA.

Hansu Gu, Haojie Hang, Qin Lv, and Dirk Grunwald. 2012. Fusing text and frienships for location inference in online social networks. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, volume 1, pages 158–165, Macau, China.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1045–1062, Mumbai, India.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H. Chi. 2011. Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 237–246, Vancouver, Canada.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st international conference on World Wide Web*, pages 769–778, Lyon, France.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Hayate Iso, Shoko Wakamiya, and Eiji Aramaki. 2017. Density estimation for geolocation via convolutional mixture density network. *arXiv preprint arXiv:1705.02750*.

Gaya Jayasinghe, Brian Jin, James Mchugh, Bella Robinson, and Stephen Wan. 2016. CSIRO Data61 at the WNUT geo shared task. In *Proceedings of the COLING 2016 Workshop on Noisy User-generated Text (W-NUT 2016)*, pages 218–226, Osaka, Japan.

David Jurgens. 2013. That's what friends are for: Inferring location in online social media platforms based on social relationships. In *Proceedings of the 7th International Conference on Weblogs and Social Media (ICWSM 2013)*, pages 273–282, Boston, USA.

David Jurgens, Tyler Finethy, James McCorriston, Yi Tian Xu, and Derek Ruths. 2015. Geolocation prediction in twitter using social networks: A critical analysis and review of current practice. In *Proceedings of the 9th International Conference on Weblogs and Social Media (ICWSM 2015)*, pages 188–197, Oxford, UK.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Longbo Kong, Zhi Liu, and Yan Huang. 2014. Spot: Locating social media users based on social network context. *Proceedings of the VLDB Endowment*, 7(13):1681–1684.

Rui Li, Shengjie Wang, and Kevin Chen-Chuan Chang. 2012a. Multiple location profiling for users and relationships from social network and content. *Proceedings of the VLDB Endowment*, 5(11):1603–1614.

Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. 2012b. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2012)*, pages 1023–1031, Beijing, China.

Michael D Lieberman, Hanan Samet, and Jagan Sankaranarayanan. 2010. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *Proceedings of the 26th International Conference on Data Engineering (ICDE 2010)*, pages 201–212, Long Beach, USA.

Eric Malmi, Arno Solin, and Aristides Gionis. 2015. The blind leading the blind: Network-based location estimation under uncertainty. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2015 (ECML PKDD 2015)*, pages 406–421, Porto, Portugal.

Jeffrey McGee, James Caverlee, and Zhiyuan Cheng. 2013. Location prediction in social media based on tie strength. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 459–468, San Fransisco, USA. ACM.

Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. 2017. Unifying text, metadata, and user network representations with a neural network for geolocation prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1260–1272, Vancouver, Canada.

Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen M Carley. 2013. Is the sample good enough? Comparing data from Twitter's streaming API with Twitter's firehose. In *Proceedings of the 7th International Conference on Weblogs and Social Media (ICWSM 2013)*, pages 400–408, Boston, USA.

Michael J. Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. In *Proceedings of the Fifth International Conference on Weblogs and Social Media (ICSWM 2011)*, pages 265–272, Barcelona, Spain.

Afshin Rahimi, Timothy Baldwin, and Trevor Cohn. 2017a. Continuous representation of location for geolocation and lexical dialectology using mixture density networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language*

Processing (EMNLP 2017), pages 167–176, Copenhagen, Denmark.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2015a. Twitter user geolocation using a unified text and network prediction model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics — 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 630–636, Beijing, China.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2017b. A neural model for user geolocation and lexical dialectology. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 207–216, Vancouver, Canada.

Afshin Rahimi, Duy Vu, Trevor Cohn, and Timothy Baldwin. 2015b. Exploiting text and network context for geolocation of social media users. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2015)*, pages 1362–1367, Denver, USA.

Erik Rauch, Michael Bukatin, and Kenneth Baker. 2003. A confidence-based framework for disambiguating geographic terms. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*, pages 50–54, Edmonton, Canada.

Kejiang Ren, Shaowu Zhang, and Hongfei Lin. 2012. Where are you settling down: Geo-locating Twitter users based on tweets and social networks. In *Proceedings of the 8th Asia Information Retrieval Societies Conference (AIRS 2012)*, pages 150–161, Tianjin, China.

Silvio Ribeiro and Gisele L. Pappa. 2017. Strategies for combining Twitter users geo-location methods. *GeoInformatica*, pages 1–25.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL 2012)*, pages 1500–1510, Jeju, South Korea.

Dominic Rout, Kalina Bontcheva, Daniel Preoţiuc-Pietro, and Trevor Cohn. 2013. Where's @wally?: A classification approach to geolocating users based on their social ties. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media (Hypertext 2013)*, pages 11–20, Paris, France.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, pages 851–860, New York, USA.

Pavel Serdyukov, Vanessa Murdock, and Roelof Van Zwol. 2009. Placing Flickr photos on a map. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 484–491, Boston, USA.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387.*

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning (ECML-PKDD 2009)*, pages 442–457, Bled, Slovenia.

Fengjiao Wang, Chun-Ta Lu, Yongzhi Qu, and S Yu Philip. 2017. Collective geographical embedding for geolocating social network users. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2017)*, pages 599–611, Jeju, South Korea.

Benjamin P Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (ACL-HLT 2011)*, pages 955–964, Portland, USA.

Benjamin P Wing and Jason Baldridge. 2014. Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 336–348, Doha, Qatar.

Antonio Jimeno Yepes, Andrew MacKinlay, and Bo Han. 2015. Investigating public health surveillance using twitter. In *Proceedings of the 2015 Workshop on Biomedical Natural Language Processing (BioNLP 2015)*, pages 164–170, Beijing, China.

Document Modeling with External Attention for Sentence Extraction

Shashi Narayan[*]
University of Edinburgh
shashi.narayan@ed.ac.uk

Ronald Cardenas[*]
Charles University in Prague
ronald.cardenas@matfyz.cz

Nikos Papasarantopoulos[*]
University of Edinburgh
nikos.papasa@ed.ac.uk

Shay B. Cohen **Mirella Lapata**
University of Edinburgh
{scohen,mlap}@inf.ed.ac.uk

Jiangsheng Yu **Yi Chang**
Huawei Technologies
{jiangsheng.yu,yi.chang}@huawei.com

Abstract

Document modeling is essential to a variety of natural language understanding tasks. We propose to use external information to improve document modeling for problems that can be framed as sentence extraction. We develop a framework composed of a hierarchical document encoder and an attention-based extractor with attention over external information. We evaluate our model on extractive document summarization (where the external information is image captions and the title of the document) and answer selection (where the external information is a question). We show that our model consistently outperforms strong baselines, in terms of both informativeness and fluency (for CNN document summarization) and achieves state-of-the-art results for answer selection on WikiQA and NewsQA.[1]

1 Introduction

Recurrent neural networks have become one of the most widely used models in natural language processing (NLP). A number of variants of RNNs such as Long Short-Term Memory networks (LSTM; Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit networks (GRU; Cho et al., 2014) have been designed to model text capturing long-term dependencies in problems such as language modeling. However, document modeling, a key to many natural language understanding tasks, is still an open challenge. Recently, some neural network architectures were proposed to capture large context for modeling text (Mikolov and Zweig, 2012; Ghosh et al., 2016; Ji et al., 2015; Wang and Cho, 2016). Lin et al. (2015) and Yang et al. (2016) proposed a hierarchical RNN network for document-level modeling as well as sentence-level modeling, at the cost of increased computational complexity. Tran et al. (2016) further proposed a contextual language model that considers information at inter-document level.

It is challenging to rely only on the document for its understanding, and as such it is not surprising that these models struggle on problems such as document summarization (Cheng and Lapata, 2016; Chen et al., 2016; Nallapati et al., 2017; See et al., 2017; Tan and Wan, 2017) and machine reading comprehension (Trischler et al., 2016; Miller et al., 2016; Weissenborn et al., 2017; Hu et al., 2017; Wang et al., 2017). In this paper, we formalize the use of external information to further guide document modeling for end goals.

We present a simple yet effective document modeling framework for sentence extraction that allows machine reading with "external attention." Our model includes a neural hierarchical document encoder (or a machine reader) and a hierarchical attention-based sentence extractor. Our hierarchical document encoder resembles the architectures proposed by Cheng and Lapata (2016) and Narayan et al. (2018) in that it derives the document meaning representation from its sentences and their constituent words. Our novel sentence extractor combines this document meaning representation with an attention mechanism (Bahdanau et al., 2015) over the external information to label sentences from the input document. Our model explicitly biases the extractor with external cues and

[*]The first three authors made equal contributions to this paper. The work was done when the second author was visiting Edinburgh.

[1]Our TensorFlow code and datasets are publicly available at https://github.com/shashiongithub/Document-Models-with-Ext-Information.

implicitly biases the encoder through training.

We demonstrate the effectiveness of our model on two problems that can be naturally framed as sentence extraction with external information. These two problems, extractive document summarization and answer selection for machine reading comprehension, both require local and global contextual reasoning about a given document. Extractive document summarization systems aim at creating a summary by identifying (and subsequently concatenating) the most important sentences in a document, whereas answer selection systems select the candidate sentence in a document most likely to contain the answer to a query. For document summarization, we exploit the title and image captions which often appear with documents (specifically newswire articles) as external information. For answer selection, we use word overlap features, such as the inverse sentence frequency (ISF, Trischler et al., 2016) and the inverse document frequency (IDF) together with the query, all formulated as external cues.

Our main contributions are three-fold: First, our model ensures that sentence extraction is done in a larger (rich) context, i.e., the full document is read first before we start labeling its sentences for extraction, and each sentence labeling is done by implicitly estimating its local and global relevance to the document and by directly attending to some external information for importance cues.

Second, while external information has been shown to be useful for summarization systems using traditional hand-crafted features (Edmundson, 1969; Kupiec et al., 1995; Mani, 2001), our model is the first to exploit such information in deep learning-based summarization. We evaluate our models automatically (in terms of ROUGE scores) on the CNN news highlights dataset (Hermann et al., 2015). Experimental results show that our summarizer, informed with title and image captions, consistently outperforms summarizers that do not use this information. We also conduct a human evaluation to judge which type of summary participants prefer. Our results overwhelmingly show that human subjects find our summaries more informative and complete.

Lastly, with the machine reading capabilities of our model, we confirm that a full document needs to be "read" to produce high quality extracts allowing a rich contextual reasoning, in contrast to previous answer selection approaches that often measure a score between each sentence in the document and the question and return the sentence with highest score in an isolated manner (Yin et al., 2016; dos Santos et al., 2016; Wang et al., 2016). Our model with ISF and IDF scores as external features achieves competitive results for answer selection. Our ensemble model combining scores from our model and word overlap scores using a logistic regression layer achieves state-of-the-art results on the popular question answering datasets WikiQA (Yang et al., 2015) and NewsQA (Trischler et al., 2016), and it obtains comparable results to the state of the art for SQuAD (Rajpurkar et al., 2016). We also evaluate our approach on the MSMarco dataset (Nguyen et al., 2016) and elaborate on the behavior of our machine reader in a scenario where each candidate answer sentence is contextually independent of each other.

2 Document Modeling For Sentence Extraction

Given a document D consisting of a sequence of n sentences $(s_1, s_2, ..., s_n)$, we aim at labeling each sentence s_i in D with a label $y_i \in \{0, 1\}$ where $y_i = 1$ indicates that s_i is extraction-worthy and 0 otherwise. Our architecture resembles those previously proposed in the literature (Cheng and Lapata, 2016; Nallapati et al., 2017). The main components include a sentence encoder, a document encoder, and a novel sentence extractor (see Figure 1) that we describe in more detail below. The novel characteristics of our model are that each sentence is labeled by implicitly estimating its (local and global) relevance to the document and by directly attending to some external information for importance cues.

Sentence Encoder A core component of our model is a convolutional sentence encoder (Kim, 2014; Kim et al., 2016) which encodes sentences into continuous representations. We use temporal narrow convolution by applying a kernel filter K of width h to a window of h words in sentence s to produce a new feature. This filter is applied to each possible window of words in s to produce a feature map $f \in R^{k-h+1}$ where k is the sentence length. We then apply max-pooling over time over the feature map f and take the maximum value as the feature corresponding to this particular filter K. We use multiple kernels of various sizes and each kernel multiple times to construct the representation of a sentence. In Figure 1, kernels of size 2 (red) and 4 (blue) are applied three times each. The max-pooling over time operation yields two feature lists f^{K_2} and $f^{K_4} \in R^3$. The final sentence embeddings have six dimensions.

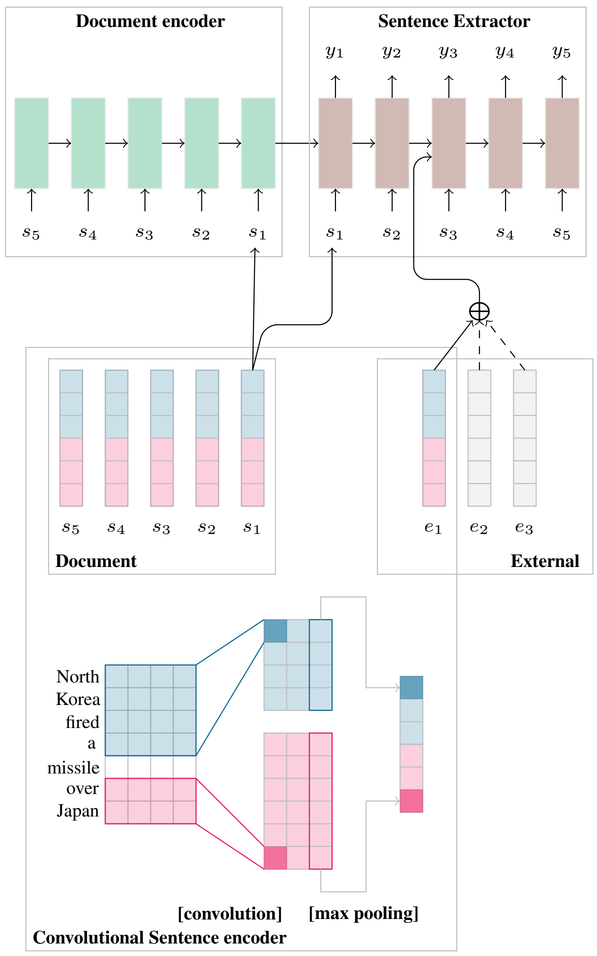
Document Encoder The document encoder composes a sequence of sentences to obtain a document representation. We use a recurrent neural network with LSTM cells to avoid the vanishing gradient problem when training long sequences (Hochreiter and Schmidhuber, 1997). Given a document D consisting of a sequence of sentences $(s_1, s_2, \ldots, s_n)$, we follow common practice and feed the sentences in reverse order (Sutskever et al., 2014; Li et al., 2015; Filippova et al., 2015).

Sentence Extractor Our sentence extractor sequentially labels each sentence in a document with 1 or 0 by implicitly estimating its relevance in the document and by directly attending to the external information for importance cues. It is implemented with another RNN with LSTM cells with an attention mechanism (Bahdanau et al., 2015) and a softmax layer. Our attention mechanism differs from the standard practice of attending intermediate states of the input (encoder). Instead, our extractor attends to a sequence of p pieces of external information $E : (e_1, e_2, ..., e_p)$ relevant for the task (e.g., e_i is a title or an image caption for summarization) for cues. At time t_i, it reads sentence s_i and makes a binary prediction, conditioned on the document representation (obtained from the document encoder), the previously labeled sentences and the external information. This way, our labeler is able to identify locally and globally important sentences within the document which correlate well with the external information.

Given sentence s_t at time step t, it returns a probability distribution over labels as:

$$p(y_t|s_t, D, E) = \text{softmax}(g(h_t, h'_t)) \quad (1)$$
$$g(h_t, h'_t) = U_o(V_h h_t + W'_h h'_t) \quad (2)$$
$$h_t = \text{LSTM}(s_t, h_{t-1})$$
$$h'_t = \sum_{i=1}^{p} \alpha_{(t,i)} e_i,$$
$$\text{where } \alpha_{(t,i)} = \frac{\exp(h_t e_i)}{\sum_j \exp(h_t e_j)}$$

where $g(\cdot)$ is a single-layer neural network with parameters U_o, V_h and W'_h. h_t is an intermediate RNN state at time step t. The dynamic context vector h'_t is essentially the weighted sum of the external information $(e_1, e_2, \ldots, e_p)$. Figure 1 summarizes our model.



Document encoder
Sentence Extractor
y_1 y_2 y_3 y_4 y_5
s_5 s_4 s_3 s_2 s_1
s_1 s_2 s_3 s_4 s_5
s_5 s_4 s_3 s_2 s_1
Document
e_1 e_2 e_3
External
North
Korea
fired
a
missile
over
Japan
[convolution]
[max pooling]
Convolutional Sentence encoder

Figure 1: Hierarchical encoder-decoder model for sentence extraction with external attention. $s_1, \ldots, s_5$ are sentences in the document and, e_1, e_2 and e_3 represent external information. For the extractive summarization task, e_is are external information such as title and image captions. For the answers selection task, e_is are the query and word overlap features.

3 Sentence Extraction Applications

We validate our model on two sentence extraction problems: extractive document summarization and answer selection for machine reading comprehension. Both these tasks require local and global contextual reasoning about a given document. As such, they test the ability of our model to facilitate document modeling using external information.

Extractive Summarization An extractive summarizer aims to produce a summary $\mathcal{S}$ by selecting m sentences from D (where $m < n$). In this setting, our sentence extractor sequentially predicts label $y_i \in \{0, 1\}$ (where 1 means that s_i should be included in the summary) by assigning score $p(y_i|s_i, D, E, \theta)$ quantifying the relevance of s_i to the summary. We assemble a summary $\mathcal{S}$ by selecting m sentences with top $p(y_i = 1|s_i, D, E, \theta)$ scores.

We formulate external information E as the sequence of the title and the image captions associated with the document. We use the convolutional sentence encoder to get their sentence-level representations.

Answer Selection Given a question q and a document D , the goal of the task is to select one candidate sentence $s_i \in D$ in which the answer exists. In this setting, our sentence extractor sequentially predicts label $y_i \in \{0, 1\}$ (where 1 means that s_i contains the answer) and assign score $p(y_i|s_i, D, E, \theta)$ quantifying s_i's relevance to the query. We return as answer the sentence s_i with the highest $p(y_i = 1|s_i, D, E, \theta)$ score.

We treat the question q as external information and use the convolutional sentence encoder to get its sentence-level representation. This simplifies Eq. (1) and (2) as follow:

$$
\begin{aligned}
p(y_t|s_t, D, q) &= \text{softmax}(g(h_t, q)) \\
g(h_t, q) &= U_o(V_h h_t + W_q q),
\end{aligned} \tag{3}
$$

where V_h and W_q are network parameters. We exploit the simplicity of our model to further assimilate external features relevant for answer selection: the inverse sentence frequency (ISF, (Trischler et al., 2016)), the inverse document frequency (IDF) and a modified version of the ISF score which we call *local ISF*. Trischler et al. (2016) have shown that a simple ISF baseline (i.e., a sentence with the highest ISF score as an answer) correlates well with the answers. The ISF score α_{s_i} for the sentence s_i is computed as $\alpha_{s_i} = \sum_{w \in s_i \cap q} \text{IDF}(w)$, where IDF is the inverse document frequency score of word w, defined as: $\text{IDF}(w) = \log \frac{N}{N_w}$, where N is the total number of sentences in the training set and N_w is the number of sentences in which w appears. Note that, $s_i \cap q$ refers to the set of words that appear both in s_i and in q. Local ISF is calculated in the same manner as the ISF score, only with setting the total number of sentences (N) to the number of sentences in the article that is being analyzed.

More formally, this modifies Eq. (3) as follows:

$$
p(y_t|s_t, D, q) = \text{softmax}(g(h_t, q, \alpha_t, \beta_t, \gamma_t)) \tag{4}
$$

where α_t, β_t and γ_t are the ISF, IDF and local ISF scores (real values) of sentence s_t respectively . The function g is calculated as follows:

$$
\begin{aligned}
g(h_t, q, \alpha_t, \beta_t, \gamma_t) = &U_o\,\big(V_h h_t + \\
&W_q q + W_{\text{isf}}(\alpha_t \cdot \mathbb{1}) + \\
&W_{\text{idf}}(\beta_t \cdot \mathbb{1}) + W_{\text{lisf}}(\gamma_t \cdot \mathbb{1})\big)\,,
\end{aligned}
$$

where W_{isf}, W_{idf} and W_{lisf} are new parameters added to the network and $\mathbb{1}$ is a vector of 1s of size equal to the sentence embedding size. In Figure 1, these external feature vectors are represented as 6-dimensional gray vectors accompanied with dashed arrows.

4 Experiments and Results

This section presents our experimental setup and results assessing our model in both the extractive summarization and answer selection setups. In the rest of the paper, we refer to our model as XNET for its ability to exploit e**X**ternal information to improve document representation.

4.1 Extractive Document Summarization

Summarization Dataset We evaluated our models on the CNN news highlights dataset (Hermann et al., 2015).[2] We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 documents). We followed previous studies (Cheng and Lapata, 2016; Nallapati et al., 2016, 2017; See et al., 2017; Tan and Wan, 2017) in assuming that the

[2]Hermann et al. (2015) have also released the DailyMail dataset, but we do not report our results on this dataset. We found that the script written by Hermann et al. to crawl DailyMail articles mistakenly extracts image captions as part of the main body of the document. As image captions often do not have sentence boundaries, they blend with the sentences of the document unnoticeably. This leads to the production of erroneous summaries.

"story highlights" associated with each article are gold-standard abstractive summaries. We trained our network on a named-entity-anonymized version of news articles. However, we generated deanonymized summaries and evaluated them against gold summaries to facilitate human evaluation and to make human evaluation comparable to automatic evaluation.

To train our model, we need documents annotated with sentence extraction information, i.e., each sentence in a document is labeled with 1 (summary-worthy) or 0 (not summary-worthy). We followed Nallapati et al. (2017) and automatically extracted ground truth labels such that all positively labeled sentences from an article collectively give the highest ROUGE (Lin and Hovy, 2003) score with respect to the gold summary.

We used a modified script of Hermann et al. (2015) to extract titles and image captions, and we associated them with the corresponding articles. All articles get associated with their titles. The availability of image captions varies from 0 to 414 per article, with an average of 3 image captions. There are 40% CNN articles with at least one image caption.

All sentences, including titles and image captions, were padded with zeros to a sentence length of 100. All input documents were padded with zeros to a maximum document length of 126. For each document, we consider a maximum of 10 image captions. We experimented with various numbers (1, 3, 5, 10 and 20) of image captions on the validation set and found that our model performed best with 10 image captions. We refer the reader to the supplementary material for more implementation details to replicate our results.

Comparison Systems We compared the output of our model against the standard baseline of simply selecting the first three sentences from each document as the summary. We refer to this baseline as LEAD in the rest of the paper.

We also compared our system against the sentence extraction system of Cheng and Lapata (2016). We refer to this system as POINTERNET as the neural attention architecture in Cheng and Lapata (2016) resembles the one of Pointer Networks (Vinyals et al., 2015).[3] It does not exploit any external information.[4] Cheng and Lapata (2016) report only on the DailyMail dataset. We used their code (`https://github.com/cheng6076/NeuralSum`) to produce results on the CNN dataset.[5]

MODELS	R1	R2	R3	R4	RL	Avg.
LEAD	49.2	18.9	9.8	6.0	43.8	25.5
POINTERNET	53.3	19.7	10.4	6.4	47.2	27.4
XNET+TITLE	55.0	21.6	11.7	7.5	48.9	28.9
XNET+CAPTION	55.3	21.3	11.4	7.2	49.0	28.8
XNET+FS	54.8	21.1	11.3	7.2	48.6	28.6
Combination Models (XNET+)						
TITLE+CAPTION	**55.4**	**21.8**	**11.8**	**7.5**	**49.2**	**29.2**
TITLE+FS	55.1	21.6	11.6	7.4	48.9	28.9
CAPTION+FS	55.3	21.5	11.5	7.3	49.0	28.9
TITLE+CAPTION+FS	55.4	21.5	11.6	7.4	49.1	29.0

Table 1: Ablation results on the validation set. We report R1, R2, R3, R4, RL and their average (Avg.). The first block of the table presents LEAD and POINTERNET which do not use any external information. LEAD is the baseline system selecting first three sentences. POINTERNET is the sentence extraction system of Cheng and Lapata. XNET is our model. The second and third blocks of the table present different variants of XNET. We experimented with three types of external information: title (TITLE), image captions (CAPTION) and the first sentence (FS) of the document. The bottom block of the table presents models with more than one type of external information. The best performing model (highlighted in boldface) is used on the test set.

Automatic Evaluation To automatically assess the quality of our summaries, we used ROUGE (Lin and Hovy, 2003), a recall-oriented metric, to compare our model-generated summaries to manually-written highlights.[6] Previous work has reported ROUGE-1 (R1) and ROUGE-2 (R2) scores to access informativeness, and ROUGE-L (RL) to access fluency. In addition to R1, R2 and RL, we also report ROUGE-3 (R3) and ROUGE-4 (R4) capturing higher order n-grams overlap to assess informativeness and fluency simultaneously.

[3]The architecture of POINTERNET is closely related to our model without external information.

[4]Adding external information to POINTERNET is an interesting direction of research but we do not pursue it here. It requires decoding with multiple types of attentions and this is not the focus of this paper.

[5]We are unable to compare our results to the extractive system of Nallapati et al. (2017) because they report their results on the DailyMail dataset and their code is not available. The abstractive systems of Chen et al. (2016) and Tan and Wan (2017) report their results on the CNN dataset, however, their results are not comparable to ours as they report on the full-length F_1 variants of ROUGE to evaluate their abstractive summaries. We report ROUGE recall scores which is more appropriate to evaluate our extractive summaries.

[6]We used `pyrouge`, a Python package, to compute all our ROUGE scores with parameters "-a -c 95 -m -n 4 -w 1.2."

We report our results on both full length (three sentences with the top scores as the summary) and fixed length (first 75 bytes and 275 bytes as the summary) summaries. For full length summaries, our decision of selecting three sentences is guided by the fact that there are 3.11 sentences on average in the gold highlights of the training set. We conduct our ablation study on the validation set with full length ROUGE scores, but we report both fixed and full length ROUGE scores for the test set.

We experimented with two types of external information: title (TITLE) and image captions (CAPTION). In addition, we experimented with the first sentence (FS) of the document as external information. Note that the latter is not external information, it is a sentence in the document. However, we wanted to explore the idea that the first sentence of the document plays a crucial part in generating summaries (Rush et al., 2015; Nallapati et al., 2016). XNET with FS acts as a baseline for XNET with title and image captions.

We report the performance of several variants of XNET on the validation set in Table 1. We also compare them against the LEAD baseline and POINTERNET. These two systems do not use any additional information. Interestingly, all the variants of XNET significantly outperform LEAD and POINTERNET. When the title (TITLE), image captions (CAPTION) and the first sentence (FS) are used separately as additional information, XNET performs best with TITLE as its external information. Our result demonstrates the importance of the title of the document in extractive summarization (Edmundson, 1969; Kupiec et al., 1995; Mani, 2001). The performance with TITLE and CAPTION is better than that with FS. We also tried possible combinations of TITLE, CAPTION and FS. All XNET models are superior to the ones without any external information. XNET performs best when TITLE and CAPTION are jointly used as external information (55.4%, 21.8%, 11.8%, 7.5%, and 49.2% for R1, R2, R3, R4, and RL respectively). It is better than the the LEAD baseline by 3.7 points on average and than POINTERNET by 1.8 points on average, indicating that external information is useful to identify the gist of the document. We use this model for testing purposes.

Our final results on the test set are shown in Table 2. It turns out that for smaller summaries (75 bytes) LEAD and POINTERNET are superior to XNET. This result could be because LEAD (always) and POINTERNET (often) include the first sentence in their summaries, whereas, XNET is better capable at selecting sentences from various document positions. This is not captured by smaller summaries of 75 bytes, but it becomes more evident with longer summaries (275 bytes and full length) where XNET performs best across all ROUGE scores. We note that POINTERNET outperforms LEAD for 75-byte summaries, then its performance drops behind LEAD for 275-byte summaries, but then it outperforms LEAD for full length summaries on the metrics R1, R2 and RL. It shows that POINTERNET with its attention over sentences in the document is capable of exploring more than first few sentences in the document, but it is still behind XNET which is better at identifying salient sentences in the document. XNET performs significantly better than POINTERNET by 0.8 points for 275-byte summaries and by 1.9 points for full length summaries, on average for all ROUGE scores.

MODELS	R1	R2	R3	R4	RL
Fixed length: 75b					
LEAD	20.1	7.1	**3.5**	2.1	14.6
POINTERNET	**20.3**	**7.2**	**3.5**	**2.2**	**14.8**
XNET	20.2	7.1	3.4	2.0	14.6
Fixed length: 275b					
LEAD	39.1	14.5	7.6	4.7	34.6
POINTERNET	38.6	13.9	7.3	4.4	34.3
XNET	**39.7**	**14.7**	**7.9**	**5.0**	**35.2**
Full length summaries					
LEAD	49.3	19.5	10.7	6.9	43.8
POINTERNET	51.7	19.7	10.6	6.6	45.7
XNET	**54.2**	**21.6**	**12.0**	**7.9**	**48.1**

Table 2: Final results on the test set. POINTERNET is the sentence extraction system of Cheng and Lapata. XNET is our best model from Table 1. Best ROUGE score in each block and each column is highlighted in boldface.

Models	1st	2nd	3rd	4th
LEAD	0.15	0.17	**0.47**	0.21
POINTERNET	0.16	0.05	0.31	**0.48**
XNET	0.28	**0.53**	0.15	0.04
HUMAN	**0.41**	0.25	0.07	0.27

Table 3: Human evaluations: Ranking of various systems. Rank 1st is best and rank 4th, worst. Numbers show the percentage of times a system gets ranked at a certain position.

Human Evaluation We complement our automatic evaluation results with human evaluation. We randomly selected 20 articles from the test set.

Annotators were presented with a news article and summaries from four different systems. These include the LEAD baseline, POINTERNET, XNET and the human authored highlights. We followed the guidelines in Cheng and Lapata (2016), and asked our participants to rank the summaries from best (1st) to worst (4th) in order of informativeness (does the summary capture important information in the article?) and fluency (is the summary written in well-formed English?). We did not allow any ties and we only sampled articles with non-identical summaries. We assigned this task to five annotators who were proficient English speakers. Each annotator was presented with all 20 articles. The order of summaries to rank was randomized per article. An example of summaries our subjects ranked is provided in the supplementary material.

The results of our human evaluation study are shown in Table 3. As one might imagine, HUMAN gets ranked 1st most of the time (41%). However, it is closely followed by XNET which ranked 1st 28% of the time. In comparison, POINTERNET and LEAD were mostly ranked at 3rd and 4th places. We also carried out pairwise comparisons between all models in Table 3 for their statistical significance using a one-way ANOVA with post-hoc Tukey HSD tests with ($p < 0.01$). It showed that XNET is significantly better than LEAD and POINTERNET, and it does not differ significantly from HUMAN. On the other hand, POINTERNET does not differ significantly from LEAD and it differs significantly from both XNET and HUMAN. The human evaluation results corroborates our empirical results in Table 1 and Table 2: XNET is better than LEAD and POINTERNET in producing informative and fluent summaries.

4.2 Answer Selection

Question Answering Datasets We run experiments on four datasets collected for open domain question-answering tasks: WikiQA (Yang et al., 2015), SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), and MSMarco (Nguyen et al., 2016).

NewsQA was especially designed to present lexical and syntactic divergence between questions and answers. It contains 119,633 questions posed by crowdworkers on 12,744 CNN articles previously collected by Hermann et al. (2015). In a similar manner, SQuAD associates 100,000+ question with a Wikipedia article's first paragraph, for 500+ previously chosen articles. WikiQA was collected by mining web-searching query logs and then associating them with the summary section of the Wikipedia article presumed to be related to the topic of the query. A similar collection procedure was followed to create MSMarco with the difference that each candidate answer is a whole paragraph from a different browsed website associated with the query.

We follow the widely used setup of leaving out unanswered questions (Trischler et al., 2016; Yang et al., 2015) and adapt the format of each dataset to our task of answer sentence selection by labeling a candidate sentence with 1 if any answer span is contained in that sentence. In the case of MSMarco, each candidate paragraph comes associated with a label, hence we treat each one as a single long sentence. Since SQuAD keeps the official test dataset hidden and MSMarco does not provide labels for its released test set, we report results on their official validation sets. For validation, we set apart 10% of each official training set.

Our dataset splits consist of 92,525, 5,165 and 5,124 samples for NewsQA; 79,032, 8,567, and 10,570 for SQuAD; 873, 122, and 237 for WikiQA; and 79,704, 9,706, and 9,650 for MSMarco, for training, validation, and testing respectively.

Comparison Systems We compared the output of our model against the ISF (Trischler et al., 2016) and LOCALISF baselines. Given an article, the sentence with the highest ISF score is selected as an answer for the ISF baseline and the sentence with the highest local ISF score for the LOCALISF baseline. We also compare our model against a neural network (PAIRCNN) that encodes (question, candidate) in an isolated manner as in previous work (Yin et al., 2016; dos Santos et al., 2016; Wang et al., 2016). The architecture uses the sentence encoder explained in earlier sections to learn the question and candidate representations. The distribution over labels is given by $p(y_t|q) = p(y_t|s_t, q) = \text{softmax}(g(s_t, q))$ where $g(s_t, q) = \text{ReLU}(W_{sq} \cdot [s_t; q] + b_{sq})$. In addition, we also compare our model against AP-CNN (dos Santos et al., 2016), ABCNN (Yin et al., 2016), L.D.C (Wang and Jiang, 2017), KV-MemNN (Miller et al., 2016), and COMPAGGR, a state-of-the-art system by Wang et al. (2017).

We experiment with several variants of our model. XNET is the vanilla version of our sen-

	SQuAD			WikiQA			NewsQA			MSMarco		
	ACC	MAP	MRR	ACC	MAP	MRR	ACC	MAP	MRR	ACC	MAP	MRR
WRD CNT	77.84	27.50	27.77	51.05	48.91	49.24	44.67	46.48	46.91	20.16	19.37	19.51
WGT WRD CNT	78.43	28.10	28.38	49.79	50.99	51.32	45.24	48.20	48.64	20.50	20.06	20.23
AP-CNN	-	-	-	-	68.86	69.57	-	-	-	-	-	-
ABCNN	-	-	-	-	69.21	71.08	-	-	-	-	-	-
L.D.C	-	-	-	-	70.58	72.26	-	-	-	-	-	-
KV-MemNN	-	-	-	-	70.69	72.65	-	-	-	-	-	-
LOCALISF	79.50	27.78	28.05	49.79	49.57	50.11	44.69	48.40	46.48	20.21	20.22	20.39
ISF	78.85	28.09	28.36	48.52	46.53	46.72	45.61	48.57	48.99	20.52	20.07	20.23
PAIRCNN	32.53	46.34	46.35	32.49	39.87	38.71	25.67	40.16	39.89	14.92	34.62	35.14
COMPAGGR	85.52	91.05	91.05	60.76	73.12	74.06	54.54	67.63	68.21	32.05	**52.82**	**53.43**
XNET	35.50	58.46	58.84	54.43	69.12	70.22	26.18	42.28	42.43	15.45	35.42	35.97
XNETTOPK	36.09	59.70	59.32	55.00	68.66	70.24	29.41	46.69	46.97	17.04	37.60	38.16
LRXNET	**85.63**	**91.10**	**91.85**	**63.29**	**76.57**	**75.10**	**55.17**	**68.92**	**68.43**	**32.92**	31.15	30.41
XNET+	79.39	87.32	88.00	57.08	70.25	71.28	47.23	61.81	61.42	23.07	42.88	43.42

Table 4: Results (in percentage) for answer selection comparing our approaches (bottom part) to baselines (top): AP-CNN (dos Santos et al., 2016), ABCNN (Yin et al., 2016), L.D.C (Wang and Jiang, 2017), KV-MemNN (Miller et al., 2016), and COMPAGGR, a state-of-the-art system by Wang et al. (2017). (WGT) WRD CNT stands for the (weighted) word count baseline. See text for more details.

tence extractor conditioned only on the query q as external information (Eq. (3)). XNET+ is an extension of XNET which uses ISF, IDF and local ISF scores in addition to the query q as external information (Eqn. (4)). We also experimented with a baseline XNETTOPK where we choose the top k sentences with highest ISF score, and then among them choose the one with the highest probability according to XNET. In our experiments, we set $k = 5$. In the end, we experimented with an ensemble network LRXNET which combines the XNET score, the COMPAGGR score and other word-overlap-based scores (tweaked and optimized for each dataset separately) for each sentence using a logistic regression classifier. It uses ISF and LocalISF scores for NewsQA, IDF and ISF scores for SQuAD, sentence length, IDF and ISF scores for WikiQA, and word overlap and ISF score for MSMarco. We refer the reader to the supplementary material for more implementation and optimization details to replicate our results.

Evaluation Metrics We consider metrics that evaluate systems that return a ranked list of candidate answers: mean average precision (MAP), mean reciprocal rank (MRR), and accuracy (ACC).

Results Table 4 gives the results for the test sets of NewsQA and WikiQA, and the original validation sets of SQuAD and MSMarco. Our first observation is that XNET outperforms PAIRCNN, supporting our claim that it is beneficial to read the whole document in order to make decisions, instead of only observing each candidate in isolation.

Secondly, we can observe that ISF is indeed a strong baseline that outperforms XNET. This means that just "reading" the document using a vanilla version of XNET is not sufficient, and help is required through a coarse filtering. Indeed, we observe that XNET+ outperforms all baselines except for COMPAGGR. Our ensemble model LRXNET can ultimately surpass COMPAGGR on majority of the datasets.

This consistent behavior validates the machine reading capabilities and the improved document representation with external features of our model for answer selection. Specifically, the combination of document reading and word overlap features is required to be done in a soft manner, using a classification technique. Using it as a hard constraint, with XNETTOPK, does not achieve the best result. We believe that often the ISF score is a better indicator of answer presence in the vicinity of certain candidate instead of in the candidate itself. As such, XNET+ is capable of using this feature in datasets with richer context.

It is worth noting that the improvement gained by LRXNET over the state-of-the-art follows a pattern. For the SQuAD dataset, the results are comparable (less than 1%). However, the improvement for WikiQA reaches ~3% and then the gap shrinks again for NewsQA, with an improvement of ~1%. This could be explained by the fact that each sample of the SQuAD is a paragraph, compared to an article summary for WikiQA, and to an entire article for NewsQA. Hence, we further strengthen our hypothesis that a richer context is needed to achieve better results, in this case expressed as document length, but as the length of the context increases the limitation of sequential models to learn from long rich sequences arises.[7]

Interestingly, our model lags behind COMPAGGR on the MSMarco dataset. It turns out this is due to contextual independence between candidates in the MSMarco dataset, i.e., each candidate is a stand-alone paragraph in this dataset, in contrast to contextually dependent candidate sentences from a document in the NewsQA, SQuAD and WikiQA datasets. As a result, our models (XNET+ and LRXNET) with document reading abilities perform poorly. This can be observed by the fact that XNET and PAIRCNN obtain comparable results. COMPAGGR performs better because comparing each candidate independently is a better strategy.

5 Conclusion

We describe an approach to model documents while incorporating external information that informs the representations learned for the sentences in the document. We implement our approach through an attention mechanism of a neural network architecture for modeling documents.

Our experiments with extractive document summarization and answer selection tasks validates our model in two ways: first, we demonstrate that external information is important to guide document modeling for natural language understanding tasks. Our model uses image captions and the title of the document for document summarization, and the query with word overlap features for answer selection and outperforms its counterparts that do not use this information. Second, our external attention mechanism successfully guides the learning of the document representation for the relevant end goal. For answer selection, we show that inserting the query with word overlap features using our external attention mechanism outperforms state-of-the-art systems that naturally also have access to this information.

Acknowledgments

We thank Jianpeng Cheng for providing us with the CNN dataset and the implementation of PointerNet. We also thank the members of the Edinburgh NLP group for participating in our human evaluation experiments. This work greatly benefitted from discussions with Jianpeng Cheng, Annie Louis, Pedro Balage, Alfonso Mendes, Sebastião Miranda, and members of the Edinburgh NLP group. We gratefully acknowledge the support of the European Research Council (Lapata; award number 681760), the European Union under the Horizon 2020 SUMMA project (Narayan, Cohen; grant agreement 688139), and Huawei Technologies (Cohen).

[7]See the supplementary material for an example supporting our hypothesis.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, California, USA.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. New York, USA, pages 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 484–494.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing*. Doha, Qatar, pages 1724–1734.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR* abs/1602.03609.

Harold P. Edmundson. 1969. New methods in automatic extracting. *Journal of the Association for Computing Machinery* 16(2):264–285.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 360–368.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. *CoRR* abs/1602.06291.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*. pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR* abs/1705.02798.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *CoRR* abs/1511.03962.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. Phoenix, Arizona USA, pages 2741–2749.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Seattle, Washington, USA, pages 406–407.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Beijing, China, pages 1106–1115.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using N-gram cooccurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Edmonton, Canada, pages 71–78.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods on Natural Language Processing*. Lisbon, Portugal, pages 899–907.

Inderjeet Mani. 2001. *Automatic Summarization*. Natural language processing. John Benjamins Publishing Company.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proceedings of the Spoken Language Technology Workshop*. IEEE, pages 234–239.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods on Natural Language Processing*. Austin, Texas, pages 1400–1409.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. San Francisco, California USA, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany, pages 280–290.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. New Orleans, US.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS Marco: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches, co-located with the 30th Annual Conference on Neural Information Processing Systems*. Barcelona, Spain.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 2383–2392.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, pages 1073–1083.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. pages 3104–3112.

Jiwei Tan and Xiaojun Wan. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, pages 1171–1181.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2016. Inter-document contextual language model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 762–766.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR* abs/1611.09830.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of the 5th International Conference on Learning Representations*. Toulon, France.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 1319–1329.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, pages 189–198.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1340–1349.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning*. Vancouver, Canada, pages 271–280.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 2013–2018.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 1480–1489.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4:259–272.

Neural Models for Documents with Metadata

Dallas Card[1] **Chenhao Tan**[2] **Noah A. Smith**[3]
[1]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, 15213, USA
[2]Department of Computer Science, University of Colorado, Boulder, CO, 80309, USA
[3]Paul G. Allen School of CSE, University of Washington, Seattle, WA, 98195, USA
dcard@cmu.edu chenhao.tan@colorado.edu
nasmith@cs.washington.edu

Abstract

Most real-world document collections involve various types of metadata, such as author, source, and date, and yet the most commonly-used approaches to modeling text corpora ignore this information. While specialized models have been developed for particular applications, few are widely used in practice, as customization typically requires derivation of a custom inference algorithm. In this paper, we build on recent advances in variational inference methods and propose a general neural framework, based on topic models, to enable flexible incorporation of metadata and allow for rapid exploration of alternative models. Our approach achieves strong performance, with a manageable tradeoff between perplexity, coherence, and sparsity. Finally, we demonstrate the potential of our framework through an exploration of a corpus of articles about US immigration.

1 Introduction

Topic models comprise a family of methods for uncovering latent structure in text corpora, and are widely used tools in the digital humanities, political science, and other related fields (Boyd-Graber et al., 2017). Latent Dirichlet allocation (LDA; Blei et al., 2003) is often used when there is no prior knowledge about a corpus. In the real world, however, most documents have non-textual attributes such as author (Rosen-Zvi et al., 2004), timestamp (Blei and Lafferty, 2006), rating (McAuliffe and Blei, 2008), or ideology (Eisenstein et al., 2011; Nguyen et al., 2015b), which we refer to as **metadata**.

Many customizations of LDA have been developed to incorporate document metadata. Two models of note are **supervised LDA** (SLDA; McAuliffe and Blei, 2008), which jointly models words and labels (e.g., ratings) as being generated from a latent representation, and **sparse additive generative** models (SAGE; Eisenstein et al., 2011), which assumes that observed covariates (e.g., author ideology) have a sparse effect on the relative probabilities of words given topics. The structural topic model (STM; Roberts et al., 2014), which adds correlations between topics to SAGE, is also widely used, but like SAGE it is limited in the types of metadata it can efficiently make use of, and how that metadata is used. Note that in this work we will distinguish **labels** (metadata that are generated jointly with words from latent topic representations) from **covariates** (observed metadata that influence the distribution of labels and words).

The ability to create variations of LDA such as those listed above has been limited by the expertise needed to develop custom inference algorithms for each model. As a result, it is rare to see such variations being widely used in practice. In this work, we take advantage of recent advances in variational methods (Kingma and Welling, 2014; Rezende et al., 2014; Miao et al., 2016; Srivastava and Sutton, 2017) to facilitate approximate Bayesian inference *without requiring model-specific derivations*, and propose a general neural framework for topic models with metadata, SCHOLAR.[1]

SCHOLAR combines the abilities of SAGE and SLDA, and allows for easy exploration of the following options for customization:

1. Covariates: as in SAGE and STM, we incorporate explicit deviations for observed covariates, as well as effects for *interactions* with topics.
2. Supervision: as in SLDA, we can use metadata as labels to help infer topics that are relevant in predicting those labels.

[1]**S**parse **C**ontextual **H**idden and **O**bserved **L**anguage Autoencode**R**.

3. Rich encoder network: we use the encoding network of a variational autoencoder (VAE) to incorporate additional prior knowledge in the form of word embeddings, and/or to provide interpretable embeddings of covariates.
4. Sparsity: as in SAGE, a sparsity-inducing prior can be used to encourage more interpretable topics, represented as sparse deviations from a background log-frequency.

We begin with the necessary background and motivation (§2), and then describe our basic framework and its extensions (§3), followed by a series of experiments (§4). In an unsupervised setting, we can customize the model to trade off between perplexity, coherence, and sparsity, with improved coherence through the introduction of word vectors. Alternatively, by incorporating metadata we can either learn topics that are more predictive of labels than SLDA, or learn explicit deviations for particular parts of the metadata. Finally, by combining all parts of our model we can meaningfully incorporate metadata in multiple ways, which we demonstrate through an exploration of a corpus of news articles about US immigration.

In presenting this particular model, we emphasize not only its ability to adapt to the characteristics of the data, but the extent to which the VAE approach to inference provides a powerful framework for latent variable modeling that suggests the possibility of many further extensions. Our implementation is available at `https://github.com/dallascard/scholar`.

2 Background and Motivation

LDA can be understood as a non-negative Bayesian matrix factorization model: the observed document-word frequency matrix, $\mathbf{X} \in \mathbb{Z}^{D \times V}$ (D is the number of documents, V is the vocabulary size) is factored into two low-rank matrices, $\mathbf{\Theta}^{D \times K}$ and $\mathbf{B}^{K \times V}$, where each row of $\mathbf{\Theta}$, $\boldsymbol{\theta}_i \in \Delta^K$ is a latent variable representing a distribution over topics in document i, and each row of $\mathbf{B}$, $\boldsymbol{\beta}_k \in \Delta^V$, represents a single topic, i.e., a distribution over words in the vocabulary.[2] While it is possible to factor the count data into unconstrained matrices, the particular priors assumed by LDA are important for interpretability (Wallach et al., 2009). For example, the neural variational document model (NVDM; Miao et al., 2016) allows $\boldsymbol{\theta}_i \in \mathbb{R}^K$ and achieves normalization by taking the softmax of $\boldsymbol{\theta}_i^\top \mathbf{B}$. However, the experiments in Srivastava and Sutton (2017) found the performance of the NVDM to be slightly worse than LDA in terms of perplexity, and dramatically worse in terms of topic coherence.

The topics discovered by LDA tend to be parsimonious and coherent groupings of words which are readily identifiable to humans as being related to each other (Chang et al., 2009), and the resulting mode of the matrix $\mathbf{\Theta}$ provides a representation of each document which can be treated as a measurement for downstream tasks, such as classification or answering social scientific questions (Wallach, 2016). LDA does not require — and cannot make use of — additional prior knowledge. As such, the topics that are discovered may bear little connection to metadata of a corpus that is of interest to a researcher, such as sentiment, ideology, or time.

In this paper, we take inspiration from two models which have sought to alleviate this problem. The first, supervised LDA (SLDA; McAuliffe and Blei, 2008), assumes that documents have labels y which are generated conditional on the corresponding latent representation, i.e., $y_i \sim p(y \mid \boldsymbol{\theta}_i)$.[3] By incorporating labels into the model, it is forced to learn topics which allow documents to be represented in a way that is useful for the classification task. Such models can be used inductively as text classifiers (Balasubramanyan et al., 2012).

SAGE (Eisenstein et al., 2011), by contrast, is an exponential-family model, where the key innovation was to replace topics with sparse deviations from the background log-frequency of words ($\boldsymbol{d}$), i.e., $p(\text{word} \mid \text{softmax}(\boldsymbol{d} + \boldsymbol{\theta}_i^\top \mathbf{B}))$. SAGE can also incorporate deviations for observed covariates, as well as interactions between topics and covariates, by including additional terms inside the softmax. In principle, this allows for inferring, for example, the effect on an author's ideology on their choice of words, as well as ideological variations on each underlying topic. Unlike the NVDM, SAGE still constrains $\boldsymbol{\theta}_i$ to lie on the simplex, as in LDA.

SLDA and SAGE provide two different ways that users might wish to incorporate prior knowl-

[2] $\mathbb{Z}$ denotes nonnegative integers, and Δ^K denotes the set of K-length nonnegative vectors that sum to one. For a proper probabilistic interpretation, the matrix to be factored is actually the matrix of latent mean parameters of the assumed data generating process, $\mathbf{X}_{ij} \sim \text{Poisson}(\mathbf{\Lambda}_{ij})$. See Cemgil (2009) or Paisley et al. (2014) for details.

[3] Technically, the model conditions on the mean of the per-word latent variables, but we elide this detail in the interest of concision.

edge as a way of guiding the discovery of topics in a corpus: SLDA incorporates labels through a distribution conditional on topics; SAGE includes explicit sparse deviations for each unique value of a covariate, in addition to topics.[4]

Because of the Dirichlet-multinomial conjugacy in the original model, efficient inference algorithms exist for LDA. Each variation of LDA, however, has required the derivation of a custom inference algorithm, which is a time-consuming and error-prone process. In SLDA, for example, each type of distribution we might assume for $p(y \mid \boldsymbol{\theta})$ would require a modification of the inference algorithm. SAGE breaks conjugacy, and as such, the authors adopted L-BFGS for optimizing the variational bound. Moreover, in order to maintain computational efficiency, it assumed that covariates were limited to a single categorical label.

More recently, the variational autoencoder (VAE) was introduced as a way to perform approximate posterior inference on models with otherwise intractable posteriors (Kingma and Welling, 2014; Rezende et al., 2014). This approach has previously been applied to models of text by Miao et al. (2016) and Srivastava and Sutton (2017). We build on their work and show how this framework can be adapted to seamlessly incorporate the ideas of both SAGE and SLDA, while allowing for greater flexibility in the use of metadata. Moreover, by exploiting automatic differentiation, we allow for modification of the model without requiring any change to the inference procedure. The result is not only a highly adaptable family of models with scalable inference and efficient prediction; it also points the way to incorporation of many ideas found in the literature, such as a gradual evolution of topics (Blei and Lafferty, 2006), and hierarchical models (Blei et al., 2010; Nguyen et al., 2013, 2015b).

3 SCHOLAR: A Neural Topic Model with Covariates, Supervision, and Sparsity

We begin by presenting the generative story for our model, and explain how it generalizes both SLDA and SAGE (§3.1). We then provide a general explanation of inference using VAEs and how it applies to our model (§3.2), as well as how to infer document representations and predict labels at test time (§3.3). Finally, we discuss how we can incorporate additional prior knowledge (§3.4).

3.1 Generative Story

Consider a corpus of D documents, where document i is a list of N_i words, $\boldsymbol{w}_i$, with V words in the vocabulary. For each document, we may have observed covariates $\boldsymbol{c}_i$ (e.g., year of publication), and/or one or more labels, $\boldsymbol{y}_i$ (e.g., sentiment).

Our model builds on the generative story of LDA, but optionally incorporates labels and covariates, and replaces the matrix product of $\boldsymbol{\Theta}$ and $\mathbf{B}$ with a more flexible generative network, f_g, followed by a softmax transform. Instead of using a Dirichlet prior as in LDA, we employ a logistic normal prior on $\boldsymbol{\theta}$ as in Srivastava and Sutton (2017) to facilitate inference (§3.2): we draw a latent variable, $\boldsymbol{r}$,[5] from a multivariate normal, and transform it to lie on the simplex using a softmax transform.[6]

The generative story is shown in Figure 1a and described in equations below:

```
For each document $i$ of length $N_i$:
    # Draw a latent representation on the simplex from a logistic normal prior:
    $\boldsymbol{r}_i \sim \mathcal{N}(\boldsymbol{r} \mid \boldsymbol{\mu}_0(\alpha), \text{diag}(\boldsymbol{\sigma}_0^2(\alpha)))$
    $\boldsymbol{\theta}_i = \text{softmax}(\boldsymbol{r}_i)$
    # Generate words, incorporating covariates:
    $\boldsymbol{\eta}_i = f_g(\boldsymbol{\theta}_i, \boldsymbol{c}_i)$
    For each word $j$ in document $i$:
        $w_{ij} \sim p(w \mid \text{softmax}(\boldsymbol{\eta}_i))$
    # Similarly generate labels:
    $\boldsymbol{y}_i \sim p(\boldsymbol{y} \mid f_y(\boldsymbol{\theta}_i, \boldsymbol{c}_i))$,
```

where $p(w \mid \text{softmax}(\boldsymbol{\eta}_i))$ is a multinomial distribution and $p(\boldsymbol{y} \mid f_y(\boldsymbol{\theta}_i, \boldsymbol{c}_i))$ is a distribution appropriate to the data (e.g., multinomial for categorical labels). f_g is a model-specific combination of latent variables and covariates, f_y is a multi-layer neural network, and $\boldsymbol{\mu}_0(\alpha)$ and $\boldsymbol{\sigma}_0^2(\alpha)$ are the mean and diagonal covariance terms of a multivariate normal prior. To approximate a symmetric Dirichlet

[4]A third way of incorporating metadata is the approach used by various "upstream" models, such as Dirichlet-multinomial regression (Mimno and McCallum, 2008), which uses observed metadata to inform the document prior. We hypothesize that this approach could be productively combined with our framework, but we leave this as future work.

[5]$\boldsymbol{r}$ is equivalent to $\boldsymbol{z}$ in the original VAE. To avoid confusion with topic assignment of words in the topic modeling literature, we use $\boldsymbol{r}$ instead of $\boldsymbol{z}$.

[6]Unlike the correlated topic model (CTM; Lafferty and Blei, 2006), which also uses a logistic-normal prior, we fix the parameters of the prior and use a diagonal covariance matrix, rather than trying to infer correlations among topics. However, it would be a straightforward extension of our framework to place a richer prior on the latent document representations, and learn correlations by updating the parameters of this prior after each epoch, analogously to the variational EM approach used for the CTM.

prior with hyperparameter α, these are given by the Laplace approximation (Hennig et al., 2012) to be $\mu_{0,k}(\alpha) = 0$ and $\sigma^2_{0,k} = (K-1)/(\alpha K)$.

If we were to ignore covariates, place a Dirichlet prior on $\mathbf{B}$, and let $\boldsymbol{\eta} = \boldsymbol{\theta}_i^\top \mathbf{B}$, this model is equivalent to SLDA with a logistic normal prior. Similarly, we can recover a model that is like SAGE, but lacks sparsity, if we ignore labels, and let

$$\boldsymbol{\eta}_i = \boldsymbol{d} + \boldsymbol{\theta}_i^\top \mathbf{B} + \boldsymbol{c}_i^\top \mathbf{B}^{cov} + (\boldsymbol{\theta}_i \otimes \boldsymbol{c}_i)^\top \mathbf{B}^{int}, \quad (1)$$

where $\boldsymbol{d}$ is the V-dimensional background term (representing the log of the overall word frequency), $\boldsymbol{\theta}_i \otimes \boldsymbol{c}_i$ is a vector of interactions between topics and covariates, and $\mathbf{B}^{cov}$ and $\mathbf{B}^{int}$ are additional weight (deviation) matrices. The background is included to account for common words with approximately the same frequency across documents, meaning that the $\mathbf{B}^*$ weights now represent both positive and negative deviations from this background. This is the form of f_g which we will use in our experiments.

To recover the full SAGE model, we can place a sparsity-inducing prior on each $\mathbf{B}^*$. As in Eisenstein et al. (2011), we make use of the compound normal-exponential prior for each element of the weight matrices, $\mathbf{B}^*_{m,n}$, with hyperparameter γ,[7]

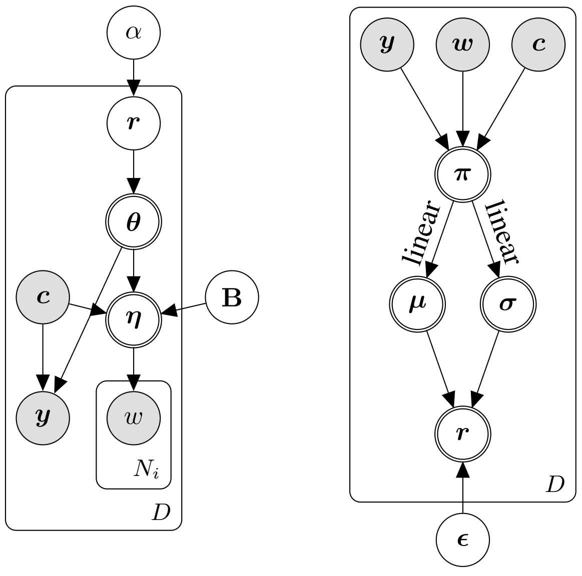
$$\tau_{m,n} \sim \text{Exponential}(\gamma), \quad (2)$$
$$\mathbf{B}^*_{m,n} \sim \mathcal{N}(0, \tau_{m,n}). \quad (3)$$

We can choose to ignore various parts of this model, if, for example, we don't have any labels or observed covariates, or we don't wish to use interactions or sparsity.[8] Other generator networks could also be considered, with additional layers to represent more complex interactions, although this might involve some loss of interpretability.

In the absence of metadata, and without sparsity, our model is equivalent to the ProdLDA model of Srivastava and Sutton (2017) with an explicit background term, and ProdLDA is, in turn, a special case of SAGE, without background log-frequencies, sparsity, covariates, or labels. In the next section we generalize the inference method used for ProdLDA; in our experiments we validate its performance and explore the effects of regularization and word-vector initialization (§3.4). The NVDM (Miao et al., 2016) uses the same approach to inference, but does not not restrict document representations to the simplex.

[7]To avoid having to tune γ, we employ an improper Jeffery's prior, $p(\tau_{m,n}) \propto 1/\tau_{m,n}$, as in SAGE. Although this causes difficulties in posterior inference for the variance terms, τ, in practice, we resort to a variational EM approach, with MAP-estimation for the weights, $\mathbf{B}$, and thus alternate between computing expectations of the τ parameters, and updating all other parameters using some variant of stochastic gradient descent. For this, we only require the expectation of each τ_{mn} for each E-step, which is given by $1/\mathbf{B}^2_{m,n}$. We refer the reader to Eisenstein et al. (2011) for additional details.

[8]We could also ignore latent topics, in which case we would get a naïve Bayes-like model of text with deviations for each covariate $p(\boldsymbol{w}_{ij} \mid \boldsymbol{c}_i) \propto \exp(\boldsymbol{d} + \boldsymbol{c}_i^\top \mathbf{B}^{cov})$.



(a) Generative model (b) Inference model

Figure 1: Figure 1a presents the generative story of our model. Figure 1b illustrates the inference network using the reparametrization trick to perform variational inference on our model. Shaded nodes are observed; double circles indicate deterministic transformations of parent nodes.

3.2 Learning and Inference

As in past work, each document i is assumed to have a latent representation $\boldsymbol{r}_i$, which can be interpreted as its relative membership in each topic (after exponentiating and normalizing). In order to infer an approximate posterior distribution over $\boldsymbol{r}_i$, we adopt the sampling-based VAE framework developed in previous work (Kingma and Welling, 2014; Rezende et al., 2014).

As in conventional variational inference, we assume a variational approximation to the posterior, $q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i \mid \boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i)$, and seek to minimize the KL divergence between it and the true posterior, $p(\boldsymbol{r}_i \mid \boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i)$, where $\boldsymbol{\Phi}$ is the set of variational parameters to be defined below. After some manipulations (given in supplementary materials), we obtain the evidence lower bound (ELBO) for a single document,

$$\mathcal{L}(\boldsymbol{w}_i) = \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i|\boldsymbol{w}_i,\boldsymbol{c}_i,\boldsymbol{y}_i)}\left[\sum_{j=1}^{N_i}\log p(w_{ij} \mid \boldsymbol{r}_i, \boldsymbol{c}_i)\right] + \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i|\boldsymbol{w}_i,\boldsymbol{c}_i,\boldsymbol{y}_i)}\left[\log p(\boldsymbol{y}_i \mid \boldsymbol{r}_i, \boldsymbol{c}_i)\right] - \mathrm{D}_{\mathrm{KL}}\left[q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i \mid \boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i) \,||\, p(\boldsymbol{r}_i \mid \alpha)\right]. \tag{4}$$

As in the original VAE, we will encode the parameters of our variational distributions using a shared multi-layer neural network. Because we have assumed a diagonal normal prior on $\boldsymbol{r}$, this will take the form of a network which outputs a mean vector, $\boldsymbol{\mu}_i = f_{\boldsymbol{\mu}}(\boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i)$ and diagonal of a covariance matrix, $\boldsymbol{\sigma}_i^2 = f_{\boldsymbol{\sigma}}(\boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i)$, such that $q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i \mid \boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$. Incorporating labels and covariates to the inference network used by Miao et al. (2016) and Srivastava and Sutton (2017), we use:

$$\boldsymbol{\pi}_i = f_e([\mathbf{W}_x \boldsymbol{x}_i; \mathbf{W}_c \boldsymbol{c}_i; \mathbf{W}_y \boldsymbol{y}_i]), \tag{5}$$
$$\boldsymbol{\mu}_i = \mathbf{W}_\mu \boldsymbol{\pi}_i + \boldsymbol{b}_\mu, \tag{6}$$
$$\log \boldsymbol{\sigma}_i^2 = \mathbf{W}_\sigma \boldsymbol{\pi}_i + \boldsymbol{b}_\sigma, \tag{7}$$

where $\boldsymbol{x}_i$ is a V-dimensional vector representing the counts of words in $\boldsymbol{w}_i$, and f_e is a multilayer perceptron. The full set of encoder parameters, $\boldsymbol{\Phi}$, thus includes the parameters of f_e and all weight matrices and bias vectors in Equations 5–7 (see Figure 1b).

This approach means that the expectations in Equation 4 are intractable, but we can approximate them using sampling. In order to maintain differentiability with respect to $\boldsymbol{\Phi}$, even after sampling, we make use of the reparameterization trick (Kingma and Welling, 2014),[9] which allows us to reparameterize samples from $q_{\boldsymbol{\Phi}}(\boldsymbol{r} \mid \boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i)$ in terms of samples from an independent source of noise, i.e.,

$$\boldsymbol{\epsilon}^{(s)} \sim \mathcal{N}(0, \mathbf{I}),$$
$$\boldsymbol{r}_i^{(s)} = g_{\boldsymbol{\Phi}}(\boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i, \boldsymbol{\epsilon}^{(s)}) = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \cdot \boldsymbol{\epsilon}^{(s)}.$$

We thus replace the bound in Equation 4 with a Monte Carlo approximation using a single sample[10] of $\boldsymbol{\epsilon}$ (and thereby of $\boldsymbol{r}$):

$$\mathcal{L}(\boldsymbol{w}_i) \approx \sum_{j=1}^{N_i}\log p(w_{ij} \mid \boldsymbol{r}_i^{(s)}, \boldsymbol{c}_i) + \log p(\boldsymbol{y}_i \mid \boldsymbol{r}_i^{(s)}, \boldsymbol{c}_i) - \mathrm{D}_{\mathrm{KL}}\left[q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i \mid \boldsymbol{w}_i, \boldsymbol{c}_i, \boldsymbol{y}_i) \,||\, p(\boldsymbol{r}_i \mid \alpha)\right]. \tag{8}$$

We can now optimize this sampling-based approximation of the variational bound with respect to $\boldsymbol{\Phi}$, $\mathbf{B}^*$, and all parameters of f_g and f_y using stochastic gradient descent. Moreover, because of this stochastic approach to inference, we are *not* restricted to covariates with a small number of unique values, which was a limitation of SAGE. Finally, the KL divergence term in Equation 8 can be computed in closed form (see supplementary materials).

3.3 Prediction on Held-out Data

In addition to inferring latent topics, our model can both infer latent representations for new documents and predict their labels, the latter of which was the motivation for SLDA. In traditional variational inference, inference at test time requires fixing global parameters (topics), and optimizing the per-document variational parameters for the test set. With the VAE framework, by contrast, the encoder network (Equations 5–7) can be used to directly estimate the posterior distribution for each test document, using only a forward pass (no iterative optimization or sampling).

If not using labels, we can use this approach directly, passing the word counts of new documents through the encoder to get a posterior $q_{\boldsymbol{\Phi}}(\boldsymbol{r}_i \mid \boldsymbol{w}_i, \boldsymbol{c}_i)$. When we also include labels to be predicted, we can first train a fully-observed model, as above, then fix the decoder, and retrain the encoder without labels. In practice, however, if we train the encoder network using one-hot encodings of document labels, it is sufficient to provide a vector of all zeros for the labels of test documents; this is what we adopt for our experiments (§4.2), and we still obtain good predictive performance.

The label network, f_y, is a flexible component which can be used to predict a wide range of outcomes, from categorical labels (such as star ratings; McAuliffe and Blei, 2008) to real-valued outputs (such as number of citations or box-office returns;

[9] The Dirichlet distribution cannot be directly reparameterized in this way, which is why we use the logistic normal prior on $\boldsymbol{\theta}$ to approximate the Dirichlet prior used in LDA.

[10] Alternatively, one can average over multiple samples.

Yogatama et al., 2011). For categorical labels, predictions are given by

$$\hat{y}_i = \underset{y \in \mathcal{Y}}{\text{argmax}}\; p(y \mid \boldsymbol{r}_i, \boldsymbol{c}_i). \tag{9}$$

Alternatively, when dealing with a small set of categorical labels, it is also possible to treat them as observed categorical covariates during training. At test time, we can then consider all possible one-hot vectors, $\boldsymbol{e}$, in place of $\boldsymbol{c}_i$, and predict the label that maximizes the probability of the words, i.e.,

$$\hat{y}_i = \underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{j=1}^{N_i} \log p(w_{ij} \mid \boldsymbol{r}_i, \boldsymbol{e}_y). \tag{10}$$

This approach works well in practice (as we show in §4.2), but does not scale to large numbers of labels, or other types of prediction problems, such as multi-class classification or regression.

The choice to include metadata as covariates, labels, or both, depends on the data. The key point is that we can incorporate metadata in two very different ways, depending on what we want from the model. Labels guide the model to infer topics that are relevant to those labels, whereas covariates induce explicit deviations, leaving the latent variables to account for the rest of the content.

3.4 Additional Prior Information

A final advantage of the VAE framework is that the encoder network provides a way to incorporate additional prior information in the form of word vectors. Although we can learn all parameters starting from a random initialization, it is also possible to initialize and fix the initial embeddings of words in the model, $\mathbf{W}_x$, in Equation 5. This leverages word similarities derived from large amounts of unlabeled data, and may promote greater coherence in inferred topics. The same could also be done for some covariates; for example, we could embed the source of a news article based on its place on the ideological spectrum. Conversely, if we choose to learn these parameters, the learned values ($\mathbf{W}_y$ and $\mathbf{W}_c$) may provide meaningful embeddings of these metadata (see section §4.3).

Other variants on topic models have also proposed incorporating word vectors, both as a parallel part of the generative process (Nguyen et al., 2015a), and as an alternative parameterization of topic distributions (Das et al., 2015), but inference is not scalable in either of these models. Because of the generality of the VAE framework, we could also modify the generative story so that word embeddings are emitted (rather than tokens); we leave this for future work.

4 Experiments and Results

To evaluate and demonstrate the potential of this model, we present a series of experiments below. We first test SCHOLAR without observed metadata, and explore the effects of using regularization and/or word vector initialization, compared to LDA, SAGE, and NVDM (§4.1). We then evaluate our model in terms of predictive performance, in comparison to SLDA and an l_2-regularized logistic regression baseline (§4.2). Finally, we demonstrate the ability to incorporate covariates and/or labels in an exploratory data analysis (§4.3).

The scores we report are generalization to held-out data, measured in terms of perplexity; coherence, measured in terms of non-negative point-wise mutual information (NPMI; Chang et al., 2009; Newman et al., 2010), and classification accuracy on test data. For coherence we evaluate NPMI using the top 10 words of each topic, both internally (using test data), and externally, using a decade of articles from the English Gigaword dataset (Graff and Cieri, 2003). Since our model employs variational methods, the reported perplexity is an upper bound based on the ELBO.

As datasets we use the familiar 20 newsgroups, the IMDB corpus of 50,000 movie reviews (Maas et al., 2011), and the UIUC Yahoo answers dataset with 150,000 documents in 15 categories (Chang et al., 2008). For further exploration, we also make use of a corpus of approximately 4,000 time-stamped news articles about US immigration, each annotated with pro- or anti-immigration tone (Card et al., 2015). We use the original author-provided implementations of SAGE[11] and SLDA,[12] while for LDA we use Mallet.[13]. Our implementation of SCHOLAR is in TensorFlow, but we have also provided a preliminary PyTorch implementation of the core of our model.[14] For additional details about datasets and implementation, please refer to the supplementary material.

It is challenging to fairly evaluate the relative computational efficiency of our approach compared to past work (due to the stochastic nature of our ap-

[11]github.com/jacobeisenstein/SAGE
[12]github.com/blei-lab/class-slda
[13]mallet.cs.umass.edu
[14]github.com/dallascard/scholar

proach to inference, choices about hyperparameters such as tolerance, and because of differences in implementation). Nevertheless, in practice, the performance of our approach is highly appealing. For all experiments in this paper, our implementation was much faster than SLDA or SAGE (implemented in C and Matlab, respectively), and competitive with Mallet.

4.1 Unsupervised Evaluation

Although the emphasis of this work is on incorporating observed labels and/or covariates, we briefly report on experiments in the unsupervised setting. Recall that, without metadata, SCHOLAR equates to ProdLDA, but with an explicit background term.[15] We therefore use the same experimental setup as Srivastava and Sutton (2017) (learning rate, momentum, batch size, and number of epochs) and find the same general patterns as they reported (see Table 1 and supplementary material): our model returns more coherent topics than LDA, but at the cost of worse perplexity. SAGE, by contrast, attains very high levels of sparsity, but at the cost of worse perplexity and coherence than LDA. As expected, the NVDM produces relatively low perplexity, but very poor coherence, due to its lack of constraints on $\boldsymbol{\theta}$.

Further experimentation revealed that the VAE framework involves a tradeoff among the scores; running for more epochs tends to result in better perplexity on held-out data, but at the cost of worse coherence. Adding regularization to encourage sparse topics has a similar effect as in SAGE, leading to worse perplexity and coherence, but it does create sparse topics. Interestingly, initializing the encoder with pretrained word2vec embeddings, and *not* updating them returned a model with the best internal coherence of any model we considered for IMDB and Yahoo answers, and the second-best for 20 newsgroups.

The background term in our model does not have much effect on perplexity, but plays an important role in producing coherent topics; as in SAGE, the background can account for common words, so they are mostly absent among the most heavily weighted words in the topics. For instance, words like *film* and *movie* in the IMDB corpus are relatively unimportant in the topics learned by our model, but would be much more heavily weighted without the background term, as they are in topics learned by LDA.

Model	Ppl. ↓	NPMI (int.) ↑	NPMI (ext.) ↑	Sparsity ↑
LDA	**1508**	0.13	0.14	0
SAGE	1767	0.12	0.12	**0.79**
NVDM	1748	0.06	0.04	0
SCHOLAR − B.G.	1889	0.09	0.13	0
SCHOLAR	1905	0.14	0.13	0
SCHOLAR + W.V.	1991	**0.18**	**0.17**	0
SCHOLAR + REG.	2185	0.10	0.12	0.58

Table 1: Performance of our various models in an unsupervised setting (i.e., without labels or covariates) on the IMDB dataset using a 5,000-word vocabulary and 50 topics. The supplementary materials contain additional results for 20 newsgroups and Yahoo answers.

4.2 Text Classification

We next consider the utility of our model in the context of categorical labels, and consider them alternately as observed covariates and as labels generated conditional on the latent representation. We use the same setup as above, but tune number of training epochs for our model using a random 20% of training data as a development set, and similarly tune regularization for logistic regression.

Table 2 summarizes the accuracy of various models on three datasets, revealing that our model offers competitive performance, both as a joint model of words and labels (Eq. 9), and a model which conditions on covariates (Eq. 10). Although SCHOLAR is comparable to the logistic regression baseline, our purpose here is not to attain state-of-the-art performance on text classification. Rather, the high accuracies we obtain demonstrate that we are learning low-dimensional representations of documents that are relevant to the label of interest, outperforming SLDA, and have the same attractive properties as topic models. Further, any neural network that is successful for text classification could be incorporated into f_y and trained end-to-end along with topic discovery.

4.3 Exploratory Study

We demonstrate how our model might be used to explore an annotated corpus of articles about immigration, and adapt to different assumptions about the data. We only use a small number of topics in this part ($K = 8$) for compact presentation.

[15]Note, however, that a batchnorm layer in ProdLDA may play a similar role to a background term, and there are small differences in implementation; please see supplementary material for more discussion of this.

	20news	IMDB	Yahoo
Vocabulary size	2000	5000	5000
Number of topics	50	50	250
SLDA	0.60	0.64	0.65
SCHOLAR (labels)	0.67	0.86	0.73
SCHOLAR (covariates)	**0.71**	**0.87**	0.72
Logistic regression	0.70	**0.87**	**0.76**

Table 2: Accuracy of various models on three datasets with categorical labels.

Tone as a label. We first consider using the annotations as a *label*, and train a joint model to infer topics relevant to the tone of the article (pro- or anti-immigration). Figure 2 shows a set of topics learned in this way, along with the predicted probability of an article being pro-immigration conditioned on the given topic. All topics are coherent, and the predicted probabilities have strong face validity, e.g., “arrested charged charges agents operation” is least associated with pro-immigration.

Tone as a covariate. Next we consider using tone as a *covariate*, and build a model using both tone and tone-topic interactions. Table 3 shows a set of topics learned from the immigration data, along with the most highly-weighted words in the corresponding tone-topic interaction terms. As can be seen, these interaction terms tend to capture different frames (e.g., “criminal” vs. “detainees”, and “illegals” vs. “newcomers”, etc).

Combined model with temporal metadata. Finally, we incorporate both the tone annotations and the year of publication of each article, treating the former as a label and the latter as a covariate. In this model, we also include an embedding matrix, $\mathbf{W}_c$, to project the one-hot *year* vectors down to a two-dimensional continuous space, with a learned deviation for each dimension. We omit the topics in the interest of space, but Figure 3 shows the learned embedding for each year, along with the top terms of the corresponding deviations. As can be seen, the model learns that adjacent years tend to produce similar deviations, *even though we have not explicitly encoded this information*. The left-right dimension roughly tracks a temporal trend with positive deviations shifting from the years of *Clinton* and *INS* on the left, to *Obama* and *ICE* on the right.[16] Meanwhile, the events of 9/11 dominate the vertical direction, with the words *sept*,

[16]The Immigration and Naturalization Service (INS) was transformed into Immigration and Customs Enforcement (ICE) and other agencies in 2003.

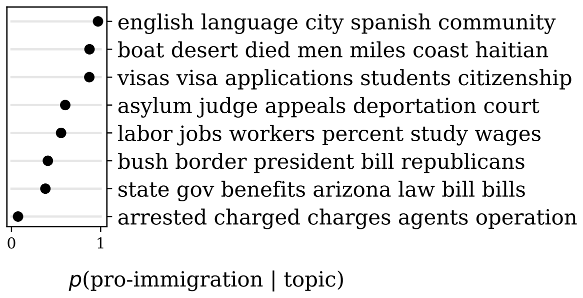


Figure 2: Topics inferred by a joint model of words and tone, and the corresponding probability of pro-immigration tone for each topic. A topic is represented by the top words sorted by word probability throughout the paper.

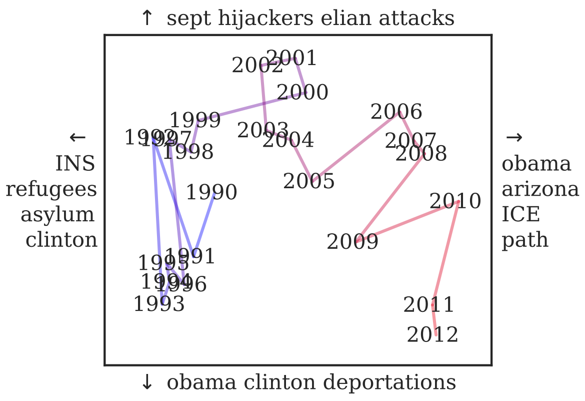


Figure 3: Learned embeddings of year-of-publication (treated as a covariate) from combined model of news articles about immigration.

hijackers, and *attacks* increasing in probability as we move up in the space. If we wanted to look at each year individually, we could drop the embedding of years, and learn a sparse set of topic-year interactions, similar to tone in Table 3.

5 Additional Related Work

The literature on topic models is vast; in addition to papers cited throughout, other efforts to incorporate metadata into topic models include Dirichlet-multinomial regression (DMR; Mimno and McCallum, 2008), Labeled LDA (Ramage et al., 2009), and MedLDA (Zhu et al., 2009). A recent paper also extended DMR by using deep neural networks to embed metadata into a richer document prior (Benton and Dredze, 2018).

A separate line of work has pursued parameterizing unsupervised models of documents using neural networks (Hinton and Salakhutdinov,

Base topics (each row is a topic)	Anti-immigration interactions	Pro-immigration interactions
ice customs agency enforcement homeland	**criminal** customs arrested	**detainees** detention center agency
population born percent americans english	jobs million **illegals** taxpayers	english **newcomers** hispanic city
judge case court guilty appeals attorney	guilty charges man charged	asylum court judge case appeals
patrol border miles coast desert boat guard	patrol border agents boat	died authorities desert border bodies
licenses drivers card visa cards applicants	foreign sept visas system	green citizenship card citizen apply
island story chinese ellis international	smuggling federal charges	island school ellis english story
guest worker workers bush labor bill	bill border house senate	workers tech skilled farm labor
benefits bill welfare republican state senate	republican california gov state	law welfare students tuition

Table 3: Top words for topics (left) and the corresponding anti-immigration (middle) and pro-immigration (right) variations when treating tone as a covariate, with interactions.

2009; Larochelle and Lauly, 2012), including non-Bayesian approaches (Cao et al., 2015). More recently, Lau et al. (2017) proposed a neural language model that incorporated topics, and He et al. (2017) developed a scalable alternative to the correlated topic model by simultaneously learning topic embeddings.

Others have attempted to extend the reparameterization trick to the Dirichlet and Gamma distributions, either through transformations (Kucukelbir et al., 2016) or a generalization of reparameterization (Ruiz et al., 2016). Black-box and VAE-style inference have been implemented in at least two general purpose tools designed to allow rapid exploration and evaluation of models (Kucukelbir et al., 2015; Tran et al., 2016).

6 Conclusion

We have presented a neural framework for generalized topic models to enable flexible incorporation of metadata with a variety of options. We take advantage of stochastic variational inference to develop a general algorithm for our framework such that variations do not require any model-specific algorithm derivations. Our model demonstrates the tradeoff between perplexity, coherence, and sparsity, and outperforms SLDA in predicting document labels. Furthermore, the flexibility of our model enables intriguing exploration of a text corpus on US immigration. We believe that our model and code will facilitate rapid exploration of document collections with metadata.

Acknowledgments

We would like to thank Charles Sutton, anonymous reviewers, and all members of Noah's ARK for helpful discussions and feedback. This work was made possible by a University of Washington Innovation award and computing resources provided by XSEDE.

References

Ramnath Balasubramanyan, William W. Cohen, Doug Pierce, and David P. Redlawsk. 2012. Modeling polarizing topics: When do different political communities respond differently to the same news? In *Proceedings of ICWSM*.

Adrian Benton and Mark Dredze. 2018. Deep Dirichlet multinomial regression. In *Proceedings of NAACL*.

David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. 2010. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *JACM*, 57(2).

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of ICML*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*, 3:993–1022.

Jordan Boyd-Graber, Yuening Hu, and David Mimno. 2017. Applications of topic models. *Foundations and Trends in Information Retrieval*, 11(2-3):143–296.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of AAAI*.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of ACL*.

Ali Taylan Cemgil. 2009. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, pages 4:1–4:17.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of NIPS*.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *Proceedings of AAAI*.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of ACL*.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of ICML.*

David Graff and C Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium.*

Junxian He, Zhiting Hu, Taylor Berg-Kirkpatrick, Ying Huang, and Eric P. Xing. 2017. Efficient correlated topic modeling with topic embedding. In *Proceedings of KDD.*

Philipp Hennig, David Stern, Ralf Herbrich, and Thore Graepel. 2012. Kernel topic models. In *Proceedings of AISTATS.*

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2009. Replicated softmax: An undirected topic model. In *Proceedings of NIPS.*

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of ICLR.*

Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David Blei. 2015. Automatic variational inference in stan. In *Proceedings of NIPS.*

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. 2016. Automatic differentiation variational inference. ArXiv:1603.00788.

John D. Lafferty and David M. Blei. 2006. Correlated topic models. In *Proceedings of NIPS.*

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Proceedings of NIPS.*

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *Proceedings of ACL.*

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL.*

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *Proceedings of NIPS.*

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of ICML.*

David Mimno and Andrew McCallum. 2008. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Proceedings of UAI.*

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of ACL.*

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015a. Improving topic models with latent feature word representations. In *Proceedings of ACL.*

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015b. Tea party in the house: A hierarchical ideal point topic model and its application to Republican legislators in the 112th congress. In *Proceedings of ACL.*

Viet-An Nguyen, Jordan L. Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Proceedings of NIPS.*

John William Paisley, David M. Blei, and Michael I. Jordan. 2014. Bayesian nonnegative matrix factorization with stochastic variational inference. In *Handbook of Mixed Membership Models and Their Applications*, pages 205–224.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of EMNLP.*

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML.*

Molly Roberts, Brandon Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Gadarian, Bethany Albertson, and David Rand. 2014. Structural topic models for open ended survey responses. *American Journal of Political Science*, 58:1064–1082.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of UAI.*

Francisco J R Ruiz, Michalis K Titsias, and David M Blei. 2016. The generalized reparameterization gradient. In *Proceedings of NIPS.*

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. In *Proceedings of ICLR.*

Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. 2016. Edward: A library for probabilistic modeling, inference, and criticism. ArXiv:1610.09787.

Hanna Wallach. 2016. Interpretability and measurement. EMNLP Workshop on Natural Language Processing and Computational Social Science.

Hanna Wallach, David M. Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of NIPS.*

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R Routledge, and Noah A Smith. 2011. Predicting a scientific community's response to an article. In *Proceedings of EMNLP.*

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2009. MedLDA: Maximum margin supervised topic models for regression and classification. In *Proceedings of ICML.*

NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing

Dinghan Shen[1*], **Qinliang Su**[2*], **Paidamoyo Chapfuwa**[1], **Wenlin Wang**[1], **Guoyin Wang**[1], **Lawrence Carin**[1], **Ricardo Henao**[1]

[1] Duke University [2] Sun Yat-sen University

dinghan.shen@duke.edu

Abstract

Semantic hashing has become a powerful paradigm for fast similarity search in many information retrieval systems. While fairly successful, previous techniques generally require two-stage training, and the binary constraints are handled *ad-hoc*. In this paper, we present an *end-to-end* Neural Architecture for Semantic Hashing (NASH), where the binary hashing codes are treated as *Bernoulli* latent variables. A neural variational inference framework is proposed for training, where gradients are directly backpropagated through the discrete latent variable to optimize the hash function. We also draw connections between proposed method and *rate-distortion theory*, which provides a theoretical foundation for the effectiveness of the proposed framework. Experimental results on three public datasets demonstrate that our method significantly outperforms several state-of-the-art models on both *unsupervised* and *supervised* scenarios.

1 Introduction

The problem of *similarity search*, also called *nearest-neighbor search*, consists of finding documents from a large collection of documents, or *corpus*, which are most similar to a query document of interest. Fast and accurate similarity search is at the core of many information retrieval applications, such as plagiarism analysis (Stein et al., 2007), collaborative filtering (Koren, 2008), content-based multimedia retrieval (Lew et al., 2006) and caching (Pandey et al., 2009). Semantic hashing is an effective approach for fast similarity search (Salakhutdinov and Hinton, 2009; Zhang et al., 2010; Wang et al., 2014). By representing every document in the corpus as a similarity-preserving discrete (binary) *hashing code*, the similarity between two documents can be evaluated by simply calculating pairwise Hamming distances between hashing codes, *i.e.*, the number of bits that are different between two codes. Given that today, an ordinary PC is able to execute millions of Hamming distance computations in just a few milliseconds (Zhang et al., 2010), this semantic hashing strategy is very computationally attractive.

While considerable research has been devoted to text (semantic) hashing, existing approaches typically require two-stage training procedures. These methods can be generally divided into two categories: (i) binary codes for documents are first learned in an unsupervised manner, then l binary classifiers are trained via supervised learning to predict the l-bit hashing code (Zhang et al., 2010; Xu et al., 2015); (ii) continuous text representations are first inferred, which are binarized as a second (separate) step during testing (Wang et al., 2013; Chaidaroon and Fang, 2017). Because the model parameters are not learned in an end-to-end manner, these two-stage training strategies may result in suboptimal local optima. This happens because different modules within the model are optimized separately, preventing the sharing of information between them. Further, in existing methods, binary constraints are typically handled *ad-hoc* by truncation, *i.e.*, the hashing codes are obtained via direct binarization from continuous representations after training. As a result, the information contained in the continuous representations is lost during the (separate) binarization process. Moreover, training different modules (mapping and classifier/binarization) separately often requires additional hyperparameter tuning for each training stage, which can be laborious and time-consuming.

[*] Equal contribution.

In this paper, we propose a simple and generic neural architecture for text hashing that learns binary latent codes for documents in an *end-to-end* manner. Inspired by recent advances in neural variational inference (NVI) for text processing (Miao et al., 2016; Yang et al., 2017; Shen et al., 2017b), we approach semantic hashing from a generative model perspective, where binary (hashing) codes are represented as either *deterministic* or *stochastic* Bernoulli latent variables. The inference (encoder) and generative (decoder) networks are optimized jointly by maximizing a variational lower bound to the marginal distribution of input documents (corpus). By leveraging a simple and effective method to estimate the gradients with respect to discrete (binary) variables, the loss term from the generative (decoder) network can be directly backpropagated into the inference (encoder) network to optimize the hash function.

Motivated by the *rate-distortion theory* (Berger, 1971; Theis et al., 2017), we propose to inject data-dependent noise into the latent codes during the decoding stage, which adaptively accounts for the tradeoff between minimizing *rate* (number of bits used, or effective code length) and *distortion* (reconstruction error) during training. The connection between the proposed method and *rate-distortion theory* is further elucidated, providing a theoretical foundation for the effectiveness of our framework.

Summarizing, the contributions of this paper are: (*i*) to the best of our knowledge, we present the first semantic hashing architecture that can be trained in an *end-to-end* manner; (*ii*) we propose a *neural variational inference* framework to learn compact (regularized) binary codes for documents, achieving promising results on both *unsupervised* and *supervised* text hashing; (*iii*) the connection between our method and *rate-distortion theory* is established, from which we demonstrate the advantage of injecting *data-dependent noise* into the latent variable during training.

2 Related Work

Models with discrete random variables have attracted much attention in the deep learning community (Jang et al., 2016; Maddison et al., 2016; van den Oord et al., 2017; Li et al., 2017; Shu and Nakayama, 2017). Some of these structures are more natural choices for language or speech data, which are inherently discrete. More specifically,

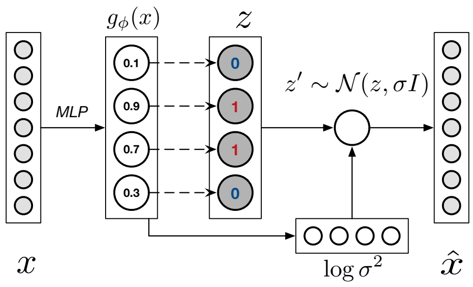


Figure 1: NASH for *end-to-end* semantic hashing. The inference network maps $x \to z$ using an MLP and the generative network recovers x as $z \to \hat{x}$.

van den Oord et al. (2017) combined VAEs with vector quantization to learn discrete latent representation, and demonstrated the utility of these learned representations on images, videos, and speech data. Li et al. (2017) leveraged both pairwise label and classification information to learn discrete hash codes, which exhibit state-of-the-art performance on image retrieval tasks.

For natural language processing (NLP), although significant research has been made to learn *continuous* deep representations for words or documents (Mikolov et al., 2013; Kiros et al., 2015; Shen et al., 2018), *discrete* neural representations have been mainly explored in learning word embeddings (Shu and Nakayama, 2017; Chen et al., 2017). In these recent works, words are represented as a vector of discrete numbers, which are very efficient storage-wise, while showing comparable performance on several NLP tasks, relative to continuous word embeddings. However, discrete representations that are learned in an *end-to-end* manner at the *sentence* or *document* level have been rarely explored. Also there is a lack of strict evaluation regarding their effectiveness. Our work focuses on learning discrete (binary) representations for text documents. Further, we employ semantic hashing (fast similarity search) as a mechanism to evaluate the quality of learned binary latent codes.

3 The Proposed Method

3.1 Hashing under the NVI Framework

Inspired by the recent success of variational autoencoders for various NLP problems (Miao et al., 2016; Bowman et al., 2015; Yang et al., 2017; Miao et al., 2017; Shen et al., 2017b; Wang et al., 2018), we approach the training of discrete (binary) latent variables from a generative perspective. Let x and z denote the input document and its corresponding binary hash code, respectively. Most of the previous text hashing methods focus on modeling the encoding distribution $p(z|x)$, or *hash function*, so the local/global pairwise similarity structure of documents in the original space is preserved in latent space (Zhang et al., 2010; Wang et al., 2013; Xu et al., 2015; Wang et al., 2014). However, the generative (decoding) process of reconstructing x from binary latent code z, *i.e.*, modeling distribution $p(x|z)$, has been rarely considered. Intuitively, latent codes learned from a model that accounts for the generative term should naturally encapsulate key semantic information from x because the generation/reconstruction objective is a function of $p(x|z)$. In this regard, the generative term provides a natural training objective for semantic hashing.

We define a generative model that simultaneously accounts for both the encoding distribution, $p(z|x)$, and decoding distribution, $p(x|z)$, by defining approximations $q_\phi(z|x)$ and $q_\theta(x|z)$, via inference and generative networks, $g_\phi(x)$ and $g_\theta(z)$, parameterized by ϕ and θ, respectively. Specifically, $x \in \mathcal{Z}_+^{|V|}$ is the bag-of-words (count) representation for the input document, where $|V|$ is the vocabulary size. Notably, we can also employ other count weighting schemes as input features x, *e.g.*, the term frequency-inverse document frequency (TFIDF) (Manning et al., 2008). For the encoding distribution, a latent variable z is first inferred from the input text x, by constructing an inference network $g_\phi(x)$ to approximate the true posterior distribution $p(z|x)$ as $q_\phi(z|x)$. Subsequently, the decoder network $g_\theta(z)$ maps z back into input space to reconstruct the original sequence x as $\hat{x}$, approximating $p(x|z)$ as $q_\theta(x|z)$ (as shown in Figure 1). This *cyclic* strategy, $x \to z \to \hat{x} \approx x$, provides the latent variable z with a better ability to generalize (Miao et al., 2016).

To tailor the NVI framework for semantic hashing, we cast z as a binary latent variable and assume a multivariate Bernoulli prior on z: $p(z) \sim \text{Bernoulli}(\gamma) = \prod_{i=1}^{l} \gamma_i^{z_i}(1-\gamma_i)^{1-z_i}$, where $\gamma_i \in [0,1]$ is component i of vector γ. Thus, the encoding (approximate posterior) distribution $q_\phi(z|x)$ is restricted to take the form $q_\phi(z|x) = \text{Bernoulli}(h)$, where $h = \sigma(g_\phi(x))$, $\sigma(\cdot)$ is the sigmoid function, and $g_\phi(\cdot)$ is the (nonlinear) inference network specified as a multilayer perceptron (MLP). As illustrated in Figure 1, we can obtain samples from the Bernoulli posterior either *deterministically* or *stochastically*. Suppose z is a l-bit hash code, for the *deterministic* binarization, we have, for $i = 1, 2, ..., l$:

$$z_i = \mathbf{1}_{\sigma(g_\phi^i(x))>0.5} = \frac{\text{sign}(\sigma(g_\phi^i(x) - 0.5) + 1}{2}, \tag{1}$$

where z is the binarized variable, and z_i and $g_\phi^i(x)$ denote the i-th dimension of z and $g_\phi(x)$, respectively. The standard Bernoulli sampling in (1) can be understood as setting a hard threshold at 0.5 for each representation dimension, therefore, the binary latent code is generated deterministically. Another strategy to obtain the discrete variable is to binarize h in a *stochastic* manner:

$$z_i = \mathbf{1}_{\sigma(g_\phi^i(x))>\mu_i} = \frac{\text{sign}(\sigma(g_\phi^i(x)) - \mu_i) + 1}{2}, \tag{2}$$

where $\mu_i \sim \text{Uniform}(0,1)$. Because of this sampling process, we do not have to assume a predefined threshold value like in (1).

3.2 Training with Binary Latent Variables

To estimate the parameters of the encoder and decoder networks, we would ideally maximize the marginal distribution $p(x) = \int p(z)p(x|z)dz$. However, computing this marginal is intractable in most cases of interest. Instead, we maximize a variational lower bound. This approach is typically employed in the VAE framework (Kingma and Welling, 2013):

$$\begin{aligned}\mathcal{L}_{\text{vae}} &= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\theta(x|z)p(z)}{q_\phi(z|x)}\right], \qquad (3)\\ &= \mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)),\end{aligned}$$

where the Kullback-Leibler (KL) divergence $D_{KL}(q_\phi(z|x)||p(z))$ encourages the approximate posterior distribution $q_\phi(z|x)$ to be close to the multivariate Bernoulli prior $p(z)$. In this case, $D_{KL}(q_\phi(z|x)|p(z))$ can be written in closed-form as a function of $g_\phi(x)$:

$$\begin{aligned}D_{KL} &= g_\phi(x)\log\frac{g_\phi(x)}{\gamma}\\ &\quad + (1-g_\phi(x))\log\frac{1-g_\phi(x)}{1-\gamma}. \qquad (4)\end{aligned}$$

Note that the gradient for the KL divergence term above can be evaluated easily.

For the first term in (3), we should in principle estimate the influence of μ_i in (2) on $q_\theta(x|z)$ by averaging over the entire (uniform) noise distribution. However, a closed-form distribution does not exist since it is not possible to enumerate all possible configurations of z, especially when the latent dimension is large. Moreover, discrete latent variables are inherently incompatible with backpropagation, since the derivative of the sign function is zero for almost all input values. As a result, the exact gradients of L_{vae} wrt the inputs before binarization would be essentially all zero.

To estimate the gradients for binary latent variables, we utilize the straight-through (ST) estimator, which was first introduced by Hinton (2012). So motivated, the strategy here is to simply backpropagate through the hard threshold by approximating the gradient $\partial z/\partial\phi$ as 1. Thus, we have:

$$\begin{aligned}
&\frac{d\mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)]}{\partial\phi} \\
&= \frac{d\mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)]}{dz}\frac{dz}{d\sigma(g_\phi^i(x))}\frac{d\sigma(g_\phi^i(x))}{d\phi} \\
&\approx \frac{d\mathbb{E}_{q_\phi(z|x)}[\log q_\theta(x|z)]}{dz}\frac{d\sigma(g_\phi^i(x))}{d\phi} \qquad (5)
\end{aligned}$$

Although this is clearly a biased estimator, it has been shown to be a fast and efficient method relative to other gradient estimators for discrete variables, especially for the Bernoulli case (Bengio et al., 2013; Hubara et al., 2016; Theis et al., 2017). With the ST gradient estimator, the first loss term in (3) can be backpropagated into the encoder network to fine-tune the hash function $g_\phi(x)$.

For the approximate generator $q_\theta(x|z)$ in (3), let x_i denote the one-hot representation of ith word within a document. Note that $x = \sum_i x_i$ is thus the bag-of-words representation for document x. To reconstruct the input x from z, we utilize a *softmax* decoding function written as:

$$q(x_i = w|z) = \frac{\exp(z^T E x_w + b_w)}{\sum_{j=1}^{|V|}\exp(z^T E x_j + b_j)}, \qquad (6)$$

where $q(x_i = w|z)$ is the probability that x_i is word $w \in V$, $q_\theta(x|z) = \prod_i q(x_i = w|z)$ and $\theta = \{E, b_1, \dots, b_{|V|}\}$. Note that $E \in \mathbb{R}^{d\times|V|}$ can be interpreted as a word embedding matrix to be learned, and $\{b_i\}_{i=1}^{|V|}$ denote bias terms. Intuitively, the objective in (6) encourages the discrete vector z to be close to the embeddings for every word that appear in the input document x. As shown in Section 5.3.1, meaningful semantic structures can be learned and manifested in the word embedding matrix E.

3.3 Injecting Data-dependent Noise to z

To reconstruct text data x from sampled binary representation z, a deterministic decoder is typically utilized (Miao et al., 2016; Chaidaroon and Fang, 2017). Inspired by the success of employing stochastic decoders in image hashing applications (Dai et al., 2017; Theis et al., 2017), in our experiments, we found that injecting random Gaussian noise into z makes the decoder a more favorable regularizer for the binary codes, which in practice leads to stronger retrieval performance. Below, we invoke the *rate-distortion theory* to perform some further analysis, which leads to interesting findings.

Learning binary latent codes z to represent a continuous distribution $p(x)$ is a classical information theory concept known as *lossy source coding*. From this perspective, semantic hashing, which compresses an input document into compact binary codes, can be casted as a conventional *rate-distortion tradeoff* problem (Theis et al., 2017; Ballé et al., 2016):

$$\min \underbrace{-\log_2 R(z)}_{\text{Rate}} + \beta \cdot \underbrace{D(x,\hat{x})}_{\text{Distortion}}, \qquad (7)$$

where *rate* and *distortion* denote the effective code length, *i.e.*, the number of bits used, and the distortion introduced by the encoding/decoding sequence, respectively. Further, $\hat{x}$ is the reconstructed input and β is a hyperparameter that controls the tradeoff between the two terms.

Considering the case where we have a Bernoulli *prior* on z as $p(z) \sim \text{Bernoulli}(\gamma)$, and x conditionally drawn from a Gaussian distribution $p(x|z) \sim \mathcal{N}(Ez, \sigma^2 I)$. Here, $E = \{e_i\}_{i=1}^{|V|}$, where $e_i \in \mathbb{R}^d$ can be interpreted as a *codebook* with $|V|$ *codewords*. In our case, E corresponds to the *word embedding matrix* as in (6).

For the case of stochastic latent variable z, the objective function in (3) can be written in a form similar to the *rate-distortion* tradeoff:

$$\min \mathbb{E}_{q_\phi(z|x)}\left[\underbrace{-\log q_\phi(z|x)}_{\text{Rate}} + \underbrace{\frac{1}{2\sigma^2}}_{\beta}\underbrace{||x - Ez||_2^2}_{\text{Distortion}} + C\right], \qquad (8)$$

where C is a constant that encapsulates the prior distribution $p(z)$ and the Gaussian distribution normalization term. Notably, the trade-off hyper-parameter $\beta = \sigma^{-2}/2$ is closely related to the variance of the distribution $p(x|z)$. In other words, by controlling the variance σ, the model can adaptively explore different trade-offs between the *rate* and *distortion* objectives. However, the optimal trade-offs for distinct samples may be different.

Inspired by the observations above, we propose to inject data-dependent noise into latent variable z, rather than to setting the variance term σ^2 to a fixed value (Dai et al., 2017; Theis et al., 2017). Specifically, $\log \sigma^2$ is obtained via a one-layer MLP transformation from $g_\phi(x)$. Afterwards, we sample z' from $\mathcal{N}(z, \sigma^2 I)$, which then replace z in (6) to infer the probability of generating individual words (as shown in Figure 1). As a result, the variances are different for every input document x, and thus the model is provided with additional flexibility to explore various trade-offs between *rate* and *distortion* for different training observations. Although our decoder is not a strictly Gaussian distribution, as in (6), we found empirically that injecting data-dependent noise into z yields strong retrieval results, see Section 5.1.

3.4 Supervised Hashing

The proposed Neural Architecture for Semantic Hashing (NASH) can be extended to supervised hashing, where a mapping from latent variable z to labels y is learned, here parametrized by a two-layer MLP followed by a fully-connected softmax layer. To allow the model to explore and balance between maximizing the variational lower bound in (3) and minimizing the discriminative loss, the following joint training objective is employed:

$$\mathcal{L} = -\mathcal{L}_{\text{vae}}(\theta, \phi; x) + \alpha \mathcal{L}_{\text{dis}}(\eta; z, y). \qquad (9)$$

where η refers to parameters of the MLP classifier and α controls the relative weight between the variational lower bound ($\mathcal{L}_{\text{vae}}$) and discriminative loss ($\mathcal{L}_{\text{dis}}$), defined as the cross-entropy loss. The parameters $\{\theta, \phi, \eta\}$ are learned end-to-end *via* Monte Carlo estimation.

4 Experimental Setup

4.1 Datasets

We use the following three standard publicly available datasets for training and evaluation: (i) *Reuters21578*, containing 10,788 news documents, which have been classified into 90 different categories. (ii) *20Newsgroups*, a collection of 18,828 newsgroup documents, which are categorized into 20 different topics. (iii) TMC (stands for SIAM text mining competition), containing air traffic reports provided by NASA. TMC consists 21,519 training documents divided into 22 different categories. To make direct comparison with prior works, we employed the TFIDF features on these datasets supplied by (Chaidaroon and Fang, 2017), where the vocabulary sizes for the three datasets are set to 10,000, 7,164 and 20,000, respectively.

4.2 Training Details

For the inference networks, we employ a feed-forward neural network with 2 hidden layers (both with 500 units) using the ReLU non-linearity activation function, which transform the input documents, *i.e.*, TFIDF features in our experiments, into a continuous representation. Empirically, we found that stochastic binarization as in (2) shows stronger performance than deterministic binarization, and thus use the former in our experiments. However, we further conduct a systematic ablation study in Section 5.2 to compare the two binarization strategies.

Our model is trained using Adam (Kingma and Ba, 2014), with a learning rate of 1×10^{-3} for all parameters. We decay the learning rate by a factor of 0.96 for every 10,000 iterations. Dropout (Srivastava et al., 2014) is employed on the output of encoder networks, with the rate selected from $\{0.7, 0.8, 0.9\}$ on the validation set. To facilitate comparisons with previous methods, we set the dimension of z, *i.e.*, the number of bits within the hashing code) as 8, 16, 32, 64, or 128.

4.3 Baselines

We evaluate the effectiveness of our framework on both unsupervised and supervised semantic hashing tasks. We consider the following *unsupervised* baselines for comparisons: Locality Sensitive Hashing (LSH) (Datar et al., 2004), Stack Restricted Boltzmann Machines (S-RBM) (Salakhutdinov and Hinton, 2009), Spectral Hashing (SpH) (Weiss et al., 2009), Self-taught Hashing (STH) (Zhang et al., 2010) and Variational Deep Semantic Hashing (VDSH) (Chaidaroon and Fang, 2017).

Method	8 bits	16 bits	32 bits	64 bits	128 bits
LSH	0.2802	0.3215	0.3862	0.4667	0.5194
S-RBM	0.5113	0.5740	0.6154	0.6177	0.6452
SpH	0.6080	0.6340	0.6513	0.6290	0.6045
STH	0.6616	0.7351	0.7554	0.7350	0.6986
VDSH	0.6859	0.7165	0.7753	0.7456	0.7318
NASH	0.7113	0.7624	0.7993	0.7812	0.7559
NASH-N	0.7352	0.7904	0.8297	0.8086	0.7867
NASH-DN	**0.7470**	**0.8013**	**0.8418**	**0.8297**	**0.7924**

Table 1: Precision of the top 100 retrieved documents on *Reuters* dataset (*Unsupervised hashing*).

For supervised semantic hashing, we also compare NASH against a number of baselines: Supervised Hashing with Kernels (KSH) (Liu et al., 2012), Semantic Hashing using Tags and Topic Modeling (SHTTM) (Wang et al., 2013) and Supervised VDSH (Chaidaroon and Fang, 2017). It is worth noting that unlike all these baselines, our NASH model is trained end-to-end in one-step.

4.4 Evaluation Metrics

To evaluate the hashing codes for similarity search, we consider each document in the testing set as a query document. Similar documents to the query in the corresponding training set need to be retrieved based on the Hamming distance of their hashing codes, *i.e.* number of different bits. To facilitate comparison with prior work (Wang et al., 2013; Chaidaroon and Fang, 2017), the performance is measured with precision. Specifically, during testing, for a query document, we first retrieve the 100 nearest/closest documents according to the Hamming distances of the corresponding hash codes (i.e., the number of different bits). We then examine the percentage of documents among these 100 retrieved ones that belong to the same label (topic) with the query document (we consider documents having the same label as relevant pairs). The ratio of the number of relevant documents to the number of retrieved documents (fixed value of 100) is calculated as the precision score. The precision scores are further averaged over all test (query) documents.

5 Experimental Results

We experimented with four variants for our NASH model: (*i*) NASH: with deterministic decoder; (*ii*) NASH-N: with *fixed* random noise injected to decoder; (*iii*) NASH-DN: with *data-dependent* noise injected to decoder; (*iv*) NASH-DN-S: NASH-DN with supervised information during training.

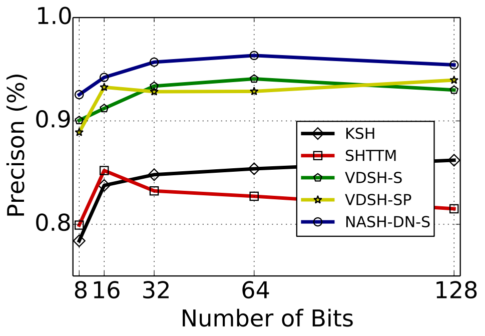


Figure 2: Precision of the top 100 retrieved documents on *Reuters* dataset (*Supervised hashing*), compared with other supervised baselines.

5.1 Semantic Hashing Evaluation

Table 1 presents the results of all models on Reuters dataset. Regarding unsupervised semantic hashing, all the NASH variants consistently outperform the baseline methods by a substantial margin, indicating that our model makes the most effective use of unlabeled data and manage to assign similar hashing codes, *i.e.*, with small Hamming distance to each other, to documents that belong to the same label. It can be also observed that the injection of noise into the decoder networks has improved the robustness of learned binary representations, resulting in better retrieval performance. More importantly, by making the variances of noise adaptive to the specific input, our NASH-DN achieves even better results, compared with NASH-N, highlighting the importance of exploring/learning the trade-off between rate and distortion objectives by the data itself. We observe the same trend and superiority of our NASH-DN models on the other two benchmarks, as shown in Tables 3 and 4.

Another observation is that the retrieval results tend to drop a bit when we set the length of hashing codes to be 64 or larger, which also happens for some baseline models. This phenomenon has been reported previously in Wang et al. (2012); Liu et al. (2012); Wang et al. (2013); Chaidaroon and Fang (2017), and the reasons could be twofold: (i) for longer codes, the number of data points that are assigned to a certain binary code decreases exponentially. As a result, many queries may fail to return any neighbor documents (Wang et al., 2012); (ii) considering the size of training data, it is likely that the model may overfit with long hash codes (Chaidaroon and Fang, 2017). However, even with longer hashing codes,

Word	**weapons**	**medical**	**companies**	**define**	**israel**	**book**
NASH	gun	treatment	company	definition	israeli	books
	guns	disease	market	defined	arabs	english
	weapon	drugs	afford	explained	arab	references
	armed	health	products	discussion	jewish	learning
	assault	medicine	money	knowledge	jews	reference
NVDM	guns	medicine	expensive	defined	israeli	books
	weapon	health	industry	definition	arab	reference
	gun	treatment	company	printf	arabs	guide
	militia	disease	market	int	lebanon	writing
	armed	patients	buy	sufficient	lebanese	pages

Table 2: The five nearest words in the semantic space learned by NASH, compared with the results from NVDM (Miao et al., 2016).

Method	**8 bits**	**16 bits**	**32 bits**	**64 bits**	**128 bits**
	Unsupervised Hashing				
LSH	0.0578	0.0597	0.0666	0.0770	0.0949
S-RBM	0.0594	0.0604	0.0533	0.0623	0.0642
SpH	0.2545	0.3200	0.3709	0.3196	0.2716
STH	0.3664	0.5237	0.5860	**0.5806**	**0.5443**
VDSH	0.3643	0.3904	0.4327	0.1731	0.0522
NASH	0.3786	0.5108	0.5671	0.5071	0.4664
NASH-N	0.3903	0.5213	0.5987	0.5143	0.4776
NASH-DN	**0.4040**	**0.5310**	**0.6225**	0.5377	0.4945
	Supervised Hashing				
KSH	0.4257	0.5559	0.6103	0.6488	0.6638
SHTTM	0.2690	0.3235	0.2357	0.1411	0.1299
VDSH-S	0.6586	0.6791	0.7564	0.6850	0.6916
VDSH-SP	**0.6609**	0.6551	0.7125	0.7045	0.7117
NASH-DN-S	0.6247	**0.6973**	**0.8069**	**0.8213**	**0.7840**

Table 3: Precision of the top 100 retrieved documents on *20Newsgroups* dataset.

Method	**8 bits**	**16 bits**	**32 bits**	**64 bits**	**128 bits**
	Unsupervised Hashing				
LSH	0.4388	0.4393	0.4514	0.4553	0.4773
S-RBM	0.4846	0.5108	0.5166	0.5190	0.5137
SpH	0.5807	0.6055	0.6281	0.6143	0.5891
STH	0.3723	0.3947	0.4105	0.4181	0.4123
VDSH	0.4330	0.6853	0.7108	0.4410	0.5847
NASH	0.5849	0.6573	0.6921	0.6548	0.5998
NASH-N	0.6233	0.6759	0.7201	0.6877	0.6314
NASH-DN	**0.6358**	**0.6956**	**0.7327**	**0.7010**	**0.6325**
	Supervised Hashing				
KSH	0.6608	0.6842	0.7047	0.7175	0.7243
SHTTM	0.6299	0.6571	0.6485	0.6893	0.6474
VDSH-S	0.7387	0.7887	0.7883	0.7967	0.8018
VDSH-SP	**0.7498**	0.7798	0.7891	0.7888	0.7970
NASH-DN-S	0.7438	**0.7946**	**0.7987**	**0.8014**	**0.8139**

Table 4: Precision of the top 100 retrieved documents on *TMC* dataset.

our NASH models perform stronger than the baselines in most cases (except for the 20Newsgroups dataset), suggesting that NASH can effectively allocate documents to informative/meaningful hashing codes even with limited training data.

We also evaluate the effectiveness of NASH in a *supervised* scenario on the Reuters dataset, where the label or topic information is utilized during training. As shown in Figure 2, our NASH-DN-S model consistently outperforms several supervised semantic hashing baselines, with various choices of hashing bits. Notably, our model exhibits higher Top-100 retrieval precision than VDSH-S and VDSH-SP, proposed by Chaidaroon and Fang (2017). This may be attributed to the fact that in VDSH models, the continuous embeddings are not optimized with their future binarization in mind, and thus could hurt the relevance of learned binary codes. On the contrary, our model is optimized in an end-to-end manner, where the gradients are directly backpropagated to the inference network (through the binary/discrete latent variable), and thus gives rise to a more robust hash function.

5.2 Ablation study

5.2.1 The effect of stochastic sampling

As described in Section 3, the binary latent variables z in NASH can be either deterministically (via (1)) or stochastically (via (2)) sampled. We compare these two types of binarization functions in the case of unsupervised hashing. As illustrated in Figure 3, stochastic sampling shows stronger retrieval results on all three datasets, indicating that endowing the sampling process of latent variables with more stochasticity improves the learned representations.

5.2.2 The effect of encoder/decoder networks

Under the variational framework introduced here, the encoder network, *i.e.*, hash function, and decoder network are jointly optimized to abstract semantic features from documents. An interesting question concerns what types of network should be leveraged for each part of our NASH model. In this regard, we further investigate the effect of

Category	Title/Subject	8-bit code	16-bit code
Baseball	*Dave Kingman for the hall of fame*	1 1 1 0 1 0 0 1	0 0 1 0 1 1 0 1 0 0 0 0 0 1 1 0
	Time of game	1 1 1 1 1 0 0 1	0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 1
	Game score report	1 1 1 0 1 0 0 1	0 0 1 0 1 1 0 1 0 0 0 0 0 1 1 0
	Why is Barry Bonds not batting 4th?	1 1 1 0 1 1 0 1	0 0 1 1 1 1 0 1 0 0 0 0 0 1 1 0
Electronics	*Building a UV flashlight*	1 0 1 1 0 1 0 0	0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 1
	How to drive an array of LEDs	1 0 1 1 0 1 0 1	0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1
	2% silver solder	1 1 0 1 0 1 0 1	0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 1
	Subliminal message flashing on TV	1 0 1 1 0 1 0 0	0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1

Table 5: Examples of learned compact hashing codes on *20Newsgroups* dataset.

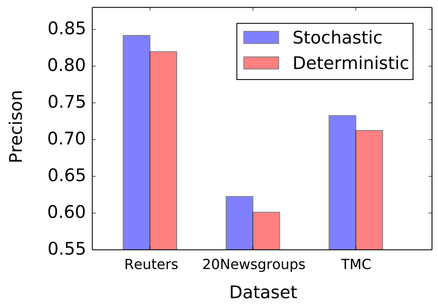


Figure 3: The precisions of the top 100 retrieved documents for NASH-DN with *stochastic* or *deterministic* binary latent variables.

using an encoder or decoder with different non-linearity, ranging from a linear transformation to two-layer MLPs. We employ a base model with an encoder of two-layer MLPs and a linear decoder (the setup described in Section 3), and the ablation study results are shown in Table 6.

Network	Encoder	Decoder
linear	0.5844	0.6225
one-layer MLP	0.6187	0.3559
two-layer MLP	0.6225	0.1047

Table 6: Ablation study with different encoder/decoder networks.

It is observed that for the encoder networks, increasing the non-linearity by stacking MLP layers leads to better empirical results. In other words, endowing the hash function with more modeling capacity is advantageous to retrieval tasks. However, when we employ a non-linear network for the decoder, the retrieval precision drops dramatically. It is worth noting that the only difference between linear transformation and one-layer MLP is whether a non-linear activation function is employed or not.

This observation may be attributed the fact that the decoder networks can be considered as a similarity measure between latent variable z and the word embeddings E_k for every word, and the probabilities for words that present in the document is maximized to ensure that z is informative. As a result, if we allow the decoder to be too expressive (*e.g.*, a one-layer MLP), it is likely that we will end up with a very flexible similarity measure but relatively less meaningful binary representations. This finding is consistent with several image hashing methods, such as SGH (Dai et al., 2017) or binary autoencoder (Carreira-Perpinán and Raziperchikolaei, 2015), where a linear decoder is typically adopted to obtain promising retrieval results. However, our experiments may not speak for other choices of encoder-decoder architectures, *e.g.*, LSTM-based sequence-to-sequence models (Sutskever et al., 2014) or DCNN-based autoencoder (Zhang et al., 2017).

5.3 Qualitative Analysis

5.3.1 Analysis of Semantic Information

To understand what information has been learned in our NASH model, we examine the matrix $E \in \mathbb{R}^{d \times l}$ in (6). Similar to (Miao et al., 2016; Larochelle and Lauly, 2012), we select the 5 nearest words according to the word vectors learned from NASH and compare with the corresponding results from NVDM.

As shown in Table 2, although our NASH model contains a binary latent variable, rather than a continuous one as in NVDM, it also effectively group semantically-similar words together in the learned vector space. This further demonstrates that the proposed generative framework manages to bypass the binary/discrete constraint and is able to abstract useful semantic information from documents.

5.3.2 Case Study

In Table 5, we show some examples of the learned binary hashing codes on *20Newsgroups* dataset. We observe that for both 8-bit and 16-bit cases, NASH typically compresses documents with shared topics into very similar binary codes. On the contrary, the hashing codes for documents with different topics exhibit much larger Hamming distance. As a result, relevant documents can be efficiently retrieved by simply computing their Hamming distances.

6 Conclusions

This paper presents a first step towards *end-to-end* semantic hashing, where the binary/discrete constraints are carefully handled with an effective gradient estimator. A neural variational framework is introduced to train our model. Motivated by the connections between the proposed method and *rate-distortion theory*, we inject data-dependent noise into the Bernoulli latent variable at the training stage. The effectiveness of our framework is demonstrated with extensive experiments.

Acknowledgments We would like to thank the ACL reviewers for their insightful suggestions. This research was supported in part by DARPA, DOE, NIH, NSF and ONR.

References

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. 2016. End-to-end optimization of nonlinear transform codes for perceptual quality. In *Picture Coding Symposium (PCS), 2016*. IEEE, pages 1–5.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* .

Toby Berger. 1971. Rate-distortion theory. *Encyclopedia of Telecommunications* .

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. 2015. Hashing with binary autoencoders. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, pages 557–566.

Suthee Chaidaroon and Yi Fang. 2017. Variational deep semantic hashing for text documents. In *Proceedings of the 40th international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Ting Chen, Martin Renqiang Min, and Yizhou Sun. 2017. Learning k-way d-dimensional discrete code for compact embedding representations. *arXiv preprint arXiv:1711.03067* .

Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. 2017. Stochastic generative hashing. *arXiv preprint arXiv:1701.02815* .

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, pages 253–262.

Geoffrey Hinton. 2012. Neural networks for machine learning, coursera. *URL: http://coursera.org/course/neuralnets* .

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Advances in neural information processing systems*. pages 4107–4115.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 426–434.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*. pages 2708–2716.

Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. 2006. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 2(1):1–19.

Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. 2017. Deep supervised discrete hashing. *arXiv preprint arXiv:1705.10999* .

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pages 2074–2081.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* .

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. *arXiv preprint arXiv:1706.00359* .

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*. pages 1727–1736.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Sandeep Pandey, Andrei Broder, Flavio Chierichetti, Vanja Josifovski, Ravi Kumar, and Sergei Vassilvitskii. 2009. Nearest-neighbor caching for content-match applications. In *Proceedings of the 18th international conference on World wide web*. ACM, pages 441–450.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50(7):969–978.

Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. 2017a. Adaptive convolutional filter generation for natural language understanding. *arXiv preprint arXiv:1709.08294* .

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017b. Deconvolutional latent-variable model for text sequence matching. *AAAI* .

Raphael Shu and Hideki Nakayama. 2017. Compressing word embeddings via deep compositional code learning. *arXiv preprint arXiv:1711.01068* .

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Benno Stein, Sven Meyer zu Eissen, and Martin Potthast. 2007. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 825–826.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy image compression with compressive autoencoders. *ICLR* .

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*. pages 6309–6318.

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* .

Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(12):2393–2406.

Qifan Wang, Dan Zhang, and Luo Si. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 213–222.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2018. Topic compositional neural language model. In *AISTATS*.

Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Advances in neural information processing systems*. pages 1753–1760.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Convolutional neural networks for text hashing. In *IJCAI*. pages 1369–1375.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139* .

Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 18–25.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*. pages 4172–4182.

Large-Scale QA-SRL Parsing

Nicholas FitzGerald* **Julian Michael*** **Luheng He** **Luke Zettlemoyer***
Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle, WA
{nfitz,julianjm,luheng,lsz}@cs.washington.edu

Abstract

We present a new large-scale corpus of Question-Answer driven Semantic Role Labeling (QA-SRL) annotations, and the first high-quality QA-SRL parser. Our corpus, QA-SRL Bank 2.0, consists of over 250,000 question-answer pairs for over 64,000 sentences across 3 domains and was gathered with a new crowd-sourcing scheme that we show has high precision and good recall at modest cost. We also present neural models for two QA-SRL subtasks: detecting argument spans for a predicate and generating questions to label the semantic relationship. The best models achieve question accuracy of 82.6% and span-level accuracy of 77.6% (under human evaluation) on the full pipelined QA-SRL prediction task. They can also, as we show, be used to gather additional annotations at low cost.

1 Introduction

Learning semantic parsers to predict the predicate-argument structures of a sentence is a long standing, open challenge (Palmer et al., 2005; Baker et al., 1998). Such systems are typically trained from datasets that are difficult to gather,[1] but recent research has explored training non-experts to provide this style of semantic supervision (Abend and Rappoport, 2013; Basile et al., 2012; Reisinger et al., 2015; He et al., 2015). In this paper, we show for the first time that it is possible to go even further by crowdsourcing a large scale dataset that can be used to train high quality parsers at modest cost.

In 1950 Alan M. Turing ***published*** "Computing machinery and intelligence" in Mind, in which he ***proposed*** that machines could be ***tested*** for intelligence ***using*** questions and answers.

Predicate		**Question**	**Answer**
published	1	Who published something?	Alan M. Turing
	2	What was published?	"Computing Machinery and Intelligence"
	3	When was something published?	In 1950
proposed	4	Who proposed something?	Alan M. Turing
	5	What did someone propose?	that machines could be tested for intelligent using questions and answers
	6	When did someone propose something?	In 1950
tested	7	What can be tested?	machines
	8	What can something be tested for?	intelligence
	9	How can something be tested?	using questions and answers
using	10	What was being used?	questions and answers
	11	Why was something being used?	tested for intelligence

Figure 1: An annotated sentence from our dataset. Question 6 was not produced by crowd workers in the initial collection, but was produced by our parser as part of Data Expansion (see Section 5.)

We adopt the Question-Answer-driven Semantic Role Labeling (QA-SRL) (He et al., 2015) annotation scheme. QA-SRL is appealing because it is intuitive to non-experts, has been shown to closely match the structure of traditional predicate-argument structure annotation schemes (He et al., 2015), and has been used for end tasks such as Open IE (Stanovsky and Dagan, 2016). In QA-SRL, each predicate-argument relationship is labeled with a question-answer pair (see Figure 1). He et al. (2015) showed that high precision QA-SRL annotations can be gathered with limited training but that high recall is challenging to achieve; it is relatively easy to gather answerable questions, but difficult to ensure that every possible question is labeled for every verb. For this reason, they hired and trained hourly annotators and only labeled a relatively small dataset (3000 sentences).

Our first contribution is a new, scalable approach for crowdsourcing QA-SRL. We introduce a streamlined web interface (including an auto-suggest mechanism and automatic quality control to boost recall) and use a validation stage to en-

* Much of this work was done while these authors were at the Allen Institute for Artificial Intelligence.

[1]The PropBank (Bonial et al., 2010) and FrameNet (Ruppenhofer et al., 2016) annotation guides are 89 and 119 pages, respectively.

sure high precision (i.e. all the questions must be answerable). With this approach, we produce QA-SRL Bank 2.0, a dataset with 133,479 verbs from 64,018 sentences across 3 domains, totaling 265,140 question-answer pairs, in just 9 days. Our analysis shows that the data has high precision with good recall, although it does not cover every possible question. Figure 1 shows example annotations.

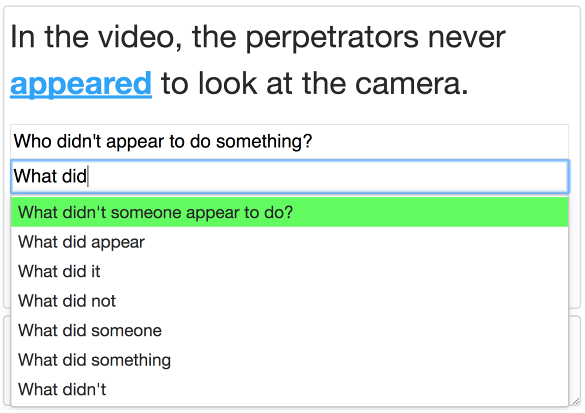
Using this data, our second contribution is a comparison of several new models for learning a QA-SRL parser. We follow a pipeline approach where the parser does (1) unlabeled *span detection* to determine the arguments of a given verb, and (2) *question generation* to label the relationship between the predicate and each detected span. Our best model uses a span-based representation similar to that introduced by Lee et al. (2016) and a custom LSTM to decode questions from a learned span encoding. Our model does not require syntactic information and can be trained directly from the crowdsourced span labels.

Experiments demonstrate that the model does well on our new data, achieving up to 82.2% span-detection F1 and 47.2% exact-match question accuracy relative to the human annotations. We also demonstrate the utility of learning to predict easily interpretable QA-SRL structures, using a simple data bootstrapping approach to expand our dataset further. By tuning our model to favor recall, we over-generate questions which can be validated using our annotation pipeline, allowing for greater recall without requiring costly redundant annotations in the question writing step. Performing this procedure on the training and development sets grows them by 20% and leads to improvements when retraining our models. Our final parser is highly accurate, achieving 82.6% question accuracy and 77.6% span-level precision in an human evaluation. Our data, code, and trained models will be made publicly available.[2]

2 Data Annotation

A QA-SRL annotation consists of a set of question-answer pairs for each verbal predicate in a sentence, where each answer is a set of contiguous spans from the sentence. QA-SRL questions are defined by a 7-slot template shown in Table 1. We introduce a crowdsourcing pipeline to collect annotations rapidly, cheaply, and at large scale.

[2]http://qasrl.org


In the video, the perpetrators never appeared to look at the camera.

Who didn't appear to do something?

What did

What didn't someone appear to do?

What did appear

What did it

What did not

What did someone

What did something

What didn't

Figure 2: Interface for the generation step. Autocomplete shows completions of the current QA-SRL slot, and auto-suggest shows fully-formed questions (highlighted green) based on the previous questions.

Pipeline Our crowdsourcing pipeline consists of a *generation* and *validation* step. In the generation step, a sentence with one of its verbs marked is shown to a single worker, who must write QA-SRL questions for the verb and highlight their answers in the sentence. The questions are passed to the validation step, where n workers answer each question or mark it as *invalid*. In each step, no two answers to distinct questions may overlap with each other, to prevent redundancy.

Instructions Workers are instructed that a *valid* question-answer pair must satisfy three criteria: 1) the question is grammatical, 2) the question-answer pair is asking about the time, place, participants, etc., of the target verb, and 3) all correct answers to each question are given.

Autocomplete We provide an autocomplete drop-down to streamline question writing. Autocomplete is implemented as a Non-deterministic Finite Automaton (NFA) whose states correspond to the 7 QA-SRL slots paired with a partial representation of the question's syntax. We use the NFA to make the menu more compact by disallowing obviously ungrammatical combinations (e.g., *What did been appeared?*), and the syntactic representation to auto-suggest complete questions about arguments that have not yet been covered (see Figure 2). The auto-suggest feature significantly reduces the number of keystrokes required to enter new questions after the first one, speeding up the annotation process and making it easier for annotators to provide higher recall.

Wh	Aux	Subj	Verb	Obj	Prep	Misc
Who			blamed	someone		
What	did	someone	blame	something	on	
Who			refused		to	do something
When	did	someone	refuse		to	do something
Who	might		put	something		somewhere
Where	might	someone	put	something		

Table 1: Example QA-SRL questions, decomposed into their slot-based representation. See He et al. (2015) for the full details. All slots draw from a small, deterministic set of options, including verb tense ($present$, $pastparticiple$, etc.) Here we have replaced the verb-tense slot with its conjugated form.

	Wikipedia	Wikinews	Science
Sentences	15,000	14,682	46,715
Verbs	32,758	34,026	66,653
Questions	75,867	80,081	143,388
Valid Qs	67,146	70,555	127,455

Table 2: Statistics for the dataset with questions written by workers across three domains.

Payment and quality control Generation pays 5c for the first QA pair (required), plus 5c, 6c, etc. for each successive QA pair (optional), to boost recall. The validation step pays 8c per verb, plus a 2c bonus per question beyond four. Generation workers must write at least 2 questions per verb and have 85% of their questions counted valid, and validators must maintain 85% answer span agreement with others, or they are disqualified from further work. A validator's answer is considered to agree with others if their answer span overlaps with answer spans provided by a majority of workers.

Preprocessing We use the Stanford CoreNLP tools (Manning et al., 2014) for sentence segmentation, tokenizing, and POS-tagging. We identify verbs by POS tag, with heuristics to filter out auxiliary verbs while retaining non-auxiliary uses of "have" and "do." We identify conjugated forms of each verb for the QA-SRL templates by finding them in Wiktionary.[3]

Dataset We gathered annotations for 133,479 verb mentions in 64,018 sentences (1.27M tokens) across 3 domains: Wikipedia, Wikinews, and science textbook text from the Textbook Question Answering (TQA) dataset (Kembhavi et al., 2017). We partitioned the source documents into train, dev, and test, sampled paragraph-wise from each document with an 80/10/10 split by sentence.

Annotation in our pipeline with $n = 2$ validators took 9 days on Amazon Mechanical Turk.[4] 1,165 unique workers participated, annotating a total of 299,308 questions. Of these, 265,140 (or 89%) were considered valid by both validators, for an average of 1.99 valid questions per verb and 4.14 valid questions per sentence. See Table 2 for a breakdown of dataset statistics by domain. The total cost was $43,647.33, for an average of 32.7c per verb mention, 14.6c per question, or 16.5c per valid question. For comparison, He et al. (2015) interviewed and hired contractors to annotate data at much smaller scale for a cost of about 50c per verb. Our annotation scheme is cheaper, far more scalable, and provides more (though noisier) supervision for answer spans.

To allow for more careful evaluation, we validated 5,205 sentences at a higher density (up to 1,000 for each domain in dev and test), re-running the generated questions through validation with $n = 3$ for a total of 6 answer annotations for each question.

Quality Judgments of question validity had moderate agreement. About 89.5% of validator judgments rated a question as valid, and the agreement rate between judgments of the same question on whether the question is invalid is 90.9%. This gives a Fleiss's Kappa of 0.51. In the higher-density re-run, validators were primed to be more critical: 76.5% of judgments considered a question valid, and agreement was at 83.7%, giving a Fleiss's Kappa of 0.55.

Despite being more critical in the denser annotation round, questions marked valid in the original dataset were marked valid by the new annotators in 86% of cases, showing our data's relatively high precision. The high precision of our annotation pipeline is also backed up by our small-scale manual evaluation (see Coverage below).

Answer spans for each question also exhibit

[3]www.wiktionary.org

[4]www.mturk.com

	P	R	F
He et al. (2015)	97.5	86.6	91.7
This work	95.7	72.4	82.4
This work (unfiltered)	94.9	85.4	89.9

Table 3: Precision and recall of our annotation pipeline on a merged and validated subset of 100 verbs. The unfiltered number represents relaxing the restriction that none of 2 validators marked the question as invalid.

good agreement. On the original dataset, each answer span has a 74.8% chance to exactly match one provided by another annotator (up to two), and on the densely annotated subset, each answer span has an 83.1% chance to exactly match one provided by another annotator (up to five).

Coverage Accurately measuring recall for QA-SRL annotations is an open challenge. For example, question 6 in Figure 1 reveals an inferred temporal relation that would not be annotated as part of traditional SRL. Exhaustively enumerating the full set of such questions is difficult, even for experts.

However, we can compare to the original QA-SRL dataset (He et al., 2015), where Wikipedia sentences were annotated with 2.43 questions per verb. Our data has lower—but loosely comparable—recall, with 2.05 questions per verb in Wikipedia.

In order to further analyze the quality of our annotations relative to (He et al., 2015), we reannotate a 100-verb subset of their data both manually (aiming for exhaustivity) and with our crowdsourcing pipeline. We merge the three sets of annotations, manually remove bad questions (and their answers), and calculate the precision and recall of the crowdsourced annotations and those of He et al. (2015) against this pooled, filtered dataset (using the span detection metrics described in Section 4). Results, shown in Table 3, show that our pipeline produces comparable precision with only a modest decrease in recall. Interestingly, re-adding the questions rejected in the validation step greatly increases recall with only a small decrease in precision, showing that validators sometimes rejected questions considered valid by the authors. However, we use the filtered dataset for our experiments, and in Section 5, we show how another crowdsourcing step can further improve recall.

3 Models

Given a sentence $\boldsymbol{X} = x_0, \dots, x_n$, the goal of a QA-SRL parser is to produce a set of tuples $(v_i, \boldsymbol{Q_i}, \mathcal{S}_i)$, where $v \in \{0, \dots, n\}$ is the index of a verbal predicate, $\boldsymbol{Q}_i$ is a question, and $\mathcal{S}_i \in \{(i,j) \mid i,j \in [0,n], j \geq i\}$ is a set of spans which are valid answers. Our proposed parsers construct these tuples in a three-step pipeline:

1. *Verbal predicates* are identified using the same POS-tags and heuristics as in data collection (see Section 2).
2. *Unlabeled span detection* selects a set $\mathcal{S}_v$ of spans as arguments for a given verb v.
3. *Question generation* predicts a question for each span in $\mathcal{S}_v$. Spans are then grouped by question, giving each question a set of answers.

We describe two models for unlabeled span detection in section 3.1, followed by question generation in section 3.2. All models are built on an LSTM encoding of the sentence. Like He et al. (2017), we start with an input $\boldsymbol{X}_v = \{\boldsymbol{x}_0 \dots \boldsymbol{x}_n\}$, where the representation $\boldsymbol{x}_i$ at each time step is a concatenation of the token w_i's embedding and an embedded binary feature $(i = v)$ which indicates whether w_i is the predicate under consideration. We then compute the output representation $\boldsymbol{H}_v = \text{BiLSTM}(\boldsymbol{X}_v)$ using a stacked alternating LSTM (Zhou and Xu, 2015) with highway connections (Srivastava et al., 2015) and recurrent dropout (Gal and Ghahramani, 2016). Since the span detection and question generation models both use an LSTM encoding, this component could in principle be shared between them. However, in preliminary experiments we found that sharing hurt performance, so for the remainder of this work each model is trained independently.

3.1 Span Detection

Given an encoded sentence $\boldsymbol{H}_v$, the goal of span detection is to select the spans $\mathcal{S}_v$ that correspond to arguments of the given predicate. We explore two models: a sequence-tagging model with BIO encoding, and a span-based model which assigns a probability to every possible span.

3.1.1 BIO Sequence Model

Our BIO model predicts a set of spans via a sequence $\boldsymbol{y}$ where each $y_i \in \{\boldsymbol{B}, \boldsymbol{I}, \boldsymbol{O}\}$, representing a token at the beginning, interior, or outside of any span, respectively. Similar to He et al.

(2017), we make independent predictions for each token at training time, and use Viterbi decoding to enforce hard BIO-constraints[5] at test time. The resulting sequences are in one-to-one correspondence with sets $\mathcal{S}_v$ of spans which are pairwise non-overlapping. The locally-normalized BIO-tag distributions are computed from the BiLSTM outputs $\boldsymbol{H}_v = \{\boldsymbol{h}_{v0}, \ldots, \boldsymbol{h}_{vn}\}$:

$$p(y_t \mid \boldsymbol{x}) \propto exp(\boldsymbol{w}_{\text{tag}}^{\intercal}\text{MLP}(\boldsymbol{h}_{vt}) + \boldsymbol{b}_{\text{tag}}) \quad (1)$$

3.1.2 Span-based Model

Our span-based model makes independent binary decisions for all $O(n^2)$ spans in the sentence. Following Lee et al. (2016), the representation of a span (i, j) is the concatenation of the BiLSTM output at each endpoint:

$$\boldsymbol{s}_{vij} = [\boldsymbol{h}_{vi}, \boldsymbol{h}_{vj}]. \quad (2)$$

The probability that the span is an argument of predicate v is computed by the sigmoid function:

$$p(y_{ij} \mid \boldsymbol{X}_v) = \sigma(\boldsymbol{w}_{\text{span}}^{\intercal}\text{MLP}(\boldsymbol{s}_{vij}) + \boldsymbol{b}_{\text{span}}) \quad (3)$$

At training time, we minimize the binary cross entropy summed over all n^2 possible spans, counting a span as a positive example if it appears as an answer to any question.

At test time, we choose a threshold τ and select every span that the model assigns probability greater than τ, allowing us to trade off precision and recall.

3.2 Question Generation

We introduce two question generation models. Given a span representation $\boldsymbol{s}_{vij}$ defined in subsubsection 3.1.2, our models generate questions by picking a word for each question slot (see Section 2). Each model calculates a joint distribution $p(\boldsymbol{y} \mid \boldsymbol{X}_v, \boldsymbol{s}_{vij})$ over values $\boldsymbol{y} = (y_1, \ldots, y_7)$ for the question slots given a span $\boldsymbol{s}_{vij}$, and is trained to minimize the negative log-likelihood of gold slot values.

3.2.1 Local Model

The local model predicts the words for each slot independently:

$$p(y_k \mid \boldsymbol{X}_v, \boldsymbol{s}_{vij}) \propto \exp(\boldsymbol{w}_k^{\intercal}\text{MLP}(\boldsymbol{s}_{vij}) + \boldsymbol{b}_k). \quad (4)$$

[5]E.g., an I-tag should only follow a B-tag.

3.2.2 Sequence Model

The sequence model uses the machinery of an RNN to share information between slots. At each slot k, we apply a multiple layers of LSTM cells:

$$\boldsymbol{h}_{l,k}, \boldsymbol{c}_{l,k} = \text{LSTMCELL}_{l,k}(\boldsymbol{h}_{l-1,k}, \boldsymbol{h}_{l,k-1}, \boldsymbol{c}_{l,k-1}) \quad (5)$$

where the initial input at each slot is a concatenation of the span representation and the embedding of the previous word of the question: $\boldsymbol{h}_{0,k} = [\boldsymbol{s}_{vij}; \boldsymbol{y}_{k-1}]$. Since each question slot predicts from a different set of words, we found it beneficial to use separate weights for the LSTM cells at each slot k. During training, we feed in the gold token at the previous slot, while at test time, we use the predicted token. The output distribution at slot k is computed via the final layers' output vector $\boldsymbol{h}_{Lk}$:

$$p(y_k \mid \boldsymbol{X}_v, \boldsymbol{s}_{vij}) \propto \exp(\boldsymbol{w}_k^{\intercal}\text{MLP}(\boldsymbol{h}_{Lk}) + \boldsymbol{b}_k) \quad (6)$$

4 Initial Results

Automatic evaluation for QA-SRL parsing presents multiple challenges. In this section, we introduce automatic metrics that can help us compare models. In Section 6, we will report human evaluation results for our final system.

4.1 Span Detection

Metrics We evaluate span detection using a modified notion of precision and recall. We count predicted spans as correct if they match any of the labeled spans in the dataset. Since each predicted span could potentially be a match to multiple questions (due to overlapping annotations) we map each predicted span to one matching question in the way that maximizes measured recall using maximum bipartite matching. We use both exact match and intersection-over-union (IOU) greater than 0.5 as matching criteria.

Results Table 4 shows span detection results on the development set. We report results for the span-based models at two threshold values τ: $\tau = 0.5$, and $\tau = \tau^*$ maximizing F1. The span-based model significantly improves over the BIO model in both precision and recall, although the difference is less pronounced under IOU matching.

4.2 Question Generation

Metrics Like all generation tasks, evaluation metrics for question generation must contend with

Exact Match			
	P	R	F
BIO	69.0	75.9	72.2
Span ($\tau = 0.5$)	81.7	80.9	81.3
Span ($\tau = \tau*$)	80.0	84.7	82.2
IOU $\geq$ 0.5			
	P	R	F
BIO	80.4	86.0	83.1
Span ($\tau = 0.5$)	87.5	84.2	85.8
Span ($\tau = \tau*$)	83.8	93.0	88.1

Table 4: Results for Span Detection on the dense development dataset. Span detection results are given with the cutoff threshold τ at 0.5, and at the value which maximizes F-score. The top chart lists precision, recall and F-score with exact span match, while the bottom reports matches where the intersection over union (IOU) is ≥ 0.5.

	EM	PM	SA
Local	44.2	62.0	83.2
Seq.	47.2	62.3	82.9

Table 5: Question Generation results on the dense development set. **EM** - Exact Match accuracy, **PM** - Partial Match Accuracy, **SA** - Slot-level accuracy

the fact that there are in general multiple possible valid questions for a given predicate-argument pair. For instance, the question "Who did someone blame something on?" may be rephrased as "Who was blamed for something?" However, due to the constrained space of possible questions defined by QA-SRL's slot format, accuracy-based metrics can still be informative. In particular, we report the rate at which the predicted question exactly matches the gold question, as well as a relaxed match where we only count the question word (WH), subject (SBJ), object (OBJ) and Miscellaneous (Misc) slots (see Table 1). Finally, we report average slot-level accuracy.

Results Table 5 shows the results for question generation on the development set. The sequential model's exact match accuracy is significantly higher, while word-level accuracy is roughly comparable, reflecting the fact that the local model learns the slot-level posteriors.

4.3 Joint results

Table 6 shows precision and recall for joint span detection and question generation, using exact match for both. This metric is exceedingly hard, but it shows that almost 40% of predictions are exactly correct in both span and question. In Section 6, we use human evaluation to get a more accurate assessment of our model's accuracy.

	P	R	F
Span + Local	37.8	43.7	40.6
Span + Seq. ($\tau = 0.5$)	39.6	45.8	42.4

Table 6: Joint span detection and question generation results on the dense development set, using exact-match for both spans and questions.

5 Data Expansion

Since our trained parser can produce full QA-SRL annotations, its predictions can be validated by the same process as in our original annotation pipeline, allowing us to focus annotation efforts towards filling potential data gaps.

By detecting spans at a low probability cutoff, we over-generate QA pairs for already-annotated sentences. Then, we filter out QA pairs whose answers overlap with answer spans in the existing annotations, or whose questions match existing questions. What remains are candidate QA pairs which fill gaps in the original annotation. We pass these questions to the validation step of our crowdsourcing pipeline with $n = 3$ validators, resulting in new labels.

We run this process on the training and development partitions of our dataset. For the development set, we use the trained model described in the previous section. For the training set, we use a relaxed version of jackknifing, training 5 models over 5 different folds. We generate 92,080 questions at a threshold of $\tau = 0.2$. Since in this case many sentences have only one question, we restructure the pay to a 2c base rate with a 2c bonus per question after the first (still paying no less than 2c per question).

Data statistics 46,017 (50%) of questions run through the expansion step were considered valid by all three annotators. In total, after filtering, the expansion step increased the number of valid questions in the train and dev partitions by 20%. However, for evaluation, since our recall metric identifies a single question for each answer span (via bipartite matching), we filter out likely question paraphrases by removing questions in the ex-

Exact Match				
	P	R	F	AUC
Original	80.8	86.8	83.7	.906
Expanded	82.9	86.4	84.6	.910
IOU ≥ 0.5				
	P	R	F	AUC
Original	87.1	93.2	90.1	.946
Expanded	87.9	93.1	90.5	.949

(a) Span Detection results with $\tau*$.

	EM	PM	WA
Original	50.5	64.4	84.1
Expanded	50.8	64.9	84.1

(b) Question Generation results

	P	R	F
Original	47.5	46.9	47.2
Expanded	44.3	55.0	49.1

(c) Joint span detection and question generation results with $\tau = 0.5$

Table 7: Results on the expanded development set comparing the full model trained on the original data, and with the expanded data.

panded development set whose answer spans have two overlaps with the answer spans of one question in the original annotations. After this filtering, the expanded development set we use for evaluation has 11.5% more questions than the original development set.

The total cost including MTurk fees was $8,210.66, for a cost of 8.9c per question, or 17.8c per valid question. While the cost per valid question was comparable to the initial annotation, we gathered many more negative examples (which may serve useful in future work), and this method allowed us to focus on questions that were missed in the first round and improve the exhaustiveness of the annotation (whereas it is not obvious how to make fully crowdsourced annotation more exhaustive at a comparable cost per question).

Retrained model We retrained our final model on the training set extended with the new valid questions, yielding modest improvements on both span detection and question generation in the development set (see Table 7). The span detection numbers are higher than on the original dataset, because the expanded development data captures true positives produced by the original model (and the resulting increase in precision can be traded off for recall as well).

6 Final Evaluation

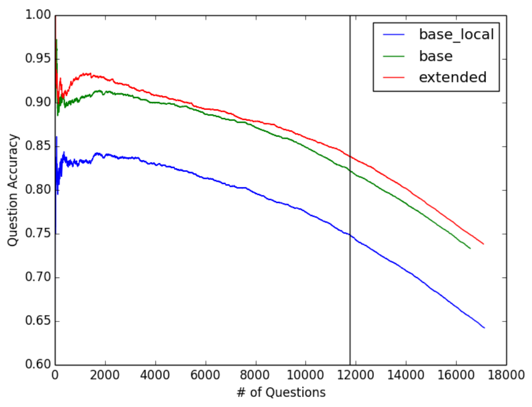
We use the crowdsourced validation step to do a final human evaluation of our models. We test 3 parsers: the span-based span detection model paired with each of the local and sequential question generation models trained on the initial dataset, and our final model (span-based span detection and sequential question generation) trained with the expanded data.

Methodology On the 5,205 sentence densely annotated subset of dev and test, we generate QA-SRL labels with all of the models using a span detection threshold of $\tau = 0.2$ and combine the questions with the existing data. We filter out questions that fail the autocomplete grammaticality check (counting them invalid) and pass the data into the validation step, annotating each question to a total of 6 validator judgments. We then compute question and span accuracy as follows: A question is considered correct if 5 out of 6 annotators consider it valid, and a span is considered correct if its generated question is correct and the span is among those selected for the question by validators. We rank all questions and spans by the threshold at which they are generated, which allows us to compute accuracy at different levels of recall.

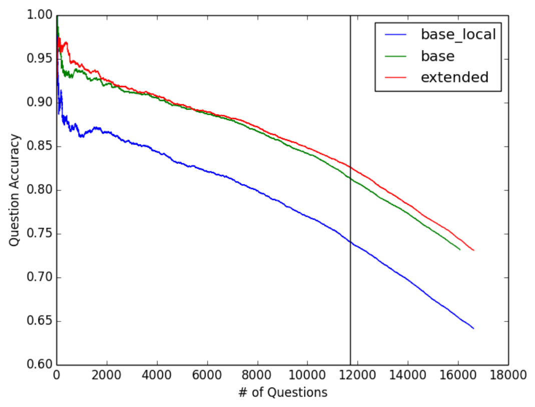
Results Figure 3 shows the results. As expected, the sequence-based question generation models are much more accurate than the local model; this is largely because the local model generated many questions that failed the grammaticality check. Furthermore, training with our expanded data results in more questions and spans generated at the same threshold. If we choose a threshold value which gives a similar number of questions per sentence as were labeled in the original data annotation (2 questions / verb), question and span accuracy are 82.64% and 77.61%, respectively.

Table 8 shows the output of our best system on 3 randomly selected sentences from our development set (one from each domain). The model was overall highly accurate—only one question and 3 spans are considered incorrect, and each mistake is nearly correct,[6] even when the sentence contains a negation.

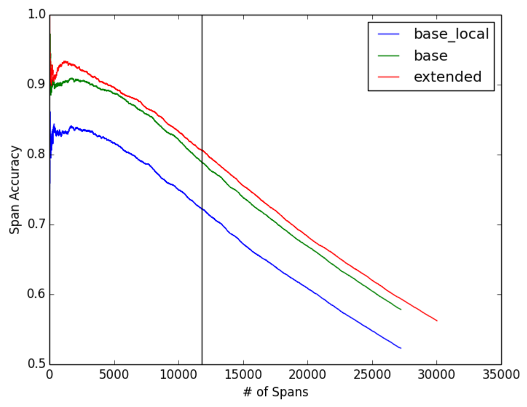
[6]The incorrect question "When did someone appear?" would be correct if the Prep and Misc slots were corrected to read "When did someone appear to do something?"



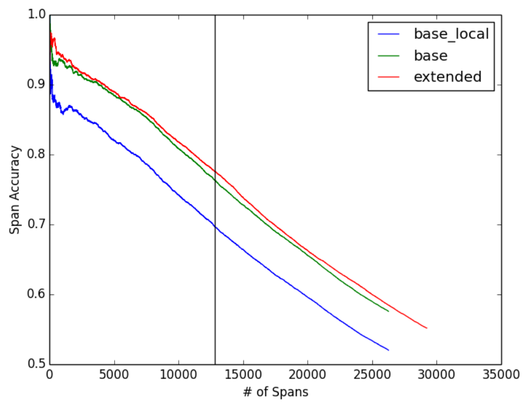
(a) Question accuracy on Dev



(b) Question accuracy on Test



(c) Span accuracy on Dev



(d) Span accuracy on Test

Figure 3: Human evaluation accuracy for questions and spans, as each model's span detection threshold is varied. Questions are considered correct if 5 out of 6 annotators consider it valid. Spans are considered correct if their question was valid, and the span was among those labeled by human annotators for that question. The vertical line indicates a threshold value where the number of questions per sentence matches that of the original labeled data (2 questions / verb).

7 Related Work

Resources and formalisms for semantics often require expert annotation and underlying syntax (Palmer et al., 2005; Baker et al., 1998; Banarescu et al., 2013). Some more recent semantic resources require less annotator training, or can be crowdsourced (Abend and Rappoport, 2013; Reisinger et al., 2015; Basile et al., 2012; Michael et al., 2018). In particular, the original QA-SRL (He et al., 2015) dataset is annotated by freelancers, while we developed streamlined crowdsourcing approaches for more scalable annotation.

Crowdsourcing has also been used for indirectly annotating syntax (He et al., 2016; Duan et al., 2016), and to complement expert annotation of SRL (Wang et al., 2018). Our crowdsourcing approach draws heavily on that of Michael et al. (2018), with automatic two-stage validation for the collected question-answer pairs.

More recently, models have been developed for these newer semantic resources, such as UCCA (Teichert et al., 2017) and Semantic Proto-Roles (White et al., 2017). Our work is the first high-quality parser for QA-SRL, which has several unique modeling challenges, such as its highly structured nature and the noise in crowdsourcing.

Several recent works have explored neural models for SRL tasks (Collobert and Weston, 2007; FitzGerald et al., 2015; Swayamdipta et al., 2017; Yang and Mitchell, 2017), many of which employ a BIO encoding (Zhou and Xu, 2015; He et al., 2017). Recently, span-based models have proven to be useful for question answering (Lee et al., 2016) and coreference resolution (Lee et al., 2017), and PropBank SRL (He et al., 2018).

A much larger super eruption in Colorado **produced** over 5,000 cubic kilometers of material.	Produced	What produced something?	A much larger super eruption
		Where did something produce something?	in Colorado
		What did something produce?	over 5,000 cubic kilometers of material
In the video, the perpetrators never **appeared** to **look** at the camera.	appeared	Where didn't someone appear to do something?	In the video
		Who didn't appear to do something?	the perpetrators
		When did someone appear?	never
		What didn't someone appear to do?	look at the camera
			to look at the camera
	look	Where didn't someone look at something?	In the video
		Who didn't look?	the perpetrators
		What didn't someone look at?	the camera
Some of the vegetarians he **met** were members of the Theosophical Society, which had been **founded** in 1875 to further universal brotherhood, and which was **devoted** to the study of Buddhist and Hindu literature.	met	Who met someone?	Some of the vegetarians
			vegetarians
		Who met?	he
		What did someone meet?	members of the Theosophical Society
	founded	What had been founded?	members of the Theosophical Society
			the Theosophical Society
		When was something founded?	in 1875
			1875
		Why has something been founded?	to further universal brotherhood
	devoted	What was devoted to something?	members of the Theosophical Society
		What was something devoted to?	the study of Buddhist and Hindu literature

Table 8: System output on 3 randomly sampled sentences from the development set (1 from each of the 3 domains). Spans were selected with $\tau = 0.5$. Questions and spans with a red background were marked incorrect during human evaluation.

8 Conclusion

In this paper, we demonstrated that QA-SRL can be scaled to large datasets, enabling a new methodology for labeling and producing predicate-argument structures at a large scale. We presented a new, scalable approach for crowdsourcing QA-SRL, which allowed us to collect QA-SRL Bank 2.0, a new dataset covering over 250,000 question-answer pairs from over 64,000 sentences, in just 9 days. We demonstrated the utility of this data by training the first parser which is able to produce high-quality QA-SRL structures. Finally, we demonstrated that the validation stage of our crowdsourcing pipeline, in combination with our parser tuned for recall, can be used to add new annotations to the dataset, increasing recall.

Acknowledgements

The crowdsourcing funds for QA-SRL Bank 2.0 was provided by the Allen Institute for Artificial Intelligence. This research was supported in part by the ARO (W911NF-16-1-0121) the NSF (IIS-1252835, IIS-1562364), a gift from Amazon, and an Allen Distinguished Investigator Award. We would like to thank Gabriel Stanovsky and Mark Yatskar for their helpful feedback.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *ACL 2013*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley Framenet project. In *ICCL 1998*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *LREC 2012*.

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. Propbank annotation guidelines.

Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *ACL 2007*.

Manjuan Duan, Ethan Hill, and Michael White. 2016. Generating disambiguating paraphrases for structurally ambiguous sentences. In *10th Linguistic Annotation Workshop*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *EMNLP 2015*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS 2016*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL 2018*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *ACL 2017*.

Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *EMNLP 2015*.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *EMNLP 2016*.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *CVPR 2017*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP 2017*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL 2014*, pages 55–60.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2018. Crowdsourcing question-answer meaning representations. In *NAACL 2018*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *TACL*.

Josef Ruppenhofer, Michael Ellsworth, Miriam RL Petruck, Christopher R Johnson, and Jan Scheffczyk. 2016. *FrameNet II: Extended theory and practice*. Institut für Deutsche Sprache, Bibliothek.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS 2015*.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *EMNLP 2016*.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Adam R Teichert, Adam Poliak, Benjamin Van Durme, and Matthew R Gormley. 2017. Semantic proto-role labeling. In *AAAI 2017*, pages 4459–4466.

Chenguang Wang, Alan Akbik, Laura Chiticariu, Yunyao Li, Fei Xia, and Anbang Xu. 2018. Crowd-in-the-loop: A hybrid approach for annotating semantic roles. In *EMNLP 2017*.

Aaron Steven White, Kyle Rawlins, and Benjamin Van Durme. 2017. The semantic proto-role linking model. In *ACL 2017*.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *EMNLP 2017*, pages 1247–1256.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL 2015*.

Syntax for Semantic Role Labeling, To Be, Or Not To Be

Shexia He[1,2,*], **Zuchao Li**[1,2,*], **Hai Zhao**[1,2,†], **Hongxiao Bai**[1,2], **Gongshen Liu**[3]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]School of Cyber Security, Shanghai Jiao Tong University, China
{heshexia, charlee}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn,
{baippa, lgshen}@sjtu.edu.cn

Abstract

Semantic role labeling (SRL) is dedicated to recognizing the predicate-argument structure of a sentence. Previous studies have shown syntactic information has a remarkable contribution to SRL performance. However, such perception was challenged by a few recent neural SRL models which give impressive performance without a syntactic backbone. This paper intends to quantify the importance of syntactic information to dependency SRL in deep learning framework. We propose an enhanced argument labeling model companying with an extended k-order argument pruning algorithm for effectively exploiting syntactic information. Our model achieves state-of-the-art results on the CoNLL-2008, 2009 benchmarks for both English and Chinese, showing the quantitative significance of syntax to neural SRL together with a thorough empirical survey over existing models.

1 Introduction

Semantic role labeling (SRL), namely semantic parsing, is a shallow semantic parsing task, which aims to recognize the predicate-argument structure of each predicate in a sentence, such as *who* did *what* to *whom*, *where* and *when*, etc. Specifically, we seek to identify arguments and label their semantic roles given a predicate. SRL is an important method to obtain semantic information beneficial to a wide range of natural language processing (NLP) tasks, including machine translation (Shi et al., 2016), question answering (Berant et al., 2013; Yih et al., 2016) and discourse relation sense classification (Mihaylov and Frank, 2016).

There are two formulizations for semantic predicate-argument structures, one is based on constituents (i.e., phrase or span), the other is based on dependencies. The latter proposed by the CoNLL-2008 shared task (Surdeanu et al., 2008) is also called semantic dependency parsing, which annotates the heads of arguments rather than phrasal arguments. Generally, SRL is decomposed into multi-step classification subtasks in pipeline systems, consisting of predicate identification and disambiguation, argument identification and classification.

In prior work of SRL, considerable attention has been paid to feature engineering that struggles to capture sufficient discriminative information, while neural network models are capable of extracting features automatically. In particular, syntactic information, including syntactic tree feature, has been show extremely beneficial to SRL since a larger scale of empirical verification of Punyakanok et al. (2008). However, all the work had to take the risk of erroneous syntactic input, leading to an unsatisfactory performance.

To alleviate the above issues, Marcheggiani et al. (2017) propose a simple but effective model for dependency SRL without syntactic input. It seems that neural SRL does not have to rely on syntactic features, contradicting with the belief that syntax is a necessary prerequisite for SRL as early as Gildea and Palmer (2002). This dramatic contradiction motivates us to make a thorough exploration on syntactic contribution to SRL.

This paper will focus on semantic dependency parsing and formulate SRL as one or two se-

[*] These authors made equal contribution.[†] Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

quence tagging tasks with predicate-specific encoding. With the help of the proposed k-order argument pruning algorithm over syntactic tree, our model obtains state-of-the-art scores on the CoNLL benchmarks for both English and Chinese.

In order to quantitatively evaluate the contribution of syntax to SRL, we adopt the ratio between labeled F_1 score for semantic dependencies (Sem-F_1) and the labeled attachment score (LAS) for syntactic dependencies introduced by CoNLL-2008 Shared Task[1] as evaluation metric. Considering that various syntactic parsers contribute different syntactic inputs with various range of quality levels, the ratio provides a fairer comparison between syntactically-driven SRL systems, which will be surveyed by our empirical study.

2 Model

To fully disclose the predicate-argument structure, typical SRL systems have to step by step perform four subtasks. Since the predicates in CoNLL-2009 (Hajič et al., 2009) corpus have been pre-identified, we need to tackle three other subtasks, which are formulized into two-step pipeline in this work, predicate disambiguation and argument labeling. Namely, we do the work of argument identification and classification in one model.

Argument structure for each known predicate will be disclosed by our argument labeler over a sequence including possible arguments (candidates). There are two ways to determine the sequence, one is to simply input the entire sentence as a syntax-agnostic SRL system does, the other is to select words according to syntactic parse tree around the predicate as most previous SRL systems did. The latter strategy usually works through a syntactic tree based argument pruning algorithm. We will use the proposed k-order argument pruning algorithm (Section 2.1) to get a sequence $w = (w_1, \dots, w_n)$ for each predicate. Then, we represent each word $w_i \in w$ as x_i (Section 2.2). Eventually, we obtain contextual features with sequence encoder (Section 2.3). The overall role labeling model is depicted in Figure 1.

2.1 Argument Pruning

As pointed out by Punyakanok et al. (2008), syntactic information is most relevant in identifying

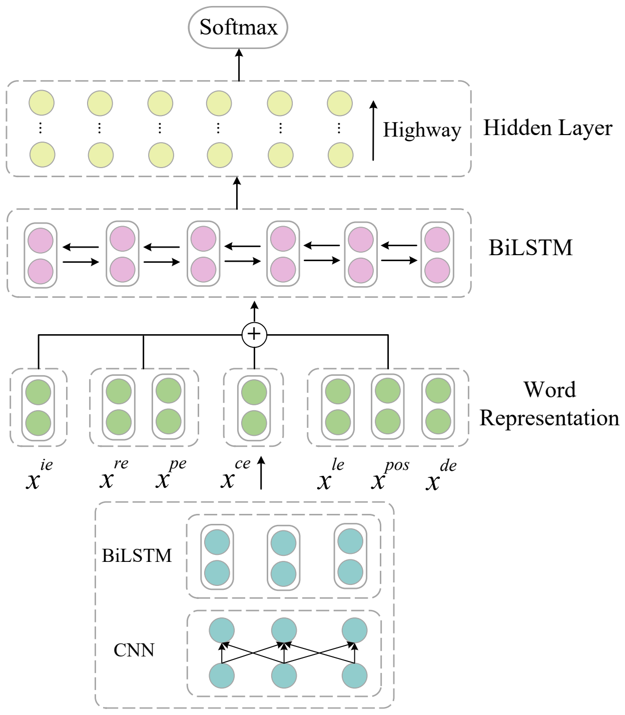


Figure 1: The Argument Labeling Model

the arguments, and the most crucial contribution of full parsing is in the pruning stage. In this paper, we propose a k-order argument pruning algorithm inspired by Zhao et al. (2009b). First of all, for node n and its descendant n_d in a syntactic dependency tree, we define the *order* to be the *distance* between the two nodes, denoted as $\mathcal{D}(n, n_d)$. Then we define k-order descendants of given node satisfying $\mathcal{D}(n, n_d) = k$, and k-order traversal that visits each node from the given node to its descendant nodes within k-th order. Note that the definition of k-order traversal is somewhat different from tree traversal in terminology.

A brief description of the proposed k-order pruning algorithm is given as follow. Initially, we set a given predicate as the current node in a syntactic dependency tree. Then, collect all its argument candidates by the strategy of k-order traversal. Afterwards, reset the current node to its syntactic head and repeat the previous step till the root of the tree. Finally, collect the root and stop. The k-order argument algorithm is presented in Algorithm 1 in detail. An example of a syntactic dependency tree for sentence *She began to trade the art for money* is shown in Figure 2.

The main reasons for applying the extended k-order argument pruning algorithm are two-fold.

[1]CoNLL-2008 is an English-only task, while CoNLL-2009 extends to a multilingual one. Their main difference is that predicates have been beforehand indicated for the latter.

Algorithm 1 k-order argument pruning algorithm

Input: A predicate p, the root node r given a syntactic dependency tree T, the order k

Output: The set of argument candidates S

```
1: initialization set $p$ as current node $c$, $c = p$
2: for each descendant $n_i$ of $c$ in $T$ do
3:     if $\mathcal{D}(c, n_i) \leq k$ and $n_i \notin S$ then
4:         $S = S + n_i$
5:     end if
6: end for
7: find the syntactic head $c_h$ of $c$, and let $c = c_h$
8: if $c = r$ then
9:     $S = S + r$
10: else
11:     goto step 2
12: end if
13: return argument candidates set $S$
```

First, previous standard pruning algorithm may hurt the argument coverage too much, even though indeed arguments usually tend to surround their predicate in a close distance. As a sequence tagging model has been applied, it can effectively handle the imbalanced distribution between arguments and non-arguments, which is hardly tackled by early argument classification models that commonly adopt the standard pruning algorithm. Second, the extended pruning algorithm provides a better trade-off between computational cost and performance by carefully tuning k.

2.2 Word Representation

We produce a predicate-specific word representation x_i for each word w_i, where i stands for the word position in an input sequence, following Marcheggiani et al. (2017). However, we differ by (1) leveraging a predicate-specific indicator embedding, (2) using deeper refined representation, including character and dependency relation embeddings, and (3) applying recent advances in RNNs, such as highway connections (Srivastava et al., 2015).

In this work, word representation x_i is the concatenation of four types of features: predicate-specific feature, character-level, word-level and linguistic features. Unlike previous work, we leverage a predicate-specific indicator embedding x_i^{ie} rather than directly using a binary flag either 0 or 1. At character level, we exploit convolutional neural network (CNN) with bidirectional LSTM (BiLSTM) to learn character embedding

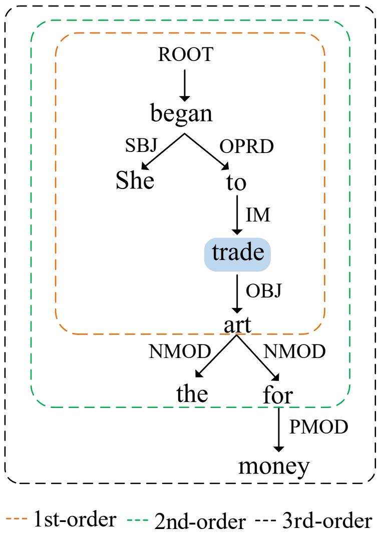


Figure 2: An example of *first*-order, *second*-order and *third*-order argument pruning. Shadow part indicates the given predicate.

x_i^{ce}. As shown in Figure 1, the representation calculated by the CNN is fed as input to BiLSTM. At word level, we use a randomly initialized word embedding x_i^{re} and a pre-trained word embedding x_i^{pe}. For linguistic features, we employ a randomly initialized lemma embedding x_i^{le} and a randomly initialized POS tag embedding x_i^{pos}. In order to incorporate more syntactic information, we adopt an additional feature, the dependency relation to syntactic head. Likewise, it is a randomly initialized embedding x_i^{de}. The resulting word representation is concatenated as $x_i = [x_i^{ie}, x_i^{ce}, x_i^{re}, x_i^{pe}, x_i^{le}, x_i^{pos}, x_i^{de}]$.

2.3 Sequence Encoder

As Long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) have shown significant representational effectiveness to NLP tasks, we thus use BiLSTM as the sentence encoder. Given an input sequence $x = (x_1, \ldots, x_n)$, BiLSTM processes the sequence in both forward and backward direction to obtain two separated hidden states, $\overrightarrow{h}_i$ which handles data from x_1 to x_i and $\overleftarrow{h}_i$ which tackles data from x_n to x_i for each word representation. Finally, we get a contextual representation $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$ by concatenating the states of BiLSTM networks.

To get the final predicted semantic roles, we exploit a multi-layer perceptron (MLP) with highway connections on the top of BiLSTM networks, which takes as input the hidden representation h_i

Hyperparameter	values
d^{ie} (indicator embedding)	16
d^{pe} (pre-trained embedding)	100
d^{ce} (character embedding)	300
d^{re} (word embedding)	100
d^{le} (lemma embedding)	100
d^{pos} (POS tag embedding)	32
d^{de} (dependency label embedding)	64
LSTM hidden sizes	512
BiLSTM layers	4
Hidden layers	10
Learning rate	0.001
Word dropout	0.1

Table 1: Hyperparameter values.

of all time steps. The MLP network consists of 10 layers with highway connections and we employ *ReLU* activations for the hidden layers. Finally, we use a softmax layer over the outputs to maximize the likelihood of labels.

2.4 Predicate Disambiguation

Although predicates have been identified given a sentence, predicate disambiguation is an indispensable task, which aims to determine the predicate-argument structure for an identified predicate in a particular context. Here, we also use the identical model (BiLSTM composed with MLP) for predicate disambiguation, in which the only difference is that we remove the syntactic dependency relation feature in corresponding word representation (Section 2.2). Exactly, given a predicate p, the resulting word representation is $p_i = [p_i^{ie}, p_i^{ce}, p_i^{re}, p_i^{pe}, p_i^{le}, p_i^{pos}]$.

3 Experiments

Our model[2] is evaluated on the CoNLL-2009 shared task both for English and Chinese datasets, following the standard training, development and test splits. The hyperparameters in our model were selected based on the development set, and are summarized in Table 1. Note that the parameters of predicate model are the same as these in argument model. All real vectors are randomly initialized, and the pre-trained word embeddings for English are GloVe vectors (Pennington et al., 2014). For Chinese, we exploit Wikipedia documents to train Word2Vec embeddings (Mikolov et al., 2013). During training procedures, we use the categorical cross-entropy as objective, with Adam optimizer (Kingma and Ba, 2015). We train models for a maximum of 20 epochs and obtain the nearly best model based on development results. For argument labeling, we preprocess corpus with k-order argument pruning algorithm. In addition, we use four CNN layers with single-layer BiLSTM to induce character representations derived from sentences. For English[3], to further enhance the representation, we adopt CNN-BiLSTM character embedding structure from AllenNLP toolkit (Peters et al., 2018).

[2]The code is available at `https://github.com/bcmi220/srl_syn_pruning`.

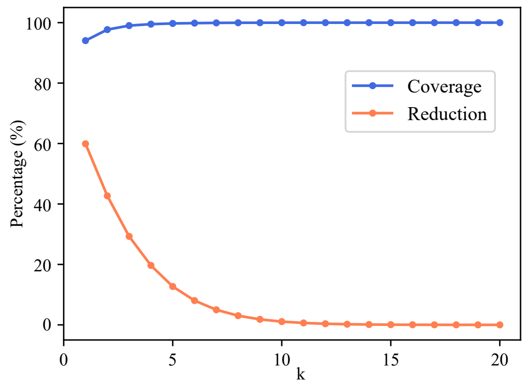


Figure 3: Changing curves of coverage and reduction with different k value on English training set. The coverage rate is the proportion of true arguments in pruning output, while the reduction is the one of pruned argument candidates in total tokens.

3.1 Preprocessing

During the pruning of argument candidates, we use the officially predicted syntactic parses provided by CoNLL-2009 shared-task organizers on both English and Chinese. Figure 3 shows changing curves of coverage and reduction following k on the English train set. According to our statistics, the number of non-arguments is ten times more than that of arguments, where the data distribution is fairly unbalanced. However, a proper pruning strategy could alleviate this problem. Accordingly, the first-order pruning reduces more than 50% candidates at the cost of missing 5.5% true ones on average, and the second-order prunes about 40% candidates with nearly 2.0% loss. The coverage of third-order has achieved 99% and it reduces approximately 1/3 corpus size.

It is worth noting that as k is larger than 19,

[3]For Chinese, we do not use character embedding.

System (syntax-aware)	P	R	F_1
Single model			
Zhao et al. (2009a)	–	–	86.2
Zhao et al. (2009c)	–	–	85.4
Björkelund et al. (2010)	87.1	84.5	85.8
Lei et al. (2015)	–	–	86.6
FitzGerald et al. (2015)	–	–	86.7
Roth and Lapata (2016)	88.1	85.3	86.7
Marcheggiani and Titov (2017)	89.1	86.8	88.0
Ours	**89.7**	**89.3**	**89.5**
Ensemble model			
FitzGerald et al. (2015)	–	–	87.7
Roth and Lapata (2016)	90.3	85.7	87.9
Marcheggiani and Titov (2017)	90.5	87.7	89.1
System (syntax-agnostic)	P	R	F_1
Marcheggiani et al. (2017)	88.7	86.8	87.7
Ours	**89.5**	**87.9**	**88.7**

Table 2: Results on the English test set (WSJ).

System (syntax-aware)	P	R	F_1
Single model			
Zhao et al. (2009a)	–	–	74.6
Zhao et al. (2009c)	–	–	73.3
Björkelund et al. (2010)	75.7	72.2	73.9
Lei et al. (2015)	–	–	75.6
FitzGerald et al. (2015)	–	–	75.2
Roth and Lapata (2016)	76.9	73.8	75.3
Marcheggiani and Titov (2017)	78.5	75.9	77.2
Ours	**81.9**	**76.9**	**79.3**
Ensemble model			
FitzGerald et al. (2015)	–	–	75.5
Roth and Lapata (2016)	79.7	73.6	76.5
Marcheggiani and Titov (2017)	80.8	77.1	78.9
System (syntax-agnostic)	P	R	F_1
Marcheggiani et al. (2017)	79.4	76.2	77.7
Ours	**81.7**	**76.1**	**78.8**

Table 3: Results on English out-of-domain test set (Brown).

there will come full coverage on all argument candidates for English training set, which let our high order pruning algorithm degrade into a syntax-agnostic setting. In this work, we use the tenth-order pruning for pursuing the best performance.

3.2 Results

Our system performance is measured with the official script from CoNLL-2009 benchmarks, combining the output of our predicate disambiguation with our semantic role labeling. Our predicate disambiguation model achieves the accuracy of 95.01% and 95.58%[4] on development and test sets, respectively. We compare our model performance with the state-of-the-art models for dependency SRL.[5] Noteworthily, our model is local and single without reranking, which neither includes global inference nor combines multiple models. The experimental results on the English in-domain (WSJ) and out-of-domain (Brown) test sets are shown in Tables 2 and 3, respectively.

For English, our syntax-aware model outperforms previously published best single model, scoring 89.5% F_1 with 1.5% absolute improvement on the in-domain (WSJ) test data. Compared with ensemble models, our single model even provides better performance (+0.4% F_1) than the system (Marcheggiani and Titov, 2017), and significantly surpasses all the rest models. In the syntax-agnostic setting (without pruning and dependency relation embedding), we also reach the new state-of-the-art, achieving a performance gain of 1% F_1.

System (syntax-aware)	P	R	F_1
Zhao et al. (2009a)	80.4	75.2	77.7
Björkelund et al. (2009)	82.4	75.1	78.6
Roth and Lapata (2016)	83.2	75.9	79.4
Marcheggiani and Titov (2017)	84.6	80.4	82.5
Ours	**84.2**	**81.5**	**82.8**
System (syntax-agnostic)	P	R	F_1
Marcheggiani et al. (2017)	83.4	79.1	81.2
Ours	**84.5**	**79.3**	**81.8**

Table 4: Results on the Chinese test set.

On the out-of-domain (Brown) test set, we achieve the new best results of 79.3% (syntax-aware) and 78.8% (syntax-agnostic) in F_1 scores. Moreover, our syntax-aware model performs better than the syntax-agnostic one.

Table 4 presents the results on Chinese test set. Even though we use the same parameters as for English, our model also outperforms the best reported results by 0.3% (syntax-aware) and 0.6% (syntax-agnostic) in F_1 scores.

[4]Note that we give a slightly better predicate model than Roth and Lapata (2016), with 94.77% and 95.47% accuracy on development and test sets, respectively.

[5]Here, we do not compare against span-based SRL models, which annotate roles for entire argument spans instead of semantic dependencies.

System(without predicate sense)	P	R	F_1
1st-order	84.4	82.6	83.5
2nd-order	84.8	83.0	83.9
3rd-order	85.1	83.3	84.2
Marcheggiani and Titov (2017)	85.2	81.6	83.3

Table 5: SRL results without predicate sense.

Our system	P	R	F_1
BiLSTM	86.5	85.1	85.8
basic model	86.3	85.7	86.0
+ indicator embedding	86.8	85.8	86.3
+ character embedding	87.2	86.6	86.9
+ both	87.7	87.0	87.3
BiLSTM + both	87.3	86.7	87.0

Table 6: Ablation on development set. The "+" denotes a specific version over the basic model.

3.3 Analysis

To evaluate the contributions of key factors in our method, a series of ablation studies are performed on the English development set.

In order to demonstrate the effectiveness of our k-order pruning algorithm, we report the SRL performance excluding predicate senses in evaluation, eliminating the performance gain from predicate disambiguation. Table 5 shows the results from our syntax-aware model with lower order argument pruning. Compared to the best previous model, our system still yields an increment in recall by more than 1%, leading to improvements in F_1 score. It demonstrates that refining syntactic parser tree based candidate pruning does help in argument recognition.

Table 6 presents the performance of our syntax-agnostic SRL system with a basic configuration, which removes components, including indicator and character embeddings. Note that the first row is the results of BiLSTM (removing MLP from basic model), whose encoding is the same as Marcheggiani et al. (2017). Experiments show that both enhanced representations improve over our basic model, and our adopted labeling model is superior to the simple BiLSTM.

Figure 4 shows F_1 scores in different k-order pruning together with our syntax-agnostic model. It also indicates that the least first-order pruning fails to give satisfactory performance, the best performing setting coming from a moderate setting of $k = 10$, and the largest k shows that our argu-

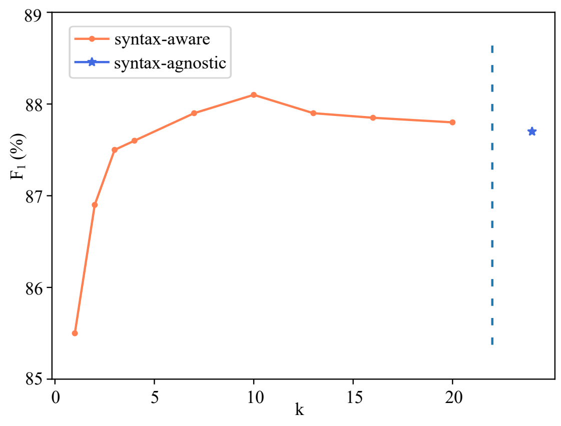


Figure 4: F_1 scores by k-order pruning and the syntax-agnostic result on English development set.

ment pruning falls back to syntax-agnostic type. Meanwhile, from the best k setting to the lower order pruning, we receive a much faster performance drop, compared to the higher order pruning until the complete syntax-agnostic case. The proposed k-order pruning algorithm always works even it reaches the syntax-agnostic setting, which empirically explains why the current syntax-aware and syntax-agnostic SRL models hold little performance difference, as maximum k-order pruning actually removes few words just like syntax-agnostic model.

3.4 End-to-end SRL

In this work, we consider additional model that integrates predicate disambiguation and argument labeling into one sequence labeling model. In order to implement an end-to-end model, we introduce a virtual root (VR) for predicate disambiguation similar to Zhao et al. (2013) who handled the entire SRL task as word pair classification. Concretely, we add a predicate sense feature to the input sequence by concatenating a VR. The word representation of VR is randomly initialized during training. In Figure 5, we give an example sequence with the labels for the given sentence.

We also report results of our end-to-end model on CoNLL-2009 test set with syntax-aware and syntax-agnostic settings. As shown in Table 7, our end-to-end model yields slightly weaker performance compared with our pipeline. A reasonable account for performance degradation is that the training data has completely different genre distributions over predicate senses and argument roles, which may be somewhat confusing for integrative model to make classification decisions.

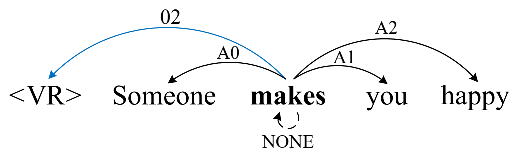


Figure 5: An example sequence with labels of end-to-end model (*makes* is the given predicate).

Our system	P	R	F_1
syntax-aware (end-to-end)	89.3	88.7	89.0
syntax-aware (pipeline)	89.7	89.3	89.5
syntax-agnostic (end-to-end)	88.9	87.9	88.4
syntax-agnostic (pipeline)	89.5	87.9	88.7

Table 7: Comparison of results on CoNLL-2009 data between our end-to-end and pipeline models.

3.5 CoNLL-2008 SRL Setting

For a full SRL task, the predicate identification subtask is also indispensable, which has been included in CoNLL-2008 shared task. We thus evaluate our model in terms of data and setting of the CoNLL-2008 benchmark (WSJ).

To identify predicates, we train the BiLSTM-MLP sequence labeling model with same parameters in Section 2.4 to tackle the predicate identification and disambiguation subtasks in one shot, and the only difference is that we remove the predicate-specific indicator feature. The F_1 score of our predicate labeling model is 90.53% on in-domain (WSJ) data. Compared with the best reported results, we observe absolute improvements in semantic F_1 of 0.8% (in Table 8). Note that as predicate identification is introduced, our same model shows about 6% performance loss for either syntax-agnostic or syntax-aware case, which indicates that predicate identification should be carefully handled, as it is very needed in a complete practical SRL system.

4 Syntactic Contribution

Syntactic information plays an informative role in semantic role labeling. However, few studies were done to quantitatively evaluate the syntactic contribution to SRL. Furthermore, we observe that most of the above compared neural SRL systems took the syntactic parser of (Björkelund et al., 2010) as syntactic inputs instead of the one from CoNLL-2009 shared task, which adopted a much weaker syntactic parser. Especially (Marcheggiani and Titov, 2017), adopted an external syntactic parser with even higher parsing accuracy. Contrarily, our SRL model is based on the automatically predicted parse with moderate performance provided by CoNLL-2009 shared task, but outperforms their models.

System	LAS	Sem-F_1
Johansson and Nugues (2008)	90.13	81.75
Zhao and Kit (2008)	87.52	77.67
Zhao et al. (2009b)	88.39	82.1 (80.53)
	89.28	82.5 (80.94)
Zhao et al. (2013)	88.39	82.5 (80.91)
	89.28	82.4 (80.88)
Ours (syntax-agnostic)	–	82.9
Ours (syntax-aware)	86.0	83.3

Table 8: Results on the CoNLL-2008 in-domain (WSJ) test set. The results in parenthesis are on WSJ + Brown test set.

This section thus attempts to explore how much syntax contributes to dependency-based SRL in deep learning framework and how to effectively evaluate relative performance of syntax-based SRL. To this end, we conduct experiments for empirical analysis with different syntactic inputs.

Syntactic Input In order to obtain different syntactic inputs, we design a faulty syntactic tree generator (refer to STG hereafter), which is able to produce random errors in the output parse tree like a true parser does. To simplify implementation, we construct a new syntactic tree based on the gold standard parse tree. Given an input error probability distribution estimated from a true parser output, our algorithm presented in Algorithm 2 stochastically modifies the syntactic heads of nodes on the premise of a valid tree.

Evaluation Measure For SRL task, the primary evaluation measure is the semantic labeled F_1 score. However, the score is influenced by the quality of syntactic input to some extent, leading to unfaithfully reflecting the competence of syntax-based SRL system. Namely, this is not the outcome of a true and fair quantitative comparison for these types of SRL models. To normalize the semantic score relative to syntactic parse, we take into account additional evaluation measure to estimate the actual overall performance of SRL. Here, we use the ratio between labeled F_1 score for semantic dependencies (Sem-F_1) and the labeled attachment score (LAS) for syntactic dependencies

System	LAS (%)	P (%)	R (%)	Sem-F_1 (%)	Sem-F_1/LAS (%)
Zhao et al. (2009c) [SRL-only]	86.0	–	–	85.4	99.3
Zhao et al. (2009a) [Joint]	89.2	–	–	86.2	96.6
Björkelund et al. (2010)	89.8	87.1	84.5	85.8	95.6
Lei et al. (2015)	90.4	–	–	86.6	95.8
Roth and Lapata (2016)	89.8	88.1	85.3	86.7	96.5
Marcheggiani and Titov (2017)	90.3*	89.1	86.8	**88.0**	97.5
Ours + CoNLL-2009 predicted	86.0	89.7	89.3	89.5	104.0
Ours + Auto syntax	90.0	90.5	89.3	**89.9**	99.9
Ours + Gold syntax	100	91.0	89.7	90.3	90.3

Table 9: Results on English test set, in terms of labeled attachment score for syntactic dependencies (LAS), semantic precision (P), semantic recall (R), semantic labeled F_1 score (Sem-F_1), the ratio Sem-F_1/LAS. A superscript * indicates LAS results from our personal communication with the authors.

Algorithm 2 Faulty Syntactic Tree Generator

```
Input: A gold standard syntactic tree $GT$, the specific error probability $p$
Output: The new generative syntactic tree $NT$
 1: $N$ denotes the number of nodes in $GT$
 2: for each node $n \in GT$ do
 3:     $r$ = random(0, 1), a random number
 4:     if $r < p$ then
 5:         $h$ = random(0, $N$), a random integer
 6:         find the syntactic head $n_h$ of $n$ in $GT$
 7:         modify $n_h = h$, and get a new tree $NT$
 8:         if $NT$ is a valid tree then
 9:             break
10:         else
11:             goto step 5
12:         end if
13:     end if
14: end for
15: return the new generative tree $NT$
```

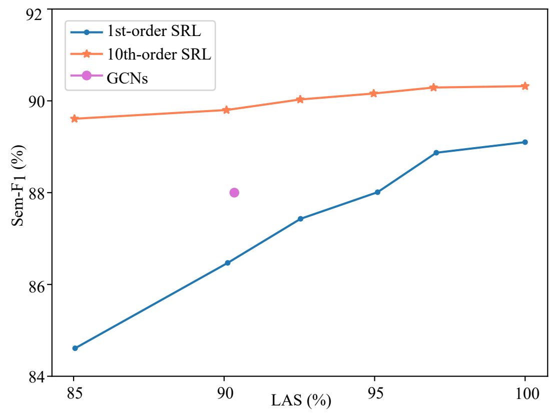


Figure 6: The Sem-F_1 scores of our models with different quality of syntactic inputs vs. GCNs (Marcheggiani and Titov, 2017) on test set.

proposed by Surdeanu et al. (2008) as evaluation metric.[6] The benefits of this measure are twofold: quantitatively evaluating syntactic contribution to SRL and impartially estimating the true performance of SRL, independent of the performance of the input syntactic parser.

Table 9 reports the performance of existing models[7] in term of Sem-F_1/LAS ratio on CoNLL-2009 English test set. Interestingly, even though our system has significantly lower scores than others by 3.8% LAS in syntactic components, we obtain the highest results both on Sem-F_1 and the Sem-F_1/LAS ratio, respectively. These results show that our SRL component is relatively much stronger. Moreover, the ratio comparison in Table 9 also shows that since the CoNLL-2009 shared task, most SRL works actually benefit from the enhanced syntactic component rather than the improved SRL component itself. All post-CoNLL SRL systems, either traditional or neural types, did not exceed the top systems of CoNLL-2009 shared task, (Zhao et al., 2009c) (SRL-only track using the provided predicated syntax) and (Zhao et al., 2009a) (Joint track using self-developed parser). We believe that this work for the first time reports both higher Sem-F_1 and higher Sem-F_1/LAS ratio since CoNLL-2009 shared task.

We also perform our first and tenth order pruning models with different erroneous syntactic inputs generated from STG and evaluate their per-

[6]The idea of ratio score in Surdeanu et al. (2008) actually was from author of this paper, Hai Zhao, which has been indicated in the acknowledgement part of Surdeanu et al. (2008).

[7]Note that several SRL systems without providing syntactic information are not listed in the table.

formance using the Sem-F_1/LAS ratio. Figure 6 shows Sem-F_1 scores at different quality of syntactic parse inputs on the English test set whose LAS varies from 85% to 100%. Compared to previous state-of-the-arts (Marcheggiani and Titov, 2017). Our tenth-order pruning model gives quite stable SRL performance no matter the syntactic input quality varies in a broad range, while our first-order pruning model yields overall lower results (1-5% F_1 drop), owing to missing too many true arguments. These results show that high-quality syntactic parses may indeed enhance dependency SRL. Furthermore, it indicates that our model with an accurate enough syntactic input as Marcheggiani and Titov (2017), namely, 90% LAS, will give a Sem-F_1 exceeding 90% for the first time in the research timeline of semantic role labeling.

5 Related Work

Semantic role labeling was pioneered by Gildea and Jurafsky (2002). Most traditional SRL models rely heavily on feature templates (Pradhan et al., 2005; Zhao et al., 2009b; Björkelund et al., 2009). Among them, Pradhan et al. (2005) combined features derived from different syntactic parses based on SVM classifier, while Zhao et al. (2009b) presented an integrative approach for dependency SRL by greedy feature selection algorithm. Later, Collobert et al. (2011) proposed a convolutional neural network model of inducing word embeddings substituting for hand-crafted features, which was a breakthrough for SRL task.

With the impressive success of deep neural networks in various NLP tasks (Zhang et al., 2016; Qin et al., 2017; Cai et al., 2017), a series of neural SRL systems have been proposed. Foland and Martin (2015) presented a dependency semantic role labeler using convolutional and time-domain neural networks, while FitzGerald et al. (2015) exploited neural network to jointly embed arguments and semantic roles, akin to the work (Lei et al., 2015), which induced a compact feature representation applying tensor-based approach. Recently, researchers consider multiple ways to effectively integrate syntax into SRL learning. Roth and Lapata (2016) introduced dependency path embedding to model syntactic information and exhibited a notable success. Marcheggiani and Titov (2017) leveraged the graph convolutional network to incorporate syntax into neural models. Differently, Marcheggiani et al. (2017) proposed a syntax-agnostic model using effective word representation for dependency SRL, which for the first time achieves comparable performance as state-of-the-art syntax-aware SRL models.

However, most neural SRL works seldom pay much attention to the impact of input syntactic parse over the resulting SRL performance. This work is thus more than proposing a high performance SRL model through reviewing the highlights of previous models, and presenting an effective syntactic tree based argument pruning. Our work is also closely related to (Punyakanok et al., 2008; He et al., 2017). Under the traditional methods, Punyakanok et al. (2008) investigated the significance of syntax to SRL system and shown syntactic information most crucial in the pruning stage. He et al. (2017) presented extensive error analysis with deep learning model for span SRL, including discussion of how constituent syntactic parser could be used to improve SRL performance.

6 Conclusion and Future Work

This paper presents a simple and effective neural model for dependency-based SRL, incorporating syntactic information with the proposed extended k-order pruning algorithm. With a large enough setting of k, our pruning algorithm will result in a syntax-agnostic setting for the argument labeling model, which smoothly unifies syntax-aware and syntax-agnostic SRL in a consistent way. Experimental results show that with the help of deep enhanced representation, our model outperforms the previous state-of-the-art models in both syntax-aware and syntax-agnostic situations.

In addition, we consider the Sem-F1/LAS ratio as a mean of evaluating syntactic contribution to SRL, and true performance of SRL independent of the quality of syntactic parser. Though we again confirm the importance of syntax to SRL with empirical experiments, we are aware that since (Pradhan et al., 2005), the gap between syntax-aware and syntax-agnostic SRL has been greatly reduced, from as high as 10% to only 1-2% performance loss in this work. However, maybe we will never reach a satisfying conclusion, as whenever one proposes a syntax-agnostic SRL system which can outperform all syntax-aware ones at then, always there comes argument that you have never fully explored creative new method to effectively exploit the syntax input.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Seattle, Washington, USA, pages 1533–1544.

Anders Björkelund, Bohnet Bernd, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics (CoLING 2010)*. Beijing, China, pages 33–36.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Boulder, Colorado, pages 43–48.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada, pages 608–615.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(1):2493–2537.

Nicholas FitzGerald, Oscar Tckstrm, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 960–970.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Joint Conference on Lexical and Computational Semantics*. pages 279–288.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, Pennsylvania, USA, pages 239–246.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Boulder, Colorado, pages 1–18.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada, pages 473–483.

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

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL)*. pages 183–187.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*. pages 1150–1160.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Copenhagen, Denmark, pages 1506–1515.

Todor Mihaylov and Anette Frank. 2016. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*. Berlin, Germany, pages 100–107.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3111–3119.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*. New Orleans, Louisiana.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. Ann Arbor, Michigan, pages 581–588.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics* 34(2):257–287.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada, pages 1006–1017.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 1192–1202.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 2245–2254.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*. pages 2377–2385.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*. Manchester, England, pages 159–177.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 201–206.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 1382–1392.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*. Boulder, Colorado, pages 61–66.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Singapore, pages 30–39.

Hai Zhao, Wenliang Chen, and Guodong Zhou. 2009c. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*. Boulder, Colorado, pages 55–60.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL)*. pages 203–207.

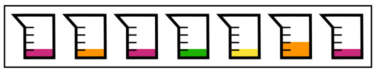
Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research* 46:203–233.

Situated Mapping of Sequential Instructions to Actions with Single-step Reward Observation

Alane Suhr and **Yoav Artzi**
Department of Computer Science and Cornell Tech
Cornell University
New York, NY, 10044
`{suhr, yoav}@cs.cornell.edu`

Abstract

We propose a learning approach for mapping context-dependent sequential instructions to actions. We address the problem of discourse and state dependencies with an attention-based model that considers both the history of the interaction and the state of the world. To train from start and goal states without access to demonstrations, we propose SESTRA, a learning algorithm that takes advantage of single-step reward observations and immediate expected reward maximization. We evaluate on the SCONE domains, and show absolute accuracy improvements of 9.8%-25.3% across the domains over approaches that use high-level logical representations.



throw out first beaker
`POP 1, STOP`
pour sixth beaker into last one
`POP 6, POP 6, PUSH 7 O, PUSH 7 O, STOP`
it turns brown
`POP 7, POP 7, POP 7, PUSH 7 B, PUSH 7 B, PUSH 7 B, STOP`
pour purple beaker into yellow one
`POP 3, PUSH 5 P, STOP`
throw out two units of brown one
`POP 7, POP 7, STOP`

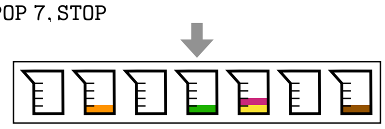


Figure 1: Example from the SCONE (Long et al., 2016) ALCHEMY domain, including a start state (top), sequence of instructions, and a goal state (bottom). Each instruction is annotated with a sequence of actions from the set of actions we define for ALCHEMY.

1 Introduction

An agent executing a sequence of instructions must address multiple challenges, including grounding the language to its observed environment, reasoning about discourse dependencies, and generating actions to complete high-level goals. For example, consider the environment and instructions in Figure 1, in which a user describes moving chemicals between beakers and mixing chemicals together. To execute the second instruction, the agent needs to resolve *sixth beaker* and *last one* to objects in the environment. The third instruction requires resolving *it* to the rightmost beaker mentioned in the second instruction, and reasoning about the set of actions required to mix the colors in the beaker to brown. In this paper, we describe a model and learning approach to map sequences of instructions to actions. Our model considers previous utterances and the world state to select actions, learns to combine simple actions to achieve complex goals, and can be trained using goal states without access to demonstrations.

The majority of work on executing sequences of instructions focuses on mapping instructions to high-level formal representations, which are then evaluated to generate actions (e.g., Chen and Mooney, 2011; Long et al., 2016). For example, the third instruction in Figure 1 will be mapped to mix(prev_arg1), indicating that the mix action should be applied to first argument of the previous action (Long et al., 2016; Guu et al., 2017). In contrast, we focus on directly generating the sequence of actions. This requires resolving references without explicitly modeling them, and learning the sequences of actions required to complete high-level actions; for example, that mixing requires removing everything in the beaker and replacing with the same number of brown items.

A key challenge in executing sequences of instructions is considering contextual cues from both the history of the interaction and the state of the world. Instructions often refer to previously mentioned objects (e.g., *it* in Figure 1) or actions (e.g., *do it again*). The world state provides the set of objects the instruction may refer to, and implicitly determines the available actions. For example, liquid can not be removed from an empty beaker. Both types of contexts continuously change during an interaction. As new instructions are given, the instruction history expands, and as the agent acts the world state changes. We propose an attention-based model that takes as input the current instruction, previous instructions, the initial world state, and the current state. At each step, the model computes attention encodings of the different inputs, and predicts the next action to execute.

We train the model given instructions paired with start and goal states without access to the correct sequence of actions. During training, the agent learns from rewards received through exploring the environment with the learned policy by mapping instructions to sequences of actions. In practice, the agent learns to execute instructions gradually, slowly correctly predicting prefixes of the correct sequences of increasing length as learning progress. A key challenge is learning to correctly select actions that are only required later in execution sequences. Early during learning, these actions receive negative updates, and the agent learns to assign them low probabilities. This results in an exploration problem in later stages, where actions that are only required later are not sampled during exploration. For example, in the ALCHEMY domain shown in Figure 1, the agent behavior early during execution of instructions can be accomplished by only using `POP` actions. As a result, the agent quickly learns a strong bias against `PUSH` actions, which in practice prevents the policy from exploring them again. We address this with a learning algorithm that observes the reward for all possible actions for each visited state, and maximizes the immediate expected reward.

We evaluate our approach on SCONE (Long et al., 2016), which includes three domains, and is used to study recovering predicate logic meaning representations for sequential instructions. We study the problem of generating a sequence of low-level actions, and re-define the set of actions for each domain. For example, we treat the beakers in the ALCHEMY domain as stacks and use only `POP` and `PUSH` actions. Our approach robustly learns to execute sequential instructions with up to 89.1% task-completion accuracy for single instruction, and 62.7% for complete sequences. Our code is available at `https://github.com/clic-lab/scone`.

2 Technical Overview

Task and Notation Let $\mathcal{S}$ be the set of all possible world states, $\mathcal{X}$ be the set of all natural language instructions, and $\mathcal{A}$ be the set of all actions. An instruction $\bar{x} \in \mathcal{X}$ of length $|\bar{x}|$ is a sequence of tokens $\langle x_1, ... x_{|\bar{x}|}\rangle$. Executing an action modifies the world state following a transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. For example, the ALCHEMY domain includes seven beakers that contain colored liquids. The world state defines the content of each beaker. We treat each beaker as a stack. The actions are `POP N` and `PUSH N C`, where $1 \leq \texttt{N} \leq 7$ is the beaker number and `C` is one of six colors. There are a total of 50 actions, including the `STOP` action. Section 6 describes the domains in detail.

Given a start state s_1 and a sequence of instructions $\langle \bar{x}_1, \ldots, \bar{x}_n\rangle$, our goal is to generate the sequence of actions specified by the instructions starting from s_1. We treat the execution of a sequence of instructions as executing each instruction in turn. The execution $\bar{e}$ of an instruction $\bar{x}_i$ starting at a state s_1 and given the history of the instruction sequence $\langle \bar{x}_1, \ldots, \bar{x}_{i-1}\rangle$ is a sequence of state-action pairs $\bar{e} = \langle (s_1, a_1), ..., (s_m, a_m)\rangle$, where $a_k \in \mathcal{A}$, $s_{k+1} = T(s_k, a_k)$. The final action a_m is the special action `STOP`, which indicates the execution has terminated. The final state is then s_m, as $T(s_k, \texttt{STOP}) = s_k$. Executing a sequence of instructions in order generates a sequence $\langle \bar{e}_1, ..., \bar{e}_n\rangle$, where $\bar{e}_i$ is the execution of instruction $\bar{x}_i$. When referring to states and actions in an indexed execution $\bar{e}_i$, the k-th state and action are $s_{i,k}$ and $a_{i,k}$. We execute instructions one after the other: $\bar{e}_1$ starts at the interaction initial state s_1 and $s_{i+1,1} = s_{i,|\bar{e}_i|}$, where $s_{i+1,1}$ is the start state of $\bar{e}_{i+1}$ and $s_{i,|\bar{e}_i|}$ is the final state of $\bar{e}_i$.

Model We model the agent with a neural network policy (Section 4). At step k of executing the i-th instruction, the model input is the current instruction $\bar{x}_i$, the previous instructions $\langle \bar{x}_1, \ldots, \bar{x}_{i-1}\rangle$, the world state s_1 at the beginning of executing $\bar{x}_i$, and the current state s_k. The model predicts the next action a_k to execute. If $a_k = \texttt{STOP}$, we switch to the next instruction, or if at the end of the instruction sequence, terminate. Otherwise, we update the state to $s_{k+1} = T(s_k, a_k)$. The model uses attention to process the different inputs and a recurrent neural network (RNN) decoder to generate actions (Bahdanau et al., 2015).

Learning We assume access to a set of N instruction sequences, where each instruction in each sequence is paired with its start and goal states. During training, we create an example for each instruction. Formally, the training set is $\{(\bar{x}_i^{(j)}, s_{i,1}^{(j)}, \langle \bar{x}_1^{(j)}, \ldots, \bar{x}_{i-1}^{(j)} \rangle, g_i^{(j)})\}_{j=1,i=1}^{N,n^{(j)}}$, where $\bar{x}_i^{(j)}$ is an instruction, $s_{i,1}^{(j)}$ is a start state, $\langle \bar{x}_1^{(j)}, \ldots, \bar{x}_{i-1}^{(j)} \rangle$ is the instruction history, $g_i^{(j)}$ is the goal state, and $n^{(j)}$ is the length of the j-th instruction sequence. This training data contains no evidence about the actions and intermediate states required to execute each instruction.[1] We use a learning method that maximizes the expected immediate reward for a given state (Section 5). The reward accounts for task-completion and distance to the goal via potential-based reward shaping.

Evaluation We evaluate exact task completion for sequences of instructions on a test set $\{(s_1^{(j)}, \langle \bar{x}_1^{(j)}, \ldots, \bar{x}_{n_j}^{(j)} \rangle, g^{(j)})\}_{j=1}^{N}$, where $g^{(j)}$ is the oracle goal state of executing instructions $\bar{x}_1^{(j)}, \ldots, \bar{x}_{n_j}^{(j)}$ in order starting from $s_1^{(j)}$. We also evaluate single-instruction task completion using per-instruction annotated start and goal states.

3 Related Work

Executing instructions has been studied using the SAIL corpus (MacMahon et al., 2006) with focus on navigation using high-level logical representations (Chen and Mooney, 2011; Chen, 2012; Artzi and Zettlemoyer, 2013; Artzi et al., 2014) and low-level actions (Mei et al., 2016). While SAIL includes sequences of instructions, the data demonstrates limited discourse phenomena, and instructions are often processed in isolation. Approaches that consider as input the entire sequence focused on segmentation (Andreas and Klein, 2015). Recently, other navigation tasks were proposed with focus on single instructions (Anderson et al., 2018; Janner et al., 2018). We focus on sequences of environment manipulation instructions and modeling contextual cues from both the changing environment and instruction history. Manipulation using single-sentence instructions has been studied using the Blocks domain (Bisk et al., 2016, 2018; Misra et al., 2017; Tan and Bansal, 2018). Our work is related to the work of Branavan et al. (2009) and Vogel and Jurafsky (2010). While both study executing sequences of instructions, similar to SAIL, the data includes limited discourse dependencies. In addition, both learn with rewards computed from surface-form similarity between text in the environment and the instruction. We do not rely on such similarities, but instead use a state distance metric.

Language understanding in interactive scenarios that include multiple turns has been studied with focus on dialogue for querying database systems using the ATIS corpus (Hemphill et al., 1990; Dahl et al., 1994). Tür et al. (2010) surveys work on ATIS. Miller et al. (1996), Zettlemoyer and Collins (2009), and Suhr et al. (2018) modeled context dependence in ATIS for generating formal representations. In contrast, we focus on environments that change during execution and directly generating environment actions, a scenario that is more related to robotic agents than database query.

The SCONE corpus (Long et al., 2016) was designed to reflect a broad set of discourse context-dependence phenomena. It was studied extensively using logical meaning representations (Long et al., 2016; Guu et al., 2017; Fried et al., 2018). In contrast, we are interested in directly generating actions that modify the environment. This requires generating lower-level actions and learning procedures that are otherwise hardcoded in the logic (e.g., mixing action in Figure 1). Except for Fried et al. (2018), previous work on SCONE assumes access only to the initial and final states during training. This form of supervision does not require operating the agent manually to acquire the correct sequence of actions, a difficult task in robotic agents with complex control. Goal state supervision has been studied for instructional language (e.g., Branavan et al., 2009; Artzi and Zettlemoyer, 2013; Bisk et al., 2016), and more extensively in question answering when learning with answer annotations only (e.g., Clarke et al., 2010; Liang et al., 2011; Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014, 2015; Liang et al., 2017).

4 Model

We map sequences of instructions $\langle \bar{x}_1, \ldots, \bar{x}_n \rangle$ to actions by executing the instructions in or-

[1]This training set is a subset of the data used in previous work (Section 6, Guu et al., 2015), in which training uses all instruction sequences of length 1 and 2.

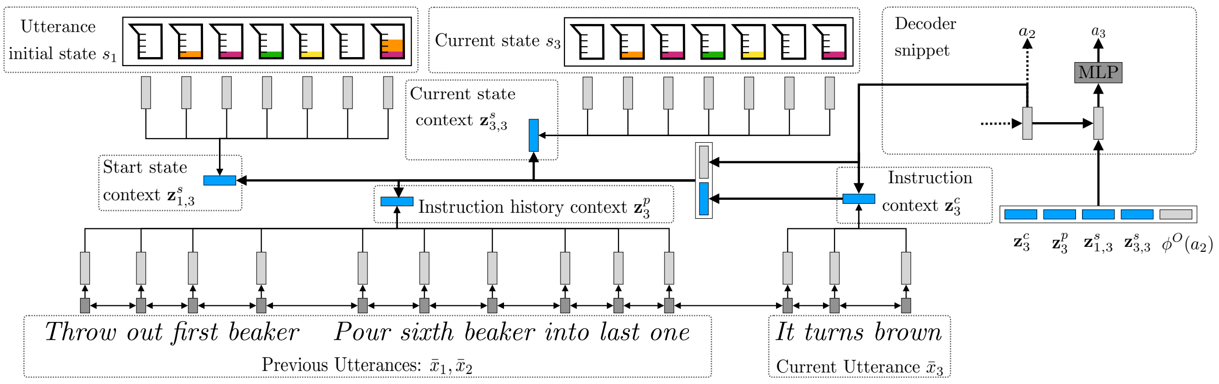

Utterance initial state s_1
Current state s_3
Decoder snippet
a_2
a_3
MLP
Current state context $\mathbf{z}^s_{3,3}$
Start state context $\mathbf{z}^s_{1,3}$
Instruction history context $\mathbf{z}^p_3$
Instruction context $\mathbf{z}^c_3$
$\mathbf{z}^c_3$ $\mathbf{z}^p_3$ $\mathbf{z}^s_{1,3}$ $\mathbf{z}^s_{3,3}$ $\phi^O(a_2)$
Throw out first beaker
Pour sixth beaker into last one
Previous Utterances: $\bar{x}_1, \bar{x}_2$
It turns brown
Current Utterance $\bar{x}_3$

Figure 2: Illustration of the model architecture while generating the third action a_3 in the third utterance $\bar{x}_3$ from Figure 1. Context vectors computed using attention are highlighted in blue. The model takes as input vector encodings from the current and previous instructions $\bar{x}_1$, $\bar{x}_2$, and $\bar{x}_3$, the initial state s_1, the current state s_3, and the previous action a_2. Instruction encodings are computed with a bidirectional RNN. We attend over the previous and current instructions and the initial and current states. We use an MLP to select the next action.

der. The model generates an execution $\bar{e} = \langle (s_1, a_1), \ldots, (s_{m_i}, a_{m_i}) \rangle$ for each instruction $\bar{x}_i$. The agent context, the information available to the agent at step k, is $\tilde{s}_k = (\bar{x}_i, \langle \bar{x}_1, \ldots, \bar{x}_{i-1} \rangle, s_k, \bar{e}[: k])$, where $\bar{e}[: k]$ is the execution up until but not including step k. In contrast to the world state, the agent context also includes instructions and the execution so far. The agent policy $\pi_\theta(\tilde{s}_k, a)$ is modeled as a probabilistic neural network parametrized by θ, where $\tilde{s}_k$ is the agent context at step k and a is an action. To generate executions, we generate one action at a time, execute the action, and observe the new world state. In step k of executing the i-th instruction, the network inputs are the current utterance $\bar{x}_i$, the previous instructions $\langle \bar{x}_1, \ldots, \bar{x}_{i-1} \rangle$, the initial state s_1 at beginning of executing $\bar{x}_i$, and the current state s_k. When executing a sequence of instructions, the initial state s_1 is either the state at the beginning of executing the sequence or the final state of the execution of the previous instruction. Figure 2 illustrates our architecture.

We generate continuous vector representations for all inputs. Each input is represented as a set of vectors that are then processed with an attention function to generate a single vector representation (Luong et al., 2015). We assume access to a domain-specific encoding function ENC(s) that, given a state s, generates a set of vectors S representing the objects in the state. For example, in the ALCHEMY domain, a vector is generated for each beaker using an RNN. Section 6 describes the different domains and their encoding functions.

We use a single bidirectional RNN with a long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) recurrence to encode the instructions. All instructions $\bar{x}_1$,…,$\bar{x}_i$ are encoded with a single RNN by concatenating them to $\bar{x}'$. We use two delimiter tokens: one separates previous instructions, and the other separates the previous instructions from the current one. The forward LSTM RNN hidden states are computed as:[2]

$$\overrightarrow{\mathbf{h}_{j+1}} = \overrightarrow{\text{LSTM}^E}\left(\phi^I(x'_{j+1}); \overrightarrow{\mathbf{h}_j}\right) \quad ,$$

where ϕ^I is a learned word embedding function and $\overrightarrow{\text{LSTM}^E}$ is the forward LSTM recurrence function. We use a similar computation to compute the backward hidden states $\overleftarrow{\mathbf{h}_j}$. For each token x'_j in $\bar{x}'$, a vector representation $\mathbf{h}'_j = \left[\overrightarrow{\mathbf{h}_j}; \overleftarrow{\mathbf{h}_j}\right]$ is computed. We then create two sets of vectors, one for all the vectors of the current instruction and one for the previous instructions:

$$\begin{aligned} X^c &= \{\mathbf{h}'_j\}_{j=J}^{J+|\bar{x}_i|} \\ X^p &= \{\mathbf{h}'_j\}_{j=0}^{j<J} \end{aligned}$$

where J is the index in $\bar{x}'$ where the current instruction $\bar{x}_i$ begins. Separating the vectors to two sets will allows computing separate attention on the current instruction and previous ones.

To compute each input representation during decoding, we use a bi-linear attention function (Luong et al., 2015). Given a set of vectors H, a query vector $\mathbf{h}^q$, and a weight matrix $\mathbf{W}$, the attention function ATTEND($H, \mathbf{h}^q, \mathbf{W}$) computes a context vector $\mathbf{z}$:

$$\begin{aligned} \alpha_i &\propto \exp(\mathbf{h}_i^\intercal \mathbf{W} \mathbf{h}^q) : i = 0, \ldots, |H| \\ \mathbf{z} &= \sum_{i=1}^{|H|} \alpha_i \mathbf{h}_i \ . \end{aligned}$$

[2]To simplify the notation, we omit the memory cell (often denoted as $\mathbf{c}_j$) from all LSTM descriptions. We use only the hidden state $\mathbf{h}_j$ to compute the intended representations (e.g., for the input text tokens). All LSTMs in this paper use zero vectors as initial hidden state $\mathbf{h}_0$ and initial cell memory $\mathbf{c}_0$.

We use a decoder to generate actions. At each time step k, we compute an input representation using the attention function, update the decoder state, and compute the next action to execute. Attention is first computed over the vectors of the current instruction, which is then used to attend over the other inputs. We compute the context vectors $\mathbf{z}_k^c$ and $\mathbf{z}_k^p$ for the current instruction and previous instructions:

$$\begin{aligned}\mathbf{z}_k^c &= \text{ATTEND}(X^c, \mathbf{h}_{k-1}^d, \mathbf{W}^c)\\ \mathbf{z}_k^p &= \text{ATTEND}(X^p, [\mathbf{h}_{k-1}^d, \mathbf{z}_k^c], \mathbf{W}^p)\ ,\end{aligned}$$

where $\mathbf{h}_{k-1}^d$ is the decoder hidden state for step $k-1$, and X^c and X^p are the sets of vector representations for the current instruction and previous instructions. Two attention heads are used over both the initial and current states. This allows the model to attend to more than one location in a state at once, for example when transferring items from one beaker to another in ALCHEMY. The current state is computed by the transition function $s_k = T(s_{k-1}, a_{k-1})$, where s_{k-1} and a_{k-1} are the state and action at step $k-1$. The context vectors for the initial state s_1 and the current state s_k are:

$$\begin{aligned}\mathbf{z}_{1,k}^s &= [\text{ATTEND}(\text{ENC}(s_1), [\mathbf{h}_{k-1}^d, \mathbf{z}_k^c], \mathbf{W}^{s_b,1});\\ &\quad\ \text{ATTEND}(\text{ENC}(s_1), [\mathbf{h}_{k-1}^d, \mathbf{z}_k^c], \mathbf{W}^{s_b,2})]\\ \mathbf{z}_{k,k}^s &= [\text{ATTEND}(\text{ENC}(s_k), [\mathbf{h}_{k-1}^d, \mathbf{z}_k^c], \mathbf{W}^{s_c,1});\\ &\quad\ \text{ATTEND}(\text{ENC}(s_k), [\mathbf{h}_{k-1}^d, \mathbf{z}_k^c], \mathbf{W}^{s_c,2})]\ ,\end{aligned}$$

where all $\mathbf{W}^{*,*}$ are learned weight matrices.

We concatenate all computed context vectors with an embedding of the previous action a_{k-1} to create the input for the decoder:

$$\begin{aligned}\mathbf{h}_k &= \tanh([\mathbf{z}_k^c; \mathbf{z}_k^p; \mathbf{z}_{1,k}^s; \mathbf{z}_{k,k}^s; \phi^O(a_{k-1})]\mathbf{W}^d + \mathbf{b}^d)\\ \mathbf{h}_k^d &= \text{LSTM}^D\left(\mathbf{h}_k; \mathbf{h}_{k-1}^d\right)\ ,\end{aligned}$$

where ϕ^O is a learned action embedding function and LSTM^D is the LSTM decoder recurrence.

Given the decoder state $\mathbf{h}_k^d$, the next action a_k is predicted with a multi-layer perceptron (MLP). The actions in our domains decompose to an action type and at most two arguments.[3] For example, the action `PUSH 1 B` in ALCHEMY has the type `PUSH` and two arguments: a beaker number and a color. Section 6 describes the actions of each domain. The probability of an action is:

$$\begin{aligned}\mathbf{h}_k^a &= \tanh(\mathbf{h}_k^d \mathbf{W}^a)\\ s_{k,a_T} &= \mathbf{h}_k^a \mathbf{b}_{a_T}\\ s_{k,a_1} &= \mathbf{h}_k^a \mathbf{b}_{a_1}\\ s_{k,a_2} &= \mathbf{h}_k^a \mathbf{b}_{a_2}\\ p(a_k = a_T(a_1, a_2) \mid \tilde{s}_k; \theta) &\propto\\ &\exp(s_{k,a_T} + s_{k,a_1} + s_{k,a_2})\ ,\end{aligned}$$

where a_T, a_1, and a_2 are an action type, first argument, and second argument. If the predicted action is `STOP`, the execution is complete. Otherwise, we execute the action a_k to generate the next state s_{k+1}, and update the agent context $\tilde{s}_k$ to $\tilde{s}_{k+1}$ by appending the pair (s_k, a_k) to the execution $\bar{e}$ and replacing the current state with s_{k+1}.

The model parameters θ include: the embedding functions ϕ^I and ϕ^O; the recurrence parameters for $\overrightarrow{\text{LSTM}^E}$, $\overleftarrow{\text{LSTM}^E}$, and LSTM^D; $\mathbf{W}^C$, $\mathbf{W}^P$, $\mathbf{W}^{s_b,1}$, $\mathbf{W}^{s_b,2}$, $\mathbf{W}^{s_c,1}$, $\mathbf{W}^{s_c,2}$, $\mathbf{W}^d$, $\mathbf{W}^a$, and $\mathbf{b}^d$; and the domain dependent parameters, including the parameters of the encoding function ENC and the action type, first argument, and second argument weights $\mathbf{b}_{a_T}$, $\mathbf{b}_{a_1}$, and $\mathbf{b}_{a_2}$.

5 Learning

We estimate the policy parameters θ using an exploration-based learning algorithm that maximizes the immediate expected reward. Broadly speaking, during learning, we observe the agent behavior given the current policy, and for each visited state compute the expected immediate reward by observing rewards for all actions. We assume access to a set of training examples $\{(\bar{x}_i^{(j)}, s_{i,1}^{(j)}, \langle \bar{x}_1^{(j)}, \ldots, \bar{x}_{i-1}^{(j)} \rangle, g_i^{(j)})\}_{j=1,i=1}^{N,n^{(j)}}$, where each instruction $\bar{x}_i^{(j)}$ is paired with a start state $s_{i,1}^{(j)}$, the previous instructions in the sequence $\langle \bar{x}_1^{(j)}, \ldots, \bar{x}_{i-1}^{(j)} \rangle$, and a goal state $g_i^{(j)}$.

Reward The reward $R_i^{(j)} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined for each example j and instruction i:

$$R_i^{(j)}(s, a, s') = P_i^{(j)}(s, a, s') + \phi_i^{(j)}(s') - \phi_i^{(j)}(s)\ ,$$

where s is a source state, a is an action, and s' is a target state.[4] $P_i^{(j)}(s, a, s')$ is a problem reward and $\phi_i^{(j)}(s') - \phi_i^{(j)}(s)$ is a shaping term. The problem reward $P_i^{(j)}(s, a, s')$ is positive for stopping at the goal $g_i^{(j)}$ and negative for stopping in an incorrect

[3]We use a `NULL` argument for unused arguments.

[4]While the reward function is defined for any state-action-state tuple, in practice, it is used during learning with tuples that follow the system dynamics, $s' = T(s, a)$.

Algorithm 1 SESTRA: **S**ingl**e**-**st**ep **R**eward Observ**a**tion.

```
Input: Training data $\{(\bar{x}_i^{(j)}, s_{i,1}^{(j)}, \langle \bar{x}_1^{(j)}, \ldots, \bar{x}_{i-1}^{(j)} \rangle, g_i^{(j)})\}_{j=1,i=1}^{N,n^{(j)}}$, learning rate $\mu$, entropy regularization coefficient $\lambda$, episode limit horizon $M$.
Definitions: $\pi_\theta$ is a policy parameterized by $\theta$, BEG is a special action to use for the first decoder step, and STOP indicates end of an execution. $T(s,a)$ is the state transition function, $H$ is an entropy function, $R_i^{(j)}(s,a,s')$ is the reward function for example $j$ and instruction $i$, and RMSPROP divides each weight by a running average of its squared gradient (Tieleman and Hinton, 2012).
Output: Parameters $\theta$ defining a learned policy $\pi_\theta$.
1: for $t = 1, \ldots, T, j = 1, \ldots, N$ do
2:   for $i = 1, \ldots, n^{(j)}$ do
3:     $\bar{e} \leftarrow \langle\,\rangle, k \leftarrow 0, a_0 \leftarrow$ BEG
4:     » Rollout up to STOP or episode limit.
5:     while $a_k \neq$ STOP $\wedge\ k < M$ do
6:       $k \leftarrow k + 1$
7:       $\tilde{s}_k \leftarrow (\bar{x}_i, \langle \bar{x}_1, \ldots, \bar{x}_{i-1} \rangle, s_k, \bar{e}[:k])$
8:       » Sample an action from policy.
9:       $a_k \sim \pi_\theta(\tilde{s}_k, \cdot)$
10:      $s_{k+1} \leftarrow T(s_k, a_k)$
11:      $\bar{e} \leftarrow [\bar{e}; \langle (s_k, a_k) \rangle]$
12:    $\Delta \leftarrow \bar{0}$
13:    for $k' = 1, \ldots, k$ do
14:      » Compute the entropy of $\pi_\theta(\tilde{s}_{k'}, \cdot)$.
15:      $\Delta \leftarrow \Delta + \lambda \nabla_\theta H(\pi_\theta(\tilde{s}_{k'}, \cdot))$
16:      for $a \in \mathcal{A}$ do
17:        $s' \leftarrow T(s_{k'}, a)$
18:        » Compute gradient for action $a$.
19:        $\Delta \leftarrow \Delta + R_i^{(j)}(s_{k'}, a, s') \nabla_\theta \pi_\theta(\tilde{s}_{k'}, a)$
20:    $\theta \leftarrow \theta + \mu \text{RMSPROP}\left(\frac{\Delta}{k}\right)$
21: return $\theta$
```

state or taking an invalid action:

$$P_i^{(j)}(s,a,s') = \begin{cases} 1.0 & a = \texttt{STOP} \wedge s' = g_i^{(j)} \\ -1.0 & a = \texttt{STOP} \wedge s' \neq g_i^{(j)} \\ -1.0 - \delta & s = s' \\ -\delta & \text{otherwise} \end{cases},$$

where δ is a verbosity penalty. The case $s = s'$ indicates that a was invalid in state s, as in this domain, all valid actions except STOP modify the state. We use a potential-based shaping term $\phi_i^{(j)}(s') - \phi_i^{(j)}(s)$ (Ng et al., 1999), where $\phi_i^{(j)}(s) = -||s - g_i^{(j)}||$ computes the edit distance between the state s and the goal, measured over the objects in each state. The shaping term densifies the reward, providing a meaningful signal for learning in nonterminal states.

Objective We maximize the immediate expected reward over all actions and use entropy regularization. The gradient is approximated by sampling an execution $\bar{e} = \langle (s_1, a_1), \ldots, (s_k, a_k) \rangle$ using our current policy:

$$\nabla_\theta \mathcal{J} = \frac{1}{k} \sum_{k'=1}^{k} \Bigg(\sum_{a \in \mathcal{A}} R(s_k, a, T(s_k, a)) \nabla_\theta \pi(\tilde{s}_k, a) + \lambda \nabla_\theta H(\pi(\tilde{s}_k, \cdot)) \Bigg),$$

where $H(\pi(\tilde{s}_k, \cdot)$ is the entropy term.

Algorithm Algorithm 1 shows the Single-step Reward Observation (SESTRA) learning algorithm. We iterate over the training data T times (line 1). For each example j and turn i, we first perform a rollout by sampling an execution $\bar{e}$ from π_θ with at most M actions (lines 5-11). If the rollout reaches the horizon without predicting STOP, we set the problem reward $P_i^{(j)}$ to -1.0 for the last step. Given the sampled states visited, we compute the entropy (line 15) and observe the immediate reward for all actions (line 19) for each step. Entropy and rewards are used to accumulate the gradient, which is applied to the parameters using RMSPROP (Dauphin et al., 2015) (line 20).

Discussion Observing the rewards for all actions for each visited state addresses an on-policy learning exploration problem. Actions that consistently receive negative reward early during learning will be visited with very low probability later on, and in practice, often not explored at all. Because the network is randomly initialized, these early negative rewards are translated into strong general biases that are not grounded well in the observed context. Our algorithm exposes the agent to such actions later on when they receive positive rewards even though the agent does not explore them during rollout. For example, in ALCHEMY, POP actions are sufficient to complete the first steps of good executions. As a result, early during learning, the agent learns a strong bias against PUSH actions. In practice, the agent then will not explore PUSH actions again. In our algorithm, as the agent learns to roll out the correct POP prefix, it is then exposed to the reward for the first PUSH even though it likely sampled another POP. It then unlearns its bias towards predicting POP.

Our learning algorithm can be viewed as a cost-sensitive variant of the oracle in DAGGER (Ross et al., 2011), where it provides the rewards for all actions instead of an oracle action. It is also related to Locally Optimal Learning to Search (LOLS; Chang et al., 2015) with two key distinctions: (a) instead of using different roll-in and roll-out policies, we use the model policy; and (b) we branch at each step, instead of once, but do not rollout

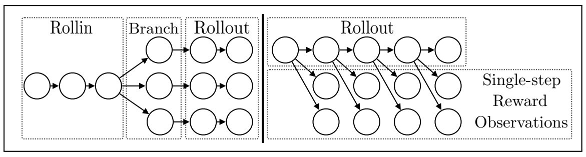


Figure 3: Illustration of LOLS (left; Chang et al., 2015) and our learning algorithm (SESTRA, right). LOLS branches a single time, and samples complete rollout for each branch to obtain the trajectory loss. SESTRA uses a complete on-policy rollout and single-step branching for all actions in each sample state.

	ALC	SCE	TAN
# Sequences (train)	3657	3352	4189
# Sequences (dev)	245	198	199
# Sequences (test)	899	1035	800
Mean instruction length	8.0±3.2	10.5±5.5	5.4±2.4
Vocabulary size	695	816	475

Table 1: Data statistics for ALCHEMY (ALC), SCENE (SCE), and TANGRAMS (TAN).

	Refs/Ex		1	2	3	4
ALCHEMY	1.4	Coref.	28	7	2	0
		Ellipsis	0	0	3	1
SCENE	2.4	Coref.	49	16	5	3
		Ellipsis	0	0	0	0
TANGRAMS	1.7	Coref.	25	14	2	1
		Ellipsis	4	0	0	0

Table 2: Counts of discourse phenomena in SCONE from 30 randomly selected development interactions for each domain. We count occurrences of coreference between instructions (e.g., *he leaves* in SCENE) and ellipsis (e.g., *then, drain 2 units* in ALCHEMY), when the last explicit mention of the referent was 1, 2, 3, or 4 turns in the past. We also report the average number of multi-turn references per interaction (Refs/Ex).

from branched actions since we only optimize the immediate reward. Figure 3 illustrates the comparison. Our summation over immediate rewards for all actions is related the summation of estimated Q-values for all actions in the Mean Actor-Critic algorithm (Asadi et al., 2017). Finally, our approach is related to Misra et al. (2017), who also maximize the immediate reward, but do not observe rewards for all actions for each state.

6 SCONE Domains and Data

SCONE has three domains: ALCHEMY, SCENE, and TANGRAMS. Each interaction contains five instructions. Table 1 shows data statistics. Table 2 shows discourse reference analysis. State encodings are detailed in the Supplementary Material.

ALCHEMY Each environment in ALCHEMY contains seven numbered beakers, each containing up to four colored chemicals in order. Figure 1 shows an example. Instructions describe pouring chemicals between and out of beakers, and mixing beakers. We treat all beakers as stacks. There are two action types: `PUSH` and `POP`. `POP` takes a beaker index, and removes the top color. `PUSH` takes a beaker index and a color, and adds the color at the top of the beaker. To encode a state, we encode each beaker with an RNN, and concatenate the last output with the beaker index embedding. The set of vectors is the state embedding.

SCENE Each environment in SCENE contains ten positions, each containing at most one person defined by a shirt color and an optional hat color. Instructions describe adding or removing people, moving a person to another position, and moving a person's hat to another person. There are four action types: `ADD_PERSON`, `ADD_HAT`, `REMOVE_PERSON`, and `REMOVE_HAT`. `ADD_PERSON` and `ADD_HAT` take a position to place the person or hat and the color of the person's shirt or hat. `REMOVE_PERSON` and `REMOVE_HAT` take the position to remove a person or hat from. To encode a state, we use a bidirectional RNN over the ordered positions. The input for each position is a concatenation of the color embeddings for the person and hat. The set of RNN hidden states is the state embedding.

TANGRAMS Each environment in TANGRAMS is a list containing at most five unique objects. Instructions describe removing or inserting an object into a position in the list, or swapping the positions of two items. There are two action types: `INSERT` and `REMOVE`. `INSERT` takes the position to insert an object, and the object identifier. `REMOVE` takes an object position. We embed each object by concatenating embeddings for its type and position. The resulting set is the state embedding.

7 Experimental Setup

Evaluation Following Long et al. (2016), we evaluate task completion accuracy using exact match between the final state and the annotated goal state. We report accuracy for complete interactions (5utts), the first three utterances of each interaction (3utts), and single instructions (Inst). For single instructions, execution starts from the annotated start state of the instruction.

Systems We report performance of ablations and two baseline systems: POLICYGRADIENT: policy gradient with cumulative episodic reward without a baseline, and CONTEXTUALBANDIT: the contextual bandit approach of Misra et al. (2017). Both systems use the reward with the

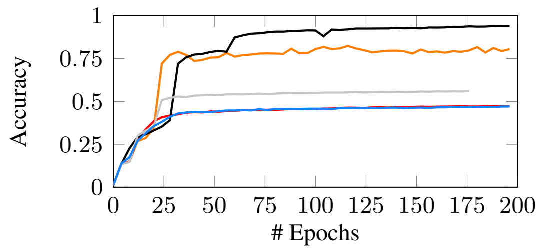


Figure 4: Instruction-level training accuracy per epoch when training five models on SCENE, demonstrating the effect of randomization in the learning method. Three of five experiments fail to learn effective models. The red and blue learning trajectories are overlapping.

shaping term and our model. We also report supervised learning results (SUPERVISED) by heuristically generating correct executions and computing maximum-likelihood estimate using context-action demonstration pairs. Only the supervised approach uses the heuristically generated labels. Although the results are not comparable, we also report the performance of previous approaches to SCONE. All three approaches generate logical representations based on lambda calculus. In contrast to our approach, this requires an ontology of hand built symbols and rules to evaluate the logical forms. Fried et al. (2018) uses supervised learning with annotated logical forms.

Training Details For test results, we run each experiment five times and report results for the model with best validation interaction accuracy. For ablations, we do the same with three experiments. We use a batch size of 20. We stop training using a validation set sampled from the training data. We hold the validation set constant for each domain for all experiments. We use patience over the average reward, and select the best model using interaction-level (5utts) validation accuracy. We tune λ, δ, and M on the development set. The selected values and other implementation details are described in the Supplementary Material.

8 Results

Table 3 shows test results. Our approach significantly outperforms POLICYGRADIENT and CONTEXTUALBANDIT, both of which suffer due to biases learned early during learning, hindering later exploration. This problem does not appear in TANGRAMS, where no action type is dominant at the beginning of executions, and all methods perform well. POLICYGRADIENT completely fails to learn ALCHEMY and SCENE due to observing only negative total rewards early during learning. Using a baseline, for example with an actor-critic method, will potentially close the gap to CONTEXTUALBANDIT. However, it is unlikely to address the on-policy exploration problem.

Table 4 shows development results, including model ablation studies. Removing previous instructions (– previous instructions) or both states (– current and initial state) reduces performance across all domains. Removing only the initial state (– initial state) or the current state (– current state) shows mixed results across the domains. Providing access to both initial and current states increases performance for ALCHEMY, but reduces performance on the other domains. We hypothesize that this is due to the increase in the number of parameters outweighing what is relatively marginal information for these domains. In our development and test results we use a single architecture across the three domains, the full approach, which has the highest interactive-level accuracy when averaged across the three domains (62.7 5utts). We also report mean and standard deviation for our approach over five trials. We observe exceptionally high variance in performance on SCENE, where some experiments fail to learn and training performance remains exceptionally low (Figure 4). This highlights the sensitivity of the model to the random effects of initialization, dropout, and ordering of training examples.

We analyze the instruction-level errors made by our best models when the agent is provided the correct initial state for the instruction. We study fifty examples in each domain to identify the type of failures. Table 5 shows the counts of major error categories. We consider multiple reference resolution errors. State reference errors indicate a failure to resolve a reference to the world state. For example, in ALCHEMY, the phrase *leftmost red beaker* specifies a beaker in the environment. If the model picked the correct action, but the wrong beaker, we count it as a state reference. We distinguish between multi-turn reference errors that should be feasible, and these that that are impossible to solve without access to states before executing previous utterances, which are not provided to our model. For example, in TANGRAMS, the instruction *put it back in the same place* refers to a previously-removed item. Because the agent only has access to the world state after following this instruction, it does not observe what kind of item was previously removed, and cannot identify the item to add. We

System	ALCHEMY Inst	ALCHEMY 3utts	ALCHEMY 5utts	SCENE Inst	SCENE 3utts	SCENE 5utts	TANGRAMS Inst	TANGRAMS 3utts	TANGRAMS 5utts
Long et al. (2016)	–	56.8	52.3	–	23.2	14.7	–	64.9	27.6
Guu et al. (2017)	–	66.9	52.9	–	64.8	46.2	–	65.8	37.1
Fried et al. (2018)	–	–	72.0	–	–	72.7	–	–	69.6
SUPERVISED	89.4	73.3	62.3	88.8	78.9	66.4	86.6	81.4	60.1
POLICYGRADIENT	0.0	0.0	0.0	0.0	1.3	0.2	84.1	77.4	54.9
CONTEXTUALBANDIT	73.8	36.0	25.7	15.1	2.9	4.4	84.8	76.9	57.9
Our approach	89.1	74.2	62.7	87.1	73.9	62.0	86.6	80.8	62.4

Table 3: Test accuracies for single instructions (Inst), first-three instructions (3utts), and full interactions (5utts).

System	ALCHEMY Inst	ALCHEMY 3utts	ALCHEMY 5utts	SCENE Inst	SCENE 3utts	SCENE 5utts	TANGRAMS Inst	TANGRAMS 3utts	TANGRAMS 5utts
SUPERVISED	92.0	83.3	71.4	85.3	72.7	60.6	86.1	81.9	58.3
POLICYGRADIENT	0.0	0.0	0.0	0.9	1.0	0.5	85.2	74.9	52.3
CONTEXTUALBANDIT	58.8	6.9	5.7	12.0	0.5	1.5	85.6	78.4	52.6
Our approach	**92.1**	**82.9**	**71.8**	**83.9**	**68.7**	56.1	88.5	82.4	60.3
– previous instructions	90.1	77.1	66.1	79.3	60.6	45.5	76.4	55.8	27.6
– current and initial state	25.7	4.5	3.3	17.5	0.0	0.0	45.4	15.1	3.5
– current state	89.8	78.0	62.9	83.0	**68.7**	54.0	87.6	78.4	60.8
– initial state	81.1	68.6	42.9	82.7	67.7	**57.1**	**88.6**	**82.9**	**63.3**
Our approach ($\mu \pm \sigma$)	91.5 ± 1.4	80.4 ± 2.6	69.5 ± 5.0	62.9 ± 17.7	37.8 ± 23.5	29.0 ± 21.1	88.2 ± 0.6	80.8 ± 2.8	59.2 ± 2.3

Table 4: Development results, including model ablations. We also report mean μ and standard deviation σ for all metrics for our approach across five experiments. We bold the best performing variations of our model.

Class	ALC	SCE	TAN
State reference	23	13	7
Multi-turn reference	12	5	13
Impossible multi-turn reference	2	5	13
Ambiguous or incorrect label	2	19	12

Table 5: Common error counts in the three domains.

also find a significant number of errors due to ambiguous or incorrect instructions. For example, the SCENE instruction *person in green appears on the right end* is ambiguous. In the annotated goal, it is interpreted as referring to a person already in the environment, who moves to the 10th position. However, it can also be interpreted as a new person in green appearing in the 10th position.

We also study performance with respect to multi-turn coreference by observing whether the model was able to identify the correct referent for each occurrence included in the analysis in Table 2. The models were able to correctly resolve 92.3%, 88.7%, and 76.0% of references in ALCHEMY, SCENE, and TANGRAMS respectively.

Finally, we include attention visualization for examples from the three domains in the Supplementary Material.

9 Discussion

We propose a model to reason about context-dependent instructional language that display strong dependencies both on the history of the interaction and the state of the world. Future modeling work may include using intermediate world states from previous turns in the interaction, which is required for some of the most complex references in the data. We propose to train our model using SESTRA, a learning algorithm that takes advantage of single-step reward observations to overcome learned biases in on-policy learning. Our learning approach requires additional reward observations in comparison to conventional reinforcement learning. However, it is particularly suitable to recovering from biases acquired early during learning, for example due to biased action spaces, which is likely to lead to incorrect blame assignment in neural network policies. When the domain and model are less susceptible to such biases, the benefit of the additional reward observations is less pronounced. One possible direction for future work is to use an estimator to predict rewards for all actions, rather than observing them.

Acknowledgements

This research was supported by the NSF (CRII-1656998), Schmidt Sciences, and cloud computing credits from Amazon. We thank John Langford and Dipendra Misra for helpful and insightful discussions with regards to our learning algorithm. We also thank the anonymous reviewers for their helpful comments.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yoav Artzi and Luke S. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Kavosh Asadi, Cameron Allen, Melrose Roderick, Abdel-rahman Mohamed, George Konidaris, and Michael L. Littman. 2017. Mean actor critic. *CoRR*, abs/1709.00503.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Yonatan Bisk, Kevin Shih, Yejin Choi, and Daniel Marcu. 2018. Learning interpretable spatial operations in a rich 3D blocks world. In *Proceedings of the Thirty-Second Conference on Artificial Intelligence*.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural language communication with robots. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the International Conference on Machine Learning*.

David Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*.

Yann Dauphin, Harm de Vries, , and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. *CoRR*, abs/1502.04390.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matthew MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Association for the Advancement of Artificial Intelligence*.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Hao Tan and Mohit Bansal. 2018. Source-target inference models for spatial instruction understanding. In *AAAI Conference on Artificial Intelligence*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Gökhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in ATIS? In *Proceedings of the Spoken Language Technology Workshop*.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Marrying Up Regular Expressions with Neural Networks: A Case Study for Spoken Language Understanding

Bingfeng Luo[1], **Yansong Feng**[*1], **Zheng Wang**[2], **Songfang Huang**[3], **Rui Yan**[1] and **Dongyan Zhao**[1]
[1]ICST, Peking University, China
[2]MetaLab, Lancaster University, UK
[3]IBM China Research Lab, China
{bf_luo,fengyansong,ruiyan,zhaody}@pku.edu.cn
z.wang@lancaster.ac.uk, huangsf@cn.ibm.com

Abstract

The success of many natural language processing (NLP) tasks is bound by the number and quality of annotated data, but there is often a shortage of such training data. In this paper, we ask the question: "Can we combine a neural network (NN) with regular expressions (RE) to improve supervised learning for NLP?". In answer, we develop novel methods to exploit the rich expressiveness of REs at different levels within a NN, showing that the combination significantly enhances the learning effectiveness when a small number of training examples are available. We evaluate our approach by applying it to spoken language understanding for intent detection and slot filling. Experimental results show that our approach is highly effective in exploiting the available training data, giving a clear boost to the RE-unaware NN.

1 Introduction

Regular expressions (REs) are widely used in various natural language processing (NLP) tasks like pattern matching, sentence classification, sequence labeling, etc. (Chang and Manning, 2014). As a technique based on human-crafted rules, it is concise, interpretable, tunable, and does not rely on much training data to generate. As such, it is commonly used in industry, especially when the available training examples are limited – a problem known as few-shot learning (GC et al., 2015).

While powerful, REs have a poor generalization ability because all synonyms and variations in a RE must be explicitly specified. As a result, REs are often ensembled with data-driven methods, such as neural network (NN) based techniques, where a set of carefully-written REs are used to handle certain cases with high precision, leaving the rest for data-driven methods.

We believe the use of REs can go beyond simple pattern matching. In addition to being a separate classifier to be ensembled, a RE also encodes a developer's knowledge for the problem domain. The knowledge could be, for example, the informative words (***clue words***) within a RE's surface form. We argue that such information can be utilized by data-driven methods to achieve better prediction results, especially in few-shot learning.

This work investigates the use of REs to improve NNs – a learning framework that is widely used in many NLP tasks (Goldberg, 2017). The combination of REs and a NN allows us to exploit the conciseness and effectiveness of REs and the strong generalization ability of NNs. This also provides us an opportunity to learn from various kinds of REs, since NNs are known to be good at tolerating noises (Xie et al., 2016).

This paper presents novel approaches to combine REs with a NN at different levels. At the input layer, we propose to use the evaluation outcome of REs as the input features of a NN (Sec.3.2). At the network module level, we show how to exploit the knowledge encoded in REs to guide the attention mechanism of a NN (Sec. 3.3). At the output layer, we combine the evaluation outcome of a RE with the NN output in a learnable manner (Sec. 3.4).

We evaluate our approach by applying it to two spoken language understanding (SLU) tasks, namely *intent detection* and *slot filling*, which respectively correspond to two fundamental NLP tasks: sentence classification and sequence labeling. To demonstrate the usefulness of REs in real-world scenarios where the available number of annotated data can vary, we explore both the few-shot learning setting and the one with full training data. Experimental results show that our approach is highly effective in utilizing the available

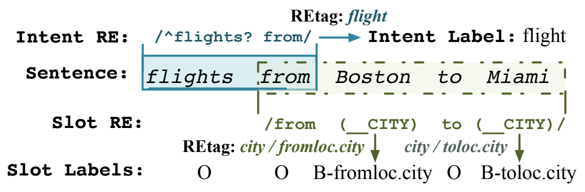

REtag: flight
Intent RE:
/^flights? from/
Intent Label: flight
Sentence:
flights from Boston to Miami
Slot RE:
/from (__CITY) to (__CITY)/
REtag: city / fromloc.city
city / toloc.city
Slot Labels:
O O B-fromloc.city O B-toloc.city

Figure 1: A sentence from the ATIS dataset. REs can be used to detect the intent and label slots.

annotated data, yielding significantly better learning performance over the RE-unaware method.

Our contributions are as follows. (1) We present the first work to systematically investigate methods for combining REs with NNs. (2) The proposed methods are shown to clearly improve the NN performance in both the few-shot learning and the full annotation settings. (3) We provide a set of guidance on how to combine REs with NNs and RE annotation.

2 Background

2.1 Typesetting

In this paper, we use italic for emphasis like *intent detection*, the Courier typeface for abbreviations like RE, bold italic for the first appearance of a concept like ***clue words***, Courier surrounded by / for regular expressions like /list(the)? __AIRLINE/, and underlined italic for words of sentences in our dataset like *Boston*.

2.2 Problem Definition

Our work targets two SLU tasks: *intent detection* and *slot filling*. The former is a sentence classification task where we learn a function to map an input sentence of n words, $\mathbf{x} = [x_1, ..., x_n]$, to a corresponding ***intent label***, c. The latter is a sequence labeling task for which we learn a function to take in an input query sentence of n words, $\mathbf{x} = [x_1, ..., x_n]$, to produce a corresponding labeling sequence, $\mathbf{y} = [y_1, ..., y_n]$, where y_i is the ***slot label*** of the corresponding word, x_i.

Take the sentence in Fig. 1 as an example. A successful intent detector would suggest the intent of the sentence as *flight*, i.e., querying about flight-related information. A slot filler, on the other hand, should identify the slots *fromloc.city* and *toloc.city* by labeling *Boston* and *Miami*, respectively, using the begin-inside-outside (BIO) scheme.

2.3 The Use of Regular Expressions

In this work, a RE defines a mapping from a text *pattern* to several ***REtags*** which are the same as or related to the ***target labels*** (i.e., intent and slot labels). A search function takes in a RE, applies it to all sentences, and returns any texts that match the pattern. We then assign the REtag (s) (that are associated with the matching RE) to either the matched sentence (for intent detection) or some matched phrases (for slot filling).

Specifically, our REtags for intent detection are the same as the intent labels. For example, in Fig. 1, we get a REtag of *flight* that is the same as the intent label *flight*.

For slot filling, we use two different sets of REs. Given the group functionality of RE, we can assign REtags to our interested ***RE groups*** (i.e., the expressions defined inside parentheses). The translation from REtags to slot labels depends on how the corresponding REs are used. (1) When REs are used at the network module level (Sec. 3.3), the corresponding REtags are the same as the target slot labels. For instance, the slot RE in Fig. 1 will assign *fromloc.city* to the first RE group and *toloc.city* to the second one. Here, __CITY is a list of city names, which can be replaced with a RE string like /Boston|Miami|LA|.../. (2) If REs are used in the input (Sec. 3.2) and the output layers (Sec. 3.4) of a NN, the corresponding REtag would be different from the target slot labels. In this context, the two RE groups in Fig. 1 would be simply tagged as *city* to capture the commonality of three related target slot labels: *fromloc.city*, *toloc.city*, *stoploc.city*. Note that we could use the target slot labels as REtags for all the settings. The purpose of abstracting REtags to a simplified version of the target slot labels here is to show that REs can still be useful when their evaluation outcome does not exactly match our learning objective. Further, as shown in Sec. 4.2, using simplified REtags can also make the development of REs easier in our tasks.

Intuitively, complicated REs can lead to better performance but require more efforts to generate. Generally, there are two aspects affecting RE complexity most: the number of RE groups[1] and *or* clauses (i.e., expressions separated by the disjunction operator |) in a RE group. Having a larger number of RE groups often leads to better

[1] When discussing complexity, we consider each semantically independent consecutive word sequence as a RE group (excluding clauses, such as \w+, that can match any word). For instance, the RE: /how long(\w+){1,2}? (it take|flight)/ has two RE groups: (how long) and (it take|flight).

precision but lower coverage on pattern matching, while a larger number of *or* clauses usually gives a higher coverage but slightly lower precision.

3 Our Approach

As depicted in Fig. 2, we propose to combine `NNs` and `REs` from three different angles.

3.1 Base Models

We use the Bi-directional LSTM (`BLSTM`) as our base `NN` model because it is effective in both intent detection and slot filling (Liu and Lane, 2016).

Intent Detection. As shown in Fig. 2, the `BLSTM` takes as input the word embeddings $[\mathbf{x}_1, ..., \mathbf{x}_n]$ of a n-word sentence, and produces a vector $\mathbf{h}_i$ for each word i. A self-attention layer then takes in the vectors produced by the `BLSTM` to compute the sentence embedding $\mathbf{s}$:

$$\mathbf{s} = \sum_i \alpha_i \mathbf{h}_i, \quad \alpha_i = \frac{\exp(\mathbf{h}_i^\intercal \mathbf{W}\mathbf{c})}{\sum_i \exp(\mathbf{h}_i^\intercal \mathbf{W}\mathbf{c})} \tag{1}$$

where α_i is the attention for word i, $\mathbf{c}$ is a randomly initialized trainable vector used to select informative words for classification, and $\mathbf{W}$ is a weight matrix. Finally, $\mathbf{s}$ is fed to a softmax classifier for intent classification.

Slot Filling. The model for slot filling is straightforward – the slot label prediction is generated by a softmax classier which takes in the `BLSTM`'s output $\mathbf{h}_i$ and produces the slot label of word i. Note that attention aggregation in Fig. 2 is only employed by the network module level method presented in Sec. 3.3.

3.2 Using REs at the Input Level

At the input level, we use the evaluation outcomes of `REs` as features which are fed to `NN` models.

Intent Detection. Our `REtag` for intent detection is the same as our target intent label. Because real-world `REs` are unlikely to be perfect, one sentence may be matched by more than one `RE`. This may result in several `REtags` that are conflict with each other. For instance, the sentence *list the Delta airlines flights to Miami* can match a `RE`: `/list( the)? __AIRLINE/` that outputs tag *airline*, and another `RE`: `/list( \w+){0,3} flights?/` that outputs tag *flight*.

To resolve the conflicting situations illustrated above, we average the randomly initialized trainable tag embeddings to form an aggregated embedding as the `NN` input. There are two ways to use the aggregated embedding. We can append the aggregated embedding to either the embedding of every input word, or the input of the softmax classifier (see ① in Fig. 2(a)). To determine which strategy works best, we perform a pilot study. We found that the first method causes the tag embedding to be copied many times; consequently, the `NN` tends to heavily rely on the `REtags`, and the resulting performance is similar to the one given by using `REs` alone in few-shot settings. Thus, we adopt the second approach.

Slot Filling. Since the evaluation outcomes of slot `REs` are word-level tags, we can simply embed and average the `REtags` into a vector $\mathbf{f}_i$ for each word, and append it to the corresponding word embedding $\mathbf{w}_i$ (as shown in ① in Fig. 2(b)). Note that we also extend the slot `REtags` into the `BIO` format, e.g., the `REtags` of phrase *New York* are *B-city* and *I-city* if its original tag is *city*.

3.3 Using REs at the Network Module Level

At the network module level, we explore ways to utilize the clue words in the surface form of a `RE` (bold blue arrows and words in ② of Fig. 2) to guide the attention module in `NNs`.

Intent Detection. Taking the sentence in Fig. 1 for example, the `RE`: `/^flights? from/` that leads to intent *flight* means that *flights from* are the key words to decide the intent *flight*. Therefore, the attention module in `NNs` should leverage these two words to get the correct prediction. To this end, we extend the base intent model by making two changes to incorporate the guidance from `REs`.

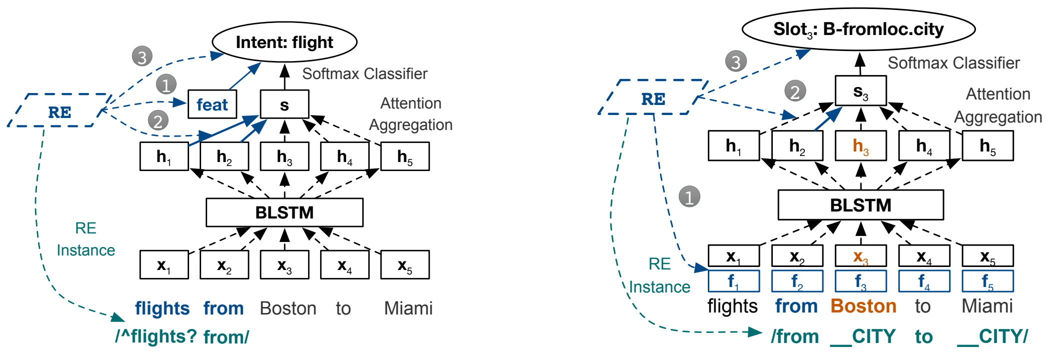
First, since each intent has its own clue words, using a single sentence embedding for all intent labels would make the attention less focused. Therefore, we let each intent label k use different attention $\mathbf{a}_k$, which is then used to generate the sentence embedding $\mathbf{s}_k$ for that intent:

$$\mathbf{s}_k = \sum_i \alpha_{ki} \mathbf{h}_i, \quad \alpha_{ki} = \frac{\exp(\mathbf{h}_i^\intercal \mathbf{W}_a \mathbf{c}_k)}{\sum_i \exp(\mathbf{h}_i^\intercal \mathbf{W}_a \mathbf{c}_k)} \tag{2}$$

where $\mathbf{c}_k$ is a trainable vector for intent k which is used to compute attention $\mathbf{a}_k$, $\mathbf{h}_i$ is the `BLSTM` output for word i, and $\mathbf{W}_a$ is a weight matrix.

The probability p_k that the input sentence expresses intent k is computed by:

$$p_k = \frac{\exp(logit_k)}{\sum_k \exp(logit_k)}, \qquad logit_k = \mathbf{w}_k \mathbf{s}_k + b_k \tag{3}$$



(a) Intent Detection (b) Slot Filling (predicting slot label for *Boston*)

Figure 2: Overview of our methods. ①, ②, ③ refers to the methods in Sec. 3.2, 3.3, 3.4 respectively.

where $\mathbf{w}_k$, $logit_k$, b_k are weight vector, logit, and bias for intent k, respectively.

Second, apart from indicating a sentence for intent k (***positive REs***), a RE can also indicate that a sentence does not express intent k (***negative REs***). We thus use a new set of attention (***negative attentions***, in contrast to ***positive attentions***), to compute another set of logits for each intent with Eqs. 2 and 3. We denote the logits computed by positive attentions as $logit_{pk}$, and those by negative attentions as $logit_{nk}$, the final logit for intent k can then be calculated as:

$$logit_k = logit_{pk} - logit_{nk} \tag{4}$$

To use REs to guide attention, we add an attention loss to the final loss:

$$loss_{att} = \sum_k \sum_i t_{ki} \log(\alpha_{ki}) \tag{5}$$

where t_{ki} is set to 0 when none of the matched REs (that leads to intent k) marks word i as a clue word – otherwise t_{ki} is set to $1/l_k$, where l_k is the number of clue words for intent k (if no matched RE leads to intent k, then $t_{k*} = 0$). We use Eq. 5 to compute the positive attention loss, $loss_{att\_p}$, for positive REs and negative attention loss, $loss_{att\_n}$, for negative ones. The final loss is computed as:

$$loss = loss_c + \beta_p loss_{att\_p} + \beta_n loss_{att\_n} \tag{6}$$

where $loss_c$ is the original classification loss, β_p and β_n are weights for the two attention losses.

Slot Filling. The ***two-side attention*** (positive and negative attention) mechanism introduced for intent prediction is unsuitable for slot filling. Because for slot filling, we need to compute attention for each word, which demands more computational and memory resources than doing that for intent detection[2].

Because of the aforementioned reason, we use a simplified version of the two-side attention, where all the slot labels share the same set of positive and negative attention. Specifically, to predict the slot label of word i, we use the following equations, which are similar to Eq. 1, to generate a sentence embedding $\mathbf{s}_{pi}$ with regard to word i from positive attention:

$$\mathbf{s}_{pi} = \sum_j \alpha_{pij}\mathbf{h}_j, \quad \alpha_{pij} = \frac{\exp(\mathbf{h}_j^\intercal \mathbf{W}_{sp}\mathbf{h}_i)}{\sum_j \exp(\mathbf{h}_j^\intercal \mathbf{W}_{sp}\mathbf{h}_i)} \tag{7}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the BLSTM outputs for word i and j respectively, $\mathbf{W}_{sp}$ is a weight matrix, and α_{pij} is the positive attention value for word j with respect to word i. Further, by replacing $\mathbf{W}_{sp}$ with $\mathbf{W}_{sn}$, we use Eq. 7 again to compute negative attention and generate the corresponding sentence embedding $\mathbf{s}_{ni}$.

Finally, the prediction $\mathbf{p}_i$ for word i can be calculated as:

$$\begin{aligned}\mathbf{p}_i = \text{softmax}((&\mathbf{W}_p[\mathbf{s}_{pi};\mathbf{h}_i] + \mathbf{b}_p) \\ &-(\mathbf{W}_n[\mathbf{s}_{ni};\mathbf{h}_i] + \mathbf{b}_n))\end{aligned} \tag{8}$$

where $\mathbf{W}_p$, $\mathbf{W}_n$, $\mathbf{b}_p$, $\mathbf{b}_n$ are weight matrices and bias vectors for positive and negative attention, respectively. Here we append the BLSTM output $\mathbf{h}_i$ to $\mathbf{s}_{pi}$ and $\mathbf{s}_{ni}$ because the word i itself also plays a crucial part in identifying its slot label.

3.4 Using REs at the Output Level

At the output level, REs are used to amend the output of NNs. At this level, we take the same

[2]Since we need to assign a label to each word, if we still compute attention for each slot label, we will have to compute $2 \times L \times n^2$ attention values for one sentence. Here, L is the number of tags and n is the sentence length. The BIO tagging format will further double the number of tags.

approach used for intent detection and slot filling (see ③ in Fig. 2).

As mentioned in Sec. 2.3, the slot `RE`s used in the output level only produce a simplified version of target slot labels, for which we can further annotate their corresponding target slot labels. For instance, a `RE` that outputs *city* can lead to three slot labels: *fromloc.city*, *toloc.city*, *stoploc.city*.

Let z_k be a 0-1 indicator of whether there is at least one matched `RE` that leads to target label k (intent or slot label), the final logits of label k for a sentence (or a specific word for slot filling) is:

$$logit_k = logit'_k + w_k z_k \tag{9}$$

where $logit'_k$ is the logit produced by the original `NN`, and w_k is a trainable weight indicating the overall confidence for `RE`s that lead to target label k. Here we do not assign a trainable weight for each `RE` because it is often that only a few sentences match a `RE`.

We modify the logit instead of the final probability because a logit is an unconstrained real value, which matches the property of $w_k z_k$ better than probability. Actually, when performing model ensemble, ensembling with logits is often empirically better than with the final probability[3]. This is also the reason why we choose to operate on logits in Sec. 3.3.

4 Evaluation Methodology

Our experiments aim to answer three questions: **Q1:** Does the use of `RE`s enhance the learning quality when the number of annotated instances is small? **Q2:** Does the use of `RE`s still help when using the full training data? **Q3:** How can we choose from different combination methods?

4.1 Datasets

We use the ATIS dataset (Hemphill et al., 1990) to evaluate our approach. This dataset is widely used in `SLU` research. It includes queries of flights, meal, etc. We follow the setup of Liu and Lane (2016) by using 4,978 queries for training and 893 for testing, with 18 intent labels and 127 slot labels. We also split words like *Miami's* into *Miami 's* during the tokenization phase to reduce the number of words that do not have a pre-trained word embedding. This strategy is useful for few-shot learning.

[3] An example can be found in the ensemble version that Juan et al. (2016) used in the Avazu Kaggle competition.

To answer **Q1** , we also exploit the ***full few-shot learning setting***. Specifically, for intent detection, we randomly select 5, 10, 20 training instances for each intent to form the few-shot training set; and for slot filling, we also explore 5, 10, 20 shots settings. However, since a sentence typically contains multiple slots, the number of mentions of frequent slot labels may inevitably exceeds the target shot count. To better approximate the target shot count, we select sentences for each slot label in ascending order of label frequencies. That is k_1-shot dataset will contain k_2-shot dataset if $k_1 > k_2$. All settings use the original test set.

Since most existing few-shot learning methods require either many few-shot classes or some classes with enough data for training, we also explore the ***partial few-shot learning setting*** for intent detection to provide a fair comparison for existing few-shot learning methods. Specifically, we let the 3 most frequent intents have 300 training instances, and the rest remains untouched. This is also a common scenario in real world, where we often have several frequent classes and many classes with limited data. As for slot filling, however, since the number of mentions of frequent slot labels already exceeds the target shot count, the original slot filling few-shot dataset can be directly used to train existing few-shot learning methods. Therefore, we do not distinguish full and partial few-shot learning for slot filling.

4.2 Preparing REs

We use the syntax of `RE`s in Perl in this work. Our `RE`s are written by a paid annotator who is familiar with the domain. It took the annotator in total less than 10 hours to develop all the `RE`s, while a domain expert can accomplish the task faster. We use the 20-shot training data to develop the `RE`s, but word lists like cities are obtained from the full training set. The development of `RE`s is considered completed when the `RE`s can cover most of the cases in the 20-shot training data with resonable precision. After that, the `RE`s are fixed throughout the experiments.

The majority of the time for writing the `RE`s is proportional to the number of `RE` groups. It took about 1.5 hours to write the 54 intent `RE`s with on average 2.2 groups per `RE`. It is straightforward to write the slot `RE`s for the input and output level methods, for which it took around 1 hour to write the 60 `RE`s with 1.7 groups on average. By contrast, writing slot REs to guide attention requires more efforts as the annotator needs to carefully select clue words and annotate the full slot label. As a result, it took about 5.5 hours to generate 115 REs with on average 3.3 groups. The performance of the REs can be found in the last line of Table 1.

In practice, a positive RE for intent (or slot) k can often be treated as negative REs for other intents (or slots). As such, we use the positive REs for intent (or slot) k as the negative REs for other intents (or slots) in our experiments.

4.3 Experimental Setup

Hyper-parameters. Our hyper-parameters for the BLSTM are similar to the ones used by Liu and Lane (2016). Specifically, we use batch size 16, dropout probability 0.5, and BLSTM cell size 100. The attention loss weight is 16 (both positive and negative) for full few-shot learning settings and 1 for other settings. We use the 100d GloVe word vectors (Pennington et al., 2014) pre-trained on Wikipedia and Gigaword (Parker et al., 2011), and the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001.

Evaluation Metrics. We report accuracy and macro-F1 for intent detection, and micro/macro-F1 for slot filling. Micro/macro-F1 are the harmonic mean of micro/macro precision and recall. Macro-precision/recall are calculated by averaging precision/recall of each label, and micro-precision/recall are averaged over each prediction.

Competitors and Naming Conventions. Here, a bold Courier typeface like **BLSTM** denotes the notations of the models that we will compare in Sec. 5.

Specifically, we compare our methods with the baseline **BLSTM** model (Sec. 3.1). Since our attention loss method (Sec. 3.3) uses two-side attention, we include the raw two-side attention model without attention loss (**+two**) for comparison as well. Besides, we also evaluate the RE output (**REO**), which uses the REtags as prediction directly, to show the quality of the REs that we will use in the experiments.[4]

As for our methods for combinging REs with NN, **+feat** refers to using REtag as input features (Sec. 3.2), **+posi** and **+neg** refer to using positive and negative attention loss respectively, **+both** refers to using both postive and negative attention losses (Sec. 3.3), and **+logit** means using REtag to modify NN output (Sec. 3.4).

Moverover, since the REs can also be formatted as first-order-logic (FOL) rules, we also compare our methods with the teacher-student framework proposed by Hu et al. (2016a), which is a general framework for distilling knowledge from FOL rules into NN (**+hu16**). Besides, since we consider few-short learning, we also include the memory module proposed by Kaiser et al. (2017), which performs well in various few-shot datasets (**+mem**)[5]. Finally, the state-of-art model on the ATIS dataset is also included (**L&L16**), which jointly models the intent detection and slot filling in a single network (Liu and Lane, 2016).

5 Experimental Results

5.1 Full Few-Shot Learning

To answer **Q1** , we first explore the full few-shot learning scenario.

Intent Detection. As shown in Table 1, except for 5-shot, all approaches improve the baseline BLSTM. Our network-module-level methods give the best performance because our attention module directly receives signals from the clue words in REs that contain more meaningful information than the REtag itself used by other methods. We also observe that since negative REs are derived from positive REs with some noises, posi performs better than neg when the amount of available data is limited. However, neg is slightly better in 20-shot, possibly because negative REs significantly outnumbers the positive ones. Besides, two alone works better than the BLSTM when there are sufficient data, confirming the advantage of our two-side attention architecture.

As for other proposed methods, the output level method (logit) works generally better than the input level method (feat), except for the 5-shot case. We believe this is due to the fewer number of RE related parameters and the shorter distance that the gradient needs to travel from the loss to these parameters – both make logit easier to train. However, since logit directly modifies the output, the final prediction is more sensitive to the insufficiently trained weights in logit, leading to the inferior results in the 5-shot setting.

[4] For slot filling, we evaluate the REs that use the target slot labels as REtags.

[5] We tune C and π_0 of hu16, and choose (0.1, 0.3) for intent, and (1, 0.3) for slot. We tune memory-size and k of mem, and choose (1024, 64) for intent, and (2048, 64) for slot.

Model Type	Model Name	Intent 5-shot	Intent 10-shot	Intent 20-shot	Slot 5-shot	Slot 10-shot	Slot 20-shot
		Macro-F1 / Accuracy			Macro-F1 / Accuracy		
Base Model	BLSTM	45.28 / 60.02	60.62 / 64.61	63.60 / 80.52	60.78 / 83.91	74.28 / 90.19	80.57 / 93.08
Input Level	+feat	49.40 / 63.72	64.34 / 73.46	65.16 / 83.20	**66.84 / 88.96**	79.67 / **93.64**	84.95 / 95.00
Output Level	+logit	46.01 / 58.68	63.51 / 77.83	69.22 / **89.25**	63.68 / 86.18	76.12 / 91.64	83.71 / 94.43
	+hu16	47.22 / 56.22	61.83 / 68.42	67.40 / 84.10	63.37 / 85.37	75.67 / 91.06	80.85 / 93.47
Network Module Level	+two	40.44 / 57.22	60.72 / 75.14	62.88 / 83.65	60.38 / 83.63	73.22 / 90.08	79.58 / 92.57
	+two+posi	50.90 / 74.47	68.69 / 84.66	72.43 / 85.78	59.59 / 83.47	73.62 / 89.28	78.94 / 92.21
	+two+neg	49.01 / 68.31	64.67 / 79.17	72.32 / 86.34	59.51 / 83.23	72.92 / 89.11	78.83 / 92.07
	+two+both	**54.86 / 75.36**	**71.23 / 85.44**	**75.58** / 88.80	59.47 / 83.35	73.55 / 89.54	79.02 / 92.22
Few-Shot Model	+mem	-	-	-	61.25 / 83.45	77.83 / 90.57	82.98 / 93.49
	+mem+feat	-	-	-	65.08 / 88.07	**80.64** / 93.47	**85.45 / 95.39**
RE Output	REO	70.31 / 68.98			42.33 / 70.79		

Table 1: Results on Full Few-Shot Learning Settings. For slot filling, we do not distinguish full and partial few-shot learning settings (see Sec. 4.1).

To compare with existing methods of combining NN and rules, we also implement the teacher-student network (Hu et al., 2016a). This method lets the NN learn from the posterior label distribution produced by FOL rules in a teacher-student framework, but requires considerable amounts of data. Therefore, although both hu16 and logit operate at the output level, logit still performs better than hu16 in these few-shot settings, since logit is easier to train.

It can also be seen that starting from 10-shot, two+both significantly outperforms pure REO. This suggests that by using our attention loss to connect the distributional representation of the NN and the clue words of REs, we can generalize RE patterns within a NN architecture by using a small amount of annotated data.

Slot Filling. Different from intent detection, as shown in Table 1, our attention loss does not work for slot filling. The reason is that the slot label of a ***target word*** (the word for which we are trying to predict a slot label) is decided mainly by the semantic meaning of the word itself, together with 0-3 phrases in the context to provide supplementary information. However, our attention mechanism can only help in recognizing clue words in the context, which is less important than the word itself and have already been captured by the BLSTM, to some extent. Therefore, the attention loss and the attention related parameters are more of a burden than a benefit. As is shown in Fig. 1, the model recognizes *Boston* as *fromloc.city* mainly because *Boston* itself is a city, and its context word *from* may have already been captured by the BLSTM and our attention mechanism does not help much. By examining the attention values of +two trained on the full dataset, we find that instead of marking informative context words, the attention tends to concentrate on the target word itself. This observation further reinforces our hypothesis on the attention loss.

On the other hand, since the REtags provide extra information, such as type, about words in the sentence, logit and feat generally work better. However, different from intent detection, feat only outperforms logit by a margin. This is because feat can use the REtags of all words to generate better context representations through the NN, while logit can only utilize the REtag of the target word before the final output layer. As a result, feat actually gathers more information from REs and can make better use of them than logit. Again, hu16 is still outperformed by logit, possibly due to the insufficient data support in this few-shot scenario. We also see that even the BLSTM outperforms REO in 5-shot, indicating while it is hard to write high-quality RE patterns, using REs to boost NNs is still feasible.

Summary. The amount of extra information that a NN can utilize from the combined REs significantly affects the resulting performance. Thus, the attention loss methods work best for intent detection and feat works best for slot filling. We also see that the improvements from REs decreases as having more training data. This is not surprising because the implicit knowledge embedded in the REs are likely to have already been captured by a sufficient large annotated dataset and in this scenario using the REs will bring in fewer benefits.

5.2 Partial Few-Shot Learning

To better understand the relationship between our approach and existing few-shot learning methods, we also implement the memory network method

Model	5-shot	10-shot	20-shot
	Macro-F1 / Accuracy		
BLSTM	64.73 / 91.71	78.55 / 96.53	82.05 / 97.20
+hu16	65.22 / 91.94	84.49 / 96.75	84.80 / 97.42
+two	65.59 / 91.04	77.92 / 95.52	81.01 / 96.86
+two+both	66.62 / 92.05	85.75 / 96.98	**87.97 / 97.76**
+mem	67.54 / 91.83	82.16 / 96.75	84.69 / 97.42
+mem+posi	**70.46 / 93.06**	**86.03 / 97.09**	86.69 / 97.65

Table 2: Intent Detection Results on Partial Few-Shot Learning Setting.

Model	Intent	Slot
	Macro-F1/Accuracy	Macro-F1/Micro-F1
BLSTM	92.50 / 98.77	85.01 / 95.47
+feat	91.86 / 97.65	86.7 / 95.55
+logit	92.48 / 98.77	86.94 / 95.42
+hu16	93.09 / 98.77	85.74 / 95.33
+two	93.64 / 98.88	84.45 / 95.05
+two+both	**96.20 / 98.99**	85.44 / 95.27
+mem	93.42 / 98.77	85.72 / 95.37
+mem+posi/feat	94.36 / **98.99**	**87.82 / 95.90**
L&L16	- / 98.43	- / 95.98

Table 3: Results on Full Dataset. The left side of '/' applies for intent, and the right side for slot.

(Kaiser et al., 2017) which achieves good results in various few-shot datasets. We adapt their open-source code, and add their memory module (mem) to our BLSTM model.

Since the memory module requires to be trained on either many few-shot classes or several classes with extra data, we expand our full few-shot dataset for intent detection, so that the top 3 intent labels have 300 sentences (partial few-shot).

As shown in Table 2, mem works better than BLSTM, and our attention loss can be further combined with the memory module (mem+posi), with even better performance. hu16 also works here, but worse than two+both. Note that, the memory module requires the input sentence to have only one embedding, thus we only use one set of positive attention for combination.

As for slot filling, since we already have extra data for frequent tags in the original few-shot data (see Sec. 4.1), we use them directly to run the memory module. As shown in the bottom of Table 1, mem also improves the base BLSTM, and gains further boost when it is combined with feat[6].

5.3 Full Dataset

To answer **Q2**, we also evaluate our methods on the full dataset. As seen in Table 3, for intent detection, while two+both still works, feat and logit no longer give improvements. This shows

[6]For compactness, we only combine the best method in each task with mem, but others can also be combined.

Model	Intent		Slot	
	Macro-F1 / Accuracy		Macro-F1 / Micro-F1	
	Complex	Simple	Complex	Simple
BLSTM	63.60 / 80.52		80.57 / 93.08	
+feat	65.16/**83.20**	**66.51**/80.40	**84.95/95.00**	83.88/94.71
+logit	**69.22/89.25**	65.09/83.09	**83.71/94.43**	83.22/93.94
+both	**75.58/88.80**	74.51/87.46	-	-

Table 4: Results on 20-Shot Data with Simple REs. +both refers to +two +both for short.

that since both REtag and annotated data provide intent labels for the input sentence, the value of the extra noisy tag from RE become limited as we have more annotated data. However, as there is no guidance on attention in the annotations, the clue words from REs are still useful. Further, since feat concatenates REtags at the input level, the powerful NN makes it more likely to overfit than logit, therefore feat performs even worse when compared to the BLSTM.

As for slot filling, introducing feat and logit can still bring further improvements. This shows that the word type information contained in the REtags is still hard to be fully learned even when we have more annotated data. Moreover, different from few-shot settings, two+both has a better macro-F1 score than the BLSTM for this task, suggesting that better attention is still useful when the base model is properly trained.

Again, hu16 outperforms the BLSTM in both tasks, showing that although the REtags are noisy, their teacher-student network can still distill useful information. However, hu16 is a general framework to combine FOL rules, which is more indirect in transferring knowledge from rules to NN than our methods. Therefore, it is still inferior to attention loss in intent detection and feat in slot filling, which are designed to combine REs.

Further, mem generally works in this setting, and can receive further improvement by combining our fusion methods. We can also see that two+both works clearly better than the state-of-art method (L&L16) in intent detection, which jointly models the two tasks. And mem+feat is comparative to L&L16 in slot filling.

5.4 Impact of the RE Complexity

We now discuss how the RE complexity affects the performance of the combination. We choose to control the RE complexity by modifying the number of groups. Specifically, we reduce the number of groups for existing REs to decrease RE complexity. To mimic the process of writing simple

REs from scratch, we try our best to keep the key RE groups. For intent detection, all the REs are reduced to at most 2 groups. As for slot filling, we also reduce the REs to at most 2 groups, and for some simples case, we further reduce them into word-list patterns, e.g., (__CITY).

As shown in Table 4, the simple REs already deliver clear improvements to the base NN models, which shows the effectiveness of our methods, and indicates that simple REs are quite cost-efficient since these simple REs only contain 1-2 RE groups and thus very easy to produce. We can also see that using complex REs generally leads to better results compared to using simple REs. This indicates that when considering using REs to improve a NN model, we can start with simple REs, and gradually increase the RE complexity to improve the performance over time[7].

6 Related Work

Our work builds upon the following techniques, while qualitatively differing from each

NN with Rules. On the initialization side, Li et al. (2017) uses important n-grams to initialize the convolution filters. On the input side, Wang et al. (2017a) uses knowledge base rules to find relevant concepts for short texts to augment input. On the output side, Hu et al. (2016a; 2016b) and Guo et al. (2017) use FOL rules to rectify the output probability of NN, and then let NN learn from the rectified distribution in a teacher-student framework. Xiao et al. (2017), on the other hand, modifies the decoding score of NN by multiplying a weight derived from rules. On the loss function side, people modify the loss function to model the relationship between premise and conclusion (Demeester et al., 2016), and fit both human-annotated and rule-annotated labels (Alashkar et al., 2017). Since fusing in initialization or in loss function often require special properties of the task, these approaches are not applicable to our problem. Our work thus offers new ways to exploit RE rules at different levels of a NN.

NNs and REs. As for NNs and REs, previous work has tried to use RE to speed up the decoding phase of a NN (Strauß et al., 2016) and generating REs from natural language specifications of the RE (Locascio et al., 2016). By contrast, our work aims to use REs to improve the prediction ability of a NN.

Few-Shot Learning. Prior work either considers few-shot learning in a metric learning framework (Koch et al., 2015; Vinyals et al., 2016), or stores instances in a memory (Santoro et al., 2016; Kaiser et al., 2017) to match similar instances in the future. Wang et al. (2017b) further uses the semantic meaning of the class name itself to provide extra information for few-shot learning. Unlike these previous studies, we seek to use the human-generated REs to provide additional information.

Natural Language Understanding. Recurrent neural networks are proven to be effective in both intent detection (Ravuri and Stoicke, 2015) and slot filling (Mesnil et al., 2015). Researchers also find ways to jointly model the two tasks (Liu and Lane, 2016; Zhang and Wang, 2016). However, no work so far has combined REs and NNs to improve intent detection and slot filling.

7 Conclusions

In this paper, we investigate different ways to combine NNs and REs for solving typical SLU tasks. Our experiments demonstrate that the combination clearly improves the NN performance in both the few-shot learning and the full dataset settings. We show that by exploiting the implicit knowledge encoded within REs, one can significantly improve the learning performance. Specifically, we observe that using REs to guide the attention module works best for intent detection, and using REtags as features is an effective approach for slot filling. We provide interesting insights on how REs of various forms can be employed to improve NNs, showing that while simple REs are very cost-effective, complex REs generally yield better results.

Acknowledgement

This work is supported by the National High Technology R&D Program of China (Grant No. 2015AA015403), the National Natural Science Foundation of China (Grant Nos. 61672057 and 61672058); the UK Engineering and Physical Sciences Research Council (EPSRC) under grants EP/M01567X/1 (SANDeRs) and EP/M015793/1 (DIVIDEND); and the Royal Society International Collaboration Grant (IE161012). For any correspondence, please contact Yansong Feng.

[7]We do not include results of both for slot filling since its REs are different from feat and logit, and we have already shown that the attention loss method does not work for slot filling.

References

Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. 2017. Examples-rules guided deep neural network for makeup recommendation. In *AAAI*, pages 941–947.

Angel X Chang and Christopher D Manning. 2014. Tokensregex: Defining cascaded regular expressions over tokens. *Tech. Rep. CSTR 2014-02*.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted rule injection for relation embeddings. *arXiv preprint arXiv:1606.08359*.

Paul Suganthan GC, Chong Sun, Haojun Zhang, Frank Yang, Narasimhan Rampalli, Shishir Prasad, Esteban Arcaute, Ganesh Krishnan, Rohit Deep, Vijay Raghavendra, et al. 2015. Why big data industrial systems need rules and what we can do about it. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 265–276. ACM.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2017. Knowledge graph embedding with iterative guidance from soft rules. *arXiv preprint arXiv:1711.11231*.

Charles T Hemphill, John J Godfrey, George R Doddington, et al. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016a. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. 2016b. Deep neural networks with massive learned knowledge. In *EMNLP*, pages 1670–1679.

Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM.

Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Shen Li, Zhe Zhao, Tao Liu, Renfen Hu, and Xiaoyong Du. 2017. Initializing convolutional filters with semantic features for text classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1885–1890.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. *arXiv preprint arXiv:1608.03000*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(3):530–539.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, linguistic data consortium. *Google Scholar*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Suman Ravuri and Andreas Stoicke. 2015. A comparative study of neural network models for lexical intent classification. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 368–374. IEEE.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Tobias Strauß, Gundram Leifert, Tobias Grüning, and Roger Labahn. 2016. Regular expressions for decoding of neural network outputs. *Neural Networks*, 79:1–11.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017a. Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2915–2921. AAAI Press.

Peng Wang, Lingqiao Liu, Chunhua Shen, Zi Huang, Anton van den Hengel, and Heng Tao Shen. 2017b. Multi-attention network for one shot learning. In

2017 IEEE conference on computer vision and pattern recognition, CVPR, pages 22–25.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2017. Symbolic priors for rnn-based semantic parsing. In *wenty-sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 4186–4192.

Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. 2016. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, pages 2993–2999.

Token-level and sequence-level loss smoothing for RNN language models

Maha Elbayad[1,2] **Laurent Besacier**[1] **Jakob Verbeek**[2]
Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG, LJK, F-38000 Grenoble France
[1] `firstname.lastname@univ-grenoble-alpes.fr`
[2] `firstname.lastname@inria.fr`

Abstract

Despite the effectiveness of recurrent neural network language models, their maximum likelihood estimation suffers from two limitations. It treats all sentences that do not match the ground truth as equally poor, ignoring the structure of the output space. Second, it suffers from "exposure bias": during training tokens are predicted given ground-truth sequences, while at test time prediction is conditioned on generated output sequences. To overcome these limitations we build upon the recent reward augmented maximum likelihood approach *i.e.* sequence-level smoothing that encourages the model to predict sentences close to the ground truth according to a given performance metric. We extend this approach to token-level loss smoothing, and propose improvements to the sequence-level smoothing approach. Our experiments on two different tasks, image captioning and machine translation, show that token-level and sequence-level loss smoothing are complementary, and significantly improve results.

1 Introduction

Recurrent neural networks (RNNs) have recently proven to be very effective sequence modeling tools, and are now state of the art for tasks such as machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015), image captioning (Kiros et al., 2014; Vinyals et al., 2015; Anderson et al., 2017) and automatic speech recognition (Chorowski et al., 2015; Chiu et al., 2017).

The basic principle of RNNs is to iteratively compute a vectorial sequence representation, by applying at each time-step the same trainable function to compute the new network state from the previous state and the last symbol in the sequence. These models are typically trained by maximizing the likelihood of the target sentence given an encoded source (text, image, speech).

Maximum likelihood estimation (MLE), however, has two main limitations. First, the training signal only differentiates the ground-truth target output from all other outputs. It treats all other output sequences as equally incorrect, regardless of their semantic proximity from the ground-truth target. While such a "zero-one" loss is probably acceptable for coarse grained classification of images, *e.g.* across a limited number of basic object categories (Everingham et al., 2010) it becomes problematic as the output space becomes larger and some of its elements become semantically similar to each other. This is in particular the case for tasks that involve natural language generation (captioning, translation, speech recognition) where the number of possible outputs is practically unbounded. For natural language generation tasks, evaluation measures typically do take into account structural similarity, *e.g.* based on n-grams, but such structural information is not reflected in the MLE criterion. The second limitation of MLE is that training is based on predicting the next token given the input and preceding *ground-truth* output tokens, while at test time the model predicts conditioned on the input and the so-far *generated* output sequence. Given the exponentially large output space of natural language sentences, it is not obvious that the learned RNNs generalize well beyond the relatively sparse distribution of ground-truth sequences used during MLE optimization. This phenomenon is known as "exposure bias" (Ranzato et al., 2016; Bengio et al., 2015).

MLE minimizes the KL divergence between a target Dirac distribution on the ground-truth sentence(s) and the model's distribution. In this paper, we build upon the "loss smoothing" approach by Norouzi et al. (2016), which smooths the Dirac target distribution over similar sentences, increasing the support of the training data in the output space. We make the following main contributions:

- We propose a token-level loss smoothing approach, using word-embeddings, to achieve smoothing among semantically similar terms, and we introduce a special procedure to promote rare tokens.
- For sequence-level smoothing, we propose to use restricted token replacement vocabularies, and a "lazy evaluation" method that significantly speeds up training.
- We experimentally validate our approach on the MSCOCO image captioning task and the WMT'14 English to French machine translation task, showing that on both tasks combining token-level and sequence-level loss smoothing improves results significantly over maximum likelihood baselines.

In the remainder of the paper, we review the existing methods to improve RNN training in Section 2. Then, we present our token-level and sequence-level approaches in Section 3. Experimental evaluation results based on image captioning and machine translation tasks are laid out in Section 4.

2 Related work

Previous work aiming to improve the generalization performance of RNNs can be roughly divided into three categories: those based on regularization, data augmentation, and alternatives to maximum likelihood estimation.

Regularization techniques are used to increase the smoothness of the function learned by the network, *e.g.* by imposing an ℓ_2 penalty on the network weights, also known as "weight decay". More recent approaches mask network activations during training, as in dropout (Srivastava et al., 2014) and its variants adapted to recurrent models (Pham et al., 2014; Krueger et al., 2017). Instead of masking, batch-normalization (Ioffe and Szegedy, 2015) rescales the network activations to avoid saturating the network's non-linearities. Instead of regularizing the network parameters or activations, it is also possible to directly regularize based on the entropy of the output distribution (Pereyra et al., 2017).

Data augmentation techniques improve the robustness of the learned models by applying transformations that might be encountered at test time to the training data. In computer vision, this is common practice, and implemented by, *e.g.*, scaling, cropping, and rotating training images (LeCun et al., 1998; Krizhevsky et al., 2012; Paulin et al., 2014). In natural language processing, examples of data augmentation include input noising by randomly dropping some input tokens (Iyyer et al., 2015; Bowman et al., 2015; Kumar et al., 2016), and randomly replacing words with substitutes sampled from the model (Bengio et al., 2015). Xie et al. (2017) introduced data augmentation schemes for RNN language models that leverage n-gram statistics in order to mimic Kneser-Ney smoothing of n-grams models. In the context of machine translation, Fadaee et al. (2017) modify sentences by replacing words with rare ones when this is plausible according to a pre-trained language model, and substitutes its equivalent in the target sentence using automatic word alignments. This approach, however, relies on the availability of additional monolingual data for language model training.

The *de facto* standard way to train RNN language models is maximum likelihood estimation (MLE) (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). The sequential factorization of the sequence likelihood generates an additive structure in the loss, with one term corresponding to the prediction of each output token given the input and the preceding ground-truth output tokens. In order to directly optimize for sequence-level structured loss functions, such as measures based on n-grams like BLEU or CIDER, Ranzato et al. (2016) use reinforcement learning techniques that optimize the expectation of a sequence-level reward. In order to avoid early convergence to poor local optima, they pre-train the model using MLE.

Leblond et al. (2018) build on the learning to search approach to structured prediction (Daumé III et al., 2009; Chang et al., 2015) and adapts it to RNN training. The model generates candidate sequences at each time-step using all possible tokens, and scores these at sequence-level to derive a training signal for each time step. This leads to an approach that is structurally close to MLE, but computationally expensive. Norouzi et al. (2016) introduce a reward augmented maximum likelihood (RAML) approach, that incorporates a notion of sequence-level reward without facing the difficulties of reinforcement learning. They define a target distribution over output sentences using a soft-max over the reward over all possible outputs. Then, they minimize the KL divergence between the target distribution and the model's output distribution. Training with a general reward distribution is similar to MLE training, except that we use multiple sentences sampled from the target distribution instead of only the ground-truth sentences.

In our work, we build upon the work of Norouzi et al. (2016) by proposing improvements to sequence-level smoothing, and extending it to token-level smoothing. Our token-level smoothing approach is related to the label smoothing approach of Szegedy et al. (2016) for image classification. Instead of maximizing the probability of the correct class, they train the model to predict the correct class with a large probability and all other classes with a small uniform probability. This regularizes the model by preventing overconfident predictions. In natural language generation with large vocabularies, preventing such "narrow" over-confident distributions is imperative, since for many tokens there are nearly interchangeable alternatives.

3 Loss smoothing for RNN training

We briefly recall standard recurrent neural network training, before presenting sequence-level and token-level loss smoothing below.

3.1 Maximum likelihood RNN training

We are interested in modeling the conditional probability of a sequence $y = (y_1, \dots, y_T)$ given a conditioning observation x,

$$p_\theta(y|x) = \prod_{t=1}^{T} p_\theta(y_t|x, y_{<t}), \tag{1}$$

where $y_{<t} = (y_1, \dots, y_{t-1})$, the model parameters are given by θ, and x is a source sentence or an image in the contexts of machine translation and image captioning, respectively.

In a recurrent neural network, the sequence y is predicted based on a sequence of states h_t,

$$p_\theta(y_t|x, y_{<t}) = p_\theta(y_t|h_t), \tag{2}$$

where the RNN state is computed recursively as

$$h_t = \begin{cases} f_\theta(h_{t-1}, y_{t-1}, x) & \text{for } t \in \{1, ..T\}, \\ g_\theta(x) & \text{for } t = 0. \end{cases} \tag{3}$$

The input is encoded by g_θ and used to initialize the state sequence, and f_θ is a non-linear function that updates the state given the previous state h_{t-1}, the last output token y_{t-1}, and possibly the input x. The state update function can take different forms, the ones including gating mechanisms such as LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Chung et al., 2014) are particularly effective to model long sequences.

In standard teacher-forced training, the hidden states will be computed by forwarding the ground truth sequence y^* *i.e.* in Eq. (3), the RNN has access to the true previous token y^*_{t-1}. In this case we will note the hidden states h^*_t.

Given a ground-truth target sequence y^*, maximum likelihood estimation (MLE) of the network parameters θ amounts to minimizing the loss

$$\ell_{\text{MLE}}(y^*, x) = -\ln p_\theta(y^*|x) \tag{4}$$

$$= -\sum_{t=1}^{T} \ln p_\theta(y^*_t|h^*_t). \tag{5}$$

The loss can equivalently be expressed as the KL-divergence between a Dirac centered on the target output (with $\delta_a(x) = 1$ at $x = a$ and 0 otherwise) and the model distribution, either at the sequence-level or at the token-level:

$$\ell_{\text{MLE}}(y^*, x) = D_{\text{KL}}\big(\delta_{y^*}||p_\theta(y|x)\big) \tag{6}$$

$$= \sum_{t=1}^{T} D_{\text{KL}}\big(\delta_{y^*_t}||p_\theta(y_t|h^*_t)\big). \tag{7}$$

Loss smoothing approaches considered in this paper consist in replacing the Dirac on the ground-truth sequence with distributions with larger support. These distributions can be designed in such a manner that they reflect which deviations from ground-truth predictions are preferred over others.

3.2 Sequence-level loss smoothing

The reward augmented maximum likelihood approach of Norouzi et al. (2016) consists in replacing the sequence-level Dirac δ_{y^*} in Eq. (6) with a distribution

$$r(y|y^*) \propto \exp r(y, y^*)/\tau, \tag{8}$$

where $r(y, y^*)$ is a "reward" function that measures the quality of sequence y w.r.t. y^*, *e.g.* metrics used for evaluation of natural language processing tasks can be used, such as BLEU (Papineni et al., 2002) or CIDEr (Vedantam et al., 2015). The temperature parameter τ controls the concentration of the distribution around y^*. When $m > 1$ ground-truth sequences are paired with the same input x, the reward function can be adapted to fit this setting and be defined as $r(y, \{y^{*(1)}, \dots, y^{*(m)}\})$. The sequence-level smoothed loss function is then given by

$$\begin{aligned}\ell_{\text{Seq}}(y^*, x) &= D_{\text{KL}}\big(r(y|y^*)||p_\theta(y|x)\big)\\ &= H(r(y|y^*)) - \mathbb{E}_r[\ln p_\theta(y|x)]\,, \quad (9)\end{aligned}$$

where the entropy term $H(r(y|y^*))$ does not depend on the model parameters θ.

In general, expectation in Eq. (9) is intractable due to the exponentially large output space, and replaced with a Monte-Carlo approximation:

$$\mathbb{E}_r[-\ln p_\theta(y|x)] \approx -\sum_{l=1}^{L} \ln p_\theta(y^l|x). \qquad (10)$$

Stratified sampling. Norouzi et al. (2016) show that when using the Hamming or edit distance as a reward, we can sample directly from $r(y|y^*)$ using a stratified sampling approach. In this case sampling proceeds in three stages. (i) Sample a distance d from $\{0, \dots, T\}$ from a prior distribution on d. (ii) Uniformly select d positions in the sequence to be modified. (iii) Sample the d substitutions uniformly from the token vocabulary.

Details on the construction of the prior distribution on d for a reward based on the Hamming distance can be found in Appendix A.

Importance sampling. For a reward based on BLEU or CIDEr , we cannot directly sample from $r(y|y^*)$ since the normalizing constant, or "partition function", of the distribution is intractable to compute. In this case we can resort to importance sampling. We first sample L sequences y^l from a tractable proposal distribution $q(y|y^*)$. We then compute the importance weights

$$\omega_l \approx \frac{r(y^l|y^*)/q(y^l|y^*)}{\sum_{k=1}^{L} r(y^k|y^*)/q(y^k|y^*)}, \qquad (11)$$

where $r(y^k|y^*)$ is the un-normalized reward distribution in Eq. (8). We finally approximate the expectation by reweighing the samples in the Monte Carlo approximation as

$$\mathbb{E}_r[-\ln p_\theta(y|x)] \approx -\sum_{l=1}^{L} \omega_l \ln p_\theta(y^l|x). \qquad (12)$$

In our experiments we use a proposal distribution based on the Hamming distance, which allows for tractable stratified sampling, and generates sentences that do not stray away from the ground truth.

We propose two modifications to the sequence-level loss smoothing of Norouzi et al. (2016): sampling to a restricted vocabulary (described in the following paragraph) and lazy sequence-level smoothing (described in section 3.4).

Restricted vocabulary sampling. In the stratified sampling method for Hamming and edit distance rewards, instead of drawing from the large vocabulary $\mathcal{V}$, containing typically in the order of 10^4 words or more, we can restrict ourselves to a smaller subset $\mathcal{V}_{sub}$ more adapted to our task. We considered three different possibilities for $\mathcal{V}_{sub}$.

$\mathcal{V}$: the full vocabulary from which we sample uniformly (default), or draw from our token-level smoothing distribution defined below in Eq. (13).

$\mathcal{V}_{refs}$: uniformly sample from the set of tokens that appear in the ground-truth sentence(s) associated with the current input.

$\mathcal{V}_{batch}$: uniformly sample from the tokens that appear in the ground-truth sentences across all inputs that appear in a given training mini-batch.

Uniformly sampling from $\mathcal{V}_{batch}$ has the effect of boosting the frequencies of words that appear in many reference sentences, and thus approximates to some extent sampling substitutions from the uni-gram statistics of the training set.

3.3 Token-level loss smoothing

While the sequence-level smoothing can be directly based on performance measures of interest such as BLEU or CIDEr, the support of the smoothed distribution is limited to the number of samples drawn during training. We propose smoothing the token-level Diracs $\delta_{y_t^*}$ in Eq. (7) to increase its support to similar tokens. Since we apply smoothing to each of the tokens independently, this approach implicitly increases the support to an exponential number of sequences, unlike the sequence-level smoothing approach. This comes at the price, however, of a naive token-level independence assumption in the smoothing.

We define the smoothed token-level distribution, similar as the sequence-level one, as a softmax over a token-level "reward" function,

$$r(y_t|y_t^*) \propto \exp r(y_t, y_t^*)/\tau, \qquad (13)$$

where τ is again a temperature parameter. As a token-level reward $r(y_t, y_t^*)$ we use the cosine similarity between y_t and y_t^* in a semantic word-embedding space. In our experiments we use GloVe (Pennington et al., 2014); preliminary experiments with word2vec (Mikolov et al., 2013) yielded somewhat worse results.

Promoting rare tokens. We can further improve the token-level smoothing by promoting rare tokens. To do so, we penalize frequent tokens when smoothing over the vocabulary, by subtracting $\beta\,\mathrm{freq}(y_t)$ from the reward, where $\mathrm{freq}(\cdot)$ denotes the term frequency and β is a non-negative weight. This modification encourages frequent tokens into considering the rare ones. We experimentally found that it is also beneficial for rare tokens to boost frequent ones, as they tend to have mostly rare tokens as neighbors in the word-embedding space. With this in mind, we define a new token-level reward as:

$$r^{\mathrm{freq}}(y_t, y_t^*) = r(y_t, y_t^*) - \beta \min\left(\frac{\mathrm{freq}(y_t)}{\mathrm{freq}(y_t^*)}, \frac{\mathrm{freq}(y_t^*)}{\mathrm{freq}(y_t)}\right), \quad (14)$$

where the penalty term is strongest if both tokens have similar frequencies.

3.4 Combining losses

In both loss smoothing methods presented above, the temperature parameter τ controls the concentration of the distribution. As τ gets smaller the distribution peaks around the ground-truth, while for large τ the uniform distribution is approached. We can, however, not separately control the spread of the distribution and the mass reserved for the ground-truth output. We therefore introduce a second parameter $\alpha \in [0, 1]$ to interpolate between the Dirac on the ground-truth and the smooth distribution. Using $\bar{\alpha} = 1 - \alpha$, the sequence-level and token-level loss functions are then defined as

$$\begin{aligned}\ell^{\alpha}_{\mathrm{Seq}}(y^*, x) &= \alpha \ell_{\mathrm{Seq}}(y^*, x) + \bar{\alpha}\ell_{\mathrm{MLE}}(y^*, x) \quad (15)\\ &= \alpha \mathbb{E}_r[\ell_{\mathrm{MLE}}(y, x)] + \bar{\alpha}\ell_{\mathrm{MLE}}(y^*, x)\\ \ell^{\alpha}_{\mathrm{Tok}}(y^*, x) &= \alpha \ell_{\mathrm{Tok}}(y^*, x) + \bar{\alpha}\ell_{\mathrm{MLE}}(y^*, x) \quad (16)\end{aligned}$$

To benefit from both sequence-level and token-level loss smoothing, we also combine them by applying token-level smoothing to the different sequences sampled for the sequence-level smoothing. We introduce two mixing parameters α_1 and α_2. The first controls to what extent sequence-level smoothing is used, while the second controls to what extent token-level smoothing is used. The combined loss is defined as

$$\begin{aligned}\ell^{\alpha_1,\alpha_2}_{\mathrm{Seq,\,Tok}}(y^*, x, r) &= \alpha_1 \mathbb{E}_r[\ell_{\mathrm{Tok}}(y, x)] + \bar{\alpha}_1 \ell_{\mathrm{Tok}}(y^*, x)\\ &= \alpha_1 \mathbb{E}_r[\alpha_2 \ell_{\mathrm{Tok}}(y, x) + \bar{\alpha}_2 \ell_{\mathrm{MLE}}(y, x)]\\ &\quad + \bar{\alpha}_1(\alpha_2 \ell_{\mathrm{Tok}}(y^*, x) + \bar{\alpha}_2 \ell_{\mathrm{MLE}}(y^*, x)). \quad (17)\end{aligned}$$

In our experiments, we use held out validation data to set mixing and temperature parameters.

Algorithm 1 Sequence-level smoothing algorithm

```
Input: $x, y^*$
Output: $\ell^{\alpha}_{\mathrm{seq}}(x, y^*)$
  Encode $x$ to initialize the RNN
  Forward $y^*$ in the RNN to compute the hidden states $h_t^*$
  Compute the MLE loss $\ell_{\mathrm{MLE}}(y^*, x)$
  for $l \in \{1, \ldots, L\}$ do
      Sample $y^l \sim r(|y^*)$
      if Lazy then
          Compute $\ell(y^l, x) = -\sum_t \log p_\theta(y_t^l | h_t^*)$
      else
          Forward $y^l$ in the RNN to get its hidden states $h_t^l$
          Compute $\ell(y^l, x) = \ell_{\mathrm{MLE}}(y^l, x)$
      end if
  end for
  $\ell^{\alpha}_{\mathrm{Seq}}(x, y^*) = \bar{\alpha}\ell_{\mathrm{MLE}}(y^*, x) + \frac{\alpha}{L}\sum_l \ell(y^l, x)$
```

Lazy sequence smoothing. Although sequence-level smoothing is computationally efficient compared to reinforcement learning approaches (Ranzato et al., 2016; Rennie et al., 2017), it is slower compared to MLE. In particular, we need to forward each of the samples y^l through the RNN in teacher-forcing mode so as to compute its hidden states h_t^l, which are used to compute the sequence MLE loss as

$$\ell_{\mathrm{MLE}}(y^l, x) = -\sum_{t=1}^{T} \ln p_\theta(y_t^l | h_t^l). \quad (18)$$

To speed up training, and since we already forward the ground truth sequence in the RNN to evaluate the MLE part of $\ell^{\alpha}_{\mathrm{Seq}}(y^*, x)$, we propose to use the same hidden states h_t^* to compute both the MLE and the sequence-level smoothed loss. In this case:

$$\ell_{\mathrm{lazy}}(y^l, x) = -\sum_{t=1}^{T} \ln p_\theta(y_t^l | h_t^*) \quad (19)$$

In this manner, we only have a single instead of $L + 1$ forwards-passes in the RNN. We provide the pseudo-code for training in Algorithm 1.

			Without attention			With attention		
Loss	Reward	$\mathcal{V}_{sub}$	BLEU-1	BLEU-4	CIDEr	BLEU-1	BLEU-4	CIDEr
MLE			70.63	30.14	93.59	73.40	33.11	101.63
MLE + γH			70.79	30.29	93.61	72.68	32.15	99.77
Tok	Glove sim		71.94	31.27	95.79	73.49	32.93	102.33
Tok	Glove sim r^{freq}		72.39	31.76	97.47	74.01	33.25	102.81
Seq	Hamming	$\mathcal{V}$	71.76	31.16	96.37	73.12	32.71	101.25
Seq	Hamming	$\mathcal{V}_{batch}$	71.46	31.15	**96.53**	73.26	**32.73**	101.90
Seq	Hamming	$\mathcal{V}_{refs}$	**71.80**	**31.63**	96.22	**73.53**	32.59	**102.33**
Seq, lazy	Hamming	$\mathcal{V}$	70.81	30.43	94.26	73.29	32.81	101.58
Seq, lazy	Hamming	$\mathcal{V}_{batch}$	71.85	31.13	**96.65**	73.43	32.95	**102.03**
Seq, lazy	Hamming	$\mathcal{V}_{refs}$	**71.96**	**31.23**	95.34	**73.53**	**33.09**	101.89
Seq	CIDEr	$\mathcal{V}$	71.05	30.46	94.40	73.08	32.51	101.84
Seq	CIDEr	$\mathcal{V}_{batch}$	71.51	31.17	95.78	**73.50**	**33.04**	**102.98**
Seq	CIDEr	$\mathcal{V}_{refs}$	**71.93**	**31.41**	**96.81**	73.42	32.91	102.23
Seq, lazy	CIDEr	$\mathcal{V}$	71.43	**31.18**	**96.32**	73.55	**33.19**	**102.94**
Seq, lazy	CIDEr	$\mathcal{V}_{batch}$	71.47	31.00	95.56	73.18	32.60	101.30
Seq, lazy	CIDEr	$\mathcal{V}_{refs}$	**71.82**	31.06	95.66	**73.92**	33.10	102.64
Tok-Seq	Hamming	$\mathcal{V}$	70.79	30.43	96.34	73.68	32.87	101.11
Tok-Seq	Hamming	$\mathcal{V}_{batch}$	72.28	31.65	96.73	73.86	33.32	102.90
Tok-Seq	Hamming	$\mathcal{V}_{refs}$	72.69	32.30	98.01	73.56	33.00	101.72
Tok-Seq	CIDEr	$\mathcal{V}$	70.80	30.55	96.89	73.31	32.40	100.33
Tok-Seq	CIDEr	$\mathcal{V}_{batch}$	72.13	31.71	96.92	73.61	32.67	101.41
Tok-Seq	CIDEr	$\mathcal{V}_{refs}$	**73.08**	**32.82**	**99.92**	**74.28**	**33.34**	**103.81**

Table 1: MS-COCO 's test set evaluation measures.

4 Experimental evaluation

In this section, we compare sequence prediction models trained with maximum likelihood (MLE) with our token and sequence-level loss smoothing on two different tasks: image captioning and machine translation.

4.1 Image captioning

4.1.1 Experimental setup.

We use the MS-COCO datatset (Lin et al., 2014), which consists of 82k training images each annotated with five captions. We use the standard splits of Karpathy and Li (2015), with 5k images for validation, and 5k for test. The test set results are generated via beam search (beam size 3) and are evaluated with the MS-COCO captioning evaluation tool. We report CIDEr and BLEU scores on this internal test set. We also report results obtained on the official MS-COCO server that additionally measures METEOR (Denkowski and Lavie, 2014) and ROUGE-L (Lin, 2004). We experiment with both non-attentive LSTMs (Vinyals et al., 2015) and the ResNet baseline of the state-of-the-art top-down attention (Anderson et al., 2017).

The MS-COCO vocabulary consists of 9,800 words that occur at least 5 times in the training set. Additional details and hyperparameters can be found in Appendix B.1.

4.1.2 Results and discussion

Restricted vocabulary sampling In this section, we evaluate the impact of the vocabulary subset from which we sample the modified sentences for sequence-level smoothing. We experiment with two rewards: CIDEr , which scores w.r.t. all five available reference sentences, and Hamming distance reward taking only a single reference into account. For each reward we train our (Seq) models with each of the three subsets detailed previously in Section 3.2, Restricted vocabulary sampling.

From the results in Table 1 we note that for the inattentive models, sampling from $\mathcal{V}_{refs}$ or $\mathcal{V}_{batch}$ has a better performance than sampling from the full vocabulary on all metrics. In fact, using these subsets introduces a useful bias to the model and improves performance. This improvement is most notable using the CIDEr reward that scores candidate sequences w.r.t. to multiple references, which stabilizes the scoring of the candidates.

With an attentive decoder, no matter the reward, re-sampling sentences with words from $\mathcal{V}_{ref}$ rather than the full vocabulary $\mathcal{V}$ is better for both reward functions, and all metrics. Additional experimental results, presented in Appendix B.2, obtained with a BLEU-4 reward, in its single and

	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr		SPICE	
	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40
Google NIC$^+$ (Vinyals et al., 2015)	71.3	89.5	54.2	80.2	40.7	69.4	30.9	58.7	25.4	34.6	53.0	68.2	94.3	94.6	18.2	63.6
Hard-Attention (Xu et al., 2015)	70.5	88.1	52.8	77.9	38.3	65.8	27.7	53.7	24.1	32.2	51.6	65.4	86.5	89.3	17.2	59.8
ATT-FCN$^+$ (You et al., 2016)	73.1	90.0	56.5	81.5	42.4	70.9	31.6	59.9	25.0	33.5	53.5	68.2	94.3	95.8	18.2	63.1
Review Net$^+$ (Yang et al., 2016)	72.0	90.0	55.0	81.2	41.4	70.5	31.3	59.7	25.6	34.7	53.3	68.6	96.5	96.9	18.5	64.9
Adaptive$^+$ (Lu et al., 2017)	74.8	92.0	58.4	84.5	44.4	74.4	33.6	63.7	26.4	35.9	55.0	70.5	104.2	105.9	19.7	67.3
SCST:Att2all$^{+\dagger}$ (Rennie et al., 2017)	78.1	93.7	61.9	86.0	47.0	75.9	35.2	64.5	27.0	35.5	56.3	70.7	114.7	116.7	-	-
LSTM-A3$^{+\dagger\circ}$ (Yao et al., 2017)	78.7	93.7	62.7	86.7	47.6	76.5	35.6	65.2	27.0	35.4	56.4	70.5	116	118	-	-
Up-Down$^{+\dagger\circ}$ (Anderson et al., 2017)	80.2	95.2	64.1	88.8	49.1	79.4	36.9	68.5	27.6	36.7	57.1	72.4	117.9	120.5	-	-
Ours: Tok-Seq CIDEr	72.6	89.7	55.7	80.9	41.2	69.8	30.2	58.3	25.5	34.0	53.5	68.0	96.4	99.4	-	-
Ours: Tok-Seq CIDEr $^+$	74.9	92.4	58.5	84.9	44.8	75.1	34.3	64.7	26.5	36.1	55.2	71.1	103.9	104.2	-	-

Table 2: MS-COCO 's server evaluation . ($^+$) for ensemble submissions, (†) for submissions with CIDEr optimization and ($^\circ$) for models using additional data.

multiple references variants, further corroborate this conclusion.

Lazy training. From the results of Table 1, we see that lazy sequence-level smoothing is competitive with exact non-lazy sequence-level smoothing, while requiring roughly equivalent training time as MLE. We provide detailed timing results in Appendix B.3.

Overall For reference, we include in Table 1 baseline results obtained using MLE, and our implementation of MLE with entropy regularization (MLE+γH) (Pereyra et al., 2017), as well as the RAML approach of Norouzi et al. (2016) which corresponds to sequence-level smoothing based on the Hamming reward and sampling replacements from the full vocabulary (Seq, Hamming, $\mathcal{V}$)

We observe that entropy smoothing is not able to improve performance much over MLE for the model without attention, and even deteriorates for the attention model. We improve upon RAML by choosing an adequate subset of vocabulary for substitutions.

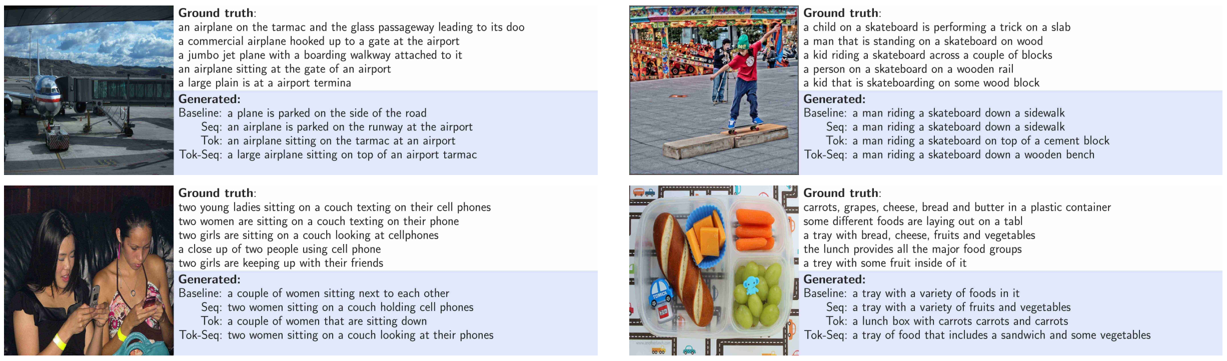
We also report the performances of token-level smoothing, where the promotion of rare tokens boosted the scores in both attentive and non-attentive models.

For sequence-level smoothing, choosing a task-relevant reward with importance sampling yielded better results than plain Hamming distance.

Moreover, we used the two smoothing schemes (Tok-Seq) and achieved the best results with CIDEr as a reward for sequence-level smoothing combined with a token-level smoothing that promotes rare tokens improving CIDEr from 93.59 (MLE) to 99.92 for the model without attention, and improving from 101.63 to 103.81 with attention.

Qualitative results. In Figure 1 we showcase captions obtained with MLE and our three variants of smoothing *i.e.* token-level (Tok), sequence-level (Seq) and the combination (Tok-Seq). We note that the sequence-level smoothing tend to generate lengthy captions overall, which is maintained in the combination. On the other hand, the token-level smoothing allows for a better recognition of objects in the image that stems from the robust training of the classifier *e.g.* the 'cement block' in the top right image or the carrots in the bottom right. More examples are available in Appendix B.4

Comparison to the state of the art. We compare our model to state-of-the-art systems on the MS-COCO evaluation server in Table 2. We submitted a single model (Tok-Seq, CIDEr , $\mathcal{V}_{refs}$) as well as an ensemble of five models with different initializations trained on the training set plus 35k images from the dev set (a total of 117k images) to the MS-COCO server. The three best results on the server (Rennie et al., 2017; Yao et al., 2017; Anderson et al., 2017) are trained in two stages where they first train using MLE, before switching to policy gradient methods based on CIDEr. Anderson et al. (2017) reported an increase of 5.8% of CIDEr on the test split after the CIDEr optimization. Moreover, Yao et al. (2017) uses additional information about image regions to train the attributes classifiers, while Anderson et al. (2017) pre-trains its bottom-up attention model on the Visual Genome dataset (Krishna et al., 2017). Lu et al. (2017); Yao et al. (2017) use the same CNN encoder as ours (ResNet-152), (Vinyals et al., 2015; Yang et al., 2016) use Inception-v3 (Szegedy et al., 2016) for image encoding and Rennie et al. (2017); Anderson et al.



Ground truth:
an airplane on the tarmac and the glass passageway leading to its doo
a commercial airplane hooked up to a gate at the airport
a jumbo jet plane with a boarding walkway attached to it
an airplane sitting at the gate of an airport
a large plain is at a airport termina

Generated:
Baseline: a plane is parked on the side of the road
Seq: an airplane is parked on the runway at the airport
Tok: an airplane sitting on the tarmac at an airport
Tok-Seq: a large airplane sitting on top of an airport tarmac

Ground truth:
a child on a skateboard is performing a trick on a slab
a man that is standing on a skateboard on wood
a kid riding a skateboard across a couple of blocks
a person on a skateboard on a wooden rail
a kid that is skateboarding on some wood block

Generated:
Baseline: a man riding a skateboard down a sidewalk
Seq: a man riding a skateboard down a sidewalk
Tok: a man riding a skateboard on top of a cement block
Tok-Seq: a man riding a skateboard down a wooden bench

Ground truth:
two young ladies sitting on a couch texting on their cell phones
two women are sitting on a couch texting on their phone
two girls are sitting on a couch looking at cellphones
a close up of two people using cell phone
two girls are keeping up with their friends

Generated:
Baseline: a couple of women sitting next to each other
Seq: two women sitting on a couch holding cell phones
Tok: a couple of women that are sitting down
Tok-Seq: two women sitting on a couch looking at their phones

Ground truth:
carrots, grapes, cheese, bread and butter in a plastic container
some different foods are laying out on a tabl
a tray with bread, cheese, fruits and vegetables
the lunch provides all the major food groups
a trey with some fruit inside of it

Generated:
Baseline: a tray with a variety of foods in it
Seq: a tray with a variety of fruits and vegetables
Tok: a lunch box with carrots carrots and carrots
Tok-Seq: a tray of food that includes a sandwich and some vegetables

Figure 1: Examples of generated captions with the baseline MLE and our models with attention.

(2017) use Resnet-101, both of which have similar performances to ResNet-152 on ImageNet classification (Canziani et al., 2016).

4.2 Machine translation

4.2.1 Experimental setup.

For this task we validate the effectiveness of our approaches on two different datasets. The first is WMT'14 English to French, in its filtered version, with 12M sentence pairs obtained after dynamically selecting a "clean" subset of 348M words out of the original "noisy" 850M words (Bahdanau et al., 2015; Cho et al., 2014; Sutskever et al., 2014). The second benchmark is IWSLT'14 German to English consisting of around 150k pairs for training. In all our experiments we use the attentive model of (Bahdanau et al., 2015) The hyperparameters of each of these models as well as any additional pre-processing can be found in Appendix C.1

To assess the translation quality we report the BLEU-4 metric.

4.2.2 Results and analysis

Loss	Reward	$\mathcal{V}_{sub}$	**WMT'14**	**IWSLT'14**
MLE			30.03	27.55
tok	Glove sim		30.16	27.69
tok	Glove sim r^{freq}		30.19	27.83
Seq	Hamming	$\mathcal{V}$	30.85	27.98
Seq	Hamming	$\mathcal{V}_{batch}$	31.18	28.54
Seq	BLEU-4	$\mathcal{V}_{batch}$	31.29	28.56
Tok-Seq	Hamming	$\mathcal{V}_{batch}$	31.36	28.70
Tok-Seq	BLEU-4	$\mathcal{V}_{batch}$	31.39	28.74

Table 3: Tokenized BLEU score on WMT'14 En-Fr evaluated on the news-test-2014 set. And Tokenzied, case-insensitive BLEU on IWSLT'14 De-En.

We present our results in Table 3. On both benchmarks, we improve on both MLE and RAML approach of Norouzi et al. (2016) (Seq, Hamming, $\mathcal{V}$): using the smaller batch-vocabulary for replacement improves results, and using importance sampling based on BLEU-4 further boosts results. In this case, unlike in the captioning experiment, token-level smoothing brings smaller improvements. The combination of both smoothing approaches gives best results, similar to what was observed for image captioning, improving the MLE BLEU-4 from 30.03 to 31.39 on WMT'14 and from 27.55 to 28.74 on IWSLT'14. The outputs of our best model are compared to the MLE in some examples showcased in Appendix C.

5 Conclusion

We investigated the use of loss smoothing approaches to improve over maximum likelihood estimation of RNN language models. We generalized the sequence-level smoothing RAML approach of Norouzi et al. (2016) to the token-level by smoothing the ground-truth target across semantically similar tokens. For the sequence-level, which is computationally expensive, we introduced an efficient "lazy" evaluation scheme, and introduced an improved re-sampling strategy. Experimental evaluation on image captioning and machine translation demonstrates the complementarity of sequence-level and token-level loss smoothing, improving over both the maximum likelihood and RAML.

Acknowledgment. This work has been partially supported by the grant ANR-16-CE23-0006 "Deep in France" and LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

References

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. 2017. Bottom-up and top-down attention for image captioning and visual question answering. *arXiv preprint arXiv:1707.07998*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. 2015. Generating sentences from a continuous space. In *CoNLL*.

A. Canziani, A. Paszke, and E. Culurciello. 2016. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, and J. Langford. 2015. Learning to search better than your teacher. In *ICML*.

C.-C. Chiu, T. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R.-J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani. 2017. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint arXiv:1712.01769*.

K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*.

J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. 2015. Attention-based models for speech recognition. In *NIPS*.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning Workshop*.

H. Daumé III, J. Langford, and D. Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

M. Denkowski and A. Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Workshop on statistical machine translation*.

M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. 2010. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338.

M. Fadaee, A. Bisazza, and C. Monz. 2017. Data augmentation for low-resource neural machine translation. In *ACL*.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

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

S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

A. Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

R. Kiros, R. Salakhutdinov, and R. Zemel. 2014. Multimodal neural language models. In *ICML*.

R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73.

A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. Ke, A. Goyal, Y. Bengio, H. Larochelle, A. Courville, and C. Pal. 2017. Zoneout: Regularizing RNNs by randomly preserving hidden activations. In *ICLR*.

A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

R. Leblond, J.-B. Alayrac, A. Osokin, and S. Lacoste-Julien. 2018. SeaRnn: Training RNNs with global-local losses. In *ICLR*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324.

C.-Y. Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *ACL Workshop Text Summarization Branches Out*.

T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. 2014. Microsoft COCO: common objects in context. In *ECCV*.

J. Lu, C. Xiong, D. Parikh, and R. Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

M. Norouzi, S. Bengio, Z. Chen, N. Jaitly, M. Schuster, Y. Wu, and D. Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *NIPS*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid. 2014. Transformation pursuit for image classification. In *CVPR*.

M. Pedersoli, T. Lucas, C. Schmid, and J. Verbeek. 2017. Areas of attention for image captioning. In *ICCV*.

J. Pennington, R. Socher, and C. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*.

G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *ICLR*.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition*.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

S. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

I. Sutskever, O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

R. Vedantam, C. Zitnick, and D. Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Z. Xie, S. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, and A. Ng. 2017. Data noising as smoothing in neural network language models. In *ICLR*.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Z. Yang, Y. Yuan, Y. Wu, R. Salakhutdinov, and W. Cohen. 2016. Encode, review, and decode: Reviewer module for caption generation. In *NIPS*.

T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei. 2017. Boosting image captioning with attributes. In *ICLR*.

Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. 2016. Image captioning with semantic attention. In *CVPR*.

Numeracy for Language Models: Evaluating and Improving their Ability to Predict Numbers

Georgios P. Spithourakis
Department of Computer Science
University College London
g.spithourakis@cs.ucl.ac.uk

Sebastian Riedel
Department of Computer Science
University College London
s.riedel@cs.ucl.ac.uk

Abstract

Numeracy is the ability to understand and work with numbers. It is a necessary skill for composing and understanding documents in clinical, scientific, and other technical domains. In this paper, we explore different strategies for modelling numerals with language models, such as memorisation and digit-by-digit composition, and propose a novel neural architecture that uses a continuous probability density function to model numerals from an open vocabulary. Our evaluation on clinical and scientific datasets shows that using hierarchical models to distinguish numerals from words improves a perplexity metric on the subset of numerals by 2 and 4 orders of magnitude, respectively, over non-hierarchical models. A combination of strategies can further improve perplexity. Our continuous probability density function model reduces mean absolute percentage errors by 18% and 54% in comparison to the second best strategy for each dataset, respectively.

1 Introduction

Language models (LMs) are statistical models that assign a probability over sequences of words. Language models can often help with other tasks, such as speech recognition (Mikolov et al., 2010; Prabhavalkar et al., 2017), machine translation (Luong et al., 2015; Gülçehre et al., 2017), text summarisation (Filippova et al., 2015; Gambhir and Gupta, 2017), question answering (Wang et al., 2017), semantic error detection (Rei and Yannakoudakis, 2017; Spithourakis et al., 2016a), and fact checking (Rashkin et al., 2017).

Numeracy and literacy refer to the ability to comprehend, use, and attach meaning to numbers and words, respectively. Language models exhibit literacy by being able to assign higher probabilities to sentences that are both grammatical and realistic, as in this example:

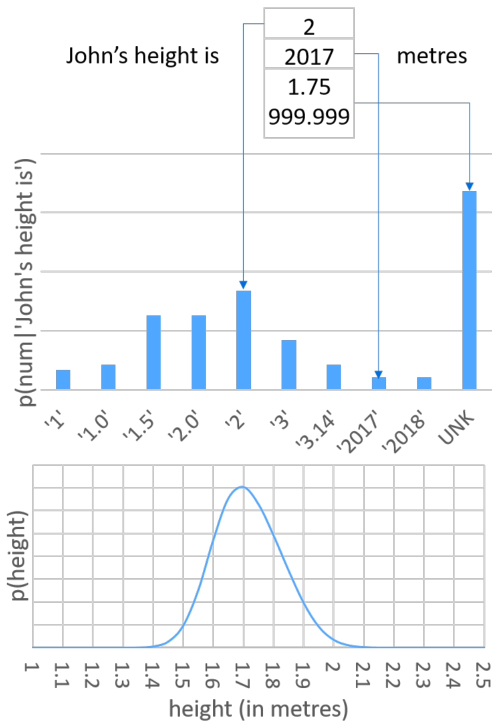


Figure 1: Modelling numerals with a categorical distribution over a fixed vocabulary maps all out-of-vocabulary numerals to the same type, e.g. UNK, and does not reflect the smoothness of the underlying continuous distribution of certain attributes.

'I eat an apple' (grammatical and realistic)
'An apple eats me' (unrealistic)
'I eats an apple' (ungrammatical)

Likewise, a numerate language model should be able to rank numerical claims based on plausibility:

'John's height is 1.75 metres' (realistic)
'John's height is 999.999 metres' (unrealistic)

Existing approaches to language modelling treat numerals similarly to other words, typically using categorical distributions over a fixed vocabulary.

However, this maps all unseen numerals to the same unknown type and ignores the smoothness of continuous attributes, as shown in Figure 1. In that respect, existing work on language modelling does not explicitly evaluate or optimise for numeracy. Numerals are often neglected and low-resourced, e.g. they are often masked (Mitchell and Lapata, 2009), and there are only $15{,}164$ (3.79%) numerals among GloVe's $400{,}000$ embeddings pretrained on 6 billion tokens (Pennington et al., 2014). Yet, numbers appear ubiquitously, from children's magazines (Joram et al., 1995) to clinical reports (Bigeard et al., 2015), and grant objectivity to sciences (Porter, 1996).

Previous work finds that numerals have higher out-of-vocabulary rates than other words and proposes solutions for representing unseen numerals as inputs to language models, e.g. using numerical magnitudes as features (Spithourakis et al., 2016b,a). Such work identifies that the perplexity of language models on the subset of numerals can be very high, but does not directly address the issue. This paper focuses on evaluating and improving the ability of language models to predict numerals. The main contributions of this paper are as follows:

1. We explore different strategies for modelling numerals, such as memorisation and digit-by-digit composition, and propose a novel neural architecture based on continuous probability density functions.

2. We propose the use of evaluations that adjust for the high out-of-vocabulary rate of numerals and account for their numerical value (magnitude).

3. We evaluate on a clinical and a scientific corpus and provide a qualitative analysis of learnt representations and model predictions. We find that modelling numerals separately from other words can drastically improve the perplexity of LMs, that different strategies for modelling numerals are suitable for different textual contexts, and that continuous probability density functions can improve the LM's prediction accuracy for numbers.

2 Language Models

Let $s_1,s_2,...,s_L$ denote a document, where s_t is the token at position t. A language model estimates the probability of the next token given previous tokens, i.e. $p(s_t|s_1,...,s_{t-1})$. Neural LMs estimate this probability by feeding embeddings, i.e. vectors that represent each token, into a Recurrent Neural Network (RNN) (Mikolov et al., 2010).

Token Embeddings Tokens are most commonly represented by a D-dimensional dense vector that is unique for each word from a vocabulary $\mathcal{V}$ of known words. This vocabulary includes special symbols (e.g. 'UNK') to handle out-of-vocabulary tokens, such as unseen words or numerals. Let w_s be the one-hot representation of token s, i.e. a sparse binary vector with a single element set to 1 for that token's index in the vocabulary, and $E \in \mathbb{R}^{D\times|\mathcal{V}|}$ be the token embeddings matrix. The token embedding for s is the vector $e_s^{\text{token}} = Ew_s$.

Character-Based Embeddings A representation for a token can be build from its constituent characters (Luong and Manning, 2016; Santos and Zadrozny, 2014). Such a representation takes into account the internal structure of tokens. Let $d_1,d_2,...,d_N$ be the characters of token s. A character-based embedding for s is the final hidden state of a D-dimensional character-level RNN: $e_s^{\text{chars}} = \text{RNN}(d_0,d_1,...d_L)$.

Recurrent and Output Layer The computation of the conditional probability of the next token involves recursively feeding the embedding of the current token e_{s_t} and the previous hidden state h_{t-1} into a D-dimensional token-level RNN to obtain the current hidden state h_t. The output probability is estimated using the softmax function, i.e.

$$\begin{aligned} p(s_t|h_t) &= \text{softmax}(\psi(s_t)) = \tfrac{1}{Z}e^{\psi(s_t)} \\ Z &= \sum_{s'\in\mathcal{V}} e^{\psi(s')}, \end{aligned} \tag{1}$$

where $\psi(.)$ is a score function.

Training and Evaluation Neural LMs are typically trained to minimise the cross entropy on the training corpus:

$$\mathcal{H}_{train} = -\frac{1}{N}\sum_{s_t \in train} \log p(s_t|s_{<t}) \tag{2}$$

A common performance metric for LMs is per token perplexity (Eq. 3), evaluated on a test corpus. It can also be interpreted as the branching factor: the size of an equally weighted distribution with equivalent uncertainty, i.e. how many sides you need on a fair die to get the same uncertainty as the model distribution.

$$PP_{test} = \exp(\mathcal{H}_{test}) \tag{3}$$

3 Strategies for Modelling Numerals

In this section we describe models with different strategies for generating numerals and propose the use of number-specific evaluation metrics that adjust for the high out-of-vocabulary rate of numerals and account for numerical values. We draw inspiration from theories of numerical cognition. The triple code theory (Dehaene et al., 2003) postulates that humans process quantities through two exact systems (verbal and visual) and one approximate number system that semantically represents a number on a mental number line. Tzelgov et al. (2015) identify two classes of numbers: i) primitives, which are holistically retrieved from long-term memory; and ii) non-primitives, which are generated online. An in-depth review of numerical and mathematical cognition can be found in Kadosh and Dowker (2015) and Campbell (2005).

3.1 Softmax Model and Variants

This class of models assumes that numerals come from a finite vocabulary that can be memorised and retrieved later. The *softmax* model treats all tokens (words and numerals) alike and directly uses Equation 1 with score function:

$$\psi(s_t) = h_t^T e_{s_t}^{\text{token}} = h_t^T E_{\text{out}} w_{s_t}, \tag{4}$$

where $E_{out} \in \mathbb{R}^{D \times |\mathcal{V}|}$ is an output embeddings matrix. The summation in Equation 1 is over the complete target vocabulary, which requires mapping any out-of-vocabulary tokens to special symbols, e.g. 'UNK$_{\text{word}}$' and 'UNK$_{\text{numeral}}$'.

Softmax with Digit-Based Embeddings The *softmax+rnn* variant considers the internal syntax of a numeral's digits by adjusting the score function:

$$\begin{aligned}\psi(s_t) &= h_t^T e_{s_t}^{\text{token}} + h_t^T e_{s_t}^{\text{chars}} \\ &= h_t^T E_{\text{out}} w_{s_t} + h_t^T E_{\text{out}}^{\text{RNN}} w_{s_t},\end{aligned} \tag{5}$$

where the columns of $E_{\text{out}}^{\text{RNN}}$ are composed of character-based embeddings for in-vocabulary numerals and token embeddings for the remaining vocabulary. The character set comprises digits (0-9), the decimal point, and an end-of-sequence character. The model still requires normalisation over the whole vocabulary, and the special unknown tokens are still needed.

Hierarchical Softmax A hierarchical softmax (Morin and Bengio, 2005a) can help us decouple the modelling of numerals from that of words. The probability of the next token s_t is decomposed to that of its class c_t and the probability of the exact token from within the class:

$$\begin{aligned}p(s_t|h_t) &= \sum_{c_t \in C} p(c_t|h_t) p(s_t|c_t, h_t) \\ p(c_t|h_t) &= \sigma\left(h_t^T b\right)\end{aligned} \tag{6}$$

where the valid token classes are $C = \{\text{word, numeral}\}$, σ is the sigmoid function and b is a D-dimensional vector. Each of the two branches of $p(s_t|c_t, h_t)$ can now be modelled by independently normalised distributions. The hierarchical variants (*h-softmax* and *h-softmax+rnn*) use two independent softmax distributions for words and numerals. The two branches share no parameters, and thus words and numerals will be embedded into separate spaces.

The hierarchical approach allows us to use any well normalised distribution to model each of its branches. In the next subsections, we examine different strategies for modelling the branch of numerals, i.e. $p(s_t|c_t = \text{numeral}, h_t)$. For simplicity, we will abbreviate this to $p(s)$.

3.2 Digit-RNN Model

Let $d_1, d_2 ... d_N$ be the digits of numeral s. A digit-by-digit composition strategy estimates the probability of the numeral from the probabilities of its digits:

$$p(s) = p(d_1) p(d_2|d_1) ... p(d_N|d_{<N}) \tag{7}$$

The *d-RNN* model feeds the hidden state h_t of the token-level RNN into a character-level RNN (Graves, 2013; Sutskever et al., 2011) to estimate this probability. This strategy can accommodate an open vocabulary, i.e. it eliminates the need for an UNK$_{\text{numeral}}$ symbol, as the probability is normalised one digit at a time over the much smaller vocabulary of digits (digits 0-9, decimal separator, and end-of-sequence).

3.3 Mixture of Gaussians Model

Inspired by the approximate number system and the mental number line (Dehaene et al., 2003), our proposed *MoG* model computes the probability of numerals from a probability density function (pdf) over real numbers, using a mixture of Gaussians for the underlying pdf:

$$\begin{aligned}q(v) &= \sum_{k=1}^{K} \pi_k \mathcal{N}_k(v; \mu_k, \sigma_k^2) \\ \pi_k &= \text{softmax}\left(B^T h_t\right),\end{aligned} \tag{8}$$

where K is the number of components, π_k are mixture weights that depend on hidden state h_t of the token-level RNN, $\mathcal{N}_k$ is the pdf of the normal distribution with mean $\mu_k \in \mathbb{R}$ and variance $\sigma_k^2 \in \mathbb{R}$, and $B \in \mathbb{R}^{D \times K}$ is a matrix.

The difficulty with this approach is that for any continuous random variable, the probability that it equals a specific value is always zero. To resolve this,

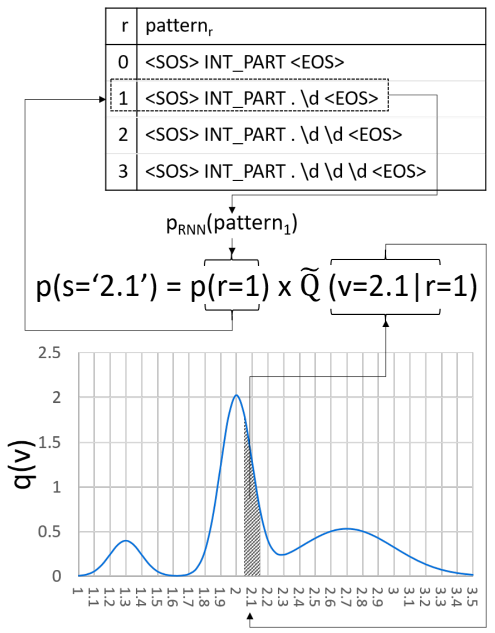


Figure 2: Mixture of Gaussians model. The probability of a numeral is decomposed into the probability of its decimal precision and the probability that an underlying number will produce the numeral when rounded at the given precision.

we consider a probability mass function (pmf) that discretely approximates the pdf:

$$\widetilde{Q}(v|r) = \int_{v-\epsilon_r}^{v+\epsilon_r} q(u)du = F(v+\epsilon_r) - F(v-\epsilon_r), \quad (9)$$

where $F(.)$ is the cumulative density function of $q(.)$, and $\epsilon_r = 0.5 \times 10^{-r}$ is the number's precision. The level of discretisation r, i.e. how many decimal digits to keep, is a random variable in $\mathbb{N}$ with distribution $p(r)$. The mixed joint density is:

$$p(s) = p(v,r) = p(r)\widetilde{Q}(v|r) \quad (10)$$

Figure 2 summarises this strategy, where we model the level of discretisation by converting the numeral into a pattern and use a RNN to estimate the probability of that pattern sequence:

$$p(r) = p(\text{SOS INT\_PART . } \overbrace{\text{\\d ... \\d}}^{r \text{ decimal digits}} \text{ EOS}) \quad (11)$$

3.4 Combination of Strategies

Different mechanisms might be better for predicting numerals in different contexts. We propose a *combination* model that can select among different strategies for modelling numerals:

$$p(s) = \sum_{\forall m \in M} \alpha_m p(s|m)$$
$$\alpha_m = \text{softmax}(A^T h_t), \quad (12)$$

where M={h-softmax, d-RNN, MoG}, and $A \in \mathbb{R}^{D \times |M|}$. Since both d-RNN and MoG are open-vocabulary models, the unknown numeral token can now be removed from the vocabulary of h-softmax.

3.5 Evaluating the Numeracy of LMs

Numeracy skills are centred around the understanding of numbers and numerals. A number is a mathematical object with a specific magnitude, whereas a numeral is its symbolic representation, usually in the positional decimal Hindu–Arabic numeral system (McCloskey and Macaruso, 1995). In humans, the link between numerals and their numerical values boosts numerical skills (Griffin et al., 1995).

Perplexity Evaluation Test perplexity evaluated only on numerals will be informative of the symbolic component of numeracy. However, model comparisons based on naive evaluation using Equation 3 might be problematic: perplexity is sensitive to out-of-vocabulary (OOV) rate, which might differ among models, e.g. it is zero for open-vocabulary models. As an extreme example, in a document where all words are out of vocabulary, the best perplexity is achieved by a trivial model that predicts everything as unknown.

Ueberla (1994) proposed Adjusted Perplexity (APP; Eq. 14), also known as unknown-penalised perplexity (Ahn et al., 2016), to cancel the effect of the out-of-vocabulary rate on perplexity. The APP is the perplexity of an adjusted model that uniformly redistributes the probability of each out-of-vocabulary class over all different types in that class:

$$p'(s) = \begin{cases} p(s)\frac{1}{|OOV_c|} & \text{if } s \in OOV_c \\ p(s) & \text{otherwise} \end{cases} \quad (13)$$

where OOV_c is an out-of-vocabulary class (e.g. words and numerals), and $|OOV_c|$ is the cardinality of each OOV set. Equivalently, adjusted perplexity can be calculated as:

$$APP_{test} = \exp\left(\mathcal{H}_{test} + \sum_c \mathcal{H}^c_{\text{adjust}}\right)$$
$$\mathcal{H}^c_{\text{adjust}} = -\sum_t \frac{|s_t \in OOV_c|}{N} \log \frac{1}{|OOV_c|} \quad (14)$$

where N is the total number of tokens in the test set and $|s \in OOV_c|$ is the count of tokens from the test set belonging in each OOV set.

Evaluation on the Number Line While perplexity looks at symbolic performance on numerals, this evaluation focuses on numbers and particularly on their numerical value, which is their most prominent semantic content (Dehaene et al., 2003; Dehaene and Cohen, 1995).

Let v_t be the numerical value of token s_t from the test corpus. Also, let $\hat{v}_t$ be the value of the most probable numeral under the model $s_t = \text{argmax}\,(p(s_t|h_t, c_t{=}\text{num}))$. Any evaluation metric from the regression literature can be used to measure the models performance. To evaluate on the number line, we can use any evaluation metric from the regression literature. In reverse order of tolerance to extreme errors, some of the most popular are Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Median Absolute Error (MdAE):

$$\begin{aligned} e_i &= v_i - \hat{v}_i \\ RMSE &= \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \\ MAE &= \frac{1}{N}\sum_{i=1}^{N} |e_i| \\ MdAE &= \text{median}\{|e_i|\} \end{aligned} \tag{15}$$

The above are sensitive to the scale of the data. If the data contains values from different scales, percentage metrics are often preferred, such as the Mean/Median Absolute Percentage Error (MAPE/MdAPE):

$$\begin{aligned} pe_i &= \frac{v_i - \hat{v}_i}{v_i} \\ MAPE &= \frac{1}{N}\sum_{i=1}^{N} |pe_i| \\ MdAPE &= \text{median}\{|pe_i|\} \end{aligned} \tag{16}$$

4 Data

To evaluate our models, we created two datasets with documents from the clinical and scientific domains, where numbers abound (Bigeard et al., 2015; Porter, 1996). Furthermore, to ensure that the numbers will be informative of some attribute, we only selected texts that reference tables.

Clinical Data Our *clinical* dataset comprises clinical records from the London Chest Hospital. The records where accompanied by tables with 20 numeric attributes (age, heart volumes, etc.) that they partially describe, as well as include numbers not found in the tables. Numeric tokens constitute only a small proportion of each sentence (4.3%), but account for a large part of the unique tokens vocabulary (>40%) and suffer high OOV rates.

Scientific Data Our *scientific* dataset comprises paragraphs from Cornell's ARXIV [1] repository of scientific articles, with more than half a million converted papers in 37 scientific sub-fields. We used the preprocessed ARXMLIV (Stamerjohanns et al., 2010; Stamerjohanns and Kohlhase, 2008) [2] version, where papers have been converted from LATEX into a custom XML format using the LATEXML [3] tool. We then kept all paragraphs with at least one reference to a table and a number.

	Clinical			Scientific		
	Train	Dev	Test	Train	Dev	Test
#inst	11170	1625	3220	14694	2037	4231
maxLen	667	594	666	2419	1925	1782
avgLen	210.1	209.1	206.9	210.1	215.9	212.1
%word	95.7	95.7	95.7	96.1	96.1	96.0
%nums	4.3	4.3	4.3	3.9	3.9	4.0
min	0.0	0.0	0.0	0.0	0.0	0.0
median	59.5	59.0	60.0	5.0	4.0	4.5
mean	300.6	147.7	464.8	$\sim 10^{21}$	$\sim 10^{7}$	$\sim 10^{7}$
max	$\sim 10^{7}$	$\sim 10^{5}$	$\sim 10^{7}$	$\sim 10^{26}$	$\sim 10^{11}$	$\sim 10^{11}$

Table 1: Statistical description of the clinical and scientific datasets: Number of instances (i.e. paragraphs), maximum and average lengths, proportions of words and numerals, descriptive statistics of numbers.

For both datasets, we lowercase tokens and normalise numerals by omitting the thousands separator ("2,000" becomes "2000") and leading zeros ("007" becomes "7"). Special mathematical symbols are tokenised separately, e.g. negation ("-1" as "-", "1"), fractions ("3/4" as "3", "/", "4"), etc. For this reason, all numbers were non-negative. Table 1 shows descriptive statistics for both datasets.

5 Experimental Results and Discussion

We set the vocabularies to the 1,000 and 5,000 most frequent token types for the clinical and scientific datasets, respectively. We use gated token-character embeddings (Miyamoto and Cho, 2016) for the input of numerals and token embeddings for the input and output of words, since the scope of our paper is numeracy. We set the models' hidden dimensions to $D = 50$ and initialise all token embeddings to pretrained GloVe (Pennington et al., 2014). All our

[1]ARXIV.ORG. Cornell University Library at http://arxiv.org/, visited December 2016

[2]ARXMLIV. Project home page at http://arxmliv.kwarc.info/, visited December 2016

[3]LATEXML. http://dlmf.nist.gov, visited December 2016

	Clinical						Scientific					
	words		numerals		total		words		numerals		total	
Model	PP	APP	PP	APP	PP	APP	PP	APP	PP	APP	PP	APP
softmax	4.08	5.99	12.04	58443.72	4.28	8.91	33.96	51.83	127.12	3505856.25	35.79	80.62
softmax+rnn	4.03	5.91	**11.57**	56164.81	4.21	8.77	**33.54**	51.20	**119.68**	3300688.50	**35.28**	79.47
h-softmax	**4.00**	4.96	11.78	495.95	**4.19**	6.05	34.73	49.81	122.67	550.98	36.51	54.80
h-softmax+rnn	4.03	4.99	11.65	490.14	4.22	6.09	34.04	48.83	120.83	542.70	35.80	53.73
d-RNN	3.99	**4.95**	263.22	263.22	4.79	5.88	34.08	48.89	519.80	**519.80**	37.98	53.70
MoG	4.03	4.99	226.46	226.46	4.79	5.88	34.14	48.97	683.16	683.16	38.45	54.37
combination	4.01	4.96	197.59	**197.59**	4.74	**5.82**	33.64	**48.25**	520.95	520.95	37.50	**53.03**

Table 2: Test set perplexities for the clinical and scientific data. Adjusted perplexities (APP) are directly comparable across all data and models, but perplexities (PP) are sensitive to the varying out-of-vocabulary rates.

	Clinical					Scientific		
Model	RMSE	MAE	MdAE	MAPE%	MdAPE%	MdAE	MAPE%	MdAPE%
mean	1043.68	294.95	245.59	2353.11	409.47	$\sim 10^{20}$	$\sim 10^{23}$	$\sim 10^{22}$
median	1036.18	120.24	34.52	425.81	52.05	4.20	8039.15	98.65
softmax	997.84	80.29	12.70	621.78	22.41	3.00	1947.44	80.62
softmax+rnn	991.38	74.44	13.00	503.57	23.91	3.50	15208.37	80.00
h-softmax	1095.01	167.19	14.00	746.50	25.00	3.00	1652.21	80.00
h-softmax+rnn	1001.04	83.19	12.30	491.85	23.44	3.00	2703.49	80.00
d-RNN	1009.34	70.21	9.00	513.81	17.90	3.00	1287.27	**52.45**
MoG	998.78	**57.11**	**6.92**	**348.10**	**13.64**	**2.10**	**590.42**	90.00
combination	**989.84**	69.47	9.00	552.06	17.86	3.00	2332.50	88.89

Table 3: Test set regression evaluation for the clinical and scientific data. Mean absolute percentage error (MAPE) is scale independent and allows for comparison across data, whereas root mean square and mean absolute errors (RMSE, MAE) are scale dependent. Medians (MdAE, MdAPE) are informative of the distribution of errors.

RNNs are LSTMs (Hochreiter and Schmidhuber, 1997) with the biases of LSTM forget gate were initialised to 1.0 (Józefowicz et al., 2015). We train using mini-batch gradient decent with the Adam optimiser (Kingma and Ba, 2014) and regularise with early stopping and 0.1 dropout rate (Srivastava, 2013) in the input and output of the token-based RNN.

For the mixture of Gaussians, we select the mean and variances to summarise the data at different granularities by fitting 7 separate mixture of Gaussian models on all numbers, each with twice as many components as the previous, for a total of $2^{7+1} - 1 = 256$ components. These models are initialised at percentile points from the data and trained with the expectation-minimisation algorithm. The means and variances are then fixed and not updated when we train the language model.

5.1 Quantitative Results

Perplexities Table 2 shows perplexities evaluated on the subsets of words, numerals and all tokens of the test data. Overall, all models performed better on the clinical than on the scientific data. On words, all models achieve similar perplexities in each dataset.

On numerals, softmax variants perform much better than other models in PP, which is an artefact of the high OOV-rate of numerals. APP is significantly worse, especially for non-hierarchical variants, which perform about 2 and 4 orders of magnitude worse than hierarchical ones.

For open-vocabulary models, i.e. d-RNN, MoG, and combination, PP is equivalent to APP. On numerals, d-RNN performed better than softmax variants in both datasets. The MoG model performed twice as well as softmax variants on the clinical dataset, but had the third worse performance in the scientific dataset. The combination model had the best overall APP results for both datasets.

Evaluations on the Number Line To factor out model specific decoding processes for finding the best next numeral, we use our models to rank a set of candidate numerals: we compose the union of in-vocabulary numbers and 100 percentile points from the training set, and we convert numbers into numerals by considering all formats up to n decimal points. We select n to represent 90% of numerals seen at training, which yields $n=3$ and $n=4$ for the clinical and scientific data, respectively.

Table 3 shows evaluation results, where we also include two naive baselines of constant predictions: with the mean and median of the training data. For both datasets, RMSE and MAE were too sensitive to extreme errors to allow drawing safe conclusions, particularly for the scientific dataset, where both metrics were in the order of 10^9. MdAE can be of some use, as 50% of the errors are absolutely smaller than that.

Along percentage metrics, MoG achieved the best MAPE in both datasets (18% and 54% better that the second best) and was the only model to perform better than the median baseline for the clinical data. However, it had the worst MdAPE, which means that MoG mainly reduced larger percentage errors. The d-RNN model came third and second in the clinical and scientific datasets, respectively. In the latter it achieved the best MdAPE, i.e. it was effective at reducing errors for 50% of the numbers. The combination model did not perform better than its constituents. This is possibly because MoG is the only strategy that takes into account the numerical magnitudes of the numerals.

5.2 Learnt Representations

Softmax versus Hierarchical Softmax Figure 3 visualises the cosine similarities of the output token embeddings of numerals for the softmax and h-softmax models. Simple softmax enforced high similarities among all numerals and the unknown numeral token, so as to make them more dissimilar to words, since the model embeds both in the same space. This is not the case for h-softmax that uses two different spaces: similarities are concentrated along the diagonal and fan out as the magnitude grows, with the exception of numbers with special meaning, e.g. years and percentile points.

Digit embeddings Figure 4 shows the cosine similarities between the digits of the d-RNN output mode. We observe that each primitive digit is mostly similar to its previous and next digit. Similar behaviour was found for all digit embeddings of all models.

5.3 Predictions from the Models

Next Numeral Figure 5 shows the probabilities of different numerals under each model for two examples from the clinical development set. Numerals are grouped by number of decimal points. The h-softmax model's probabilities are spiked, d-RNNs are saw-tooth like and MoG's are smooth, with the occasional spike, whenever a narrow component allows for it. Probabilities rapidly decrease for more decimal digits, which is reminiscent of the theoretical expectation that the probability of en exact value for a continuous variable is zero.

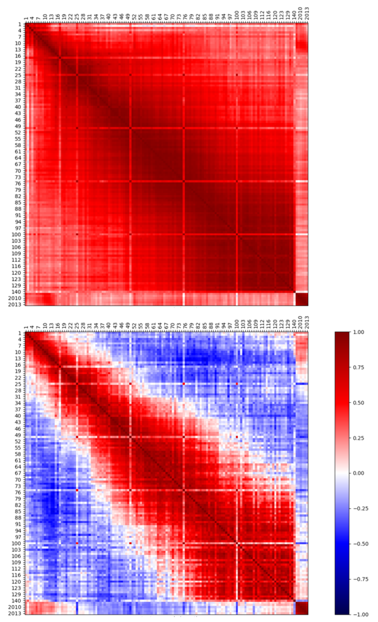


Figure 3: Numeral embeddings for the softmax (top) and h-softmax (bottom) models on the clinical data. Numerals are sorted by value.

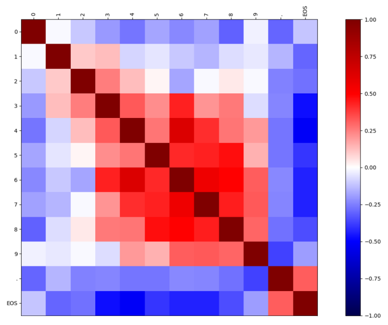


Figure 4: Cosine similarities for d-RNN's output digit embeddings trained on the scientific data.

Selection of Strategy in Combination Model Table 4 shows development set examples with high selection probabilities for each strategy of the combination model, along with numerals with the highest average selection per mode. The h-softmax model is responsible for mostly integers with special functions,

	Clinical	Scientific
h-softmax	**Examples**: *"late enhancement (> **75** %)", "late gadolinium enhancement (< **25** %)", "infarction (**2** out of **17** segments)", "infarct with **4** out of **17** segments nonviable", "adenosine stress perfusion @ **140** mcg", "stress perfusion (adenosine **140** mcg"* **Numerals**: 50, 17, 100, 75, 25, 1, 140, 2012, 2010, 2011, 8, 5, 2009, 2013, 7, 6, 2, 3, 2008, 4...	**Examples**: *"sharp et al . **2004**", "li et al . **2003**", "3.5 × **10**ˆ4", "0.3 × **10**ˆ16"* **Numerals**: 1992, 2001, 1995, 2003, 2009, 1993, 2010, 1994, 1998, 2002, 2006, 1997, 2005, 1990, 10, 2008, 2007, 2004, 1983, 1991...
d-RNN	**Examples**: *"aortic root is dilated (measured **37** x **37** mm", "ascending aorta is not dilated (**32** x **31** mm"* **Numerals**: 42, 33, 31, 43, 44, 21, 38, 36, 46, 37, 32, 39, 26, 28, 23, 29, 45, 40, 49, 94...	**Examples**: *"ngc **6334** stars", "ngc **2366** shows a wealth of small structures"* **Numerals**: 294, 4000, 238, 6334, 2363, 1275, 2366, 602, 375, 1068, 211, 6.4, 8.7, 600, 96, 0.65, 700, 1.17, 4861, 270...
MoG	**Examples**: *"stroke volume **46.1** ml", "stroke volume **65.6** ml", "stroke volume **74.5** ml", "end diastolic volume **82.6** ml", "end diastolic volume **99.09** ml", "end diastolic volume **138.47** ml"* **Numerals**: 74.5, 69.3, 95.9, 96.5, 72.5, 68.6, 82.1, 63.7, 78.6, 69.6, 69.5, 82.2, 68.3, 73.2, 63.2, 82.6, 77.7, 80.7, 70.7, 70.4...	**Examples**: *"hip **12961** and gl **676** a are orbited by giant planets," "velocities of gl **676**", "velocities of hip **12961**"* **Numerals**: 12961, 766, 7409, 4663, 44.3, 1819, 676, 1070, 5063, 323, 264, 163296, 2030, 77, 1.15, 196, 0.17, 148937, 0.43, 209458...

Table 4: Examples of numerals with highest probability in each strategy of the combination model.

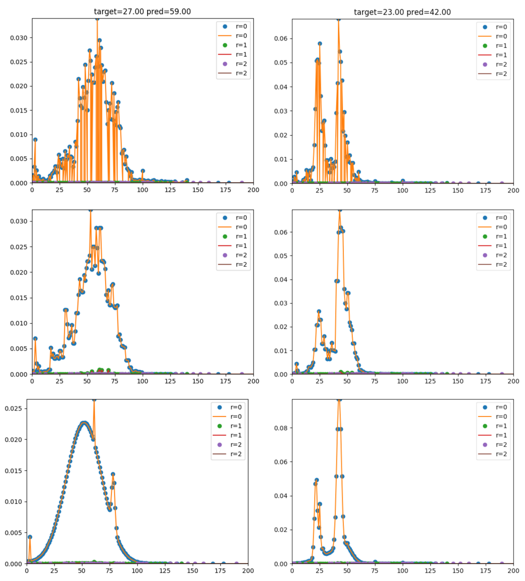


Figure 5: Example model predictions for the h-softmax (top), d-RNN (middle) and MoG (bottom) models. Examples from the clinical development set.

e.g. years, typical drug dosages, percentile points, etc. In the clinical data, d-RNN picks up two-digit integers (mostly dimensions) and MoG is activated for continuous attributes, which are mostly out of vocabulary. In the scientific data, d-RNN and MoG

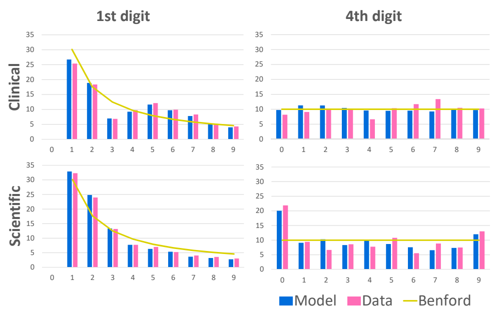


Figure 6: Distributions of significant digits from d-RNN model, data, and theoretical expectation (Benford's law).

showed affinity to different indices from catalogues of astronomical objects: d-RNN mainly to NGC (Dreyer, 1888) and MoG to various other indices, such as GL (Gliese, 1988) and HIP (Perryman et al., 1997). In this case, MoG was wrongly selected for numerals with a labelling function, which also highlights a limitation of evaluating on the number line, when a numeral is not used to represent its magnitude.

Significant Digits Figure 5 shows the distributions of the most significant digits under the d-RNN model and from data counts. The theoretical estimate has been overlayed, according to Benford's law (Benford, 1938), also called the first-digit law, which applies to many real-life numerals. The law predicts that the first digit is 1 with higher probability (about 30%) than 9 ($< 5\%$) and weakens towards uniformity at higher digits. Model probabilities closely follow estimates from the data. Violations from Benford's law can be due to rounding (Beer, 2009) and can be used as evidence for fraud detection (Lu et al., 2006).

6 Related Work

Numerical quantities have been recognised as important for textual entailment (Lev et al., 2004; Dagan et al., 2013). Roy et al. (2015) proposed a quantity entailment sub-task that focused on whether a given quantity can be inferred from a given text and, if so, what its value should be. A common framework for acquiring common sense about numerical attributes of objects has been to collect a corpus of numerical values in pre-specified templates and then model attributes as a normal distribution (Aramaki et al., 2007; Davidov and Rappoport, 2010; Iftene and Moruz, 2010; Narisawa et al., 2013; de Marneffe et al., 2010). Our model embeds these approaches into a LM that has a sense for numbers.

Other tasks that deal with numerals are numerical information extraction and solving mathematical problems. Numerical relations have at least one argument that is a number and the aim of the task is to extract all such relations from a corpus, which can range from identifying a few numerical attributes (Nguyen and Moschitti, 2011; Intxaurrondo et al., 2015) to generic numerical relation extraction (Hoffmann et al., 2010; Madaan et al., 2016). Our model does not extract values, but rather produces an probabilistic estimate.

Much work has been done in solving arithmetic (Mitra and Baral, 2016; Hosseini et al., 2014; Roy and Roth, 2016), geometric (Seo et al., 2015), and algebraic problems (Zhou et al., 2015; Koncel-Kedziorski et al., 2015; Upadhyay et al., 2016; Upadhyay and Chang, 2016; Shi et al., 2015; Kushman et al., 2014) expressed in natural language. Such models often use mathematical background knowledge, such as linear system solvers. The output of our model is not based on such algorithmic operations, but could be extended to do so in future work.

In language modelling, generating rare or unknown words has been a challenge, similar to our unknown numeral problem. Gulcehre et al. (2016) and Gu et al. (2016) adopted pointer networks (Vinyals et al., 2015) to copy unknown words from the source in translation and summarisation tasks. Merity et al. (2016) and Lebret et al. (2016) have models that copy from context sentences and from Wikipedia's infoboxes, respectively. Ahn et al. (2016) proposed a LM that retrieves unknown words from facts in a knowledge graph. They draw attention to the inappropriateness of perplexity when OOV-rates are high and instead propose an adjusted perplexity metric that is equivalent to APP. Other methods aim at speeding up LMs to allow for larger vocabularies (Chen et al., 2015), such as hierarchical softmax (Morin and Bengio, 2005b), target sampling (Jean et al., 2014), etc., but still suffer from the unknown word problem. Finally, the problem is resolved when predicting one character at a time, as done by the character-level RNN (Graves, 2013; Sutskever et al., 2011) used in our d-RNN model.

7 Conclusion

In this paper, we investigated several strategies for LMs to model numerals and proposed a novel open-vocabulary generative model based on a continuous probability density function. We provided the first thorough evaluation of LMs on numerals on two corpora, taking into account their high out-of-vocabulary rate and numerical value (magnitude). We found that modelling numerals separately from other words through a hierarchical softmax can substantially improve the perplexity of LMs, that different strategies are suitable for different contexts, and that a combination of these strategies can help improve the perplexity further. Finally, we found that using a continuous probability density function can improve prediction accuracy of LMs for numbers by substantially reducing the mean absolute percentage metric.

Our approaches in modelling and evaluation can be used in future work in tasks such as approximate information extraction, knowledge base completion, numerical fact checking, numerical question answering, and fraud detection. Our code and data are available at: `https://github.com/uclmr/numerate-language-models`.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments and also Steffen Petersen for providing the clinical dataset and advising us on the clinical aspects of this work. This research was supported by the Farr Institute of Health Informatics Research and an Allen Distinguished Investigator award.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318* .

Eiji Aramaki, Takeshi Imai, Kengo Miyo, and Kazuhiko Ohe. 2007. Uth: Svm-based semantic relation classification using physical sizes. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, pages 464–467.

TW Beer. 2009. Terminal digit preference: beware of benford's law. *Journal of clinical pathology* 62(2):192–192.

Frank Benford. 1938. The law of anomalous numbers. *Proceedings of the American philosophical society* pages 551–572.

Elise Bigeard, Vianney Jouhet, Fleur Mougin, Frantz Thiessard, and Natalia Grabar. 2015. Automatic extraction of numerical values from unstructured data in ehrs. In *MIE*. pages 50–54.

Jamie ID Campbell. 2005. *Handbook of mathematical cognition*. Psychology Press.

Welin Chen, David Grangier, and Michael Auli. 2015. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906* .

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies* 6(4):1–220.

Dmitry Davidov and Ari Rappoport. 2010. Extraction and approximation of numerical attributes from the web. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1308–1317.

Marie-Catherine de Marneffe, Christopher D Manning, and Christopher Potts. 2010. Was it good? it was provocative. learning the meaning of scalar adjectives. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 167–176.

Stanislas Dehaene and Laurent Cohen. 1995. Towards an anatomical and functional model of number processing. *Mathematical cognition* 1(1):83–120.

Stanislas Dehaene, Manuela Piazza, Philippe Pinel, and Laurent Cohen. 2003. Three parietal circuits for number processing. *Cognitive neuropsychology* 20(3-6):487–506.

John Louis Emil Dreyer. 1888. A new general catalogue of nebulæ and clusters of stars, being the catalogue of the late sir john fw herschel, bart, revised, corrected, and enlarged. *Memoirs of the Royal Astronomical Society* 49:1.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. pages 360–368.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artif. Intell. Rev.* 47(1):1–66.

Wilhelm Gliese. 1988. The third catalogue of nearby stars. *Stand. Star Newsl. 13, 13* 13.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Sharon Griffin, Robbie Case, and Allesandra Capodilupo. 1995. Teaching for understanding: The importance of the central conceptual structures in the elementary mathematics curriculum. .

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech & Language* 45:137–148.

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

Raphael Hoffmann, Congle Zhang, and Daniel S Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 286–295.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 523–533.

Adrian Iftene and Mihai-Alex Moruz. 2010. Uaic participation at rte-6 .

Ander Intxaurrondo, Eneko Agirre, Oier Lopez De Lacalle, and Mihai Surdeanu. 2015. Diamonds in the rough: Event extraction from imperfect microblog data. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 641–650.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* .

Elana Joram, Lauren B Resnick, and Anthony J Gabriele. 1995. Numeracy as cultural practice: An examination of numbers in magazines for children, teenagers, and adults. *Journal for Research in Mathematics Education* pages 346–361.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*. pages 2342–2350.

Roi Cohen Kadosh and Ann Dowker. 2015. *The Oxford handbook of numerical cognition*. Oxford Library of Psychology.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics* 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 271–281.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771* .

Iddo Lev, Bill MacCartney, Christopher D Manning, and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Proceedings of the 2nd Workshop on Text Meaning and Interpretation*. Association for Computational Linguistics, pages 9–16.

Fletcher Lu, J. Efrim Boritz, and H. Dominic Covvey. 2006. Adaptive fraud detection using benford's law. In *Advances in Artificial Intelligence, 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006*. pages 347–358.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1054–1063.

Thang Luong, Michael Kayser, and Christopher D. Manning. 2015. Deep neural language models for machine translation. In *Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015*. pages 305–309.

Aman Madaan, Ashish Mittal, Ganesh Ramakrishnan, Sunita Sarawagi, et al. 2016. Numerical relation extraction with minimal supervision. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Michael McCloskey and Paul Macaruso. 1995. Representing and using numerical information. *American Psychologist* 50(5):351.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* .

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*. pages 1045–1048.

Jeff Mitchell and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, pages 430–439.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *ACL*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700* .

Frederic Morin and Yoshua Bengio. 2005a. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005*.

Frederic Morin and Yoshua Bengio. 2005b. Hierarchical probabilistic neural network language model. In *Aistats*. Citeseer, volume 5, pages 246–252.

Katsuma Narisawa, Yotaro Watanabe, Junta Mizuno, Naoaki Okazaki, and Kentaro Inui. 2013. Is a 204 cm man tall or small? acquisition of numerical common sense from the web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 382–391.

Truc-Vien T Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 277–282.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Michael AC Perryman, L Lindegren, J Kovalevsky, E Hoeg, U Bastian, PL Bernacca, M Crézé, F Donati, M Grenon, M Grewing, et al. 1997. The hipparcos catalogue. *Astronomy and Astrophysics* 323:L49–L52.

Theodore M Porter. 1996. *Trust in numbers: The pursuit of objectivity in science and public life*. Princeton University Press.

Rohit Prabhavalkar, Kanishka Rao, Tara N. Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly. 2017. A comparison of sequence-to-sequence models for speech recognition. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*. pages 939–943.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*. pages 2931–2937.

Marek Rei and Helen Yannakoudakis. 2017. Auxiliary objectives for neural error detection models. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, BEA@EMNLP 2017*. pages 33–43.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413* .

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics* 3:1–13.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1466–1476.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*.

Georgios P. Spithourakis, Isabelle Augenstein, and Sebastian Riedel. 2016a. Numerically grounded language models for semantic error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*. pages 987–992.

Georgios P Spithourakis, Steffen E Petersen, and Sebastian Riedel. 2016b. Clinical text prediction with numerically grounded conditional language models. *EMNLP 2016* page 6.

Nitish Srivastava. 2013. Improving neural networks with dropout. *University of Toronto* 182.

Heinrich Stamerjohanns and Michael Kohlhase. 2008. Transforming the arχiv to xml. In *International Conference on Intelligent Computer Mathematics*. Springer, pages 574–582.

Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce Miller. 2010. Transforming large collections of scientific publications to xml. *Mathematics in Computer Science* 3(3):299–307.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.

Joseph Tzelgov, Dana Ganor-Stern, Arava Y Kallai, and Michal Pinhas. 2015. Primitives and non-primitives of numerical representations. *Oxford library of psychology. The Oxford handbook of numerical cognition* pages 45–66.

Joerg Ueberla. 1994. Analysing a simple language model· some general conclusions for language models for speech recognition. *Computer Speech & Language* 8(2):153–176.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *arXiv preprint arXiv:1609.07197* .

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 297–306.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *CoRR* abs/1706.01450.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics (Lisbon, Portugal*. pages 817–822.

To Attend or not to Attend: A Case Study on Syntactic Structures for Semantic Relatedness

Amulya Gupta
Iowa State University
guptaam@iastate.edu

Zhu Zhang
Iowa State University
zhuzhang@iastate.edu

Abstract

With the recent success of Recurrent Neural Networks (RNNs) in Machine Translation (MT), attention mechanisms have become increasingly popular. The purpose of this paper is two-fold; firstly, we propose a novel attention model on Tree Long Short-Term Memory Networks (Tree-LSTMs), a tree-structured generalization of standard LSTM. Secondly, we study the interaction between attention and syntactic structures, by experimenting with three LSTM variants: bidirectional-LSTMs, Constituency Tree-LSTMs, and Dependency Tree-LSTMs. Our models are evaluated on two semantic relatedness tasks: semantic relatedness scoring for sentence pairs (SemEval 2012, Task 6 and SemEval 2014, Task 1) and paraphrase detection for question pairs (Quora, 2017).[1]

1 Introduction

Recurrent Neural Networks (RNNs), in particular Long Short-Term Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997), have demonstrated remarkable accomplishments in Natural Language Processing (NLP) in recent years. Several tasks such as information extraction, question answering, and machine translation have benefited from them. However, in their vanilla forms, these networks are constrained by the sequential order of tokens in a sentence. To mitigate this limitation, structural (*dependency* or *constituency*) information in a sentence was exploited and witnessed partial success in various tasks (Goller and Kuchler, 1996; Yamada and Knight, 2001; Quirk et al., 2005; Socher et al., 2011; Tai et al., 2015).

On the other hand, alignment techniques (Brown et al., 1993) and attention mechanisms (Bahdanau et al., 2014) act as a catalyst to augment the performance of classical Statistical Machine Translation (SMT) and Neural Machine Translation (NMT) models, respectively. In short, both approaches focus on sub-strings of source sentence which are significant for predicting target words while translating. Currently, the combination of linear RNNs/LSTMs and attention mechanisms has become a *de facto* standard architecture for many NLP tasks.

At the intersection of sentence encoding and attention models, some interesting questions emerge: Can attention mechanisms be employed on tree structures, such as Tree-LSTMs (Tai et al., 2015)? If yes, what are the possible tree-based attention models? Do different tree structures (in particular constituency vs. dependency) have different behaviors in such models? With these questions in mind, we present our investigation and findings in the context of semantic relatedness tasks.

2 Background

2.1 Long Short-Term Memory Networks (LSTMs)

Concisely, an LSTM network (Hochreiter and Schmidhuber, 1997) (Figure 1) includes a *memory cell* at each time step which controls the amount of information being penetrated into the cell, neglected, and yielded by the cell. Various LSTM networks (Greff et al., 2017) have been explored till now; we focus on one representative form. To be more precise, we consider a LSTM memory cell involving: an *input gate* i_t, a forget gate f_t, and an output gate o_t at time step t. Apart from

[1]Our code for experiments on the SICK dataset is publicly available at https://github.com/amulyahwr/acl2018

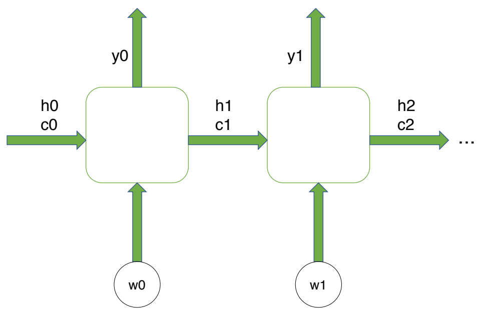


Figure 1: A linear LSTM network. w_t is the word embedding, h_t is the hidden state vector, c_t is the memory cell vector and y_t is the final processed output at time step *t*.

the hidden state h_{t-1} and input embedding w_t of the current word, the recursive function in LSTM also takes the previous time's *memory cell state*, c_{t-1}, into account, which is not the case in simple RNN. The following equations summarize a LSTM memory cell at time step *t*:

$$i_t = \sigma(w_t W^i + h_{t-1} R^i + b^i) \tag{1}$$

$$f_t = \sigma(w_t W^f + h_{t-1} R^f + b^f) \tag{2}$$

$$o_t = \sigma(w_t W^o + h_{t-1} R^o + b^o) \tag{3}$$

$$u_t = \tanh(w_t W^u + h_{t-1} R^u + b^u) \tag{4}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where:

- $(\boldsymbol{W^i}, \boldsymbol{W^f}, \boldsymbol{W^o}, \boldsymbol{W^u}) \in \mathbb{R}^{\mathrm{D\ x\ d}}$ represent input weight matrices, where $\boldsymbol{d}$ is the dimension of the hidden state vector and $\boldsymbol{D}$ is the dimension of the input word embedding, $\boldsymbol{w_t}$.
- $(\boldsymbol{R^i}, \boldsymbol{R^f}, \boldsymbol{R^o}, \boldsymbol{R^u}) \in \mathbb{R}^{\mathrm{d\ x\ d}}$ represent recurrent weight matrices and $(\boldsymbol{b^i}, \boldsymbol{b^f}, \boldsymbol{b^o}, \boldsymbol{b^u}) \in \mathbb{R}^{\mathrm{d}}$ represent biases.
- $\boldsymbol{c_t} \in \mathbb{R}^{\mathrm{d}}$ is the new memory cell vector at time step *t*.

As can be seen in Eq. 5, the input gate i_t limits the new information, u_t, by employing the element wise multiplication operator $\odot$. Moreover, the forget gate f_t regulates the amount of information from the previous state c_{t-1}. Therefore, the current memory state c_t includes both new and previous time step's information but partially.

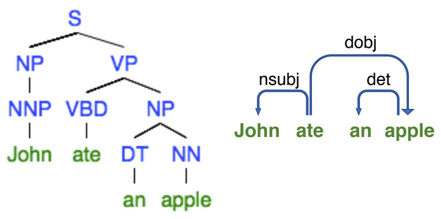


Figure 2: a. **Left**: A constituency tree; b. **Right**: A dependency tree

A natural extension of LSTM network is a *bidirectional* LSTM (bi-LSTM), which lets the sequence pass through the architecture in both directions and aggregate the information at each time step. Again, it strictly preserves the sequential nature of LSTMs.

2.2 Linguistically Motivated Sentence Structures

Most computational linguists have developed a natural inclination towards hierarchical structures of natural language, which follow guidelines collectively referred to as *syntax*. Typically, such structures manifest themselves in *parse trees*. We investigate two popular forms: *Constituency* and *Dependency* trees.

2.2.1 Constituency structure

Briefly, constituency trees (Figure 2:a) indicate a hierarchy of syntactic units and encapsulate phrase grammar rules. Moreover, these trees explicitly demonstrate groups of phrases (e.g., Noun Phrases) in a sentence. Additionally, they discriminate between terminal (lexical) and non-terminal nodes (non-lexical) tokens.

2.2.2 Dependency structure

In short, dependency trees (Figure 2:b) describe the syntactic structure of a sentence in terms of the words (lemmas) and associated grammatical relations among the words. Typically, these dependency relations are explicitly *typed*, which makes the trees valuable for practical applications such as information extraction, paraphrase detection and semantic relatedness.

2.3 Tree Long Short-Term Memory Network (Tree-LSTM)

Child-Sum Tree-LSTM (Tai et al., 2015) is an epitome of structure-based neural network which explicitly capture the structural information in a sentence. Tai et al. demonstrated that information at

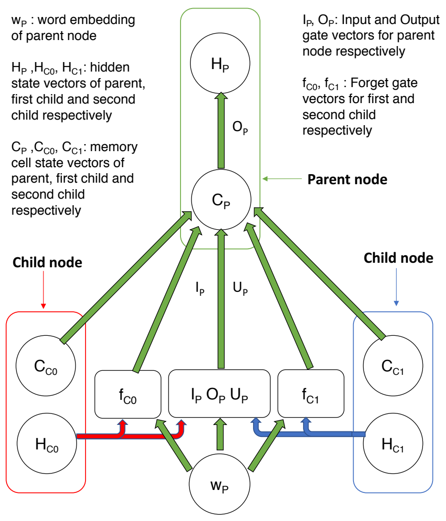

w_P : word embedding of parent node
H_P ,H_C0, H_C1: hidden state vectors of parent, first child and second child respectively
C_P ,C_C0, C_C1: memory cell state vectors of parent, first child and second child respectively
I_P, O_P: Input and Output gate vectors for parent node respectively
f_C0, f_C1 : Forget gate vectors for first and second child respectively
Parent node
Child node
Child node
H_P
O_P
C_P
I_P
U_P
C_C0
H_C0
f_C0
I_P O_P U_P
f_C1
C_C1
H_C1
w_P

Figure 3: A compositional view of parent node in Tree-LSTM network.

a parent node can be consolidated selectively from each of its child node. Architecturally, each gated vector and memory state update of the head node is dependent on the hidden states of its children in the Tree-LSTM. Assuming a good tree structure of a sentence, each node *j* of the structure incorporates the following equations.:

$$\tilde{h_j} = \sum_{k \in C(j)} h_k \tag{7}$$

$$i_j = \sigma(w_j W^i + \tilde{h_j} R^i + b^i) \tag{8}$$

$$f_{jk} = \sigma(w_j W^f + h_k R^f + b^f) \tag{9}$$

$$o_j = \sigma(w_j W^o + \tilde{h_j} R^o + b^o) \tag{10}$$

$$u_j = \tanh(w_j W^u + \tilde{h_j} R^u + b^u) \tag{11}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \tag{12}$$

$$h_j = o_j \odot \tanh(c_j) \tag{13}$$

where:

- $\boldsymbol{w_j} \in \mathbb{R}^{\mathrm{D}}$ represents word embedding of all nodes in Dependency structure and only terminal nodes in Constituency structure. [2]
- $(\boldsymbol{W^i}, \boldsymbol{W^f}, \boldsymbol{W^o}, \boldsymbol{W^u}) \in \mathbb{R}^{\mathrm{D\ x\ d}}$ represent input weight matrices.
- $(\boldsymbol{R^i}, \boldsymbol{R^f}, \boldsymbol{R^o}, \boldsymbol{R^u}) \in \mathbb{R}^{\mathrm{d\ x\ d}}$ represent recurrent weight matrices, and $(\boldsymbol{b^i}, \boldsymbol{b^f}, \boldsymbol{b^o}, \boldsymbol{b^u}) \in \mathbb{R}^{\mathrm{d}}$ represent biases.

[2] $\boldsymbol{w_j}$ is ignored for non-terminal nodes in a Constituency structure by removing the $\boldsymbol{wW}$ terms in Equations 8-11.

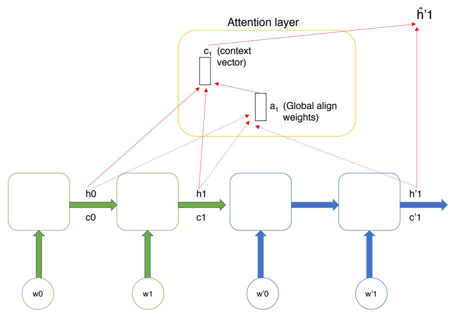


Figure 4: Global attention model

- $\boldsymbol{c_j} \in \mathbb{R}^{\mathrm{d}}$ is the new memory state vector of node *j*.
- $\boldsymbol{C(j)}$ is the set of children of node *j*.
- $\boldsymbol{f_{jk}} \in \mathbb{R}^{\mathrm{d}}$ is the forget gate vector for child *k* of node *j*.

Referring to Equation 12, the new memory cell state, c_j of node *j*, receives new information, u_j, partially. More importantly, it includes the partial information from each of its direct children, set *C(j)*, by employing the corresponding forget gate, f_{jk}.

When the Child-Sum Tree model is deployed on a dependency tree, it is referred to as *Dependency Tree-LSTM*, whereas a constituency-tree-based instantiation is referred to as *Constituency Tree-LSTM*.

2.4 Attention Mechanisms

Alignment models were first introduced in statistical machine translation (SMT) (Brown et al., 1993), which connect sub-strings in the source sentence to sub-strings in the target sentence.

Recently, attention techniques (which are effectively soft alignment models) in neural machine translation (NMT) (Bahdanau et al., 2014) came into prominence, where *attention* scores are calculated by considering words of source sentence while decoding words in target language. Although effective attention mechanisms (Luong et al., 2015) such as Global Attention Model (GAM) (Figure 4) and Local Attention Model (LAM) have been developed, such techniques have not been explored over Tree-LSTMs.

3 Inter-Sentence Attention on Tree-LSTMs

We present two types of tree-based attention models in this section. With trivial adaptation, they can be deployed in the sequence setting (degenerated trees).

3.1 Modified Decomposable Attention (MDA)

Parikh et al. (2016)'s original decomposable inter-sentence attention model only used word embeddings to construct the attention matrix, without any structural encoding of sentences. Essentially, the model incorporated three components:

Attend: Input representations (without sequence or structural encoding) of both sentences, L and R, are soft-aligned.

Compare: A set of vectors is produced by separately comparing each sub-phrase of L to sub-phrases in R. Vector representation of each sub-phrase in L is a non-linear combination of representation of word in sentence L and its aligned sub-phrase in sentence R. The same holds true for the set of vectors for sentence R.

Aggregate: Both sets of sub-phrases vectors are summed up separately to form final sentence representation of sentence L and sentence R.

We decide to augment the original decomposable inter-sentence attention model and generalize it into the tree (and sequence) setting. To be more specific, we consider two input sequences: L = (l_1, $l_2....l_{len_L}$), R = (r_1, $r_2....r_{len_R}$) and their corresponding input representations: $\bar{L}$ = ($\bar{l}_1$, $\bar{l}_2....\bar{l}_{len_L}$), $\bar{R}$ = ($\bar{r}_1$, $\bar{r}_2....\bar{r}_{len_R}$); where len_L and len_R represents number of words in L and R, respectively.

3.1.1 MDA on dependency structure

Let's assume sequences L and R have dependency tree structures D_L and D_R. In this case, len_L and len_R represents number of nodes in D_L and D_R, respectively. ***After*** using a Tree-LSTM to encode tree representations, which results in: $D_L^{'}$ = ($\bar{l}_1^{'}$, $\bar{l}_2^{'}....\bar{l}_{len_L}^{'}$), $D_R^{'}$ = ($\bar{r}_1^{'}$, $\bar{r}_2^{'}....\bar{r}_{len_R}^{'}$), we gather unnormalized attention weights, e_{ij} and normalize them as follows:

$$e_{ij} = \bar{l}_i^{'}(\bar{r}_j^{'})^T \tag{14}$$

$$\beta_i = \sum_{j=1}^{len_R} \frac{exp(e_{ij})}{\sum_{k=1}^{len_R} exp(e_{ik})} * \bar{r}_j^{'} \tag{15}$$

$$\alpha_j = \sum_{i=1}^{len_L} \frac{exp(e_{ij})}{\sum_{k=1}^{len_L} exp(e_{kj})} * \bar{l}_i^{'} \tag{16}$$

From the equations above, we can infer that the attention matrix will have a dimension len_L x len_R. In contrast to the original model, we compute the final representations of the each sentence by concatenating the LSTM-encoded representation of root with the attention-weighted representation of the root [3]:

$$h_L^{''} = G([\bar{l}_{root_L}^{'}; \beta_{root_L}]) \tag{17}$$

$$h_R^{''} = G([\bar{r}_{root_R}^{'}; \alpha_{root_R}]) \tag{18}$$

where G is a feed-forward neural network. $h_L^{''}$ and $h_R^{''}$ are final vector representations of input sequences L and R, respectively.

3.1.2 MDA on constituency structure

Let's assume sequences L and R have constituency tree structures C_L and C_R. Moreover, assume C_L and C_R have total number of nodes as N_L ($> len_L$) and N_R ($> len_R$), respectively. As in 3.1.1, the attention mechanism is employed **after** encoding the trees C_L and C_R. While encoding trees, terminal and non-terminal nodes are handled in the same way as in the original Tree-LSTM model (see 2.3).

It should be noted that we collect hidden states of all the nodes (N_L and N_R) individually in C_L and C_R during the encoding process. Hence, hidden states matrix will have dimension N_L x *d* for tree C_L whereas for tree C_R, it will have dimension N_R x *d*; where *d* is dimension of each hidden state. Therefore, attention matrix will have a dimension N_L x N_R. Finally, we employ Equations 14-18 to compute the final representations of sequences L and R.

3.2 Progressive Attention (PA)

In this section, we propose a novel attention mechanism on Tree-LSTM, inspired by (Quirk et al., 2005) and (Yamada and Knight, 2001).

3.2.1 PA on dependency structure

Let's assume a dependency tree structure of sentence L = (l_1, $l_2....l_{len_L}$) is available as D_L; where len_L represents number of nodes in D_L. Similarly, tree D_R corresponds to the sentence R = (r_1, $r_2....r_{len_R}$); where len_R represents number of nodes in D_R.

In PA, the objective is to produce the final vector representation of tree D_R conditional on the hidden state vectors of *all* nodes of D_L. Similar to

[3]In the sequence setting, we compute the corresponding representations for the *last* word in the sentence.

the encoding process in NMT, we encode R by *attending* each node of D_R to all nodes in D_L. Let's name this process *Phase1*. Next, *Phase2* is performed where L is encoded in the similar way to get the final vector representation of D_L.

Referring to Figure 5 and assuming Phase1 is being executed, a hidden state matrix, H_L, is obtained by concatenating the hidden state vector of every node in tree D_L, where the number of nodes in D_L = 3. Next, tree D_R is processed by calculating the hidden state vector at every node. Assume that the current node being processed is n_{R2} of D_R, which has a hidden state vector, h_{R2}. Before further processing, normalized weights are calculated based on h_{R2} and H_L. Formally,

$$H_{pj} = stack[h_{pj}] \tag{19}$$

$$con_{pj} = concat[H_{pj}, H_q] \tag{20}$$

$$a_{pj} = softmax(tanh(con_{pj}W_c + b) * W_a) \tag{21}$$

where:

- $\boldsymbol{p}, \boldsymbol{q} \in \{\boldsymbol{L}, \boldsymbol{R}\}$ and $\boldsymbol{q} \neq \boldsymbol{p}$
- $\boldsymbol{H_q} \in \mathbb{R}^{\boldsymbol{x} \text{ x d}}$ represents a matrix obtained by concatenating hidden state vectors of nodes in tree $\boldsymbol{D_q}$; $\boldsymbol{x}$ is $\boldsymbol{len_q}$ of sentence $\boldsymbol{q}$.
- $\boldsymbol{H_{pj}} \in \mathbb{R}^{\boldsymbol{x} \text{ x d}}$ represents a matrix obtained by stacking hidden state, $\boldsymbol{h_{pj}}$, vertically $\boldsymbol{x}$ times.
- $\boldsymbol{con_{pj}} \in \mathbb{R}^{\boldsymbol{x} \text{ x 2d}}$ represents the concatenated matrix.
- $\boldsymbol{a_{pj}} \in \mathbb{R}^{\boldsymbol{x}}$ represents the normalized attention weights at node *j* of tree $\boldsymbol{D_p}$; where $\boldsymbol{D_p}$ is the dependency structure of sentence $\boldsymbol{p}$.
- $\boldsymbol{W_c} \in \mathbb{R}^{\text{2d x d}}$ and $\boldsymbol{W_a} \in \mathbb{R}^{\text{d}}$ represent learned weight matrices.

The normalized attention weights in above equations provide an opportunity to align the subtree at the current node, n_{R2}, in D_R to sub-trees available at all nodes in D_L. Next, a gated mechanism is employed to compute the final vector representation at node n_{R2}.

Formally,

$$h'_{pj} = \sum_{0}^{(x-1)} ((1 - a_{pj}) * H_q + (a_{pj}) * H_{pj}) \tag{22}$$

where:

- $\boldsymbol{h'_{pj}} \in \mathbb{R}^{\text{d}}$ represents the final vector representation of node *j* in tree $\boldsymbol{D_p}$
- $\sum_{\boldsymbol{0}}^{\boldsymbol{(x-1)}}$ represents column-wise sum

Assuming the final vector representation of tree D_R is h'_R, the exact same steps are followed for Phase2 with the exception that the entire process is now conditional on tree D_R. As a result, the final vector representation of tree D_L, h'_L, is computed.

Lastly, the following equations are applied to vectors h'_L and h'_R, before calculating the *angle* and *distance* similarity (see Section 4).

$$h''_L = tanh(h'_L + h_L) \tag{23}$$

$$h''_R = tanh(h'_R + h_R) \tag{24}$$

where:

- $\boldsymbol{h_L} \in \mathbb{R}^{\text{d}}$ represents the vector representation of tree $\boldsymbol{D_L}$ without attention.
- $\boldsymbol{h_R} \in \mathbb{R}^{\text{d}}$ represents the vector representation of tree $\boldsymbol{D_R}$ without attention.

3.2.2 PA on constituency structure

Let C_L and C_R represent constituency trees of L and R, respectively; where C_L and C_R have total number of nodes N_L ($> len_L$) and N_R ($> len_R$). Additionally, let's assume that trees C_L and C_R have the same configuration of nodes as in Section 3.1.2, and the encoding of *terminal* and *non-terminal* nodes follow the same process as in Section 3.1.2. Assuming we have already encoded all N_L nodes of tree C_L using Tree-LSTM, we will have the hidden state matrix, H_L, with dimension N_L x d. Next, while encoding any node of C_R, we consider H_L which results in an attention vector having shape N_L. Using Equations 19-22 [4], we retrieve the final hidden state of the current node. Finally, we compute the representation of sentence R based on attention to sentence L. We perform Phase2 with the same process, except that we now condition on sentence R.

In summary, the *progressive attention* mechanism refers to all nodes in the other tree ***while*** encoding a node in the current tree, instead of waiting till the end of the structural encoding to establish cross-sentence attention, as was done in the *decomposable attention* model.

[4]At this point, we will consider $\boldsymbol{C_q}$ and $\boldsymbol{C_p}$ instead of $\boldsymbol{D_q}$ and $\boldsymbol{D_p}$, respectively, in Equations 19-22. Additionally, $\boldsymbol{x}$ will be equal to total number of nodes in the constituency tree.

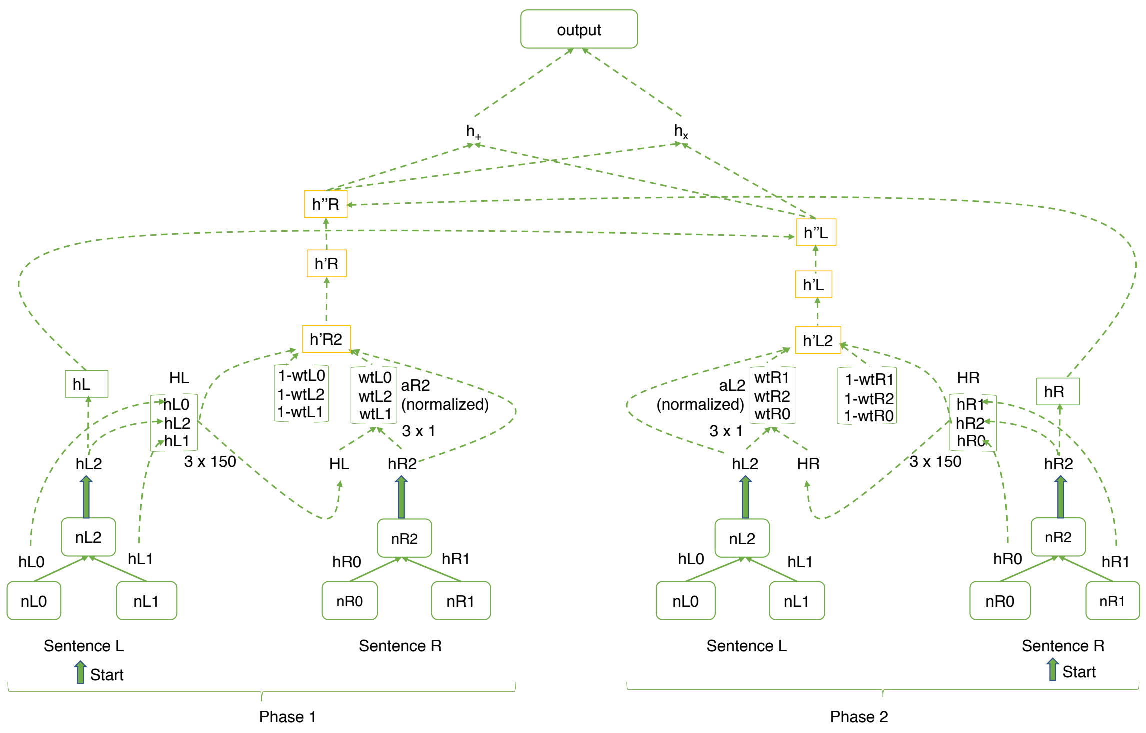


Figure 5: Progressive Attn-Tree-LSTM model

4 Evaluation Tasks

We evaluate our models on two tasks: (1) semantic relatedness scoring for sentence pairs (SemEval 2012, Task 6 and SemEval 2014, Task 1) and (2) paraphrase detection for question pairs (Quora, 2017).

4.1 Semantic Relatedness for Sentence Pairs

In SemEval 2012, Task 6 and SemEval 2014, Task 1, every sentence pair has a real-valued score that depicts the extent to which the two sentences are semantically related to each other. Higher score implies higher semantic similarity between the two sentences. Vector representations h''_L and h''_R are produced by using our *Modified Decomp-Attn* or *Progressive-Attn* models. Next, a similarity score, $\hat{y}$ between h''_L and h''_R is computed using the same neural network (see below), for the sake of fair comparison between our models and the original *Tree-LSTM* (Tai et al., 2015).

$$h_x = h''_L \odot h''_R \tag{25}$$

$$h_+ = |h''_L - h''_R| \tag{26}$$

$$h_s = \sigma(h_x W^x + h_+ W^+ + b^h) \tag{27}$$

$$\hat{p_\theta} = softmax(h_s W^p + b^p) \tag{28}$$

$$\hat{y} = r^T \hat{p_\theta} \tag{29}$$

where:

- $r^T = [1, 2..S]$
- $h_x \in \mathbb{R}^d$ measures the sign similarity between h''_L and h''_R
- $h_+ \in \mathbb{R}^d$ measures the absolute distance between h''_L and h''_R

Following (Tai et al., 2015), we convert the regression problem into a soft classification. We also use the same sparse distribution, p, which was defined in the original Tree-LSTM to transform the gold rating for a sentence pair, such that $y = r^T p$ and $\hat{y} = r^T \hat{p_\theta} \approx y$. The loss function is the KL-divergence between p and $\hat{p}$:

$$J(\theta) = \frac{\sum_{k=1}^{m} KL(p^k || \hat{p}_\theta^k)}{m} + \frac{\lambda ||\theta||_2^2}{2} \tag{30}$$

- m is the number of sentence pairs in the dataset.
- λ represents the regularization penalty.

4.2 Paraphrase Detection for Question Pairs

In this task, each question pair is labeled as either paraphrase or not, hence the task is binary classification. We use Eqs. 25 - 28 to compute the predicted distribution $\hat{p}_\theta$. The predicted label, $\hat{y}$, will be:

$$\hat{y} = \arg\max_y \hat{p}_\theta \tag{31}$$

The loss function is the negative log-likelihood:

$$J(\theta) = -\frac{\sum_{k=1}^{m} y^k \log \hat{y}^k}{m} + \frac{\lambda ||\theta||_2^2}{2} \tag{32}$$

5 Experiments

5.1 Semantic Relatedness for Sentence Pairs

We utilized two different datasets:

- The Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al. (2014)), which contains a total of 9,927 sentence pairs. Specifically, the dataset has a split of 4500/500/4927 among training, dev, and test. Each sentence pair has a score $S \in [1,5]$, which represents an average of 10 different human judgments collected by crowd-sourcing techniques.

- The MSRpar dataset (Agirre et al., 2012), which consists of 1,500 sentence pairs. In this dataset, each pair is annotated with a score $S \in [0,5]$ and has a split of 750/750 between training and test.

We used the Stanford Parsers (Chen and Manning, 2014; Bauer) to produce dependency and constituency parses of sentences. Moreover, we initialized the word embeddings with 300-dimensional Glove vectors (Pennington et al., 2014); the word embeddings were held fixed during training. We experimented with different optimizers, among which AdaGrad performed the best. We incorporated a learning rate of 0.025 and regularization penalty of 10^{-4} without dropout.

5.2 Paraphrase Detection for Question Pairs

For this task, we utilized the Quora dataset (Iyer; Kaggle, 2017). Given a pair of questions, the objective is to identify whether they are semantic duplicates. It is a binary classification problem where a duplicate question pair is labeled as 1 otherwise as 0. The training set contains about 400,000 labeled question pairs, whereas the test set consists of 2.3 million unlabeled question pairs. Moreover, the training dataset has only 37% positive samples; average length of a question is 10 words. Due to hardware and time constraints, we extracted 50,000 pairs from the original training while maintaining the same positive/negative ratio. A stratified 80/20 split was performed on this subset to produce the training/test set. Finally, 5% of the training set was used as a validation set in our experiments.

We used an identical training configuration as for the semantic relatedness task since the essence of both the tasks is practically the same. We also performed pre-processing to clean the data and then parsed the sentences using Stanford Parsers.

6 Results

6.1 Semantic Relatedness for Sentence Pairs

Table 1 summarizes our results. According to (Marelli et al., 2014), we compute three evaluation metrics: Pearson's *r*, Spearman's ρ and Mean Squared Error (MSE). We compare our attention models against the original Tree-LSTM (Tai et al., 2015), instantiated on both constituency trees and dependency trees. We also compare earlier baselines with our models, and the best results are in bold. Since Tree-LSTM is a generalization of Linear LSTM, we also implemented our attention models on Linear *Bidirectional* LSTM (BiLSTM). All results are average of 5 runs. It is witnessed that the *Progressive-Attn* mechanism combined with *Constituency Tree-LSTM* is overall the strongest contender, but PA failed to yield any performance gain on Dependency Tree-LSTM in either dataset.

6.2 Paraphrase Detection for Question Pairs

Table 2 summarizes our results where best results are highlighted in bold within each category. It should be noted that Quora is a new dataset and we have done our analysis on only 50,000 samples. Therefore, to the best of our knowledge, there is no published baseline result yet. For this task, we considered four standard evaluation metrics: Accuracy, F1-score, Precision and Recall. The *Progressive-Attn + Constituency Tree-LSTM* model still exhibits the best performance by a small margin, but the *Progressive-Attn* mechanism works surprisingly well on the linear bi-LSTM.

6.3 Effect of the Progressive Attention Model

Table 3 illustrates how various models operate on two sentence pairs from SICK test dataset. As we can infer from the table, the first pair demonstrates an instance of the active-passive voice phenomenon. In this case, the linear LSTM and vanilla Tree-LSTMs really struggle to perform.

Table 1: Results on test dataset for SICK and MSRpar semantic relatedness task. Mean scores are presented based on 5 runs (standard deviation in parenthesis). Categories of results: (1) Previous models (2) Dependency structure (3) Constituency structure (4) Linear structure

Dataset	Model	Pearson's *r*	Spearman's ρ	MSE
SICK	Illinois-LH (2014)	0.7993	0.7538	0.3692
	UNAL-NLP (2014)	0.8070	0.7489	0.3550
	Meaning factory (2014)	0.8268	0.7721	0.3224
	ECNU (2014)	0.8414	-	-
	Dependency Tree-LSTM (2015)	**0.8676 (0.0030)**	**0.8083 (0.0042)**	**0.2532 (0.0052)**
	Decomp-Attn (Dependency)	0.8239 (0.0120)	0.7614 (0.0103)	0.3326 (0.0223)
	Progressive-Attn (Dependency)	0.8424 (0.0042)	0.7733 (0.0066)	0.2963 (0.0077)
	Constituency Tree-LSTM (2015)	0.8582 (0.0038)	0.7966 (0.0053)	0.2734 (0.0108)
	Decomp-Attn (Constituency)	0.7790 (0.0076)	0.7074 (0.0091)	0.4044 (0.0152)
	Progressive-Attn (Constituency)	**0.8625 (0.0032)**	**0.7997 (0.0035)**	**0.2610 (0.0057)**
	Linear Bi-LSTM	0.8398 (0.0020)	0.7782 (0.0041)	0.3024 (0.0044)
	Decomp-Attn (Linear)	0.7899 (0.0055)	0.7173 (0.0097)	0.3897 (0.0115)
	Progressive-Attn (Linear)	**0.8550 (0.0017)**	**0.7873 (0.0020)**	**0.2761 (0.0038)**
MSRpar	ParagramPhrase (2015)	0.426	-	-
	Projection (2015)	0.437	-	-
	GloVe (2015)	0.477	-	-
	PSL (2015)	0.416	-	-
	ParagramPhrase-XXL (2015)	0.448	-	-
	Dependency Tree-LSTM	**0.4921 (0.0112)**	**0.4519 (0.0128)**	**0.6611 (0.0219)**
	Decomp-Attn (Dependency)	0.4016 (0.0124)	0.3310 (0.0118)	0.7243 (0.0099)
	Progressive-Attn (Dependency)	0.4727 (0.0112)	0.4216 (0.0092)	0.6823 (0.0159)
	Constituency Tree-LSTM	0.3981 (0.0176)	0.3150 (0.0204)	0.7407 (0.0170)
	Decomp-Attn (Constituency)	0.3991 (0.0147)	0.3237 (0.0355)	0.7220 (0.0185)
	Progressive-Attn (Constituency)	**0.5104 (0.0191)**	**0.4764 (0.0112)**	**0.6436 (0.0346)**
	Linear Bi-LSTM	0.3270 (0.0303)	0.2205 (0.0111)	0.8098 (0.0579)
	Decomp-Attn (Linear)	0.3763 (0.0332)	0.3025 (0.0587)	0.7290 (0.0206)
	Progressive-Attn (Linear)	**0.4773 (0.0206)**	**0.4453 (0.0250)**	**0.6758 (0.0260)**

Table 2: Results on test dataset for Quora paraphrase detection task. Mean scores are presented based on 5 runs (standard deviation in parenthesis). Categories of results: (1) Dependency structure (2) Constituency structure (3) Linear structure

Model	Accuracy	F-1 score (class=1)	Precision (class=1)	Recall (class=1)
Dependency Tree-LSTM	**0.7897 (0.0009)**	0.7060 (0.0050)	**0.7298 (0.0055)**	0.6840 (0.0139)
Decomp-Attn (Dependency)	0.7803 (0.0026)	0.6977 (0.0074)	0.7095 (0.0083)	0.6866 (0.0199)
Progressive-Attn (Dependency)	0.7896 (0.0025)	**0.7113 (0.0087)**	0.7214 (0.0117)	**0.7025 (0.0266)**
Constituency Tree-LSTM	0.7881 (0.0042)	0.7065 (0.0034)	0.7192 (0.0216)	0.6846 (0.0380)
Decomp-Attn (Constituency)	0.7776 (0.0004)	0.6942 (0.0050)	0.7055 (0.0069)	0.6836 (0.0164)
Progressive-Attn (Constituency)	**0.7956 (0.0020)**	**0.7192 (0.0024)**	**0.7300 (0.0079)**	**0.7089 (0.0104)**
Linear Bi-LSTM	0.7859 (0.0024)	0.7097 (0.0047)	0.7112 (0.0129)	0.7089 (0.0219)
Decomp-Attn (Linear)	0.7861 (0.0034)	0.7074 (0.0109)	0.7151 (0.0135)	0.7010 (0.0315)
Progressive-Attn (Linear)	**0.7949 (0.0031)**	**0.7182 (0.0162)**	**0.7298 (0.0115)**	**0.7092 (0.0469)**

However, when our progressive attention mechanism is integrated into syntactic structures (dependency or constituency), we witness a boost in the semantic relatedness score. Such desirable behavior is consistently observed in multiple active-passive voice pairs. The second pair points to a possible issue in data annotation. Despite the presence of strong negation, the gold-standard score is 4 out of 5 (indicating high relatedness). Interestingly, the *Progressive-Attn + Dependency Tree-LSTM* model favors the negation facet and outputs a low relatedness score.

7 Discussion

In this section, let's revisit our research questions in light of the experimental results.

First, can attention mechanisms be built for Tree-LSTMs? Does it work? The answer is yes. Our novel progressive-attention Tree-LSTM model, when instantiated on constituency trees,

Table 3: Effect of the progressive attention model

ID	Test Pair	Gold	BiLSTM		Const. Tree		Dep. Tree	
			(no attn)	(PA)	(no attn)	(PA)	(no attn)	(PA)
1	S1: The badger is burrowing a hole. S2: A hole is being burrowed by the badger.	4.9	2.60	3.02	3.52	4.34	3.41	4.63
2	S1: There is no man screaming. S2: A man is screaming.	4	3.44	3.20	3.65	3.50	3.51	2.15

significantly outperforms its counterpart without attention. The same model can also be deployed on sequences (degenerated trees) and achieve quite impressive results.

Second, the performance gap between the two attention models is quite striking, in the sense that the progressive model completely dominate its decomposable counterpart. The difference between the two models is the *pacing* of attention, i.e., when to refer to nodes in the other tree while encoding a node in the current tree. The progressive attention model garners it's empirical superiority by *attending while encoding*, instead of waiting till the end of the structural encoding to establish cross-sentence attention. In retrospect, this may justify why the original decomposable attention model in (Parikh et al., 2016) achieved competitive results without any LSTM-type encoding. Effectively, they implemented a naive version of our progressive attention model.

Third, do structures matter/help? The overall trend in our results is quite clear: the tree-based models exhibit convincing empirical strength; linguistically motivated structures are valuable. Admittedly though, on the relatively large Quora dataset, we observe some diminishing returns of incorporating structural information. It is not counter-intuitive that the sheer size of data can possibly allow structural patterns to emerge, hence lessen the need to explicitly model syntactic structures in neural architectures.

Last but not least, in trying to assess the impact of attention mechanisms (in particular the progressive attention model), we notice that the extra mileage gained on different structural encodings is different. Specifically, performance lift on Linear Bi-LSTM > performance lift on Constituency Tree-LSTM, and PA struggles to see performance lift on dependency Tree-LSTM. Interestingly enough, this observation is echoed by an earlier study (Gildea, 2004), which showed that tree-based alignment models work better on constituency trees than on dependency trees.

In summary, our results and findings lead to several intriguing questions and conjectures, which call for investigation beyond the scope of our study:

- Is it reasonable to conceptualize attention mechanisms as an implicit form of structure, which complements the representation power of explicit syntactic structures?

- If yes, does there exist some trade-off between the modeling efforts invested into syntactic and attention structures respectively, which seemingly reveals itself in our empirical results?

- The marginal impact of attention on dependency Tree-LSTMs suggests some form of saturation effect. Does that indicate a closer affinity between dependency structures (relative to constituency structures) and compositional semantics (Liang et al., 2013)?

- If yes, *why* is dependency structure a better stepping stone for compositional semantics? Is it due to the strongly lexicalized nature of the grammar? Or is it because the dependency relations (grammatical functions) embody more semantic information?

8 Conclusion

In conclusion, we proposed a novel *progressive attention* model on syntactic structures, and demonstrated its superior performance in semantic relatedness tasks. Our work also provides empirical ingredients for potentially profound questions and debates on syntactic structures in linguistics.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ICLR'2015*.

John Bauer. Shift-reduce constituency parser.

Johannes Bjerva, Johan Bos, Rob Van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Daniel Gildea. 2004. Dependencies vs. constituents for tree-based alignment. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*.

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

Csernai Iyer, Dandekar. First quora dataset release: Question pairs.

Sergio Jimenez, George Duenas, Julia Baquero, and Alexander Gelbukh. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742.

Kaggle. 2017. Quora question pairs.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Comput. Linguist.*, 39(2):389–446.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 271–279. Association for Computational Linguistics.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277.

What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties

Alexis Conneau
Facebook AI Research
Université Le Mans
aconneau@fb.com

German Kruszewski
Facebook AI Research
germank@fb.com

Guillaume Lample
Facebook AI Research
Sorbonne Universités
glample@fb.com

Loïc Barrault
Université Le Mans
loic.barrault@univ-lemans.fr

Marco Baroni
Facebook AI Research
mbaroni@fb.com

Abstract

Although much effort has recently been devoted to training high-quality sentence embeddings, we still have a poor understanding of what they are capturing. "Downstream" tasks, often based on sentence classification, are commonly used to evaluate the quality of sentence representations. The complexity of the tasks makes it however difficult to infer what kind of information is present in the representations. We introduce here 10 probing tasks designed to capture simple linguistic features of sentences, and we use them to study embeddings generated by three different encoders trained in eight distinct ways, uncovering intriguing properties of both encoders and training methods.

1 Introduction

Despite Ray Mooney's quip that you cannot cram the meaning of a whole %&!$# sentence into a single $&!#* vector, sentence embedding methods have achieved impressive results in tasks ranging from machine translation (Sutskever et al., 2014; Cho et al., 2014) to entailment detection (Williams et al., 2018), spurring the quest for "universal embeddings" trained once and used in a variety of applications (e.g., Kiros et al., 2015; Conneau et al., 2017; Subramanian et al., 2018). Positive results on concrete problems suggest that embeddings capture important linguistic properties of sentences. However, real-life "downstream" tasks require complex forms of inference, making it difficult to pinpoint the information a model is relying upon. Impressive as it might be that a system can tell that the sentence "A movie that doesn't aim too high, but it doesn't need to" (Pang and Lee, 2004) expresses a subjective viewpoint, it is hard to tell *how* the system (or even a human) comes to this conclusion. Complex tasks can also carry hidden biases that models might lock onto (Jabri et al., 2016). For example, Lai and Hockenmaier (2014) show that the simple heuristic of checking for explicit negation words leads to good accuracy in the SICK sentence entailment task.

Model introspection techniques have been applied to sentence encoders in order to gain a better understanding of which properties of the input sentences their embeddings retain (see Section 5). However, these techniques often depend on the specifics of an encoder architecture, and consequently cannot be used to compare different methods. Shi et al. (2016) and Adi et al. (2017) introduced a more general approach, relying on the notion of what we will call *probing tasks*. A probing task is a classification problem that focuses on simple linguistic properties of sentences. For example, one such task might require to categorize sentences by the tense of their main verb. Given an encoder (e.g., an LSTM) pre-trained on a certain task (e.g., machine translation), we use the sentence embeddings it produces to train the tense classifier (without further embedding tuning). If the classifier succeeds, it means that the pre-trained encoder is storing readable tense information into the embeddings it creates. Note that: (i) The probing task asks a simple question, minimizing interpretability problems. (ii) Because of their simplicity, it is easier to control for biases in probing tasks than in downstream tasks. (iii) The probing task methodology is agnostic with respect to the encoder architecture, as long as it produces a vector representation of sentences.

We greatly extend earlier work on probing tasks as follows. First, we introduce a larger set of probing tasks (10 in total), organized by the type of linguistic properties they probe. Second, we systematize the probing task methodology, controlling for a number of possible nuisance factors, and framing all tasks so that they only require single sentence representations as input, for maximum generality and to ease result interpretation. Third, we use our probing tasks to explore a wide range of state-of-the-art encoding architectures and training methods, and further relate probing and downstream task performance. Finally, we are publicly releasing our probing data sets and tools, hoping they will become a standard way to study the linguistic properties of sentence embeddings.[1]

2 Probing tasks

In constructing our probing benchmarks, we adopted the following criteria. First, for generality and interpretability, the task classification problem should only require single sentence embeddings as input (as opposed to, e.g., sentence and word embeddings, or multiple sentence representations). Second, it should be possible to construct large training sets in order to train parameter-rich multi-layer classifiers, in case the relevant properties are non-linearly encoded in the sentence vectors. Third, nuisance variables such as lexical cues or sentence length should be controlled for. Finally, and most importantly, we want tasks that address an interesting set of linguistic properties. We thus strove to come up with a set of tasks that, while respecting the previous constraints, probe a wide range of phenomena, from superficial properties of sentences such as which words they contain to their hierarchical structure to subtle facets of semantic acceptability. We think the current task set is reasonably representative of different linguistic domains, but we are not claiming that it is exhaustive. We expect future work to extend it.

The sentences for all our tasks are extracted from the Toronto Book Corpus (Zhu et al., 2015), more specifically from the random pre-processed portion made available by Paperno et al. (2016). We only sample sentences in the 5-to-28 word range. We parse them with the Stanford Parser (2017-06-09 version), using the pre-trained PCFG model (Klein and Manning, 2003), and we rely on the part-of-speech, constituency and dependency parsing information provided by this tool where needed. For each task, we construct training sets containing 100k sentences, and 10k-sentence validation and test sets. All sets are balanced, having an equal number of instances of each target class.

Surface information These tasks test the extent to which sentence embeddings are preserving surface properties of the sentences they encode. One can solve the surface tasks by simply looking at tokens in the input sentences: no linguistic knowledge is called for. The first task is to predict the *length* of sentences in terms of number of words (**SentLen**). Following Adi et al. (2017), we group sentences into 6 equal-width bins by length, and treat SentLen as a 6-way classification task. The *word content* (**WC**) task tests whether it is possible to recover information about the original words in the sentence from its embedding. We picked 1000 mid-frequency words from the source corpus vocabulary (the words with ranks between 2k and 3k when sorted by frequency), and sampled equal numbers of sentences that contain one and only one of these words. The task is to tell which of the 1k words a sentence contains (1k-way classification). This setup allows us to probe a sentence embedding for word content without requiring an auxiliary word embedding (as in the setup of Adi and colleagues).

Syntactic information The next batch of tasks test whether sentence embeddings are sensitive to syntactic properties of the sentences they encode. The *bigram shift* (**BShift**) task tests whether an encoder is sensitive to legal word orders. In this binary classification problem, models must distinguish intact sentences sampled from the corpus from sentences where we inverted two random adjacent words ("What *you are* doing out there?").

The *tree depth* (**TreeDepth**) task checks whether an encoder infers the hierarchical structure of sentences, and in particular whether it can group sentences by the depth of the longest path from root to any leaf. Since tree depth is naturally correlated with sentence length, we de-correlate these variables through a structured sampling procedure. In the resulting data set, tree depth values range from 5 to 12, and the task is to categorize sentences into the class corresponding to their depth (8 classes). As an example, the following is a long (22 tokens) but shallow (max depth: 5) sentence: "[$_1$ [$_2$ But right now, for the time being, my past, my fears, and my thoughts [$_3$ were [$_4$ my [$_5$business]]].]]" (the outermost brackets correspond to the ROOT and S nodes in the parse).

[1]https://github.com/facebookresearch/SentEval/tree/master/data/probing

In the top constituent task (**TopConst**), sentences must be classified in terms of the sequence of top constituents immediately below the sentence (S) node. An encoder that successfully addresses this challenge is not only capturing latent syntactic structures, but clustering them by constituent types. TopConst was introduced by Shi et al. (2016). Following them, we frame it as a 20-way classification problem: 19 classes for the most frequent top constructions, and one for all other constructions. As an example, "[Then] [very dark gray letters on a black screen] [appeared] [.]" has top constituent sequence: "ADVP NP VP .".

Note that, while we would not expect an untrained human subject to be explicitly aware of tree depth or top constituency, similar information must be implicitly computed to correctly parse sentences, and there is suggestive evidence that the brain tracks something akin to tree depth during sentence processing (Nelson et al., 2017).

Semantic information These tasks also rely on syntactic structure, but they further require some understanding of what a sentence denotes. The **Tense** task asks for the tense of the main-clause verb (VBP/VBZ forms are labeled as present, VBD as past). No target form occurs across the train/dev/test split, so that classifiers cannot rely on specific words (it is not clear that Shi and colleagues, who introduced this task, controlled for this factor). The *subject number* (**SubjNum**) task focuses on the number of the subject of the main clause (number in English is more often explicitly marked on nouns than verbs). Again, there is no target overlap across partitions. Similarly, *object number* (**ObjNum**) tests for the number of the direct object of the main clause (again, avoiding lexical overlap). To solve the previous tasks correctly, an encoder must not only capture tense and number, but also extract structural information (about the main clause and its arguments). We grouped Tense, SubjNum and ObjNum with the semantic tasks, since, at least for models that treat words as unanalyzed input units (without access to morphology), they must rely on what a sentence denotes (e.g., whether the described event took place in the past), rather than on structural/syntactic information. We recognize, however, that the boundary between syntactic and semantic tasks is somewhat arbitrary.

In the *semantic odd man out* (**SOMO**) task, we modified sentences by replacing a random noun or verb o with another noun or verb r. To make the task more challenging, the bigrams formed by the replacement with the previous and following words in the sentence have frequencies that are comparable (on a log-scale) with those of the original bigrams. That is, if the original sentence contains bigrams $w_{n-1}o$ and ow_{n+1}, the corresponding bigrams $w_{n-1}r$ and rw_{n+1} in the modified sentence will have comparable corpus frequencies. No sentence is included in both original and modified format, and no replacement is repeated across train/dev/test sets. The task of the classifier is to tell whether a sentence has been modified or not. An example modified sentence is: " No one could see this Hayes and I wanted to know if it was real or a *spoonful* (orig.: *ploy*)." Note that judging plausibility of a syntactically well-formed sentence of this sort will often require grasping rather subtle semantic factors, ranging from selectional preference to topical coherence.

The coordination inversion (**CoordInv**) benchmark contains sentences made of two coordinate clauses. In half of the sentences, we inverted the order of the clauses. The task is to tell whether a sentence is intact or modified. Sentences are balanced in terms of clause length, and no sentence appears in both original and inverted versions. As an example, original "They might be only memories, but I can still feel each one" becomes: "I can still feel each one, but they might be only memories." Often, addressing CoordInv requires an understanding of broad discourse and pragmatic factors.

Row **Hum. Eval.** of Table 2 reports human-validated "reasonable" upper bounds for all the tasks, estimated in different ways, depending on the tasks. For the surface ones, there is always a straightforward correct answer that a human annotator with enough time and patience could find. The upper bound is thus estimated at 100%. The TreeDepth, TopConst, Tense, SubjNum and ObjNum tasks depend on automated PoS and parsing annotation. In these cases, the upper bound is given by the proportion of sentences correctly annotated by the automated procedure. To estimate this quantity, one linguistically-trained author checked the annotation of 200 randomly sampled test sentences from each task. Finally, the BShift, SOMO and CoordInv manipulations can accidentally generate acceptable sentences. For example, one modified SOMO sentence is: "He pulled out the large round *onion* (orig.: *cork*) and saw the amber balm inside.", that is arguably not more anomalous than the original. For these tasks, we ran Amazon Mechanical Turk experiments in which subjects were asked to judge whether 1k randomly sampled test sentences were acceptable or not. Reported human accuracies are based on majority voting. See Appendix for details.

3 Sentence embedding models

In this section, we present the three sentence encoders that we consider and the seven tasks on which we train them.

3.1 Sentence encoder architectures

A wide variety of neural networks encoding sentences into fixed-size representations exist. We focus here on three that have been shown to perform well on standard NLP tasks.

BiLSTM-last/max For a sequence of T words $\{w_t\}_{t=1,\ldots,T}$, a bidirectional LSTM computes a set of T vectors $\{h_t\}_t$. For $t \in [1, \ldots, T]$, h_t is the concatenation of a forward LSTM and a backward LSTM that read the sentences in two opposite directions. We experiment with two ways of combining the varying number of $(h_1, \ldots, h_T)$ to form a fixed-size vector, either by selecting the last hidden state of h_T or by selecting the maximum value over each dimension of the hidden units. The choice of these models are motivated by their demonstrated efficiency in seq2seq (Sutskever et al., 2014) and universal sentence representation learning (Conneau et al., 2017), respectively.[2]

Gated ConvNet We also consider the non-recurrent convolutional equivalent of LSTMs, based on stacked gated temporal convolutions. Gated convolutional networks were shown to perform well as neural machine translation encoders (Gehring et al., 2017) and language modeling decoders (Dauphin et al., 2017). The encoder is composed of an input word embedding table that is augmented with positional encodings (Sukhbaatar et al., 2015), followed by a stack of temporal convolutions with small kernel size. The output of each convolutional layer is filtered by a gating mechanism, similar to the one of LSTMs. Finally, max-pooling along the temporal dimension is performed on the output feature maps of the last convolution (Collobert and Weston, 2008).

[2]We also experimented with a unidirectional LSTM, with consistently poorer results.

3.2 Training tasks

Seq2seq systems have shown strong results in machine translation (Zhou et al., 2016). They consist of an *encoder* that encodes a source sentence into a fixed-size representation, and a *decoder* which acts as a conditional language model and that generates the target sentence. We train **Neural Machine Translation** systems on three language pairs using about 2M sentences from the Europarl corpora (Koehn, 2005). We pick **English-French**, which involves two similar languages, **English-German**, involving larger syntactic differences, and **English-Finnish**, a distant pair. We also train with an **AutoEncoder** objective (Socher et al., 2011) on Europarl source English sentences. Following Vinyals et al. (2015), we train a seq2seq architecture to generate linearized grammatical parse trees (see Table 1) from source sentences (**Seq2Tree**). We use the Stanford parser to generate trees for Europarl source English sentences. We train **SkipThought** vectors (Kiros et al., 2015) by predicting the next sentence given the current one (Tang et al., 2017), on 30M sentences from the Toronto Book Corpus, excluding those in the probing sets. Finally, following Conneau et al. (2017), we train sentence encoders on **Natural Language Inference** using the concatenation of the SNLI (Bowman et al., 2015) and MultiNLI (Bowman et al., 2015) data sets (about 1M sentence pairs). In this task, a sentence encoder is trained to encode two sentences, which are fed to a classifier and whose role is to distinguish whether the sentences are contradictory, neutral or entailed. Finally, as in Conneau et al. (2017), we also include **Untrained** encoders with random weights, which act as random projections of pre-trained word embeddings.

3.3 Training details

BiLSTM encoders use 2 layers of 512 hidden units ($\sim$4M parameters), Gated ConvNet has 8 convolutional layers of 512 hidden units, kernel size 3 ($\sim$12M parameters). We use pre-trained fastText word embeddings of size 300 (Mikolov et al., 2018) without fine-tuning, to isolate the impact of encoder architectures and to handle words outside the training sets. Training task performance and further details are in Appendix.

task	source	target
AutoEncoder	I myself was out on an island in the Swedish archipelago , at Sandhamn .	I myself was out on an island in the Swedish archipelago , at Sand@ ham@ n .
NMT En-Fr	I myself was out on an island in the Swedish archipelago , at Sandhamn .	Je me trouvais ce jour là sur une île de l' archipel suédois , à Sand@ ham@ n .
NMT En-De	We really need to up our particular contribution in that regard .	Wir müssen wirklich unsere spezielle Hilfs@ leistung in dieser Hinsicht aufstocken .
NMT En-Fi	It is too early to see one system as a universal panacea and dismiss another .	Nyt on liian aikaista nostaa yksi järjestelmä jal@ usta@ lle ja antaa jollekin toiselle huono arvo@ sana .
SkipThought	the old sami was gone , and he was a different person now .	the new sami didn 't mind standing barefoot in dirty white , sans ra@ y-@ bans and without beautiful women following his every move .
Seq2Tree	Dikoya is a village in Sri Lanka .	(ROOT (S (NP NNP)NP (VP VBZ (NP (NP DT NN)NP (PP IN (NP NNP NNP)NP)PP)NP)VP .)S)ROOT

Table 1: **Source and target examples for seq2seq training tasks.**

4 Probing task experiments

Baselines Baseline and human-bound performance are reported in the top block of Table 2. **Length** is a linear classifier with sentence length as sole feature. **NB-uni-tfidf** is a Naive Bayes classifier using words' tfidf scores as features, **NB-bi-tfidf** its extension to bigrams. Finally, **BoV-fastText** derives sentence representations by averaging the fastText embeddings of the words they contain (same embeddings used as input to the encoders).[3]

Except, trivially, for Length on SentLen and the NB baselines on WC, there is a healthy gap between top baseline performance and human upper bounds. NB-uni-tfidf evaluates to what extent our tasks can be addressed solely based on knowledge about the distribution of words in the training sentences. Words are of course to some extent informative for most tasks, leading to relatively high performance in Tense, SubjNum and ObjNum. Recall that the words containing the probed features are disjoint between train and test partitions, so we are not observing a confound here, but rather the effect of the redundancies one expects in natural language data. For example, for Tense, since sentences often contain more than one verb in the same tense, NB-uni-tfidf can exploit non-target verbs as cues: the NB features most associated to the past class are verbs in the past tense (e.g "sensed", "lied", "announced"), and similarly for present (e.g "uses", "chuckles", "frowns"). Using bigram features (NB-bi-tfidf) brings in general little or no improvement with respect to the unigram baseline, except, trivially, for the BShift task, where NB-bi-tfidf can easily detect unlikely bigrams. NB-bi-tfidf has below-random performance on SOMO, confirming that the semantic intruder is not given away by superficial bigram cues.

[3]Similar results are obtained summing embeddings, and using GloVe embeddings (Pennington et al., 2014).

Our first striking result is the good overall performance of Bag-of-Vectors, confirming early insights that aggregated word embeddings capture surprising amounts of sentence information (Pham et al., 2015; Arora et al., 2017; Adi et al., 2017). BoV's good WC and SentLen performance was already established by Adi et al. (2017). Not surprisingly, word-order-unaware BoV performs randomly in BShift and in the more sophisticated semantic tasks SOMO and CoordInv. More interestingly, BoV is very good at the Tense, SubjNum, ObjNum, and TopConst tasks (much better than the word-based baselines), and well above chance in TreeDepth. The good performance on Tense, SubjNum and ObjNum has a straightforward explanation we have already hinted at above. Many sentences are naturally "redundant", in the sense that most tensed verbs in a sentence are in the same tense, and similarly for number in nouns. In 95.2% Tense, 75.9% SubjNum and 78.7% ObjNum test sentences, the target tense/number feature is also the majority one for the whole sentence. Word embeddings capture features such as number and tense (Mikolov et al., 2013), so aggregated word embeddings will naturally track these features' majority values in a sentence. BoV's TopConst and TreeDepth performance is more surprising. Accuracy is well above NB, showing that BoV is exploiting cues beyond specific words strongly associated to the target classes. We conjecture that more abstract word features captured

Task	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Baseline representations										
Majority vote	20.0	0.5	17.9	5.0	50.0	50.0	50.0	50.0	50.0	50.0
Hum. Eval.	100	100	84.0	84.0	98.0	85.0	88.0	86.5	81.2	85.0
Length	**100**	0.2	18.1	9.3	50.6	56.5	50.3	50.1	50.2	50.0
NB-uni-tfidf	22.7	**97.8**	24.1	41.9	49.5	77.7	68.9	64.0	38.0	50.5
NB-bi-tfidf	23.0	95.0	24.6	53.0	**63.8**	75.9	69.1	65.4	39.9	**55.7**
BoV-fastText	66.6	91.6	**37.1**	**68.1**	50.8	**89.1**	**82.1**	**79.8**	**54.2**	54.8
BiLSTM-last encoder										
Untrained	36.7	43.8	28.5	76.3	49.8	84.9	84.7	74.7	51.1	64.3
AutoEncoder	**99.3**	23.3	35.6	78.2	62.0	84.3	84.7	82.1	49.9	65.1
NMT En-Fr	83.5	**55.6**	42.4	81.6	62.3	88.1	89.7	89.5	52.0	71.2
NMT En-De	83.8	53.1	42.1	81.8	60.6	88.6	89.3	87.3	51.5	**71.3**
NMT En-Fi	82.4	52.6	40.8	81.3	58.8	88.4	86.8	85.3	52.1	71.0
Seq2Tree	94.0	14.0	**59.6**	**89.4**	**78.6**	**89.9**	**94.4**	**94.7**	49.6	67.8
SkipThought	68.1	35.9	33.5	75.4	60.1	89.1	80.5	77.1	**55.6**	67.7
NLI	75.9	47.3	32.7	70.5	54.5	79.7	79.3	71.3	53.3	66.5
BiLSTM-max encoder										
Untrained	73.3	**88.8**	46.2	71.8	70.6	89.2	85.8	81.9	73.3	68.3
AutoEncoder	**99.1**	17.5	45.5	74.9	71.9	86.4	87.0	83.5	73.4	71.7
NMT En-Fr	80.1	58.3	51.7	81.9	73.7	89.5	90.3	89.1	73.2	75.4
NMT En-De	79.9	56.0	52.3	82.2	72.1	90.5	90.9	89.5	73.4	**76.2**
NMT En-Fi	78.5	58.3	50.9	82.5	71.7	90.0	90.3	88.0	73.2	75.4
Seq2Tree	93.3	10.3	**63.8**	**89.6**	**82.1**	**90.9**	**95.1**	**95.1**	73.2	71.9
SkipThought	66.0	35.7	44.6	72.5	73.8	90.3	85.0	80.6	**73.6**	71.0
NLI	71.7	87.3	41.6	70.5	65.1	86.7	80.7	80.3	62.1	66.8
GatedConvNet encoder										
Untrained	90.3	17.1	30.3	47.5	62.0	78.2	72.2	70.9	61.4	59.6
AutoEncoder	**99.4**	16.8	46.3	75.2	71.9	87.7	88.5	86.5	**73.5**	72.4
NMT En-Fr	84.8	41.3	44.6	77.6	67.9	87.9	88.8	86.6	66.1	72.0
NMT En-De	89.6	49.0	50.5	81.7	72.3	90.4	91.4	89.7	72.8	**75.1**
NMT En-Fi	89.3	**51.5**	49.6	81.8	70.9	90.4	90.9	89.4	72.4	**75.1**
Seq2Tree	96.5	8.7	**62.0**	**88.9**	**83.6**	**91.5**	**94.5**	**94.3**	**73.5**	73.8
SkipThought	79.1	48.4	45.7	79.2	73.4	90.7	86.6	81.7	72.4	72.3
NLI	73.8	29.2	43.2	63.9	70.7	81.3	77.5	74.4	73.3	71.0

Table 2: **Probing task accuracies.** Classification performed by a MLP with sigmoid nonlinearity, taking pre-learned sentence embeddings as input (see Appendix for details and logistic regression results).

by the embeddings (such as the part of speech of a word) might signal different syntactic structures. For example, sentences in the "WHNP SQ ." top constituent class (e.g., "*How* long before you leave us again?") must contain a wh word, and will often feature an auxiliary or modal verb. BoV can rely on this information to noisily predict the correct class.

Encoding architectures Comfortingly, proper encoding architectures clearly outperform BoV. An interesting observation in Table 2 is that different encoder architectures trained with the same objective, and achieving similar performance on the training task,[4] can lead to linguistically different embeddings, as indicated by the probing tasks. Coherently with the findings of Conneau et al. (2017) for the downstream tasks, this suggests that the prior imposed by the encoder architecture strongly preconditions the nature of the embeddings. Complementing recent evidence that convolutional architectures are on a par with recurrent ones in seq2seq tasks (Gehring et al., 2017), we find that Gated ConvNet's overall probing task performance is comparable to that of the best LSTM architecture (although, as shown in Appendix, the LSTM has a slight edge on downstream tasks). We also replicate the finding of Conneau et al. (2017) that BiLSTM-max outperforms BiLSTM-last both in the downstream tasks (see Appendix) and in the probing tasks (Table 2). Interestingly, the latter only outperforms the former in SentLen, a task that captures a superficial aspect of sentences (how many words they contain), that could get in the way of inducing more useful linguistic knowledge.

[4]See Appendix for details on training task performance.

Training tasks We focus next on how different training tasks affect BiLSTM-max, but the patterns are generally representative across architectures. NMT training leads to encoders that are more linguistically aware than those trained on the NLI data set, despite the fact that we confirm the finding of Conneau and colleagues that NLI is best for downstream tasks (Appendix). Perhaps, NMT captures richer linguistic features useful for the probing tasks, whereas shallower or more *ad-hoc* features might help more in our current downstream tasks. Suggestively, the one task where NLI clearly outperforms NMT is WC. Thus, NLI training is better at preserving shallower word features that might be more useful in downstream tasks (cf. Figure 2 and discussion there).

Unsupervised training (SkipThought and AutoEncoder) is not on a par with supervised tasks, but still effective. AutoEncoder training leads, unsurprisingly, to a model excelling at SentLen, but it attains low performance in the WC prediction task. This curious result might indicate that the latter information is stored in the embeddings in a complex way, not easily readable by our MLP. At the other end, Seq2Tree is trained to predict annotation from the same parser we used to create some of the probing tasks. Thus, its high performance on TopConst, Tense, SubjNum, ObjNum and TreeDepth is probably an artifact. Indeed, for most of these tasks, Seq2Tree performance is above the human bound, that is, Seq2Tree learned to mimic the parser errors in our benchmarks. For the more challenging SOMO and CoordInv tasks, that only indirectly rely on tagging/parsing information, Seq2Tree is comparable to NMT, that does not use explicit syntactic information.

Perhaps most interestingly, BiLSTM-max already achieves very good performance without any training (Untrained row in Table 2). Untrained BiLSTM-max also performs quite well in the downstream tasks (Appendix). This architecture must encode priors that are intrinsically good for sentence representations. Untrained BiLSTM-max exploits the input fastText embeddings, and multiplying the latter by a random recurrent matrix provides a form of positional encoding. However, good performance in a task such as SOMO, where BoV fails and positional information alone should not help (the intruder is randomly distributed across the sentence), suggests that other architectural biases are at work. Intriguingly, a preliminary comparison of untrained BiLSTM-max and human subjects on the SOMO sentences evaluated by both reveals that, whereas humans have a bias towards finding sentences acceptable (62% sentences are rated as untampered with, vs. 48% ground-truth proportion), the model has a strong bias in the opposite direction (it rates 83% of the sentences as modified). A cursory look at contrasting errors confirms, unsurprisingly, that those made by humans are perfectly justified, while model errors are opaque. For example, the sentence "I didn't come here to *reunite* (orig. *undermine*) you" seems perfectly acceptable in its modified form, and indeed subjects judged it as such, whereas untrained BiLSTM-max "correctly" rated it as a modified item. Conversely, it is difficult to see any clear reason for the latter tendency to rate perfectly acceptable originals as modified. We leave a more thorough investigation to further work. See similar observations on the effectiveness of untrained ConvNets in vision by Ulyanov et al. (2017).

Probing task comparison A good encoder, such as NMT-trained BiLSTM-max, shows generally good performance across probing tasks. At one extreme, performance is not particularly high on the surface tasks, which might be an indirect sign of the encoder extracting "deeper" linguistic properties. At the other end, performance is still far from the human bounds on TreeDepth, BShift, SOMO and CoordInv. The last 3 tasks ask if a sentence is syntactically or semantically anomalous. This is a daunting job for an encoder that has not been explicitly trained on acceptability, and it is interesting that the best models are, at least to a certain extent, able to produce reasonable anomaly judgments. The asymmetry between the difficult TreeDepth and easier TopConst is also interesting. Intuitively, TreeDepth requires more nuanced syntactic information (down to the deepest leaf of the tree) than TopConst, that only requires identifying broad chunks.

Figure 1 reports how probing task accuracy changes in function of encoder training epochs. The figure shows that NMT probing performance is largely independent of target language, with strikingly similar development patterns across French, German and Finnish. Note in particular the similar probing accuracy curves in French and Finnish, while the corresponding BLEU scores (in lavender) are consistently higher in the former lan-

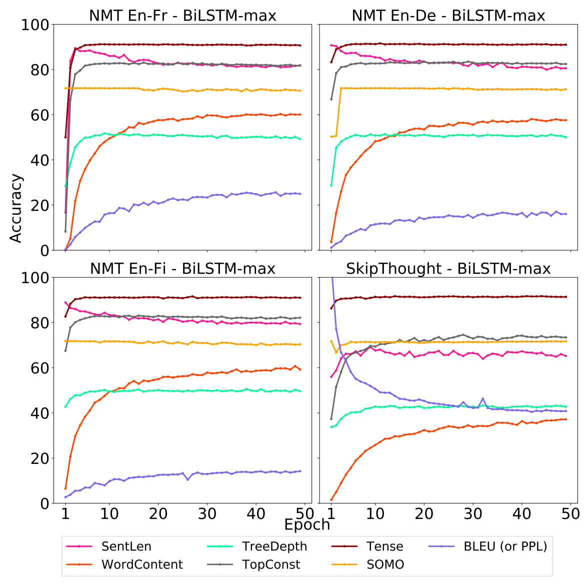


Figure 1: **Probing task scores after each training epoch, for NMT and SkipThought.** We also report training score evolution: BLEU for NMT; perplexity (PPL) for SkipThought.

guage. For both NMT and SkipThought, WC performance keeps increasing with epochs. For the other tasks, we observe instead an early flattening of the NMT probing curves, while BLEU performance keeps increasing. Most strikingly, SentLen performance is actually *decreasing*, suggesting again that, as a model captures deeper linguistic properties, it will tend to forget about this superficial feature. Finally, for the challenging SOMO task, the curves are mostly flat, suggesting that what BiLSTM-max is able to capture about this task is already encoded in its architecture, and further training doesn't help much.

Probing vs. downstream tasks Figure 2 reports correlation between performance on our probing tasks and the downstream tasks available in the SentEval[5] suite, which consists of classification (MR, CR, SUBJ, MPQA, SST2, SST5, TREC), natural language inference (SICK-E), semantic relatedness (SICK-R, STSB), paraphrase detection (MRPC) and semantic textual similarity (STS 2012 to 2017) tasks. Strikingly, WC is significantly positively correlated with all downstream tasks. This suggests that, at least for current models, the latter do not require extracting particularly abstract knowledge from the data. Just relying on the *words* contained in the input sentences can get you a long way. Conversely, there is a significant negative correlation between SentLen and most downstream tasks. The number of words in a sentence is not informative about its linguistic contents. The more models abstract away from such information, the more likely it is they will use their capacity to capture more interesting features, as the decrease of the SentLen curve along training (see Figure 1) also suggests. CoordInv and, especially, SOMO, the tasks requiring the most sophisticated semantic knowledge, are those that positively correlate with the largest number of downstream tasks after WC. We observe intriguing asymmetries: SOMO correlates with the SICK-E sentence entailment test, but not with SICK-R, which is about modeling sentence relatedness intuitions. Indeed, logical entailment requires deeper semantic analysis than modeling similarity judgments. TopConst and the number tasks negatively correlate with various similarity and sentiment data sets (SST, STS, SICK-R). This might expose biases in these tasks: SICK-R, for example, deliberately contains sentence pairs with opposite voice, that will have different constituent structure but equal meaning (Marelli et al., 2014). It might also mirrors genuine factors affecting similarity judgments (e.g., two sentences differing only in object number are very similar). Remarkably, TREC question type classification is the downstream task correlating with most probing tasks. Question classification is certainly an outlier among our downstream tasks, but we must leave a full understanding of this behaviour to future work (this is exactly the sort of analysis our probing tasks should stimulate).

5 Related work

Adi et al. (2017) introduced SentLen, WC and a word order test, focusing on a bag-of-vectors baseline, an autoencoder and skip-thought (all trained on the same data used for the probing tasks). We recast their tasks so that they only require a sentence embedding as input (two of their tasks also require word embeddings, polluting sentence-level evaluation), we extend the evaluation to more tasks, encoders and training objectives, and we relate performance on the probing tasks with that on downstream tasks. Shi et al. (2016) also use 3 probing tasks, including Tense and TopConst. It is not clear that they controlled for the same factors we considered (in particular, lexical overlap and

[5] https://github.com/facebookresearch/SentEval

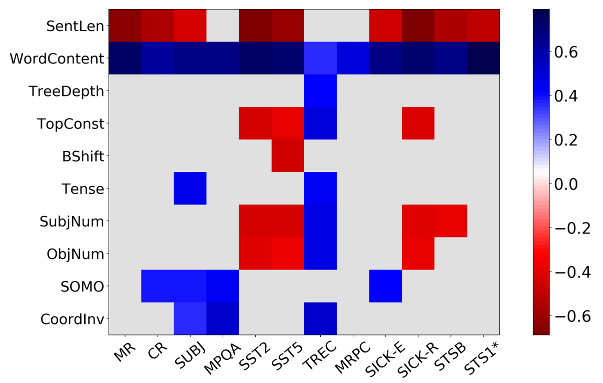


Figure 2: **Spearman correlation matrix between probing and downstream tasks.** Correlations based on all sentence embeddings we investigated (more than 40). Cells in gray denote task pairs that are not significantly correlated (after correcting for multiple comparisons).

sentence length), and they use much smaller training sets, limiting classifier-based evaluation to logistic regression. Moreover, they test a smaller set of models, focusing on machine translation.

Belinkov et al. (2017a), Belinkov et al. (2017b) and Dalvi et al. (2017) are also interested in understanding the type of linguistic knowledge encoded in sentence and word embeddings, but their focus is on word-level morphosyntax and lexical semantics, and specifically on NMT encoders and decoders. Sennrich (2017) also focuses on NMT systems, and proposes a contrastive test to assess how they handle various linguistic phenomena. Other work explores the linguistic behaviour of recurrent networks and related models by using visualization, input/hidden representation deletion techniques or by looking at the word-by-word behaviour of the network (e.g., Nagamine et al., 2015; Hupkes et al., 2017; Li et al., 2016; Linzen et al., 2016; Kàdàr et al., 2017; Li et al., 2017). These methods, complementary to ours, are not agnostic to encoder architecture, and cannot be used for general-purpose cross-model evaluation.

Finally, Conneau et al. (2017) propose a large-scale, multi-task evaluation of sentence embeddings, focusing entirely on downstream tasks.

6 Conclusion

We introduced a set of tasks probing the linguistic knowledge of sentence embedding methods. Their purpose is not to encourage the development of *ad-hoc* models that attain top performance on them, but to help exploring what information is captured by different pre-trained encoders.

We performed an extensive linguistic evaluation of modern sentence encoders. Our results suggest that the encoders are capturing a wide range of properties, well above those captured by a set of strong baselines. We further uncovered interesting patterns of correlation between the probing tasks and more complex "downstream" tasks, and presented a set of intriguing findings about the linguistic properties of various embedding methods. For example, we found that Bag-of-Vectors is surprisingly good at capturing sentence-level properties, thanks to redundancies in natural linguistic input. We showed that different encoder architectures trained with the same objective with similar performance can result in different embeddings, pointing out the importance of the architecture prior for sentence embeddings. In particular, we found that BiLSTM-max embeddings are already capturing interesting linguistic knowledge before training, and that, after training, they detect semantic acceptability without having been exposed to anomalous sentences before. We hope that our publicly available probing task set will become a standard benchmarking tool of the linguistic properties of new encoders, and that it will stir research towards a better understanding of what they learn.

In future work, we would like to extend the probing tasks to other languages (which should be relatively easy, given that they are automatically generated), investigate how multi-task training affects probing task performance and leverage our probing tasks to find more linguistically-aware universal encoders.

Acknowledgments

We thank David Lopez-Paz, Holger Schwenk, Hervé Jégou, Marc'Aurelio Ranzato and Douwe Kiela for useful comments and discussions.

References

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

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of ICLR Conference Track*. Toulon, France. Published online: https://openreview.net/group?id=ICLR.cc/2017/conference.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Advances in neural information processing systems (NIPS)* .

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of ACL*. Vancouver, Canada, pages 861–872.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of IJCNLP*. Taipei, Taiwan, pages 1–10.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of EMNLP* .

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*. ACM, pages 160–167.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*. Copenhagen, Denmark, pages 670–680.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of IJCNLP*. Taipei, Taiwan, pages 142–151.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. *Proceedings of the 34th International Conference on Machine Learning* .

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of ICML*. Sydney, Australia, pages 1243–1252.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2017. Visualisation and diagnostic classifiers reveal how recurrent and recursive neural networks process hierarchical structure. http://arxiv.org/abs/1711.10203.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *Proceedings of ECCV*. Amsterdam, the Netherlands, pages 727–739.

Àkos Kàdàr, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics* 43(4):761–780.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*. Sapporo, Japan, pages 423–430.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of SemEval*. Dublin, Ireland, pages 329–334.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of NAACL*. San Diego, CA, pages 681–691.

Jiwei Li, Monroe Will, and Dan Jurafsky. 2017. Efficient estimation of word representations in vector space. https://arxiv.org/abs/1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*. Rekjavik, Iceland, pages 216–223.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of LREC*. Miyazaki, Japan.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*. Atlanta, Georgia, pages 746–751.

Tasha Nagamine, Michael L. Seltzer, and Nima Mesgarani. 2015. Exploring how deep neural networks form phonemic categories. In *Proceedings of INTERSPEECH*. Dresden, Germany, pages 1912–1916.

Matthew Nelson, Imen El Karoui, Kristof Giber, Xiaofang Yang, Laurent Cohen, Hilda Koopman, Sydney Cash, Lionel Naccache, John Hale, Christophe Pallier, and Stanislas Dehaene. 2017. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences* 114(18):E3669–E3678.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*. Barcelona, Spain, pages 271–278.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of ACL*. Berlin, Germany, pages 1525–1534.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. Doha, Qatar, pages 1532–1543.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In *Proceedings of ACL*. Beijing, China, pages 971–981.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of EACL (Short Papers)*. Valencia, Spain, pages 376–382.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of EMNLP*. Austin, Texas, pages 1526–1534.

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*. Granada, Spain, pages 801–809.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*. Montreal, Canada, pages 3104–3112.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R de Sa. 2017. Trimming and improving skip-thought vectors. *Proceedings of the 2nd Workshop on Representation Learning for NLP* .

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. Deep image prior. https://arxiv.org/abs/1711.10925.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199* .

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*. Santiago, Chile, pages 19–27.

Robust Distant Supervision Relation Extraction via Deep Reinforcement Learning

Pengda Qin[♯], **Weiran Xu**[♯], **William Yang Wang**[♭]
[♯]Beijing University of Posts and Telecommunications, China
[♭]University of California, Santa Barbara, USA
`{qinpengda, xuweiran}@bupt.edu.cn`
`{william}@cs.ucsb.edu`

Abstract

Distant supervision has become the standard method for relation extraction. However, even though it is an efficient method, it does not come at no cost—The resulted distantly-supervised training samples are often very noisy. To combat the noise, most of the recent state-of-the-art approaches focus on selecting one-best sentence or calculating soft attention weights over the set of the sentences of one specific entity pair. However, these methods are suboptimal, and the **false positive** problem is still a key stumbling bottleneck for the performance. We argue that those incorrectly-labeled candidate sentences must be treated with a hard decision, rather than being dealt with soft attention weights. To do this, our paper describes a radical solution—We explore a deep reinforcement learning strategy to generate the false-positive indicator, where we automatically recognize false positives for each relation type without any supervised information. Unlike the removal operation in the previous studies, we redistribute them into the negative examples. The experimental results show that the proposed strategy significantly improves the performance of distant supervision comparing to state-of-the-art systems.

1 Introduction

Relation extraction is a core task in information extraction and natural language understanding. The goal of relation extraction is to predict relations for entities in a sentence (Zelenko et al., 2003; Bunescu and Mooney, 2005; GuoDong et al., 2005). For example, given a sentence *"**Barack Obama** is married to **Michelle Obama**."*, a relation classifier aims at predicting the relation of *"**spouse**"*. In downstream applications, relation extraction is the key module for constructing knowledge graphs, and it is a vital component of many natural language processing applications such as structured search, sentiment analysis, question answering, and summarization.

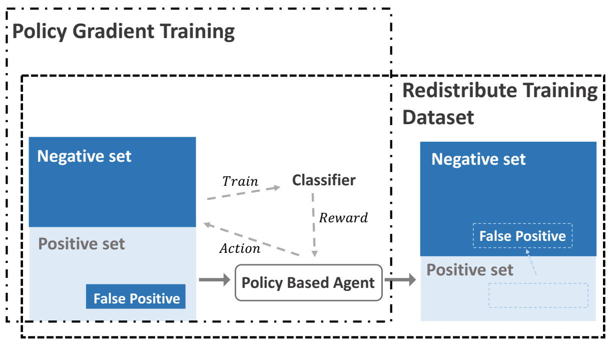


Figure 1: Our deep reinforcement learning framework aims at dynamically recognizing false positive samples, and moving them from the positive set to the negative set during distant supervision.

A major issue encountered in the early development of relation extraction algorithms is the data sparsity issue—It is extremely expensive, and almost impossible for human annotators to go through a large corpus of millions of sentences to provide a large amount of labeled training instances. Therefore, distant supervision relation extraction (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012) becomes popular, because it uses entity pairs from knowledge bases to select a set of noisy instances from unlabeled data. In recent years, neural network approaches (Zeng et al., 2014, 2015) have been proposed to train the relation extractor under these noisy conditions. To suppress the noisy(Roth et al., 2013), recent studies (Lin et al., 2016) have proposed the use of attention mechanisms to place soft weights on a set of noisy sentences, and select samples. However, we argue that only selecting one example or based on soft attention weights are not the optimal strategy: To improve the robustness, we need a systematic solution to make use of more instances, while removing false positives and placing them in the right place.

In this paper, we investigate the possibility of using dynamic selection strategies for robust distant supervision. More specifically, we design a deep reinforcement learning agent, whose goal is to learn to choose whether to remove or remain the distantly supervised candidate instance based on the performance change of the relation classifier. Intuitively, our agent would like to remove false positives, and reconstruct a cleaned set of distantly supervised instances to maximize the reward based on the classification accuracy. Our proposed method is classifier-independent, and it can be applied to any existing distant supervision model. Empirically, we show that our method has brought consistent performance gains in various deep neural network based models, achieving strong performances on the widely used New York Times dataset (Riedel et al., 2010). Our contributions are three-fold:

- We propose a novel deep reinforcement learning framework for robust distant supervision relation extraction.
- Our method is model-independent, meaning that it could be applied to any state-of-the-art relation extractors.
- We show that our method can boost the performances of recently proposed neural relation extractors.

In Section 2, we will discuss related works on distant supervision relation extraction. Next, we will describe our robust distant supervision framework in Section 3. In Section 4, empirical evaluation results are shown. And finally, we conclude in Section 5.

2 Related Work

Mintz et al. (2009) is the first study that combines dependency path and feature aggregation for distant supervision. However, this approach would introduce a lot of false positives, as the same entity pair might have multiple relations. To alleviate this issue, Hoffmann et al. (2011) address this issue, and propose a model to jointly learn with multiple relations. Surdeanu et al. (2012) further propose a multi-instance multi-label learning framework to improve the performance. Note that these early approaches do not explicitly remove noisy instances, but rather hope that the model would be able to suppress the noise.

Recently, with the advance of neural network techniques, deep learning methods (Zeng et al., 2014, 2015) are introduced, and the hope is to model noisy distant supervision process in the hidden layers. However, their approach only selects one most plausible instance per entity pair, inevitably missing out a lot of valuable training instances. Recently, Lin et al. (2016) propose an attention mechanism to select plausible instances from a set of noisy instances. However, we believe that soft attention weight assignment might not be the optimal solution, since the false positives should be completely removed and placed in the negative set. Ji et al. (2017) combine the external knowledge to rich the representation of entity pair, in which way to improve the accuracy of attention weights. Even though these above-mentioned methods can select high-quality instances, they ignore the false positive case: all the sentences of one entity pair belongs to the false positives. In this work, we take a radical approach to solve this problem—We will make use of the distantly labeled resources as much as possible, while learning a independent false-positive indicator to remove false positives, and place them in the right place. After our ACL submission, we notice that a contemporaneous study Feng et al. (2018) also adopts reinforcement learning to learn an instance selector, but their reward is calculated from the prediction probabilities. In contrast, while in our method, the reward is intuitively reflected by the performance change of the relation classifier. Our approach is also complement to most of the approaches above, and can be directly applied on top of any existing relation extraction classifiers.

3 Reinforcement Learning for Distant Supervision

We introduce a performance-driven, policy-based reinforcement learning method to heuristically recognize false positive samples. Comparing to a prior study that has underutilized the distantly-supervised samples (Lin et al., 2016), we consider an RL agent for robust distant supervision relation extraction. We first describe the definitions of our RL method, including the policy-based agent, external environment, and pre-training strategy. Next, we describe the retraining strategy for our RL agent. The goal of our agent is to determine whether to **retain** or **remove** a distantly-supervised sentence, based on the performance change of relation classifier. Finally, we describe the noisy-suppression method, where we teach our policy-based agent to make a redistribution for a cleaner distant supervision training dataset.

Distant supervision relation extraction is to predict the relation type of entity pair under the automatically-generated training set. However, the issue is that these distantly-supervised sentences that mention this entity pair may not express the desired relation type. Therefore, what our RL agent should do is to determine whether the distantly-supervised sentence is a true positive instance for this relation type. For reinforcement learning, external environment and RL agent are two necessary components, and a robust agent is trained from the dynamic interaction between these two parts (Arulkumaran et al., 2017). First, the prerequisite of reinforcement learning is that the external environment should be modeled as a Markov decision process (MDP). However, the traditional setting of relation extraction cannot satisfy this condition: the input sentences are independent of each other. In other words, we cannot merely use the information of the sentence being processed as the state. Thus, we add the information from the early states into the representation of the current state, in which way to model our task as a MDP problem (Fang et al., 2017). The other component, RL agent, is parameterized with a policy network $\pi_\theta(s, a) = p(a|s; \theta)$. The probability distribution of actions $\mathcal{A} = \{a_{remove}, a_{remain}\}$ is calculated by policy network based on state vectors. What needs to be noted is that, Deep Q Network (DQN) (Mnih et al., 2013) is also a widely-used RL method; however, it is not suitable for our case, even if our action space is small. First, we cannot compute the immediate reward for every operation; In contrast, the accurate reward can only be obtained after finishing processing the whole training dataset. Second, the stochastic policy of the policy network is capable of preventing the agent from getting stuck in an intermediate state. The following subsections detailedly introduce the definitions of the fundamental components in the proposed RL method.

States In order to satisfy the condition of MDP, the state s includes the information from the current sentence and the sentences that have been removed in early states. The semantic and syntactic information of sentence is represented by a continuous real-valued vector. According to some state-of-the-art supervised relation extraction approaches (Zeng et al., 2014; Nguyen and Grishman, 2015), we utilize both word embedding and position embedding to convert sentence into vector. With this sentence vector, the current state is the concatenation of the current sentence vector and the average vector of the removed sentences in early states. We give relatively larger weight for the vector of the current sentence, in which way to magnify the dominating influence of the current sentence information for the decision of action.

Actions At each step, our agent is required to determine whether the instance is false positive for target relation type. Each relation type has a agent[1]. There are two actions for each agent: whether to remove or retain the current instance from the training set. With the initial distantly-supervised dataset that is blended with incorrectly-labeled instances, we hope that our agent is capable of using the policy network to filter noisy instances; Under this cleaned dataset, distant supervision is then expected to achieve better performance.

Rewards As previously mentioned, the intuition of our model is that, when the incorrectly-labeled instances are filtered, the better performance of relation classifier will achieve. Therefore, we use the change of performance as the result-driven reward for a series of actions decided by the agent. Compared to accuracy, we adopt the F_1 score as the evaluation criterion, since accuracy might not be an indicative metric in a multi-class classification setting where the data distribution could be imbalanced. Thus, the reward can be formulated as the

[1]We also tried the strategy that just builds a single agent for all relation types: a binary classifier(TP/FP) or a multi-class classifier(rela1/rela2/.../FP). But, it has the limitation in the performance. We found that our one-agent-for-one-relation strategy obtained better performance than the single agent strategy.

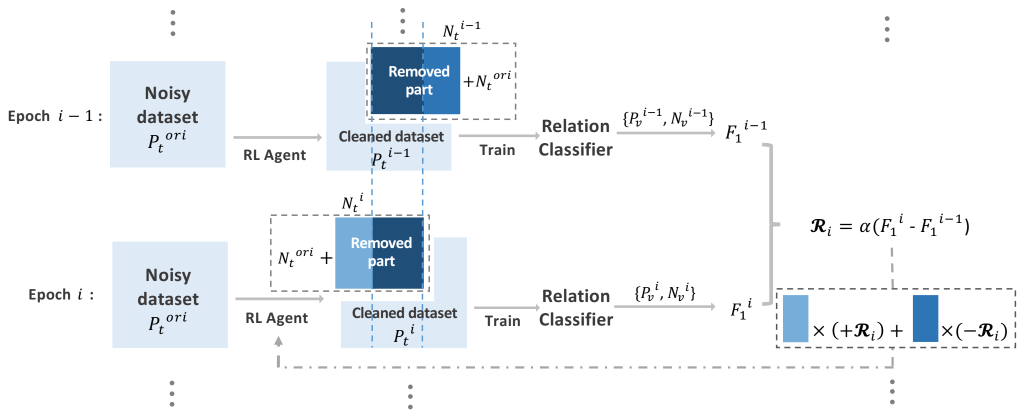


Figure 2: The proposed policy-based reinforcement learning framework. The agent tries to remove the wrong-labeled sentences from the distantly-supervised positive dataset P^{ori}. In order to calculate the reward, P^{ori} is split into the training part P_t^{ori} and the validation part P_v^{ori}; their corresponding negative part are represented as N_t^{ori} and N_v^{ori}. In each epoch i, the agent performs a series of actions to recognize the false positive samples from P_t^{ori} and treat them as negative samples. Then, a new relation classifier is trained under the new dataset $\{P_t^i, N_t^i\}$. With this relation classifier, F_1 score is calculated from the new validation set $\{P_v^i, N_v^i\}$, where P_v^i is also filtered by the current agent. After that, the current reward is measured as the difference of F_1 between the adjacent epochs.

difference between the adjacent epochs:

$$R_i = \alpha(F_1^i - F_1^{i-1}) \tag{1}$$

As this equation shows, in step i, our agent is given a positive reward only if F_1 gets improved; otherwise, the agent will receive a negative reward. Under this setting, the value of reward is proportional to the difference of F_1, and α is used to convert this difference into a rational numeric range. Naturally, the value of the reward is in a continuous space, which is more reasonable than a binary reward (-1 and 1), because this setting can reflect the number of wrong-labeled instance that the agent has removed. In order to avoid the randomness of F_1, we use the average F_1 of last five epochs to calculate the reward.

Policy Network For each input sentence, our policy network is to determine whether it expresses the target relation type and then make removal action if it is irrelevant to the target relation type. Thus, it is analogous to a binary relation classifier. CNN is commonly used to construct relation classification system (Santos et al., 2015; Xu et al., 2015; Shen and Huang, 2016), so we adopt a simple CNN with window size c_w and kernel size c_k, to model policy network $\pi(s;\theta)$. The reason why we do not choice the variants of CNN (Zeng et al., 2015; Lin et al., 2016) that are well-designed for distant supervision is that these two models belong to bag-level models (dealing with a bag of sentences simultaneously) and deal with the multi-classification problem; We just need a model to do binary sentence-level classification. Naturally, the simpler network is adopted.

3.1 Training Policy-based Agent

Unlike the goal of distant supervision relation extraction, our agent is to determine whether an annotated sentence expresses the target relation type rather than predict the relationship of entity pair, so sentences are treated independently despite belonging to the same entity pair. In distant supervision training dataset, one relation type contains several thousands or ten thousands sentences; moreover, reward R can only be calculated after processing the whole positive set of this relation type. If we randomly initialize the parameters of policy network and train this network by trial and errors, it will waste a lot of time and be inclined to poor convergence properties. In order to overcome this problem, we adopt a supervised learning procedure to pre-train our policy network, in which way to provide a general learning direction for our policy-based agent.

3.1.1 Pre-training Strategy

The pre-training strategy, inspired from *AlphaGo* (Silver et al., 2016), is a common strategy in RL related works to accelerate the training of RL agents. Normally, they utilize a small part of the annotated dataset to train policy networks before reinforcement learning. For example, *AlphaGo* uses the collected experts moves to do a supervised learning for *Go* RL agent. However, in distant supervision relation extraction task, there is not any supervised information that can be used unless let linguistic experts to do some manual annotations for part of the entity pairs. However, this is expensive, and it is not the original intention of distant supervision. Under this circumstance, we propose a compromised solution. With well-aligned corpus, the true positive samples should have evident advantage in quantity compared with false positive samples in the distantly-supervised dataset. So, for a specific relation type, we directly treat the distantly-supervised positive set as the positive set, and randomly extract part of distantly-supervised negative set as the negative set. In order to better consider prior information during this pre-training procedure, the amount of negative samples is 10 times of the number of positive samples. It is because, when learning with massive negative samples, the agent is more likely to develop toward a better direction. Cross-entropy cost function is used to train this binary classifier, where the negative label corresponds to the removing action, and the positive label corresponds to the retaining action.

$$\begin{aligned} J(\theta) = \sum_i & y_i \log[\pi(a = y_i|s_i;\theta)] \\ & + (1 - y_i)\log[1 - \pi(a = y_i|s_i;\theta)] \end{aligned} \tag{2}$$

Due to the noisy nature of the distantly-labeled instances, if we let this pre-training process overfit this noisy dataset, the predicted probabilities of most samples tend to be close to 0 or 1, which is difficult to be corrected and unnecessarily increases the training cost of reinforcement learning. So, we stop this training process when the accuracy reaches $85\% \sim 90\%$. Theoretically, our approach can be explained as increasing the entropy of the policy gradient agent, and preventing the entropy of the policy being too low, which means that the lack of exploration may be a concern.

3.1.2 Retraining Agent with Rewards

As shown in Figure 2, in order to discover incorrectly-labeled instances without any supervised information, we introduce a policy-based RL method. What our agent tries to deal with is the noisy samples from the distantly-supervised positive dataset; Here we call it as the *DS positive dataset*. We split it into the training positive set P_t^{ori} and the validation positive set P_v^{ori}; naturally, both of these two set are noisy. Correspondingly, the training negative set N_t^{ori} and the validation negative set N_v^{ori} are constructed by randomly selected from the DS negative dataset. In every epoch, the agent removes a noisy sample set Ψ_i from P_t^{ori} according to the stochastic policy $\pi(a|s)$, and we obtain a new positive set $P_t = P_t^{ori} - \Psi_i$. Because Ψ_i is recognized as the wrong-labeled samples, we redistribute it into the negative set $N_t = N_t^{ori} + \Psi_i$. Under this setting, the scale of training set is constant for each epoch. Now we utilize the cleaned data $\{P_t, N_t\}$ to train a relation classifier. The desirable situation is that RL agent has the capacity to increase the performance of relation classifier through relocating incorrectly-labeled false positive instances. Therefore, we use the validation set $\{P_v^{ori}, N_v^{ori}\}$ to measure the performance of the current agent. First, this validation set is filtered and redistributed by the current agent as $\{P_v, N_v\}$; the F_1 score of the current relation classifier is calculated from it. Finally, the difference of F_1 scores between the current and previous epoch is used to calculate reward. Next, we will introduce several strategies to train a more robust RL agent.

Removing the fixed number of sentences in each epoch In every epoch, we let the RL agent to remove a fixed number of sentences or less (when the number of the removed sentences in one epoch does not reach this fixed number during training), in which way to prevent the case that the agent tries to remove more false positive instances by removing more instances. Under the restriction of fixed number, if the agent decides to remove the current state, it means the chance of removing other states decrease. Therefore, in order to obtain a better reward, the agent should try to remove a instance set that includes more negative instances.

Loss function The quality of the RL agent is reflected by the quality of the removed part. After the pre-training process, the agent just possesses

Algorithm 1 Retraining agent with rewards for relation k. For a clearer expression, k is omitted in the following algorithm.

```
Require: Positive set $\{P_t^{ori}, P_v^{ori}\}$, Negative set $\{N_t^{ori}, N_v^{ori}\}$, the fixed number of removal $\gamma_t, \gamma_v$
 1: Load parameters $\theta$ from pre-trained policy network
 2: Initialize $s^*$ as the all-zero vector with the same dimension of $s_j$
 3: for epoch $i = 1 \to N$ do
 4:     for $s_j \in P_t^{ori}$ do
 5:         $\widetilde{s_j} = concatenation(s_j, s^*)$
 6:         Randomly sample $a_j \sim \pi(a|\widetilde{s_j};\theta)$; compute $p_j = \pi(a = 0|\widetilde{s_j};\theta)$
 7:         if $a_j == 0$ then
 8:             Save tuple $t_j = (\widetilde{s_j}, p_j)$ in $T$ and recompute the average vector of removed sentences $s^*$
 9:         end if
10:     end for
11:     Rank $T$ based on $p_j$ from high to low, obtain $T_{rank}$
12:     for $t_i$ in $T_{rank}[:\gamma_t]$ do
13:         Add $t_i[0]$ into $\Psi_i$
14:     end for
15:     $P_t^i = P_t^{ori} - \Psi_i, N_t^i = N_t^{ori} + \Psi_i$, and generate the new validation set $\{P_v^i, N_v^i\}$ with current agent
16:     Train the relation classifier based on $\{P_t^i, N_t^i\}$
17:     Calculate $F_1^i$ on the new validation set $\{P_v^i, N_v^i\}$, and Save $F_1^i$, $\Psi_i$
18:     $\mathcal{R} = \alpha(F_1^i - F_1^{i-1})$
19:     $\Omega_{i-1} = \Psi_{i-1} - \Psi_i \cap \Psi_{i-1}$; $\Omega_i = \Psi_i - \Psi_i \cap \Psi_{i-1}$
20:
21:     Updata $\theta$: $g \propto \triangledown_\theta \sum^{\Omega_i} \log \pi(a|s;\theta)\mathcal{R} + \triangledown_\theta \sum^{\Omega_{i-1}} \log \pi(a|s;\theta)(-\mathcal{R})$
22: end for
```

the ability to distinguish the obvious false positive instances, which means the discrimination of the indistinguishable wrong-labeled instances are still ambiguous. Particularly, this indistinguishable part is the criterion to reflect the quality of the agent. Therefore, regardless of these easy-distinguished instances, the different parts of the removed parts in different epochs are the determinant of the change of F_1 scores. Therefore, we definite two sets:

$$\Omega_{i-1} = \Psi_{i-1} - (\Psi_i \cap \Psi_{i-1}) \quad (3)$$

$$\Omega_i = \Psi_i - (\Psi_i \cap \Psi_{i-1}) \quad (4)$$

where Ψ_i is the removed part of epoch i. Ω_{i-1} and Ω_i are represented with the different colors in Figure 2. If F_1 score increases in the epoch i, it means the actions of the epoch i is more reasonable than that in the epoch $i-1$. In other words, Ω_i is more negative than Ω_{i-1}. Thus, we assign the positive reward to Ω_i and the negative reward to Ω_{i-1}, and vice versa. In summary, the ultimate loss function is formulated as follow:

$$\begin{aligned} J(\theta) = &\sum^{\Omega_i} \log \pi(a|s;\theta)R \\ &+ \sum^{\Omega_{i-1}} \log \pi(a|s;\theta)(-R) \end{aligned} \quad (5)$$

3.2 Redistributing Training Dataset with Policy-based Agents

Through the above reinforcement learning procedure, for each relation type, we obtain a agent as the false-positive indicator. These agents possess the capability of recognizing incorrectly-labeled instances of the corresponding relation types. We adopt these agents as classifiers to recognize false positive samples in the noisy distantly-supervised training dataset. For one entity pair, if all the sentence aligned from corpus are classified as false positive, then this entity pair is redistributed into the negative set.

4 Experiments

We adopt a policy-based RL method to generate a series of relation indicators and use them to redistribute training dataset by moving false positive samples to negative sample set. Therefore, our experiments are intended to demonstrate that our RL agents possess this capability.

4.1 Datast and Evaluation Metrics

We evaluate the proposed method on a commonly-used dataset[2], which is first presented in Riedel et al. (2010). This dataset is generated by aligning entity pairs from Freebase with New York Times corpus(NYT). Entity mentions of NYT corpus are recognized by the Stanford named entity recognizer (Finkel et al., 2005). The sentences from the years 2005-2006 are used as the training corpus and sentences from 2007 are used as the testing corpus. There are 52 actual relations and a special relation NA which indicates there is no relation between the head and tail entities. The sentences of NA are from the entity pairs that exist in the same sentence of the actual relations but do not appear in the Freebase.

Similar to the previous works, we adopt the held-out evaluation to evaluate our model, which can provide an approximate measure of the classification ability without costly human evaluation. Similar to the generation of the training set, the entity pairs in test set are also selected from Freebase, which will be predicted under the sentences discovered from the NYT corpus.

4.2 Experimental Settings

4.2.1 Policy-based Agent

The action space of our RL agent just includes two actions. Therefore, the agent can be modeled as a binary classifier. We adopt a single-window CNN as this policy network. The detailed hyperparameter settings are presented in Table 1. As for word embeddings, we directly use the word embedding file released by Lin et al. (2016)[3], which just keeps the words that appear more than 100 times in NYT. Moreover, we have the same dimension setting of the position embedding, and the maximum length of relative distance is -30 and 30 ("-" and "+" represent the left and right side of the entities). The learning rate of reinforcement learning is $2e^{-5}$. For each relation type, the fixed number γ_t, γ_v are according to the pre-trained agent. When one relation type has too many distant-supervised positive sentences (for example, /location/location/contains has 75768 sentences), we sample a subset of size 7,500 sentences to train the agent. For the average vector of the removed sentences, in the pre-training process and the first state of the retraining process, it is set as all-zero vector.

Hyperparameter	**Value**
Window size c_w	3
Kernel size c_k	100
Batch size	64
Regulator α	100

Table 1: Hyperparameter settings.

ID	**Relation**	**Original**	**Pretrain**	**RL**
1	/peo/per/pob	55.60	53.63	**55.74**
2	/peo/per/n	78.85	80.80	**83.63**
3	/peo/per/pl	86.65	89.62	**90.76**
4	/loc/loc/c	80.78	83.79	**85.39**
5	/loc/cou/ad	**90.9**	88.1	89.86
6	/bus/per/c	81.03	82.56	**84.22**
7	/loc/cou/c	88.10	93.78	**95.19**
8	/loc/adm/c	86.51	85.56	**86.63**
9	/loc/nei/n	96.51	97.20	**98.23**
10	/peo/dec/p	82.2	83.0	**84.6**

Table 2: Comparison of F_1 scores among three cases: the relation classifier is trained with the original dataset, the redistributed dataset generated by the pre-trained agent, and the redistributed dataset generated by our RL agent respectively. The name of relation types are abbreviated: $/peo/per/pob$ represents $/people/person/place\_of\_birth$

4.2.2 Relation Classifier for Calculating Reward

In order to evaluate a series of actions by agent, we use a simple CNN model, because the simple network is more sensitive to the quality of the training set. The proportion between P_t^{ori} and P_v^{ori} is 2:1, and they are all derived from the training set of Riedel dataset; the corresponding negative sample sets N_t^{ori} and N_v^{ori} are randomly selected from the Riedel negative dataset, whose size is twice that of their corresponding positive sets.

4.3 The Effectiveness of Reinforcement Learning

In Table 2, we list the F_1 scores before and after adopting the proposed RL method. Even though there are 52 actual relation types in Riedel dataset, only 10 relation types have more than 1000 pos-

[2]http://iesl.cs.umass.edu/riedel/ecml/

[3]https://github.com/thunlp/NRE

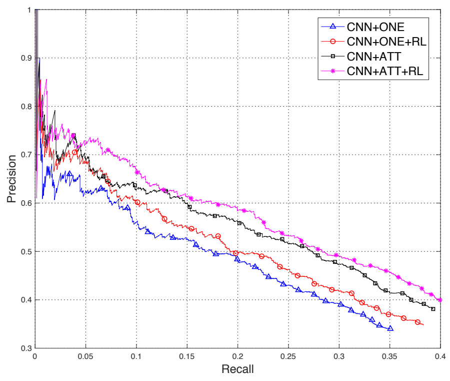


Figure 3: Aggregate PR curves of CNN˙based model.

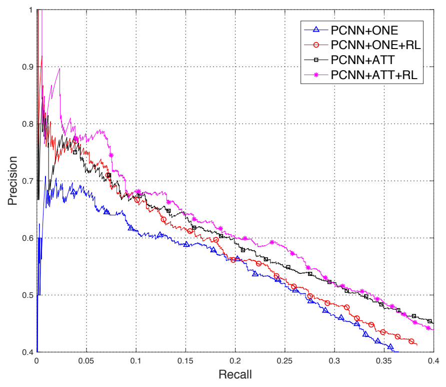


Figure 4: Aggregate PR curves of PCNN˙based model.

itive instances[4]. Because of the randomness of deep neural network on the small-scale dataset, we just train policy-based agents for these 10 relation types. First, compared with *O*riginal case, most of the *Pretrain* agents yield obvious improvements: It not only demonstrates the rationality of our pretraining strategy, but also verifies our hypothesis that most of the positive samples in Riedel dataset are true positive. More significantly, after retraining with the proposed policy-based RL method, the F_1 scores achieve further improvement, even for the case the *Pretrain* agents perform bad. These comparable results illustrate that the proposed policy-based RL method is capable of making agents develop towards a good direction.

[4]The supervised relation classification task Semeval-2010 Task 8 (Hendrickx et al., 2009) annotates nearly 1,000 instances for each relation type.

Model	**-**	**+RL**	**p-value**
CNN+ONE	0.177	**0.190**	1.24e-4
CNN+ATT	0.219	**0.229**	7.63e-4
PCNN+ONE	0.206	**0.220**	8.35e-6
PCNN+ATT	0.253	**0.261**	4.36e-3

Table 3: Comparison of AUC values between previous studies and our RL method, and the p-value of t-test.

4.4 Impact of False Positive Samples

Zeng et al. (2015) and Lin et al. (2016) are both the robust models to solve wrong labeling problem of distant supervision relation extraction. Zeng et al. (2015) combine at-least-one multi-instance learning with deep neural network to extract only one active sentence to predict the relation between entity pair; Lin et al. (2016) combine all sentences of one entity pair and assign soft attention weights to them, in which way to generate a compositive relation representation for this entity pair. However, the false positive phenomenon also includes the case that all the sentences of one entity pair are wrong, which is because the corpus is not completely aligned with the knowledge base. This phenomenon is also common between Riedel dataset and Freebase through our manual inspection. Obviously, there is nothing the above two methods can do in this case.

The proposed RL method is to tackle this problem. We adopt our RL agents to redistribute Riedel dataset by moving false positive samples into the negative sample set. Then we use Zeng et al. (2015) and Lin et al. (2016) to predict relations on this cleaned dataset, and compare the performance with that on the original Riedel dataset. As shown in Figure 3 and Figure 4, under the assistant of our RL agent, the same model can achieve obvious improvement with more reasonable training dataset. In order to give the more intuitive comparison, we calculate the AUC value of each PR curve, which reflects the area size under these curves. These comparable results also indicate the effectiveness of our policy-based RL method. Moreover, as can be seen from the result of t-test evaluation, all the p-values are less than 5e-02, so the improvements are significant.

4.5 Case Study

Figure 5 indicates that, for different relations, the scale of the detected false positive samples is not

Relation	**/people/person/place_of_birth**
FP	1. GHETTO SUPERSTAR (THE MAN THAT I AM) – Ranging from **Pittsburgh** to Broadway, **Billy Porter** performs his musical memoir.
FP	1. "They are trying to create a united front at home in the face of the pressures **Syria** is facing," said **Sami Moubayed**, a political analyst and writer here. 2. "Iran injected **Syria** with a lot of confidence: stand up, show defiance," said **Sami Moubayed**, a political analyst and writer in Damascus.
Relation	**/people/deceased_person/place_of_death**
FP	1. Some **New York city** mayors – **William O'Dwyer**, Vincent R. Impellitteri and Abraham Beame – were born abroad. 2. Plenty of local officials have, too, including two **New York city** mayors, James J. Walker, in 1932, and **William O'Dwyer**, in 1950.

Table 4: Some examples of the false positive samples detected by our policy-based agent. Each row denotes the annotated sentences of one entity pair.

proportional to the original scale, which is in accordance with the actual accident situation. At the same time, we analyze the correlation between the false positive phenomenon and the number of sentences of entity pairs : With this the number ranging from 1 to 5, the corresponding percentages are [55.9%, 32.0%, 3.7%, 4.4%, 0.7%]. This distribution is consistent with our assumption. Because Freebase is, to some extent, not completely aligned with the NYT corpus, entity pairs with fewer sentences are more likely to be false positive, which is the major factor hindering the performance of the previous systems. In Table 4, we present some false positive examples selected by our agents. Taking entity pair (*Sami Moubayed*, *Syria*) as an example, it is obvious that there is not any valuable information reflecting relation */people/person/place_of_birth*. Both of these sentences talks about the situation analysis of *Syria* from the political analyst *Sami Moubayed*. We also found that, for some entity pairs, even though there are multiple sentences, all of them are identical. This phenomenon also increases the probability of the appearance of false positive samples.

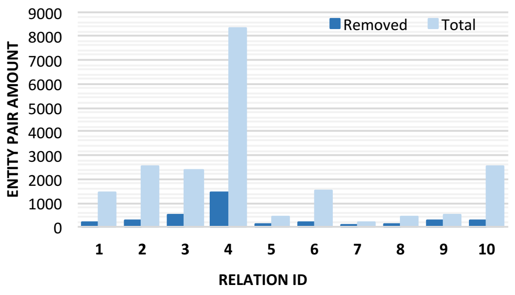


Figure 5: This figure presents the scale of the removed part for each relation type, where the horizontal axis corresponds to the IDs in Table 2.

5 Conclusion

In this work, we propose a deep reinforcement learning framework for robust distant supervision. The intuition is that, in contrast to prior works that utilize only one instance per entity pair and use soft attention weights to select plausible distantly supervised examples, we describe a policy-based framework to systematically learn to relocate the false positive samples, and better utilize the unlabeled data. More specifically, our goal is to teach the reinforcement agent to optimize the selection/redistribution strategy that maximizes the reward of boosting the performance of relation classification. An important aspect of our work is that our framework does not depend on a specific form of the relation classifier, meaning that it is a plug-and-play technique that could be potentially applied to any relation extraction pipeline. In experiments, we show that our framework boosts the performance of distant supervision relation extraction of various strong deep learning baselines on the widely used New York Times - Freebase dataset.

Acknowledge

This work was supported by National Natural Science Foundation of China (61702047), Beijing Natural Science Foundation (4174098), the Fundamental Research Funds for the Central Universities (2017RC02) and National Natural Science Foundation of China (61703234)

References

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

Razvan Bunescu and Raymond J Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS*, pages 171–178.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL (1)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL (2)*, pages 365–371.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 73–78. ACM.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv preprint arXiv:1506.07650*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 17–21.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Interpretable and Compositional Relation Learning by Joint Training with an Autoencoder

Ryo Takahashi*[1] and **Ran Tian*[1]** and **Kentaro Inui[1,2]**
(* equal contribution)
[1]Tohoku University [2] RIKEN, Japan
{ryo.t, tianran, inui}@ecei.tohoku.ac.jp

Abstract

Embedding models for entities and relations are extremely useful for recovering missing facts in a knowledge base. Intuitively, a relation can be modeled by a matrix mapping entity vectors. However, relations reside on low dimension sub-manifolds in the parameter space of arbitrary matrices – for one reason, composition of two relations M_1, M_2 may match a third M_3 (e.g. composition of relations currency_of_country and country_of_film usually matches currency_of_film_budget), which imposes compositional constraints to be satisfied by the parameters (i.e. $M_1 \cdot M_2 \approx M_3$). In this paper we investigate a dimension reduction technique by training relations jointly with an autoencoder, which is expected to better capture compositional constraints. We achieve state-of-the-art on Knowledge Base Completion tasks with strongly improved Mean Rank, and show that joint training with an autoencoder leads to interpretable sparse codings of relations, helps discovering compositional constraints and benefits from compositional training. Our source code is released at github.com/tianran/glimvec.

1 Introduction

Broad-coverage knowledge bases (KBs) such as Freebase (Bollacker et al., 2008) and DBPedia (Auer et al., 2007) store a large amount of facts in the form of ⟨head entity, relation, tail entity⟩ triples (e.g. ⟨*The Matrix*, country_of_film, *Australia*⟩), which could support a wide range of reasoning and question answering applications. The Knowledge Base Completion (KBC) task aims

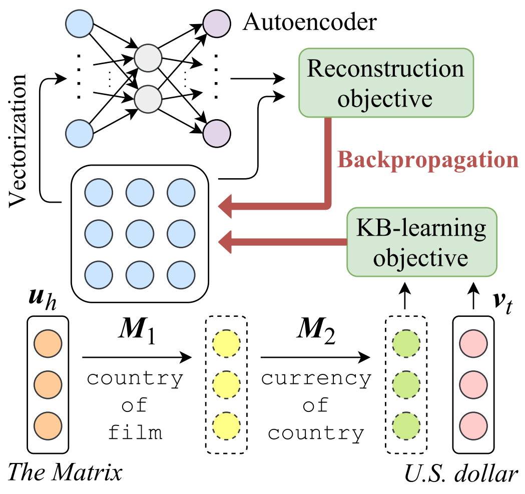


Figure 1: In joint training, relation parameters (e.g. M_1) receive updates from both a *KB-learning objective*, trying to predict entities in the KB; and a *reconstruction objective* from an autoencoder, trying to recover relations from low dimension codings.

to predict the missing part of an incomplete triple, such as ⟨*Finding Nemo*, country_of_film, ?⟩, by reasoning from known facts stored in the KB.

As a most common approach (Wang et al., 2017), modeling entities and relations to operate in a low dimension vector space helps KBC, for three conceivable reasons. First, when dimension is low, entities modeled as vectors are forced to share parameters, so "similar" entities which participate in many relations in common get close to each other (e.g. *Australia* close to *US*). This could imply that an entity (e.g. *US*) "type matches" a relation such as country_of_film. Second, relations may share parameters as well, which could transfer facts from one relation to other similar relations, for example from ⟨x, award_winner, y⟩ to ⟨x, award_nominated, y⟩. Third, spatial positions might be used to implement *composition* of relations, as relations can be regarded as mappings from head to tail entities, and the composition of two maps can match a third (e.g. the composition of `currency_of_country` and `country_of_film` matches the relation `currency_of_film_budget`), which could be captured by modeling composition in a space.

However, modeling relations as mappings naturally requires more parameters – a general linear map between d-dimension vectors is represented by a matrix of d^2 parameters – which are less likely to be shared, impeding transfers of facts between similar relations. Thus, it is desired to reduce dimensionality of relations; furthermore, the existence of a composition of two relations (assumed to be modeled by matrices $\boldsymbol{M}_1, \boldsymbol{M}_2$) matching a third ($\boldsymbol{M}_3$) also justifies dimension reduction, because it implies a *compositional constraint* $\boldsymbol{M}_1 \cdot \boldsymbol{M}_2 \approx \boldsymbol{M}_3$ that can be satisfied only by a lower dimension sub-manifold in the parameter space[1].

Previous approaches reduce dimensionality of relations by imposing pre-designed hard constraints on the parameter space, such as constraining that relations are translations (Bordes et al., 2013) or diagonal matrices (Yang et al., 2015), or assuming they are linear combinations of a small number of prototypes (Xie et al., 2017). However, pre-designed hard constraints do not seem to cope well with compositional constraints, because it is difficult to know *a priori* which two relations compose to which third relation, hence difficult to choose a pre-design; and compositional constraints are not always exact (e.g. the composition of `currency_of_country` and `headquarter_location` usually matches `business_operation_currency` but not always), so hard constraints are less suited.

In this paper, we investigate an alternative approach by training relation parameters jointly with an autoencoder (Figure 1). During training, the autoencoder tries to reconstruct relations from low dimension codings, with the reconstruction objective back-propagating to relation parameters as well. We show this novel technique promotes parameter sharing between different relations, and drives them toward low dimension manifolds (Sec.6.2). Besides, we expect the technique to cope better with compositional constraints, because it discovers low dimension manifolds posteriorly from data, and it does not impose any explicit hard constraints.

[1]It is noteworthy that similar compositional constraints apply to most modeling schemes of relations, not just matrices.

Yet, joint training with an autoencoder is not simple; one has to keep a subtle balance between gradients of the reconstruction and KB-learning objectives throughout the training process. We are not aware of any theoretical principles directly addressing this problem; but we found some important settings after extensive pre-experiments (Sec.4). We evaluate our system using standard KBC datasets, achieving state-of-the-art on several of them (Sec.6.1), with strongly improved Mean Rank. We discuss detailed settings that lead to the performance (Sec.4.1), and we show that joint training with an autoencoder indeed helps discovering compositional constraints (Sec.6.2) and benefits from compositional training (Sec.6.3).

2 Base Model

A knowledge base (KB) is a set $\mathcal{T}$ of triples of the form $\langle h, r, t\rangle$, where $h, t \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ is a relation (e.g. $\langle$*The Matrix*, `country_of_film`, *Australia*$\rangle$). A relation r has its inverse $r^{-1} \in \mathcal{R}$ so that for every $\langle h, r, t\rangle \in \mathcal{T}$, we regard $\langle t, r^{-1}, h\rangle$ as also in the KB. Under this assumption and given $\mathcal{T}$ as training data, we consider the Knowledge Base Completion (KBC) task that predicts candidates for a missing tail entity in an incomplete $\langle h, r, ?\rangle$ triple.

Most approaches tackle this problem by training a *score function* measuring the plausibility of triples being facts. The model we implement in this work represents entities h, t as d-dimension vectors $\boldsymbol{u}_h, \boldsymbol{v}_t$ respectively, and relation r as a $d \times d$ matrix $\boldsymbol{M}_r$. If $\boldsymbol{u}_h, \boldsymbol{v}_t$ are one-hot vectors with dimension $d = |\mathcal{E}|$ corresponding to each entity, one can take $\boldsymbol{M}_r$ as the adjacency matrix of entities joined by relation r, so the set of tail entities filling into $\langle h, r, ?\rangle$ is calculated by $\boldsymbol{u}_h^\top \boldsymbol{M}_r$ (with each nonzero entry corresponds to an answer). Thus, we have $\boldsymbol{u}_h^\top \boldsymbol{M}_r \boldsymbol{v}_t > 0$ if and only if $\langle h, r, t\rangle \in \mathcal{T}$. This motivates us to use $\boldsymbol{u}_h^\top \boldsymbol{M}_r \boldsymbol{v}_t$ as a natural parameter to model plausibility of $\langle h, r, t\rangle$, even in a low dimension space with $d \ll |\mathcal{E}|$. Thus, we define the score function as

$$s(h, r, t) := \exp(\boldsymbol{u}_h^\top \boldsymbol{M}_r \boldsymbol{v}_t) \qquad (1)$$

for the basic model. This is similar to the bilinear model of Nickel et al. (2011), except that we distinguish $\boldsymbol{u}_h$ (the vector for head entities) from $\boldsymbol{v}_t$ (the vector for tail entities). It has also been proposed in Tian et al. (2016), but for modeling dependency trees rather than KBs.

More generally, we consider *composition* of relations $r_1/\ldots/r_l$ to model *path*s in a KB (Guu et al., 2015), as defined by $r_1, \ldots, r_l$ participating in a sequence of facts such that the head entity of each fact coincides with the tail of its previous. For example, a sequence of two facts ⟨*The Matrix*, `country_of_film`, *Australia*⟩ and ⟨*Australia*, `currency_of_country`, *Australian Dollar*⟩ form a path of composition `country_of_film/currency_of_country`, because the head of the second fact (i.e. *Australia*) coincides with the tail of the first. Using the previous $d = |\mathcal{E}|$ analogue, one can verify that composition of relations is represented by multiplication of adjacency matrices, so we accordingly define

$$s(h, r_1/\ldots/r_l, t) := \exp(\boldsymbol{u}_h^\top \boldsymbol{M}_{r_1} \cdots \boldsymbol{M}_{r_l} \boldsymbol{v}_t)$$

to measure the plausibility of a path. It is explored in Guu et al. (2015) to learn a score function not only for single facts but also for paths. This *compositional training* scheme is shown to bring valuable information about the structure of the KB and may help KBC. In this work, we conduct experiments both with and without compositional training.

In order to learn parameters $\boldsymbol{u}_h, \boldsymbol{v}_t, \boldsymbol{M}_r$ of the score function, we follow Tian et al. (2016) using a Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012) objective. For each path (or triple) $\langle h, r_1/\ldots, t\rangle$ taken from the KB, we generate negative samples by replacing the tail entity t with some random noise t^*. Then, we maximize

$$\mathcal{L}_1 := \sum_{\text{path}} \ln \frac{s(h, r_1/\ldots, t)}{k + s(h, r_1/\ldots, t)} + \sum_{\text{noise}} \ln \frac{k}{k + s(h, r_1/\ldots, t^*)}$$

as our *KB-learning objective*. Here, k is the number of noises generated for each path. When the score function is regarded as probability, $\mathcal{L}_1$ represents the log-likelihood of "$\langle h, r_1/\ldots, t\rangle$ being actual path and $\langle h, r_1/\ldots, t^*\rangle$ being noise". Maximizing $\mathcal{L}_1$ increases the scores of actual paths and decreases the scores of noises.

3 Joint Training with an Autoencoder

Autoencoders learn efficient codings of high-dimensional data while trying to reconstruct the original data from the coding. By joint training relation matrices with an autoencoder, we also expect it to help reducing the dimensionality of the original data (i.e. relation matrices).

Formally, we define a *vectorization* $\boldsymbol{m}_r$ for each relation matrix $\boldsymbol{M}_r$, and use it as input to the autoencoder. $\boldsymbol{m}_r$ is defined as a reshape of $\boldsymbol{M}_r$ flattened into a d^2-dimension vector, and normalized such that $\|\boldsymbol{m}_r\| = \sqrt{d}$. We define

$$\boldsymbol{c}_r := \mathrm{ReLU}(\boldsymbol{A}\boldsymbol{m}_r) \qquad (2)$$

as the coding. Here $\boldsymbol{A}$ is a $c \times d^2$ matrix with $c \ll d^2$, and ReLU is the Rectified Linear Unit function (Nair and Hinton, 2010). We reconstruct the input from $\boldsymbol{c}_r$ by multiplying a $d^2 \times c$ matrix $\boldsymbol{B}$. We want $\boldsymbol{B}\boldsymbol{c}_r$ to be more similar to $\boldsymbol{m}_r$ than other relations. For this purpose, we define a similarity

$$g(r_1, r_2) := \exp(\frac{1}{\sqrt{dc}} \boldsymbol{m}_{r_1}^\top \boldsymbol{B}\boldsymbol{c}_{r_2}), \qquad (3)$$

which measures the length of $\boldsymbol{B}\boldsymbol{c}_{r_2}$ projected to the direction of $\boldsymbol{m}_{r_1}$. In order to learn the parameters $\boldsymbol{A}, \boldsymbol{B}$, we adopt the Noise Contrastive Estimation scheme as in Sec.2, generate random noises r^* for each relation r and maximize

$$\mathcal{L}_2 := \sum_{r \in \mathcal{R}} \ln \frac{g(r, r)}{k + g(r, r)} + \sum_{r^* \sim \mathcal{R}} \ln \frac{k}{k + g(r, r^*)}$$

as our *reconstruction objective*. Maximizing $\mathcal{L}_2$ increases $\boldsymbol{m}_r$'s similarity with $\boldsymbol{B}\boldsymbol{c}_r$, and decreases it with $\boldsymbol{B}\boldsymbol{c}_{r^*}$.

During joint training, both $\mathcal{L}_1$ and $\mathcal{L}_2$ are simultaneously maximized, and the gradient $\nabla\mathcal{L}_2$ propagates to relation matrices as well. Since $\nabla\mathcal{L}_2$ depends on $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\boldsymbol{A}, \boldsymbol{B}$ interact with all relations, they promote indirect parameter sharing between different relation matrices. In Sec.6.2, we further show that joint training drives relations toward a low dimension manifold.

4 Optimization Tricks

Joint training with an autoencoder is not simple. Relation matrices receive updates from both $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$, but if they update $\nabla\mathcal{L}_1$ too much, the autoencoder has no effect; conversely, if they update $\nabla\mathcal{L}_2$ too often, all relation matrices crush into one cluster. Furthermore, an autoencoder should learn from genuine patterns of relation matrices that emerge from fitting the KB, but not the reverse – in which the autoencoder imposes arbitrary patterns to relation matrices according to random initialization. Therefore, it is not surprising that a naive optimization of $\mathcal{L}_1 + \mathcal{L}_2$ does not work.

After extensive pre-experiments, we have found some crucial settings for successful training. The most important "magic" is the scaling factor $\frac{1}{\sqrt{dc}}$ in definition of the similarity function (3), perhaps being combined with other settings as we discuss below. We have tried different factors 1, $\frac{1}{\sqrt{d}}$, $\frac{1}{\sqrt{c}}$ and $\frac{1}{dc}$ instead, with various combinations of d and c; but the autoencoder failed to learn meaningful codings in other settings. When the scaling factor is too small (e.g. $\frac{1}{dc}$), all relations get almost the same coding; conversely if the factor is too large (e.g. 1), all codings get very close to 0.

The next important rule is to keep a balance between the updates coming from $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$. We use Stochastic Gradient Descent (SGD) for optimization, and the common practice (Bottou, 2012) is to set the learning rate as

$$\alpha(\tau) := \frac{\eta}{1 + \eta\lambda\tau}. \tag{4}$$

Here, η, λ are hyper-parameters and τ is a counter of processed data points. In this work, in order to control the updates in detail to keep a balance, we modify (4) to use a a step counter τ_r for each relation r, counting "number of updates" instead of data points[2]. That is, whenever $\boldsymbol{M}_r$ gets a nonzero update from a gradient calculation, τ_r increases by 1. Furthermore, we use different hyper-parameters for different "types of updates", namely η_1, λ_1 for updates coming from $\nabla\mathcal{L}_1$, and η_2, λ_2 for updates coming from $\nabla\mathcal{L}_2$. Thus, let Δ_1 be the partial gradient of $\nabla\mathcal{L}_1$, and Δ_2 the partial gradient of $\nabla\mathcal{L}_2$, we update $\boldsymbol{M}_r$ by $\alpha_1(\tau_r)\Delta_1 + \alpha_2(\tau_r)\Delta_2$ at each step, where

$$\alpha_1(\tau_r) := \frac{\eta_1}{1 + \eta_1\lambda_1\tau_r}, \quad \alpha_2(\tau_r) := \frac{\eta_2}{1 + \eta_2\lambda_2\tau_r}.$$

The rule for setting η_1, λ_1 and η_2, λ_2 is that, η_2 should be much smaller than η_1, because η_1, η_2 control the magnitude of learning rates at the early stage of training, with the autoencoder still largely random and Δ_2 not making much sense; on the other hand, one has to choose λ_1 and λ_2 such that $\|\Delta_1\|/\lambda_1$ and $\|\Delta_2\|/\lambda_2$ are at the same scale, because the learning rates approach $1/(\lambda_1\tau_r)$ and $1/(\lambda_2\tau_r)$ respectively, as the training proceeds. In this way, the autoencoder will not impose random patterns to relation matrices according to its initialization at the early stage, and a balance is kept between $\alpha_1(\tau_r)\Delta_1$ and $\alpha_2(\tau_r)\Delta_2$ later.

But how to estimate $\|\Delta_1\|$ and $\|\Delta_2\|$? It seems that we can approximately calculate their scales from initialization. In this work, we use i.i.d. Gaussians of variance $1/d$ to initialize parameters, so the initial Euclidean norms are $\|\boldsymbol{u}_h\| \approx 1$, $\|\boldsymbol{v}_t\| \approx 1$, $\|\boldsymbol{M}_r\| \approx \sqrt{d}$, and $\|\boldsymbol{BAm}_r\| \approx \sqrt{dc}$. Thus, by calculating $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$ using (1) and (3), we have approximately

$$\|\Delta_1\| \approx \|\boldsymbol{u}_h\boldsymbol{v}_t^\top\| \approx 1, \quad \text{and}$$
$$\|\Delta_2\| \approx \|\frac{1}{\sqrt{dc}}\boldsymbol{Bc}_r\| \approx \frac{1}{\sqrt{dc}}\|\boldsymbol{BAm}_r\| \approx 1.$$

It suggests that, because of the scaling factor $\frac{1}{\sqrt{dc}}$ in (3), we have $\|\Delta_1\|$ and $\|\Delta_2\|$ at the same scale, so we can set $\lambda_1 = \lambda_2$. This might not be a mere coincidence.

4.1 Training the Base Model

Besides the tricks for joint training, we also found settings that significantly improve the base model on KBC, as briefly discussed below. In Sec.6.3, we will show performance gains by these settings using the FB15k-237 validation set.

Normalization It is better to normalize relation matrices to $\|\boldsymbol{M}_r\| = \sqrt{d}$ during training. This might reduce fluctuations in entity vector updates.

Regularizer It is better to minimize $\|\boldsymbol{M}_r^\top\boldsymbol{M}_r - \frac{1}{d}\operatorname{tr}(\boldsymbol{M}_r^\top\boldsymbol{M}_r)I\|$ during training. This regularizer drives $\boldsymbol{M}_r$ toward an orthogonal matrix (Tian et al., 2016) and might reduce fluctuations in entity vector updates. As a result, all relation matrices trained in this work are very close to orthogonal.

Initialization Instead of pure Gaussian, it is better to initialize matrices as $(I + G)/2$, where G is random. The identity matrix I helps passing information from head to tail (Tian et al., 2016).

Negative Sampling Instead of a unigram distribution, it is better to use a *uniform* distribution for generating noises. This is somehow counter-intuitive compared to training word embeddings.

5 Related Works

KBs have a wide range of applications (Berant et al., 2013; Hixon et al., 2015; Nickel et al., 2016a) and KBC has inspired a huge amount of research (Bordes et al., 2013; Riedel et al., 2013; Socher et al., 2013; Wang et al., 2014b,a; Xiao et al., 2016; Nguyen et al., 2016; Toutanova et al., 2016; Das et al., 2017; Hayashi and Shimbo, 2017).

[2]Similarly, we set separate step counters for all head and tail entities, and the autoencoder as well.

Among the previous works, TransE (Bordes et al., 2013) is the classic method which represents a relation as a translation of the entity vector space, and is partially inspired by Mikolov et al. (2013)'s vector arithmetic method of solving word analogy tasks. Although competitive in KBC, it is speculated that this method is well-suited for 1-to-1 relations but might be too simple to represent N-to-N relations accurately(Wang et al., 2017). Thus, extensions such as TransR (Lin et al., 2015b) and STransE (Nguyen et al., 2016) are proposed to map entities into a relation-specific vector space before translation. The ITransF model (Xie et al., 2017) further enhances this approach by imposing a hard constraint that the relation-specific maps should be linear combinations of a small number of prototypical matrices. Our work inherits the same motivation with ITransF in terms of promoting parameter-sharing among relations.

On the other hand, the base model used in this work originates from RESCAL (Nickel et al., 2011), in which relations are naturally represented as analogue to the adjacency matrices (Sec.2). Further developments include HolE (Nickel et al., 2016b) and ConvE (Dettmers et al., 2018) which improve this approach in terms of parameter-efficiency, by introducing low dimension factorizations of the matrices. We inherit the basic model of RESCAL but draw additional training techniques from Tian et al. (2016), and show that the base model already can achieve near state-of-the-art performance (Sec.6.1,6.3). This sends a message similar to Kadlec et al. (2017), saying that training tricks might be as important as model designs.

Nevertheless, we emphasize the novelty of this work in that the previous models mostly achieve dimension reduction by imposing some pre-designed hard constraints (Bordes et al., 2013; Yang et al., 2015; Trouillon et al., 2016; Nickel et al., 2016b; Xie et al., 2017; Dettmers et al., 2018), whereas the constraints themselves are not learned from data; in contrast, our approach by jointly training an autoencoder does not impose any explicit hard constraints, so it leads to more flexible modeling.

Moreover, we additionally focus on leveraging composition in KBC. Although this idea has been frequently explored before (Guu et al., 2015; Neelakantan et al., 2015; Lin et al., 2015a), our discussion about the concept of compositional constraints and its connection to dimension reduction has not been addressed similarly in previous research. In experiments, we will show (Sec.6.2,6.3) that joint training with an autoencoder indeed helps finding compositional constraints and benefits from compositional training.

Autoencoders have been used solo for learning distributed representations of syntactic trees (Socher et al., 2011), words and images (Silberer and Lapata, 2014), or semantic roles (Titov and Khoddam, 2015). It is also used for pretraining other deep neural networks (Erhan et al., 2010). However, when combined with other models, the learning of autoencoders, or more generally *sparse codings* (Rubinstein et al., 2010), is usually conveyed in an alternating manner, fixing one part of the model while optimizing the other, such as in Xie et al. (2017). To our knowledge, joint training with an autoencoder is not widely used previously for reducing dimensionality.

Jointly training an autoencoder is not simple because it takes non-stationary inputs. In this work, we modified SGD so that it shares traits with some modern optimization algorithms such as Adagrad (Duchi et al., 2011), in that they both set different learning rates for different parameters. While Adagrad sets them adaptively by keeping track of gradients for all parameters, our modification of SGD is more efficient and allows us to grasp a rough intuition about which parameter gets how much update. We believe our techniques and findings in joint training with an autoencoder could be helpful to reducing dimensionality and improving interpretability in other neural network architectures as well.

6 Experiments

We evaluate on standard KBC datasets, including WN18 and FB15k (Bordes et al., 2013), WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). The statistical information of these datasets are shown in Table 1.

WN18 collects word relations from WordNet (Miller, 1995), and FB15k is taken from Freebase (Bollacker et al., 2008); both have filtered out low frequency entities. However, it is reported in Toutanova and Chen (2015) that both WN18 and FB15k have information leaks because the inverses of some test triples appear in the training set. FB15k-237 and WN18RR fix this problem by deleting such triples from training and test data. In this work, we do evaluate on WN18 and FB15k, but our models are mainly tuned on FB15k-237.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#Train	#Valid	#Test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

Table 1: Statistical information of the KBC datasets. $|\mathcal{E}|$ and $|\mathcal{R}|$ denote the number of entities and relation types, respectively; #Train, #Valid, and #Test are the numbers of triples in the training, validation, and test sets, respectively.

For all datasets, we set the dimension $d = 256$ and $c = 16$, the SGD hyper-parameters $\eta_1 = 1/64$, $\eta_2 = 2^{-14}$ and $\lambda_1 = \lambda_2 = 2^{-14}$. The training batch size is 32 and the triples in each batch share the same head entity. We compare the base model (BASE) to our joint training with an autoencoder model (JOINT), and the base model with compositional training (BASE+COMP) to our joint model with compositional training (JOINT+COMP). When compositional training is enabled (BASE+COMP, JOINT+COMP), we use random walk to sample paths of length $1 + X$, where X is drawn from a Poisson distribution of mean $\lambda = 1.0$.

For any incomplete triple $\langle h, r, ?\rangle$ in KBC test, we calculate a score $s(h, r, e)$ from (1), for every entity $e \in \mathcal{E}$ such that $\langle h, r, e\rangle$ *does not appear in any of the training, validation, or test sets* (Bordes et al., 2013). Then, the calculated scores together with $s(h, r, t)$ for the gold triple is converted to ranks, and the rank of the gold entity t is used for evaluation. Evaluation metrics include Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits at 10 (H10). Lower MR, higher MRR, and higher H10 indicate better performance.

We consult MR and MRR on validation sets to determine training epochs; we stop training when both MR and MRR have stopped improving.

6.1 KBC Results

The results are shown in Table 2. We found that joint training with an autoencoder mostly improves performance, and the improvement becomes more clear when compositional training is enabled (i.e., JOINT $\geq$ BASE and JOINT+COMP $>$ BASE+COMP). This is convincing because generally, joint training contributes with its regularizing effects, and drastic improvements are less expected[3]. When compositional training is enabled, the system usually achieves better MR, though not always improves in other measures. The performance gains are more obvious on the WN18RR and FB15k-237 datasets, possibly because WN18 and FB15k contain a lot of easy instances that can be solved by a simple rule (Dettmers et al., 2018).

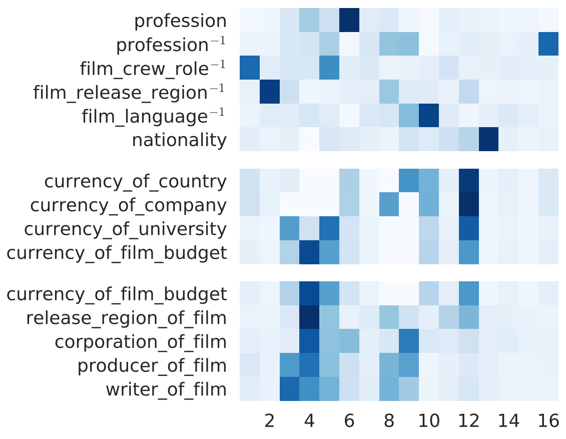


Figure 2: Examples of relation codings learned from FB15k-237. Each row shows a 16 dimension vector encoding a relation. Vectors are normalized such that their entries sum to 1.

Furthermore, the numbers demonstrated by our joint and base models are among the strongest in the literature. We have conducted re-experiments of several representative algorithms, and also compare with state-of-the-art published results. For re-experiments, we use Lin et al. (2015b)'s implementation[4] of TransE (Bordes et al., 2013) and TransR, which represent relations as vector translations; and Nickel et al. (2016b)'s implementation[5] of RESCAL (Nickel et al., 2011) and HolE, where RESCAL is most similar to the BASE model and HolE is a more parameter-efficient variant. We experimented with the default settings, and found that our models outperform most of them.

Among the published results, STransE (Nguyen et al., 2016) and ITransF (Xie et al., 2017) are more complicated versions of TransR, achieving the previous highest MR on WN18 but are outperformed by our JOINT+COMP model. ITransF is most similar to our JOINT model in that they both learn sparse codings for relations. On WN18RR and FB15k-237, Dettmers et al. (2018)'s report of ComplEx

[3]The source code and trained models are publicly released at `https://github.com/tianran/glimvec`, where we also show the mean performance and deviations of multiple random initializations, to give a more complete picture.

[4]`https://github.com/thunlp/KB2E`

[5]`https://github.com/mnick/holographic-embeddings`

Model	WN18		FB15k		WN18RR			FB15k-237		
	MR	H10	MR	H10	MR	MRR	H10	MR	MRR	H10
JOINT	**277**	**95.8**	**53**	**82.5**	**4233**	**.461***	**53.4**	**212**	.336	**52.3***
BASE	286	**95.8**	**53**	**82.5**	4371	.459	52.9	215	**.337***	**52.3***
JOINT+COMP	**191***	**94.8**	**53**	**69.7**	**2268***	**.343**	**54.8***	**197***	**.331**	**51.6**
BASE+COMP	195	**94.8**	54	69.4	2447	.310	54.1	203	.328	51.5
TransE (Bordes et al., 2013)	292	92.0	**66**	70.4	4311	.202	45.6	**278**	.236	41.6
TransR (Lin et al., 2015b)	**281**	93.6	76	**74.4**	**4222**	.210	**47.1**	320	**.282**	**45.9**
RESCAL (Nickel et al., 2011)	911	58.0	163	41.0	9689	.105	20.3	457	.178	31.9
HolE (Nickel et al., 2016b)	724	**94.3**	293	66.8	8096	**.376**	40.0	1172	.169	30.9
STransE (Nguyen et al., 2016)	206	93.4	69	79.9	-	-	-	-	-	-
ITransF (Xie et al., 2017)	**205**	94.2	65	81.0	-	-	-	-	-	-
ComplEx (Trouillon et al., 2016)	-	94.7	-	84.0	**5261**	.44	**51**	339	.247	42.8
Ensemble DistMult (Kadlec et al., 2017)	457	95.0	35.9	90.4	-	-	-	-	-	-
IRN (Shen et al., 2017)	249	95.3	38	**92.7***	-	-	-	-	-	-
ConvE (Dettmers et al., 2018)	504	95.5	64	87.3	5277	**.46**	48	**246**	**.316**	**49.1**
R-GCN+ (Schlichtkrull et al., 2017)	-	**96.4***	-	84.2	-	-	-	-	.249	41.7
ProjE (Shi and Weninger, 2017)	-	-	**34***	88.4	-	-	-	-	-	-

Table 2: KBC results on the WN18, FB15k, WN18RR, and FB15k-237 datasets. The first and second sectors compare our joint to the base models with and without compositional training, respectively; the third sector shows our re-experiments and the fourth shows previous published results. Bold numbers are the best in each sector, and (*) indicates the best of all.

(Trouillon et al., 2016) and ConvE were previously the best results. Our models mostly outperform them. Other results include Kadlec et al. (2017)'s simple but strong baseline and several recent models (Schlichtkrull et al., 2017; Shi and Weninger, 2017; Shen et al., 2017) which achieve best results on FB15k or WN18 in some measure. Our models have comparable results.

6.2 Intuition and Insight

What does the autoencoder look like? How does joint training affect relation matrices? We address these questions by analyses showing that **(i)** the autoencoder learns sparse and interpretable codings of relations, **(ii)** the joint training drives relation matrices toward a low dimension manifold, and **(iii)** it helps discovering compositional constraints.

Sparse Coding and Interpretability

Due to the ReLU function in (2), our autoencoder learns sparse coding, with most relations having large code values at only two or three dimensions. This sparsity makes it easy to find patterns in the model that to some extent explain the semantics of relations. Figure 2 shows some examples.

In the first group of Figure 2, we show a small number of relations that are almost always assigned a near one-hot coding, regardless of initialization. These are high frequency relations joining two large categories (e.g. film and language), which probably constitute the skeleton of a KB.

In the second group, we found the 12th dimension strongly correlates with `currency`; and in the third group, we found the 4th dimension strongly correlates with `film`. As for the relation `currency_of_film_budget`, it has large code values at both dimensions. This kind of relation clustering also seems independent of initialization. Intuitively, it shows that the autoencoder may discover similarities between relations and promote indirect parameter sharing among them. Yet, as the autoencoder only reconstructs *approximations* of relation matrices but never constrain them to be exactly equal to the original, relation matrices with very similar codings may still differ considerably. For example, `producer_of_film` and `writer_of_film` have codings of cosine similarity 0.973, but their relation matrices only have[6] a cosine similarity 0.338.

Low dimension manifold

In order to visualize the relation matrices learned by our joint and base models, we use UMAP[7] (McInnes and Healy, 2018) to embed M_r into a 2D plane[8]. We use relation matrices trained on

[6]Cosine similarity 0.338 is still high for matrices, due to the high dimensionality of their parameter space.

[7]`https://github.com/lmcinnes/umap`

[8]UMAP is a recently proposed manifold learning algorithm based on the fuzzy topological structure. We also tried

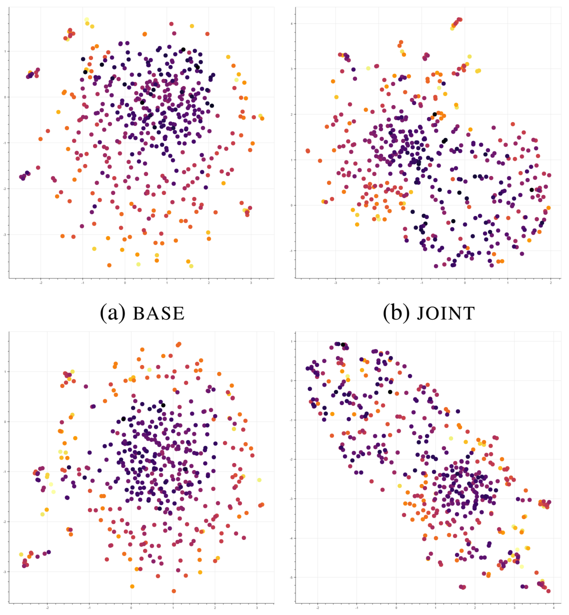

(a) BASE (b) JOINT (c) BASE+COMP (d) JOINT+COMP

Figure 3: By UMAP, relation matrices are embedded into a 2D plane. Colors show frequencies of relations; and lighter color means more frequent.

FB15k-237, and compare models trained by the same number of epochs. The results are shown in Figure 3.

We can see that Figure 3a and Figure 3c are mostly similar, with high frequency relations scattered randomly around a low frequency cluster, suggesting that they come from various directions of a high dimension space, with frequent relations probably being pulled further by the training updates. On the other hand, in Figure 3b and Figure 3d we found less frequent relations being clustered with frequent ones, and multiple traces of low dimension structures. It suggests that joint training with an autoencoder indeed drives relations toward a low dimension manifold. In addition, Figure 3d shows different structures against Figure 3b, which we conjecture could be related to compositional constraints discovered by compositional training.

Compositional constraints

In order to directly evaluate a model's ability to find compositional constraints, we extracted from FB15k-237 a list of $(r_1/r_2, r_3)$ pairs such that r_1/r_2 matches r_3. Formally, the list is constructed as below. For any relation r, we define a *content set* $C(r)$ as the set of (h, t) pairs such that $\langle h, r, t\rangle$ is a fact in the KB. Similarly, we define $C(r_1/r_2)$

t-SNE (van der Maaten and Hinton, 2008) but found UMAP more insightful.

Model	MR	MRR
JOINT+COMP	**130±27**	**.0481±.0090**
BASE+COMP	150±3	.0280±.0010
RANDOMM2	181±19	.0356±.0100

Table 3: Performance at discovering compositional constraints extracted from FB15k-237

as the set of (h, t) pairs such that $\langle h, r_1/r_2, t\rangle$ is a path. We regard $(r_1/r_2, r_3)$ as a compositional constraint if their content sets are similar; that is, if $|C(r_1/r_2) \cap C(r_3)| \geq 50$ and the Jaccard similarity between $C(r_1/r_2)$ and $C(r_3)$ is ≥ 0.4. Then, after filtering out degenerated cases such as $r_1 = r_3$ or $r_2 = r_1^{-1}$, we obtained a list of 154 compositional constraints, e.g. (`currency_of_country/country_of_film`, `currency_of_film_budget`).

For each compositional constraint $(r_1/r_2, r_3)$ in the list, we take the matrices $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ corresponding to r_1, r_2 and r_3 respectively, and rank $\boldsymbol{M}_3$ according to its cosine similarity with $\boldsymbol{M}_1\boldsymbol{M}_2$, among all relation matrices. Then, we calculate MR and MRR for evaluation. We compare the JOINT+COMP model to BASE+COMP, as well as a randomized baseline where M_2 is selected randomly from the relation matrices in JOINT+COMP instead (RANDOMM2). The results are shown in Table 3. We have evaluated 5 different random initializations for each model, trained by the same number of epochs, and we report the mean and standard deviation. We verify that JOINT+COMP performs better than BASE+COMP, indicating that joint training with an autoencoder indeed helps discovering compositional constraints. Furthermore, the random baseline RANDOMM2 tests a hypothesis that joint training might be just clustering M_3 and M_1 here, to the extent that M_3 and M_1 are so close that even a random M_2 can give the correct answer; but as it turns out, JOINT+COMP largely outperforms RANDOMM2, excluding this possibility. Thus, joint training performs better not simply because it clusters relation matrices; it learns compositions indeed.

6.3 Losses and Gains

In the KBC task, where are the losses and what are the gains of different settings? With additional evaluations, we show **(i)** some crucial settings for the base model, and **(ii)** joint training with an autoencoder benefits more from compositional training.

Settings	MR	MRR	H10
BASE	**214**	**.338**	**52.5**
no normalization	309	.326	49.9
no regularizer	400	.328	51.3
pure Gaussian	221	.336	52.1
unigram distribution	215	.324	50.6

Table 4: Ablation of the four settings of the base model as described in Sec.4.1

Crucial settings for the base model

It is noteworthy that our base model already achieves strong results. This is due to several detailed but crucial settings as we discussed in Sec.4.1; Table 4 shows their gains on the FB15k-237 validation data. The most dramatic improvement comes from the regularizer that drives matrices to orthogonal.

Gains with compositional training

One can force a model to focus more on (longer) compositions of relations, by sampling longer paths in compositional training. Since joint training with an autoencoder helps discovering compositional constraints, we expect it to be more helpful when the sampled paths are longer. In this work, path lengths are sampled from a Poisson distribution, we thus vary the mean λ of the Poisson to control the strength of compositional training. The results on FB15k-237 are shown in Table 5.

We can see that, as λ gets larger, MR improves much but MRR slightly drops. It suggests that in FB15k-237, composition of relations might mainly help finding more appropriate candidates for a missing entity, rather than pinpointing a correct one. Yet, joint training improves base models even more as the paths get longer, especially in MR. It further supports our conjecture that joint training with an autoencoder may strongly interact with compositional training.

7 Conclusion

We have investigated a dimension reduction technique which trains a KB embedding model jointly with an autoencoder. We have developed new training techniques and achieved state-of-the-art results on several KBC tasks with strong improvements in Mean Rank. Furthermore, we have shown that the autoencoder learns low dimension sparse codings that can be easily explained; the joint training technique drives high-dimensional data toward low dimension manifolds; and the reduction of dimensionality may interact strongly with composition, help discovering compositional constraints and benefit from compositional training. We believe these findings provide insightful understandings of KB embedding models and might be applied to other neural networks beyond the KBC task.

Model	λ	Valid			Test		
		MR	MRR	H10	MR	MRR	H10
BASE	0	209	.341	52.9	215	.337	52.3
JOINT	0	+1	-.001	-.2	**-3**	-.001	0
BASE	0.5	204	.337	52.2	211	.332	51.7
JOINT	0.5	**-3**	**+.002**	**+.1**	+1	**+.002**	**+.2**
BASE	1.0	191	.334	52.0	203	.328	51.5
JOINT	1.0	**-5**	**+.002**	-.1	**-6**	**+.003**	**+.1**

Table 5: Evaluation of BASE and gains by JOINT, on FB15k-237 with different strengths of compositional training. Bold numbers are improvements.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1301, Japan. We thank Pontus Stenetorp, Makoto Miwa, and the anonymous reviewers for many helpful advices and comments.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information*

Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pages 2787–2795.

Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141, Valencia, Spain. Association for Computational Linguistics.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, Lisbon, Portugal. Association for Computational Linguistics.

Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 554–559, Vancouver, Canada. Association for Computational Linguistics.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–861, Denver, Colorado. Association for Computational Linguistics.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, Vancouver, Canada. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714, Lisbon, Portugal. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2181–2187.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

L. McInnes and J. Healy. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China. Association for Computational Linguistics.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California. Association for Computational Linguistics.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 809–816, USA. Omnipress.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

R. Rubinstein, A. M. Bruckstein, and M. Elad. 2010. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103.

Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. 2017. Modeling large-scale structured relationships with shared memory for knowledge base completion. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 57–68, Vancouver, Canada. Association for Computational Linguistics.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1236–1242.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–732, Baltimore, Maryland. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Learning semantically and additively compositional distributional representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1277–1287, Berlin, Germany. Association for Computational Linguistics.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, Denver, Colorado. Association for Computational Linguistics.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1434–1444, Berlin, Germany. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2071–2080.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1112–1119.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1315–1321.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962, Vancouver, Canada. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the 3rd International Conference on Learning Representations*, pages 1–12.

A Out-of-vocabulary Entities in KBC

Occasionally, a KBC test set may contain entities that never appear in the training data. Such out-of-vocabulary (OOV) entities pose a challenge to KBC systems; while some systems address this issue by explicitly learn an OOV entity vector (Dettmers et al., 2018), our approach is described below. For an incomplete triple $\langle h, r, ?\rangle$ in the test, if h is OOV, we replace it with the most frequent entity that has ever appeared as a head of relation r in the training data. If the gold tail entity is OOV, we use the zero vector for computing the score and the rank of the gold entity.

Usually, OOV entities are rare and negligible in evaluation; except for the WN18RR test data which contains about 6.7% triples with OOV entities. Here, we also report adjusted scores on WN18RR in the setting that all triples with OOV entities are removed from the test. The results are shown in Table 6.

Model	MR	MRR	H10
JOINT	**3317**	**.493**	**57.2**
BASE	3435	.492	56.7
JOINT+COMP	**1507**	**.367**	**58.7**
BASE+COMP	1629	.332	58.0

Table 6: Adjusted scores on WN18RR.

Zero-Shot Transfer Learning for Event Extraction

Lifu Huang[1], **Heng Ji**[1], **Kyunghyun Cho**[2],
Ido Dagan[3], **Sebastian Riedel**[4], **Clare R. Voss**[5]
[1] Rensselaer Polytechnic Institute, [2] New York University,
[3] Bar-Ilan University, [4] University College London,
[5] US Army Research Lab
[1] {huangl7, jih}@rpi.edu, [2] kyunghyun.cho@nyu.edu
[3] dagan@cs.biu.ac.il, [4] s.riedel@ucl.ac.uk
[5] clare.r.voss.civ@mail.mil

Abstract

Most previous supervised event extraction methods have relied on features derived from manual annotations, and thus cannot be applied to new event types without extra annotation effort. We take a fresh look at event extraction and model it as a generic grounding problem: mapping each event mention to a specific type in a target event ontology. We design a transferable architecture of structural and compositional neural networks to jointly represent and map event mentions and types into a shared semantic space. Based on this new framework, we can select, for each event mention, the event type which is semantically closest in this space as its type. By leveraging manual annotations available for a small set of existing event types, our framework can be applied to new unseen event types without additional manual annotations. When tested on 23 unseen event types, this zero-shot framework, without manual annotations, achieves performance comparable to a supervised model trained from 3,000 sentences annotated with 500 event mentions.[1]

1 Introduction

The goal of event extraction is to identify event triggers and their arguments in unstructured text data, and then to assign an event type to each trigger and a semantic role to each argument. An example is shown in Figure 1. Traditional supervised methods have typically modeled this task of event extraction as a classification problem, by assigning event triggers to event types from a pre-defined fixed set. These methods rely heavily on manual annotations and features specific to each event type, and thus are not easily adapted to new event types without extra annotation effort. Handling new event types may even entail starting over, without being able to re-use annotations from previous event types.

To make event extraction effective as new real-world scenarios emerge, we take a look at this task from the perspective of zero-shot learning, ZSL (Frome et al., 2013; Norouzi et al., 2013; Socher et al., 2013a). ZSL, as a type of transfer learning, makes use of separate, pre-existing classifiers to build a semantic, cross-concept space that maps between their respective classes. The resulting shared semantic space then allows for building a novel "zero-shot" classifier, i,e,, requiring no (zero) additional training examples, to handle *unseen* cases. We observe that each event mention has a structure consisting of a candidate trigger and arguments, with corresponding predefined name labels for the event type and argument roles. We propose to enrich the semantic representations of each event mention and event type with rich structures, and determine the type based on the semantic similarity between an event mention and each event type defined in a target ontology. Let's consider two example sentences:

E1. The Government of <u>*China*</u> has ruled Tibet since 1951 after **dispatching** <u>*troops*</u> to the <u>*Himalayan*</u> region in <u>*1950*</u>.

E2. Iranian state television stated that the **conflict** between the <u>*Iranian police*</u> and the drug <u>*smugglers*</u> took place near the town of <u>*mirjaveh*</u>.

In E1, as also diagrammed in Figure 1, **dis-**

[1]The programs are publicly available for research purpose at: https://github.com/wilburOne/ZeroShotEvent

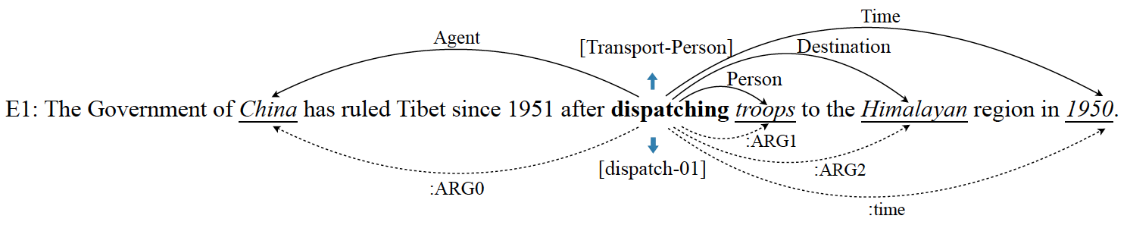

Agent
[Transport-Person]
Time
Destination
Person
E1: The Government of China has ruled Tibet since 1951 after dispatching troops to the Himalayan region in 1950.
:ARG1
[dispatch-01]
:ARG2
:ARG0
:time

Figure 1: Event Mention Example: **dispatching** is the trigger of a *Transport-Person* event with four arguments: the solid lines show the event annotations for the sentence while the dotted lines show the Abstract Meaning Representation parsing output.

patching is the trigger for the event mention of type *Transport_Person* and in E2, **conflict** is the trigger for the event mention of type *Attack*. We make use of Abstract Meaning Representations (AMR) (Banarescu et al., 2013) to identify the candidate arguments and construct event mention structures as shown in Figure 2 (top). Figure 2 (bottom) also shows event type structures defined in the Automatic Content Extraction (ACE) guideline.[2] We can see that a trigger and its event type name usually have some shared meaning. Furthermore, their structures also tend to be similar: a *Transport_Person* event typically involves a *Person* as its *patient* role, while an *Attack* event involves a *Person* or *Location* as an *Attacker*. This observation matches the theory by Pustejovsky (1991): "the semantics of an event structure can be generalized and mapped to event mention structures in a systematic and predictable way".

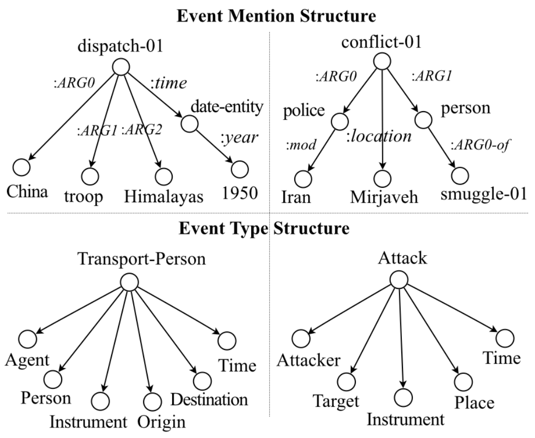


Figure 2: Examples of Event Mention Structures and Type Structures from ACE.

Inspired by this theory, for the first time, we model event extraction as a generic grounding problem, by mapping each mention to its semantically closest event type. Given an event ontology, where each event type structure is well-defined, we will refer to the event types for which we have annotated event mentions as *seen* types, while those without annotations as *unseen* types. Our goal is to learn a generic mapping function independent of event types, which can be trained from annotations for a limited number of seen event types and then used for any new unseen event types. We design a transferable neural architecture, which jointly learns and maps the structural representations of event mentions and types into a shared semantic space, by minimizing the distance between each event mention and its corresponding type. For event mentions with *unseen* types, their structures will be projected into the same semantic space using the same framework and assigned types with top-ranked similarity values.

To summarize, to apply our new zero-shot transfer learning framework to any new unseen event types, we only need (1) a structured definition of the unseen event type (its type name along with role names for its arguments, from the event ontology); and (2) some annotations for one or a few seen event types. Without requiring any additional manual annotations for the new unseen types, our ZSL framework achieves performance comparable to supervised methods trained from a substantial amount of training data for the same types.

2 Approach Overview

Briefly here, we overview the phases involved in building our framework's shared semantic space that, in turn, is the basis for the ZSL framework. Given a sentence s, we start by identifying candidate triggers and arguments based on AMR parsing (Wang et al., 2015b). For the example shown in Figure 1, we identify *dispatching* as a trigger, and its candidate arguments: *China*, *troops*, *Himalayan* and *1950*. The details will be described in Section 3.

[2]https://en.wikipedia.org/wiki/Automatic_content_extraction

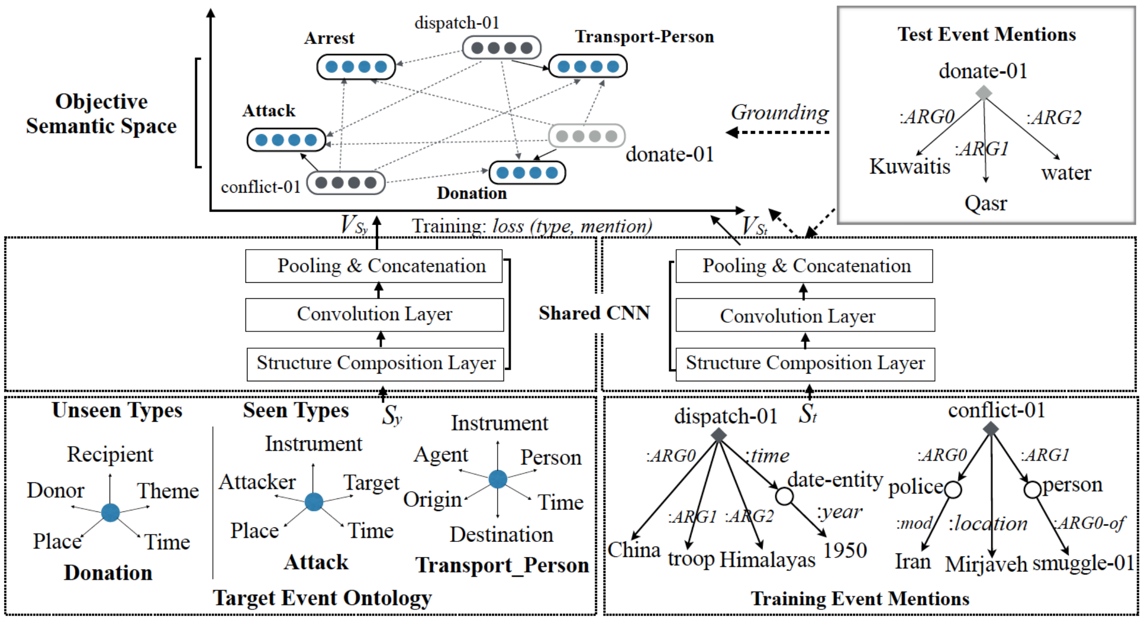


Figure 3: Architecture Overview. The blue circles denote event types and event type representations. The dark grey diamonds and circles denote triggers and trigger representations from training set. The light grey diamonds and circles denote triggers and trigger representations from testing set.

After this identification phase, we use our new neural architecture, as depicted in Figure 3, to classify triggers into event types. (The classification of arguments into roles follows the same pipeline.) For each trigger t, e.g., *dispatch-01*, we determine its type by comparing its semantic representation with that of any event type in the event ontology. In order to incorporate the contexts into the semantic representation of t, we build a structure S_t using AMR as shown in Figure 3. Each structure is composed of a set of tuples, e.g, ⟨*dispatch-01*, *:ARG0*, *China*⟩. We use a matrix to represent each AMR relation, composing its semantics with two concepts for each tuple (in Section 4), and feed all tuple representations into a CNN to generate a dense vector representation V_{S_t} for the event mention structure (in Section 5.1).

Given a target event ontology, for each type y, e.g., *Transport_Person*, we construct a type structure S_y consisting of its predefined roles, and use a tensor to denote the implicit relation between any type and argument role. We compose the semantics of type and argument role with the tensor for each tuple, e.g., ⟨*Transport_Person*, *Destination*⟩ (in Section 4). Then we generate the event type structure representation V_{S_y} using the same CNN (in Section 5.1). By minimizing the semantic distance between *dispatch-01* and *Transport_Person* using their dense vectors, V_{S_t} and V_{S_y} respectively, we jointly map the representations of event mention and event types into a shared semantic space, where each mention is closest to its annotated type.

After training that completes the construction of the semantic space, the compositional functions and CNNs are then used to project any new event mention (e.g., *donate-01*) into the semantic space and find its closest event type (e.g., *Donation*) (in Section 5.3). In the next sections we will elaborate each step in great detail.

3 Trigger and Argument Identification

Similar to Huang et al. (2016), we identify candidate triggers and arguments based on AMR Parsing (Wang et al., 2015b) and apply the same word sense disambiguation (WSD) tool (Zhong and Ng, 2010) to disambiguate word senses and link each sense to OntoNotes, as shown in Figure 1.

Given a sentence, we consider all noun and verb concepts that can be mapped to OntoNotes senses by WSD as candidate event triggers. In addition, the concepts that can be matched with verbs or nominal lexical units in FrameNet (Baker et al., 1998) are also considered as candidate triggers. For each candidate trigger, we consider any concepts that are involved in a subset of AMR relations as candidate arguments [3]. We manually select this subset of AMR relations that are useful for capturing generic relations between event triggers and arguments, as shown in Table 1.

Categories	Relations
Core roles	ARG0, ARG1, ARG2, ARG3, ARG4
Non-core roles	mod, location, instrument, poss, manner, topic, medium, prep-X
Temporal	year, duration, decade, weekday, time
Spatial	destination, path, location

Table 1: Event-Related AMR Relations.

4 Trigger and Type Structure Composition

As Figure 3 shows, for each candidate trigger t, we construct its event mention structure S_t based on its candidate arguments and AMR parsing. For each type y in the target event ontology, we construct a structure S_y by including its pre-defined roles and using its type as the root.

Each S_t or S_y is composed of a collection of tuples. For each event mention structure, a tuple consists of two AMR concepts and an AMR relation. For each event type structure, a tuple consists of a type name and an argument role name. Next we will describe how to compose semantic representations for event mention and event type respectively based on these structures.

Event Mention Structure For each tuple $u = \langle w_1, \lambda, w_2 \rangle$ in an event mention structure, we use a matrix to represent each AMR relation λ, and compose the semantics of λ between two concepts w_1 and w_2 as:

$$V_u = [V'_{w_1}; V'_{w_2}] = f([V_{w_1}; V_{w_2}] \cdot M_\lambda)$$

where V_{w_1}, $V_{w_2} \in \mathbb{R}^d$ are the vector representations of words w_1 and w_2. d is the dimension size of each word vector. $[\,;\,]$ denotes the concatenation of two vectors. $M_\lambda \in \mathbb{R}^{2d \times 2d}$ is the matrix representation for AMR relation λ. V_u is the composition representation of tuple u, which consists of two updated vector representations V'_{w_1}, V'_{w_2} for w_1 and w_2 by incorporating the semantics of λ.

Event Type Structure For each tuple $u' = \langle y, r \rangle$ in an event type structure, where y denotes the event type and r denotes an argument role, following Socher et al. (2013b), we assume an implicit relation exists between any pair of type and argument, and use a single and powerful tensor to represent the implicit relation:

$$V_{u'} = [V'_y; V'_r] = f([V_y; V_r]^T \cdot U^{[1:2d]} \cdot [V_y; V_r])$$

where V_y and V_r are vector representations for y and r. $U^{[1:2d]} \in \mathbb{R}^{2d \times 2d \times 2d}$ is a 3-order tensor. V'_u is the composition representation of tuple u', which consists of two updated vector representations V'_y, V'_r for y and r by incorporating the semantics of their implicit relation $U^{[1:2d]}$.

5 Trigger and Argument Classification

5.1 Trigger Classification for Seen Types

Both event mention and event type structures are relatively simple and can be represented with a set of tuples. CNNs have been demonstrated effective at capturing sentence level information by aggregating compositional n-gram representations. In order to generate structure-level representations, we use CNN to learn to aggregate all edge and tuple representations.

Input layer is a sequence of tuples, where the order of tuples is from top to bottom in the structure. Each tuple is represented by a $d \times 2$ dimensional vector, thus each mention structure and each type structure are represented as a feature map of dimensionality $d \times 2h^*$ and $d \times 2p^*$ respectively, where h^* and p^* are the maximal number of tuples for event mention and type structures. We use zero-padding to the right to make the volume of all input structures consistent.

Convolution layer Take S_t with h^* tuples: $u_1, u_2, ..., u_{h^*}$ as an example. The input matrix of S_t is a feature map of dimensionality $d \times 2h^*$. We make c_i as the concatenated embeddings of n continuous columns from the feature map, where n is the filter width and $0 < i < 2h^* + n$. A convolution operation involves a filter $W \in \mathbb{R}^{nd}$, which is applied to each sliding window c_i:

$$c'_i = \tanh(W \cdot c_i + b)$$

where c'_i is the new feature representation, and $b \in \mathbb{R}^d$ is a biased vector. We set filter width as 2 and stride as 2 to make the convolution function operate on each tuple with two input columns.

[3]On the whole ACE2005 corpus, using the AMR parser (Wang et al., 2015b), the coverage for trigger identification is 89.4% and the coverage for argument candidate identification is 66.0%.

Max-Pooling: All tuple representations $c_i^{'}$ are used to generate the representation of the input sequence by max-pooling.

Learning: For each event mention t, we name the correct type as *positive* and all the other types in the target event ontology as *negative*. To train the composition functions and CNN, we first consider the following hinge ranking loss:

$$L_1(t, y) = \sum_{j \in Y,\, j \neq y} \max\{0, m - C_{t,y} + C_{t,j}\}$$

$$C_{t,y} = \cos([V_t; V_{S_t}], [V_y; V_{S_y}])$$

where y is the positive event type for t. Y is the type set of the target event ontology. $[V_t; V_{S_t}]$ denotes the concatenation of representations of t and S_t. j is a negative event type for t from Y. m is a margin. $C_{t,y}$ denotes the cosine similarity between t and y.

The hinge loss is commonly used in zero-shot visual object classification task. However, it tends to overfit the seen types in our experiments. While clever data augmentation can help alleviate overfitting, we design two strategies: (1) we add "negative" event mentions into the training process. Here a "negative" event mention means that the mention has no positive event type among all seen types, namely it belongs to *Other*. (2) we design a new loss function as follows:

$$L_1^d(t, y) = \begin{cases} \max\limits_{j \in Y, j \neq y} \max\{0, m - C_{t,y} + C_{t,j}\}, & y \neq Other \\ \max\limits_{j \in Y^{'}, j \neq y^{'}} \max\{0, m - C_{t,y^{'}} + C_{t,j}\}, & y = Other \end{cases}$$

where Y is the type set of the event ontology. $Y^{'}$ is the seen type set. y is the annotated type. $y^{'}$ is the type which ranks the highest among all event types for event mention t, while t belongs to *Other*.

By minimizing L_1^d, we can learn the optimized model which can compose structure representations and map both event mention and types into a shared semantic space, where the positive type ranks the highest for each mention.

5.2 Argument Classification for Seen Types

For each mention, we map each candidate argument to a specific role based on the semantic similarity of the argument path. Take E1 as an example. *China* is matched to *Agent* based on the semantic similarity between *dispatch-01→ :ARG0→ China* and *Transport-Person→Agent*.

Given a trigger t and a candidate argument a, we first extract a path $S_a = (u_1, u_2, ..., u_p)$, which connects t and a and consists of p tuples. Each predefined role r is also represented as a structure by incorporating the event type, $S_r = \langle y, r \rangle$. We apply the same framework to take the sequence of tuples contained in S_a and S_r into a weight-sharing CNN to rank all possible roles for a.

$$L_2^d(a, r) = \begin{cases} \max\limits_{j \in R_y, j \neq r} \max\{0, m - C_{a,r} + C_{a,j}\} & r \neq Other \\ \max\limits_{j \in R_{Y^{'}}, j \neq r^{'}} \max\{0, m - C_{a,r^{'}} + C_{a,j}\} & r | y = Other \end{cases}$$

where R_y and $R_{Y^{'}}$ are the set of argument roles which are predefined for trigger type y and all seen types $Y^{'}$. r is the annotated role and $r^{'}$ is the argument role which ranks the highest for a when a or y is annotated as *Other*.

In our experiments, we sample various size of "negative" training data for trigger and argument labeling respectively. In the following section, we describe how the negative training instances are generated.

5.3 Zero-Shot Classification for Unseen Types

During test, given a new event mention $t^{'}$, we compute its mention structure representation for $S_{t^{'}}$ and all event type structure representations for $S_Y = \{S_{y_1}, S_{y_2}, ..., S_{y_n}\}$ using the same parameters trained from seen types. Then we rank all event types based on their similarity scores with mention $t^{'}$. The top ranked prediction for $t^{'}$ from the event type set, denoted as $\widehat{y}(t^{'}, 1)$, is given by:

$$\widehat{y}(t^{'}, 1) = \arg\max_{y \in Y} \cos([V_{t^{'}}; V_{S_{t^{'}}}], [V_y; V_{S_y}])$$

Moreover, $\widehat{y}(t^{'}, k)$ denotes the k^{th} most probable event type predicted for $t^{'}$. We will investigate the event extraction performance based on the top-k predicted event types.

After determining the type $y^{'}$ for mention $t^{'}$, for each candidate argument, we adopt the same ranking function to find the most appropriate role from the role set defined for $y^{'}$.

6 Experiments

6.1 Hyper-Parameters

We used the English Wikipedia dump to learn trigger sense and argument embeddings based on the Continuous Skip-gram model (Mikolov et al., 2013). Table 2 shows the hyper-parameters we used to train models.

Parameter Name	Value
Word Sense Embedding Size	200
Initial Learning Rate	0.1
# of Filters in Convolution Layer	500
Maximal # of Tuples for Mention Structure	10
Maximal # of Tuples for Argument Path	5
Maximal # of Tuples for Event Type Structure	5
Maximal # of Tuples for Argument Role Path	1

Table 2: Hyper-parameters.

6.2 ACE Event Classification

Setting	N	Seen Types for Training/Dev
A	1	Attack
B	3	Attack, Transport, Die
C	5	Attack, Transport, Die, Meet, Arrest-Jail
D	10	Attack, Transport, Die, Meet, Sentence, Arrest-Jail, Transfer-Money, Elect, Transfer-Ownership, End-Position

Table 3: Seen Types in Each Experiment Setting.

We first used the ACE event schema [4] as our target event ontology and assumed the boundaries of triggers and arguments as given. Of the 33 ACE event types, we selected the top-N most popular event types from ACE05 data as "seen" types, and used 90% event annotations of these for training and 10% for development. We set N as 1, 3, 5, 10 respectively. We tested the zero-shot classification performance on the annotations for the remaining 23 unseen types. Table 3 shows the types that we selected for training in each experiment setting.

The negative event mentions and arguments that belong to *Other* were sampled from the output of the system developed by Huang et al. (2016) based on ACE05 training sentences, which groups all candidate triggers and arguments into clusters based on semantic representations and assigns a type/role name to each cluster. We sampled the negative event mentions from the clusters (e.g., *Build*, *Threaten*) which do not map to ACE event types. We sampled the negative arguments from the arguments of negative event mentions. Table 4 shows the statistics of the training, development and testing data sets.

To show the effectiveness of structural similarity in our approach, we designed a baseline, WSD-Embedding, which directly maps event mentions and arguments to their candidate types and roles using our pre-trained word sense embeddings. Table 5 makes the contrast clear: structural similarity (our approach) is much more effective than lexical similarity (baseline) for both trigger and argument classification. Also, as the number of seen types in training increases, the performance of the transfer model improves.

We further evaluated the performance of our transfer approach on similar and distinct unseen types. The 33 subtypes defined in ACE fall within 8 coarse-grained main types, such as *Life* and *Justice*. Each subtype belongs to one main type. Subtypes that belong to the same main type tend to have similar structures. For example, *Trial-Hearing* and *Charge-Indict* have the same set of argument roles. For training our transfer model, we selected 4 subtypes of *Justice*: **Arrest-Jail**, **Convict**, **Charge-Indict**, **Execute**. For testing, we selected 3 other subtypes of *Justice*: *Sentence, Appeal, Release-Parole*. Additionally, we selected one subtype from each of the other seven main types for comparison. Table 6 shows that, when testing on a new unseen type, the more similar it is to the seen types, the better performance is achieved.

6.3 ACE Event Identification & Classification

The ACE2005 corpus includes the richest event annotations currently available for 33 types. However, in real-world scenarios, there may be thousands of event types of interest. To enrich the target event ontology and assess our transferable neural architecture on a large number of unseen types, when trained on limited annotations of seen types, we manually constructed a new event ontology which combined 33 ACE event types and argument roles, and 1,161 frames from FrameNet, except for the most generic frames such as *Entity* and *Locale*. Some ACE event types were easily aligned to frames, e.g., *Die* aligned to *Death*. Some frames were instead more accurately treated as inheritors of ACE types, such as *Suicide-Attack*, which inherits from *Attack*. We manually mapped the selected frames to ACE types.

We then compared our approach with the following state-of-the-art *supervised* methods:

- LSTM: A long short-term memory neural network (Hochreiter and Schmidhuber, 1997) based on distributed semantic features, similar

[4]ACE event schema specification is at: https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf

Setting	Training			Development		Test		
	# of Types, Roles	# of Events	# of Arguments	# of Events	# of Arguments	# of Types/Roles	# of Events	# of Arguments
A	1, 5	953/900	894/1,097	105/105	86/130	23/59	753	879
B	3, 14	1,803/1,500	2,035/1,791	200/200	191/237			
C	5, 18	2,033/1,300	2,281/1,503	225/225	233/241			
D	10, 37	2537/700	2,816/879	281/281	322/365			

Table 4: Statistics for Positive/Negative Instances in Training, Dev, and Test Sets for Each Experiment.

Setting	Method	Hit@k Trigger Classification (%)			Hit@k Argument Classification (%)		
		k=1	k=3	k=5	k=1	k=3	k=5
	WSD-Embedding	1.7	13.0	22.8	2.4	2.8	2.8
A	Our Approach	4.0	23.8	32.5	1.3	3.4	3.6
B		7.0	12.5	36.8	3.5	6.0	6.3
C		20.1	34.7	46.5	9.6	14.7	15.7
D		33.5	51.4	68.3	14.7	26.5	27.7

Table 5: Comparison between Structural Representation (Our Approach) and Word Sense Embedding based Approaches on Hit@K Accuracy (%) for Trigger and Argument Classification.

Type	Subtype	Hit@k Trigger Classification		
		1	3	5
Justice	Sentence	68.3	68.3	69.5
Justice	Appeal	67.5	97.5	97.5
Justice	Release-Parole	73.9	73.9	73.9
Conflict	Attack	26.5	44.5	46.7
Transaction	Transfer-Money	48.4	68.9	79.5
Business	Start-Org	0	33.3	66.7
Movement	Transport	2.6	3.7	7.8
Personnel	End-Position	9.1	50.4	53.7
Contact	Phone-Write	60.8	88.2	90.2
Life	Injure	87.6	91.0	91.0

Table 6: Performance on Various Types Using Justice Subtypes for Training

to (Feng et al., 2016).

- Joint: A structured perceptron model based on symbolic semantic features (Li et al., 2013).

For our approach, we followed the experiment setting *D* in the previous section, using the same training and development data sets for the 10 seen types, but targeted all 1,194 event types in our new event ontology, instead of just the 33 ACE event types. For evaluation, we sampled 150 sentences from the remaining ACE05 data, including 129 annotated event mentions for the 23 unseen types. For both LSTM and Joint approaches, we used the entire ACE05 annotated data for 33 ACE event types for training except for the held-out 150 evaluation sentences.

We first identified the candidate triggers and arguments, then mapped each of these to the target event ontology. We evaluated our model on their extracting of event mentions which were classified into 23 testing ACE types. Table 7 shows the performance.

To further demonstrate the effectiveness of zero-shot learning in our framework and its impact in saving human annotation effort, we used the supervised LSTM approach for comparison. The training data of LSTM contained 3,464 sentences with 905 annotated event mentions for the 23 unseen event types. We divided these event annotations into 10 subsets and successively added one subset at a time (10% of annotations) into the training data of LSTM. Figure 4 shows the LSTM learning curve. By contrast, without any annotated mentions on the 23 unseen test event types in its training set, our transfer learning approach achieved performance comparable to that of the LSTM, which was trained on 3,000 sentences[5] with 500 annotated event mentions.

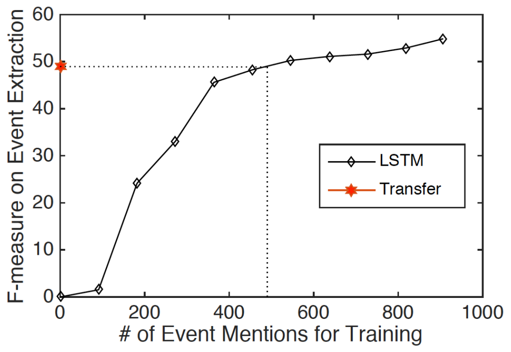


Figure 4: Comparison between Our Approach and Supervised LSTM model on 23 Unseen Event Types.

[5]The 3,000 sentences included all the sentences which even have not any event annotations.

Method	Trigger Identification			Trigger Identification + Classification			Arg Identification			Arg Identification + Classification		
	P	R	F	P	R	F	P	R	F	P	R	F
Supervised LSTM	94.7	41.8	58.0	89.4	39.5	54.8	47.8	22.6	30.6	28.9	13.7	18.6
Supervised Joint	55.8	67.4	61.1	50.6	61.2	55.4	36.4	28.1	31.7	33.3	25.7	29.0
Transfer	85.7	41.2	55.6	75.5	36.3	49.1	28.2	27.3	27.8	16.1	15.6	15.8

Table 7: Event Trigger and Argument Extraction Performance (%) on Unseen ACE Types.

6.4 Impact of AMR

Recall that we used AMR parsing output to identify triggers and arguments in constructing event structures. To assess the impact of the AMR parser (Wang et al., 2015a) on event extraction, we chose a subset of the ERE (Entity, Relation, Event) corpus (Song et al., 2015) which has ground-truth AMR annotations. This subset contains 304 documents with 1,022 annotated event mentions of 40 types. We selected the top-6 most popular event types (*Arrest-Jail, Execute, Die, Meet, Sentence, Charge-Indict*) with manual annotations of 548 event mentions as seen types. We sampled 500 negative event mentions from distinct types of clusters generated from the system (Huang et al., 2016) based on ERE training sentences. We combined the annotated events for seen types and the negative event mentions, and used 90% for training and 10% for development. For evaluation, we selected 200 sentences from the remaining ERE subset, which contains 128 *Attack* event mentions and 40 *Convict* event mentions. Table 8 shows the event extraction performances based on ground-truth AMR and system AMR respectively.

We also compared AMR analyses with Semantic Role Labeling (SRL) output (Palmer et al., 2010) by keeping only the core roles (e.g., *:ARG0, :ARG1*) from AMR annotations. As Table 8 shows, comparing the full AMR (top row) to this SRL proxy (middle row), the fine-grained AMR semantic relations such as *:location*, *:instrument* appear to be more informative for inferring event argument role labeling.

Method	Trigger Labeling			Argument Labeling		
	P	R	F_1	P	R	F_1
Perfect AMR	79.1	47.1	59.1	25.4	21.4	23.2
Perfect AMR with Core Roles only (SRL)	77.1	47.0	58.4	19.7	16.9	18.2
System AMR	85.7	32.0	46.7	22.6	15.8	18.6

Table 8: Impact of AMR and Semantic Roles on Trigger and Argument Extraction (%).

7 Related Work

Most previous event extraction methods have been based on supervised learning, using either symbolic features (Ji and Grishman, 2008; Miwa et al., 2009; Liao and Grishman, 2010; Liu et al., 2010; Hong et al., 2011; McClosky et al., 2011; Riedel and McCallum, 2011; Li et al., 2013; Liu et al., 2016) or distributional features (Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016; Nguyen et al., 2016) derived from a large amount of training data, and treating event types and argument role labels as symbols. These approaches can achieve high quality for known event types, but cannot be applied to new types without additional annotation effort. In contrast, we provide a new angle on event extraction, modeling it as a generic grounding task by taking advantage of rich semantics of event types.

Some other IE paradigms such as Open IE (Etzioni et al., 2005; Banko et al., 2007, 2008; Etzioni et al., 2011; Ritter et al., 2012), Preemptive IE (Shinyama and Sekine, 2006), On-demand IE (Sekine, 2006), Liberal IE (Huang et al., 2016, 2017), and semantic frame-based event discovery (Kim et al., 2013) can discover many events without pre-defined event schema. These paradigms however rely on information redundancy, and so they are not effective when the input data only consists of a few sentences. Our work can discover events from any size of input corpus and can also be complementary with these paradigms.

Our event extraction paradigm is similar to the task of entity linking (Ji and Grishman, 2011) in semantic mapping. However, entity linking aims to map entity mentions to the same concept, while our framework maps each event mention to a specific category. In addition, Bronstein et al. (2015) and Peng et al. (2016) employ an event-independent similarity-based function for event trigger detection, which follows few-shot learning setting and requires some trigger examples as seeds. Lu and Roth (2012) design a structure preference modeling framework, which can automatically predict argument roles without any annotated data, but it relies on manually constructed patterns.

Zero-Shot learning has been widely applied in visual object classification (Frome et al., 2013; Norouzi et al., 2013; Socher et al., 2013a; Chen et al., 2017; Li et al., 2017; Xian et al., 2017; Changpinyo et al., 2017), fine-grained name tagging (Ma et al., 2016; Qu et al., 2016), relation extraction (Verga et al., 2016; Levy et al., 2017), semantic parsing (Bapna et al., 2017) and domain adaptation (Romera-Paredes and Torr, 2015; Kodirov et al., 2015; Peng et al., 2017). In contrast to these tasks, for our case, the number of seen types in event extraction with manual annotations is quite limited. The most popular event schemas, such as ACE, define 33 event types while most visual object training sets contain more than 1,000 types. Therefore, methods proposed for zero-shot visual-object classification cannot be directly applied to event extraction due to overfitting. In this work, we designed a new loss function by creating "negative" training instances to avoid overfitting.

8 Conclusions and Future Work

In this work, we take a fresh look at the event extraction task and model it as a generic grounding problem. We propose a transferable neural architecture, which leverages existing human-constructed event schemas and manual annotations for a small set of seen types, and transfers the knowledge from the existing types to the extraction of unseen types, to improve the scalability of event extraction as well as to save human effort. To the best of our knowledge, this work is the first time that zero-shot learning has been applied to event extraction. Without any annotation, our approach can achieve performance comparable to state-of-the-art supervised models trained on a large amount of labeled data. In the future, we will extend this framework to other Information Extraction problems.

Acknowledgments

This material is based upon work supported by United States Air Force under Contract No. FA8650-17-C-7715 and ARL NS-CTA No. W911NF-09-2-0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force. or the United States Government. The United States Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proc. COLING1998*.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. ACL2013 Workshop on Linguistic Annotation and Interoperability with Discourse*.

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction for the web. In *Proc. IJCAI2007*.

M. Banko, O. Etzioni, and T. Center. 2008. The tradeoffs between open and traditional relation extraction. In *Proc. ACL-HLT2008*.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363* .

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *Proc. ACL2015*.

Soravit Changpinyo, Wei-Lun Chao, and Fei Sha. 2017. Predicting visual exemplars of unseen classes for zero-shot learning. In *Proc. ICCV2017*.

Long Chen, Hanwang Zhang, Jun Xiao, Wei Liu, and Shih-Fu Chang. 2017. Zero-shot visual recognition using semantics-preserving adversarial embedding network. *arXiv preprint arXiv:1712.01928* .

Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proc. ACL2015*.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* .

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. 2011. Open information extraction: The second generation. In *Proc. IJCAI2011*.

X. Feng, L. Huang, D. Tang, B. Qin, H. Ji, and T. Liu. 2016. A language-independent neural network for event detection. In *Proc. ACL2016*.

A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, and T. Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Proc. NIPS2013*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proc. ACL2011*.

L. Huang, T. Cassidy, X. Feng, H. Ji, C. Voss, J. Han, and A. Sil. 2016. Liberal event extraction and event schema induction. In *Proc. ACL2016*.

L. Huang, J. May, X. Pan, H. Ji, X. Ren, J. Han, L. Zhao, and J. Hendler. 2017. Liberal entity extraction: Rapid construction of fine-grained entity typing systems. *Big Data* .

H. Ji and R. Grishman. 2008. Refining event extraction through cross-document inference. In *Proc. ACL2008*.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proc. ACL-HLT2011*.

H. Kim, X. Ren, Y. Sun, C. Wang, and J. Han. 2013. Semantic frame-based document representation for comparable corpora. In *Proc. ICDM2013*.

Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. 2015. Unsupervised domain adaptation for zero-shot learning. In *Proc. ICCV2015*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115* .

Q. Li, H. Ji, and L. Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL2013*.

Yanan Li, Donghui Wang, Huanhang Hu, Yuetan Lin, and Yueting Zhuang. 2017. Zero-shot recognition using dual visual-semantic mapping paths. *arXiv preprint arXiv:1703.05002* .

S. Liao and R. Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proc. ACL2010*.

B. Liu, L. Qian, H. Wang, and G. Zhou. 2010. Dependency-driven feature-based learning for extracting protein-protein interactions from biomedical text. In *Proc. COLING2010*.

S. Liu, Y. Chen, S. He, K. Liu, and J. Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proc. ACL2016*.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proc. ACL2012*.

Y. Ma, E. Cambria, and S. Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proc. COLING2016*.

D. McClosky, M. Surdeanu, and C. D. Manning. 2011. Event extraction as dependency parsing. In *Proc. ACL2011*.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

M. Miwa, R. Stre, Y. Miyao, and J. Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proc. EMNLP2009*.

T. Nguyen, K. Cho, and R. Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proc. NAACL-HLT2016*.

T. Nguyen and R. Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proc. ACL2015*.

M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. Corrado, and J. Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650* .

M. Palmer, D. Gildea, and N. Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies* .

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proc. EMNLP2016*.

Kuan-Chuan Peng, Ziyan Wu, and Jan Ernst. 2017. Zero-shot deep domain adaptation. *arXiv preprint arXiv:1707.01922* .

J. Pustejovsky. 1991. The syntax of event structure. *Cognition* .

L. Qu, G. Ferraro, L. Zhou, W. Hou, and T. Baldwin. 2016. Named entity recognition for novel types by transfer learning. In *Proc. ACL2016*.

S. Riedel and A. McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. EMNLP2011*.

A. Ritter, O. Etzioni, and S. Clark. 2012. Open domain event extraction from twitter. In *Proc. SIGKDD2012*.

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *Proc. ICML2015*.

S. Sekine. 2006. On-demand information extraction. In *Proc. COLING-ACL2006*.

Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proc. HLT-NAACL2006*.

R. Socher, M. Ganjoo, C. Manning, and A. Ng. 2013a. Zero-shot learning through cross-modal transfer. In *Proc. NIPS2013*.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP2013*.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: Annotation of entities, relations, and events. In *Proc. NAACL-HLT2015 Workshop on on EVENTS*.

P. Verga, D. Belanger, E. Strubell, B. Roth, and A. McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proc. NAACL2016*.

C. Wang, N. Xue, and S. Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proc. ACL2015*.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *HLT-NAACL*.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. *arXiv preprint arXiv:1703.04394* .

Z. Zhong and H. T. Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proc. ACL2010*.

Recursive Neural Structural Correspondence Network for Cross-domain Aspect and Opinion Co-Extraction

Wenya Wang[†‡] **and Sinno Jialin Pan**[†]
[†]Nanyang Technological University, Singapore
[‡]SAP Innovation Center Singapore
{wa0001ya, sinnopan}@ntu.edu.sg

Abstract

Fine-grained opinion analysis aims to extract aspect and opinion terms from each sentence for opinion summarization. Supervised learning methods have proven to be effective for this task. However, in many domains, the lack of labeled data hinders the learning of a precise extraction model. In this case, unsupervised domain adaptation methods are desired to transfer knowledge from the source domain to any unlabeled target domain. In this paper, we develop a novel recursive neural network that could reduce domain shift effectively in word level through syntactic relations. We treat these relations as invariant "pivot information" across domains to build structural correspondences and generate an auxiliary task to predict the relation between any two adjacent words in the dependency tree. In the end, we demonstrate state-of-the-art results on three benchmark datasets.

1 Introduction

The problem of fine-grained opinion analysis involves extraction of opinion targets (or aspect terms) and opinion expressions (or opinion terms) from each review sentence. For example, in the sentence: "*They offer good appetizers*", the aspect and opinion terms are *appetizers* and *good* correspondingly. Many supervised deep models have been proposed for this problem (Liu et al., 2015; Yin et al., 2016; Wang et al., 2017), and obtained promising results. However, these methods fail to adapt well across domains, because the aspect terms from two different domains are usually disjoint, e.g., *laptop* v.s. *restaurant*, leading to large domain shift in the feature vector space. Though unsupervised methods (Hu and Liu, 2004; Qiu et al., 2011) can deal with data with few labels, their performance is unsatisfactory compared with supervised ones.

There have been a number of domain adaptation methods for coarse-grained sentiment classification problems across domains, where an overall sentiment polarity of a sentence or document is being predicted. Nevertheless, very few approaches exist for cross-domain fine-grained opinion analysis due to the difficulties in fine-grained adaptation, which is more challenging than coarse-grained problems. Li et al. (2012) proposed a bootstrap method based on the TrAdaBoost algorithm (Dai et al., 2007) to iteratively expand opinion and aspect lexicons in the target domain by exploiting source-domain labeled data and cross-domain common relations between aspect terms and opinion terms. However, their model requires a seed opinion lexicon in the target domain and pre-mined syntactic patterns as a bridge. Ding et al. (2017) proposed to use rules to generate auxiliary supervision on top of a recurrent neural network to learn domain-invariant hidden representation for each word. The performance highly depends on the quality of the manually defined rules and the prior knowledge of a sentiment lexicon. In addition, the recurrent structure fails to capture the syntactic interactions among words intrinsically for opinion extraction. The requirement for rules makes the above methods non-flexible.

In this paper, we propose a novel cross-domain Recursive Neural Network (RNN)[1] for aspect and opinion terms co-extraction across domains. Our motivations are twofold: 1) The dependency relations capture the interactions among different words. These relations are especially important for identifying aspect terms and opinion terms (Qiu et al., 2011; Wang et al., 2016), which are also domain-invariant within the same language. Therefore, they can be used as "pivot" information to

[1]Here, we use RNN to denote recursive neural networks, rather than recurrent neural networks.

bridge the gap between different domains. 2) Inspired by the idea of *structural learning* (Ando and Zhang, 2005), the success of target task depends on the ability of finding good predictive structures learned from other related tasks, e.g., structural correspondence learning (SCL) (Blitzer et al., 2006) for coarse-grained cross-domain sentiment classification. Here, we aim to generate an auxiliary task on dependency relation classification. Different from previous approaches, our auxiliary task and the target extraction task are of heterogeneous label spaces. We aim to integrate this auxiliary task with distributed relation representation learning into a recursive neural network.

Specifically, we generate a dependency tree for each sentence from the dependency parser and construct a unified RNN that integrates an auxiliary task into the computation of each node. The auxiliary task is to classify the dependency relation for each direct edge in the dependency tree by learning a relation feature vector. To reduce label noise brought by inaccurate parsing trees, we further propose to incorporate an autoencoder into the auxiliary task to group the relations into different clusters. Finally, to model the sequential context interaction, we develop a joint architecture that combines RNN with a sequential labeling model for aspect and opinion terms extraction. Extensive experiments are conducted to demonstrate the advantage of our proposed model.

2 Related Work

Existing works for single-domain aspect/opinion terms extraction include unsupervised methods based on association rule mining (Hu and Liu, 2004), syntactic rule propagation (Qiu et al., 2011) or topic modeling (Titov and McDonald, 2008; Lu et al., 2009; Zhang et al., 2010), as well as supervised methods based on extensive feature engineering with graphical models (Jin and Ho, 2009; Li et al., 2010) or deep learning (Liu et al., 2015; Zhang et al., 2015; Wang et al., 2017; Yin et al., 2016). Among exiting deep models, improved results are obtained using dependency relations (Yin et al., 2016; Wang et al., 2016), which indicates the significance of syntactic word interactions for target term extraction. In cross-domain setting, there are very few works for aspect/opinion terms extraction including a pipelined approach (Li et al., 2012) and a recurrent neural network (Ding et al., 2017). Both of the methods require manual construction of common and pivot syntactic patterns or rules, which are indicative of aspect or opinion words.

There have been a number of domain adaptation approaches proposed for coarse-grained sentiment classification. Among existing methods, one active line focuses on projecting original feature spaces of two domains into the same low-dimensional space to reduce domain shift using pivot features as a bridge (Blitzer et al., 2007; Pan et al., 2010; Bollegala et al., 2015; Yu and Jiang, 2016). Another line learns domain-invariant features via autoencoders (Glorot et al., 2011; Chen et al., 2012; Zhou et al., 2016). Our work is more related to the first line by utilizing pivot information to transfer knowledge across domains, but we integrate the idea into a unified deep structure that can fully utilize syntactic structure for domain adaptation in fine-grained sentiment analysis.

3 Problem Definition & Motivation

Our task is to extract opinion and aspect terms within each review sentence. We denote a sentence by a sequence of tokens $\mathbf{x} = (w_1, w_2, ..., w_n)$. The output is a sequence of token-level labels $\mathbf{y} = (y_1, y_2, ..., y_n)$, with $y_i \in \{\text{BA}, \text{IA}, \text{BO}, \text{IO}, \text{N}\}$ that represents beginning of an aspect (BA), inside of an aspect (IA), beginning of an opinion (BO), inside of an opinion (IO) or none of the above (N). A subsequence of labels started with "BA" and followed by "IA" indicates a multi-word aspect term. In unsupervised domain adaptation, we are given a set of labeled review sentences from a source domain $\mathcal{D}_S = \{(\mathbf{x}_{S_i}, \mathbf{y}_{S_i})\}_{i=1}^{n_S}$, and a set of unlabeled sentences from a target domain $\mathcal{D}_T = \{\mathbf{x}_{T_j}\}_{j=1}^{n_T}$. Our goal is to predict token-level labels on $\mathcal{D}_T$.

Existing works for cross-domain aspect and/or opinion terms extraction require hand-coded rules and a sentiment lexicon in order to transfer knowledge across domains. For example in Figure 1, given a review sentence "*They offer good appetizers*" in the source domain and "*The laptop has a nice screen*" in the target domain. If *nice* has been extracted as a common sentiment word, and "OPINION-amod-ASPECT" has been identified as a common syntactic pattern from the source domain, *screen* could be deduced as an aspect term using the identified syntactic pattern (Li et al., 2012). Similarly, Ding et al. (2017) used a set of predefined rules based on syntactic relations and a sentiment lexicon to generate auxiliary labels to learn high-level feature representations through a

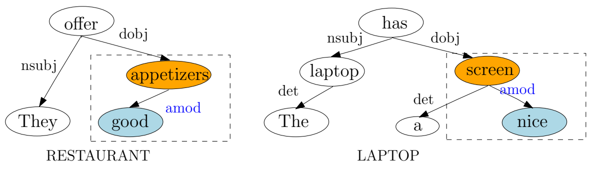


Figure 1: An example of two reviews with similar syntactic patterns.

recurrent neural network.

On one hand, these previous attempts have verified that syntactic information between words, which can be used as a bridge between domains, is crucial for domain adaptation. On the other hand, dependency-tree-based RNN (Socher et al., 2010) has proven to be effective to learn high-level feature representation of each word by encoding syntactic relations between aspect terms and opinion terms (Wang et al., 2016). With the above findings, we propose a novel RNN named **R**ecursive **N**eural **S**tructural **C**orrespondence **N**etwork (RNSCN) to learn high-level representation for each word across different domains. Our model is built upon dependency trees generated from a dependency parser. Different from previous approaches, we do not require any hand-coded rules or pre-selected pivot features to construct correspondences, but rather focus on the automatically generated dependency relations as the pivots. The model associates each direct edge in the tree with a relation feature vector, which is used to predict the corresponding dependency relation as an auxiliary task.

Note that the relation vector is the key in the model: it associates with the two interacting words and is used to construct structural correspondences between two different domains. Hence, the auxiliary task guides the learning of relation vectors, which in turn affects their correspondingly interactive words. Specifically in Figure 1, the relation vector for "amod" is computed from the features of its child and parent words, and also used to produce the hidden representation of its parent. For this relation path in both sentences, the auxiliary task enforces close proximity for these two relation vectors. This pushes the hidden representations for their parent nodes *appetizers* and *screen* closer to each other, provided that *good* and *nice* have similar representations. In a word, the auxiliary task bridges the gap between two different domains by drawing the words with similar syntactic properties closer to each other.

However, the relation vectors may be sensitive to the accuracy of the dependency parser. It might harm the learning process when some noise exists for certain relations, especially for informal texts. This problem of noisy labels has been addressed using perceptual consistency (Reed et al., 2015). Inspired by the taxonomy of dependency relations (de Marneffe and Manning, 2008), relations with similar functionalities could be grouped together, e.g., *dobj*, *iobj* and *pobj* all indicate objects. We propose to use an auto-encoder to automatically group these relations in an unsupervised manner. The reconstruction loss serves as the consistency objective that reduces label noise by aligning relation features with their intrinsic relation group.

4 Proposed Methodology

Our model consists of two components. The first component is a Recursive Neural Structural Correspondence Network (RNSCN), and the second component is a sequence labeling classifier. In this paper, we focus on Gated Recurrent Unit (GRU) as an implementation for the sequence labeling classifier. We choose GRU because it is able to deal with long-term dependencies compared to a simple Recurrent neural network and requires less parameters making it easier to train than LSTM. The resultant deep learning model is denoted by RNSCN-GRU. We also implement Conditional Random Field as the sequence labeling classifier, and denote the model by RNSCN-CRF accordingly.

The overall architecture of RNSCN-GRU without auto-encoder on relation denoising is shown in Figure 2. The left and right are two example sentences from the source and the target domain, respectively. In the first component, RNSCN, an auxiliary task to predict the dependency relation for each direct edge is integrated into a dependency-tree-based RNN. We generate a relation vector for each direct edge from its child node to parent node, and use it to predict the relation and produce the hidden representation for the parent node in the dependency tree. To address the issues of noisy relation labels, we further incorporate an auto-encoder into RNSCN, as will be shown in Figure 3.

While RNSCN mainly focuses on syntactic interactions among the words, the second component, GRU, aims to compute linear-context interactions. GRU takes the hidden representation of each word computed from RNSCN as inputs and further produces final representation of each word by taking linear contexts into consideration. We describe each component in detail in the following sections.

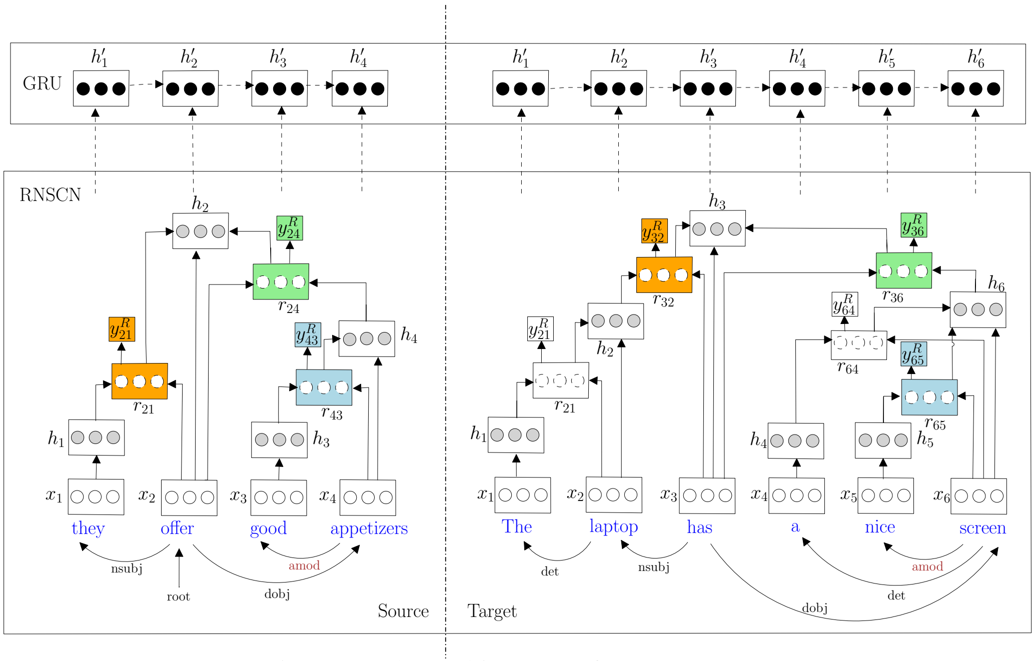


Figure 2: The architecture of RNSCN-GRU.

4.1 Recursive Neural Structural Correspondence Network

RNSCN is built on the dependency tree of each sentence, which is pre-generated from a dependency parser. Specifically, each node in the tree is associated with a word w_n, an input word embedding $\mathbf{x}_n \in \mathbb{R}^d$ and a transformed hidden representation $\mathbf{h}_n \in \mathbb{R}^d$. Each direct edge in the dependency tree associates with a relation feature vector $\mathbf{r}_{nm} \in \mathbb{R}^d$ and a true relation label vector $\mathbf{y}^R_{nm} \in \mathbb{R}^K$, where K is the total number of dependency relations, n and m denote the indices of the parent and child word of the dependency edge, respectively. Based on the dependency tree, the hidden representations are generated in a recursive manner from leaf nodes until reaching the root node. Consider the source-domain sentence shown in Figure 2 as an illustrative example, we first compute hidden representations for leaf nodes *they* and *good*:

$$\mathbf{h}_1 = \tanh(\mathbf{W}_x\mathbf{x}_1 + \mathbf{b}), \quad \mathbf{h}_3 = \tanh(\mathbf{W}_x\mathbf{x}_3 + \mathbf{b}),$$

where $\mathbf{W}_x \in \mathbb{R}^{d\times d}$ transforms word embeddings to hidden space. For non-leaf node *appetizer*, we first generate the relation vector $\mathbf{r}_{43}$ for the dependency edge $\mathbf{x}_4$ (appetizers) $\xrightarrow{\text{amod}}$ $\mathbf{x}_3$ (good) by

$$\mathbf{r}_{43} = \tanh(\mathbf{W}_h\mathbf{h}_3 + \mathbf{W}_x\mathbf{x}_4),$$

where $\mathbf{W}_h \in \mathbb{R}^{d\times d}$ transforms the hidden representation to the relation vector space. We then compute the hidden representation for *appetizer*:

$$\mathbf{h}_4 = \tanh(\mathbf{W}_{\text{amod}}\mathbf{r}_{43} + \mathbf{W}_x\mathbf{x}_4 + \mathbf{b}).$$

Moreover, the relation vector $\mathbf{r}_{43}$ is used for the auxiliary task on relation prediction:

$$\hat{\mathbf{y}}^R_{43} = \text{softmax}(\mathbf{W}_R\mathbf{r}_{43} + \mathbf{b}_R),$$

where $\mathbf{W}_R \in \mathbb{R}^{K\times d}$ is the relation classification matrix. The supervised relation classifier enforces close proximity of similar $\{\mathbf{r}_{nm}\}$'s in the distributed relation vector space. The relation features bridge the gap of word representations in different domains by incorporating them into the forward computations. In general, the hidden representation $\mathbf{h}_n$ for a non-leaf node is produced through

$$\mathbf{h}_n = \tanh\Big(\sum_{m\in\mathcal{M}_n} \mathbf{W}_{R_{nm}}\mathbf{r}_{nm} + \mathbf{W}_x\mathbf{x}_n + \mathbf{b}\Big), \quad (1)$$

where $\mathbf{r}_{nm} = \tanh(\mathbf{W}_h \cdot \mathbf{h}_m + \mathbf{W}_x \cdot \mathbf{x}_n)$, $\mathcal{M}_n$ is the set of child nodes of w_n, and $\mathbf{W}_{R_{nm}}$ is the relation transformation matrix tied with each relation R_{nm}. The predicted label vector $\hat{\mathbf{y}}^R_{nm}$ for $\mathbf{r}_{nm}$ is

$$\hat{\mathbf{y}}^R_{nm} = \text{softmax}(\mathbf{W}_R \cdot \mathbf{r}_{nm} + \mathbf{b}_R). \quad (2)$$

Here we adopt the the cross-entropy loss for relation classification between the predicted label vector $\hat{\mathbf{y}}^R_{nm}$ and the ground-truth $\mathbf{y}^R_{nm}$ to encode relation side information into feature learning:

$$\ell_R = \sum_{k=1}^{K} -\mathbf{y}^R_{nm[k]} \log \hat{\mathbf{y}}^R_{nm[k]}. \quad (3)$$

Through the auxiliary task, similar relations enforce participating words close to each other so that words with similar syntactic functionalities are clustered across domains. On the other hand, the pre-trained word embeddings group semantically-similar words. By taking them as input to RNN, together with the auxiliary task, our model encodes both semantic and syntactic information.

4.2 Reduce Label Noise with Auto-encoders

As discussed in Section 3, it might be hard to learn an accurate relation classifier when each class is a unique relation, because the dependency parser may generate incorrect relations as noisy labels. To address it, we propose to integrate an autoencoder into RNSCN. Suppose there is a set of latent groups of relations: $G = \{1, 2, ..., |G|\}$, where each relation belongs to only one group. For each relation vector, $\mathbf{r}_{nm}$, an autoencoder is performed before feeding it into the auxiliary classifier (2). The goal is to encode the relation vector to a probability distribution of assigning this relation to any group. As can be seen Figure 3, each relation vector $\mathbf{r}_{nm}$ is first passed through the autoencoder as follows,

$$p(G_{nm} = i|\mathbf{r}_{nm}) = \frac{\exp(\mathbf{r}_{nm}^\top \mathbf{W}_{enc}\mathbf{g}_i)}{\sum_{j\in G} \exp(\mathbf{r}_{nm}^\top \mathbf{W}_{enc}\mathbf{g}_j)}, \quad (4)$$

where G_{nm} denotes the inherent relation group for $\mathbf{r}_{nm}$, $\mathbf{g}_i \in \mathbb{R}^d$ represents the feature embedding for group i, and $\mathbf{W}_{enc} \in \mathbb{R}^{d\times d}$ is the encoding matrix that computes bilinear interactions between relation vector $\mathbf{r}_{nm}$ and relation group embedding $\mathbf{g}_i$. Thus, $p(G_{nm} = i|\mathbf{r}_{nm})$ represents the probability of $\mathbf{r}_{nm}$ being mapped to group i. An accumulated relation group embedding is computed as:

$$\mathbf{g}_{nm} = \sum_{i=1}^{|G|} p(G_{nm} = i|\mathbf{r}_{nm})\mathbf{g}_i. \quad (5)$$

For decoding, the decoder takes $\mathbf{g}_{nm}$ as input and tries to reconstruct the relation feature input $\mathbf{r}_{nm}$. Moreover, $\mathbf{g}_{nm}$ is also used as the higher-level feature vector for $\mathbf{r}_{nm}$ for predicting the relation label. Therefore, the objective for the auxiliary task in (3) becomes:

$$\ell_R = \ell_{R_1} + \alpha\ell_{R_2} + \beta\ell_{R_3}, \quad (6)$$

where

$$\ell_{R_1} = \|\mathbf{r}_{nm} - \mathbf{W}_{dec}\mathbf{g}_{nm}\|_2^2, \quad (7)$$

$$\ell_{R_2} = \sum_{k=1}^{K} -\mathbf{y}^R_{nm[k]} \log \hat{\mathbf{y}}^R_{nm[k]}, \quad (8)$$

$$\ell_{R_3} = \left\|\mathbf{I} - \bar{\mathbf{G}}^\top\bar{\mathbf{G}}\right\|_F^2. \quad (9)$$

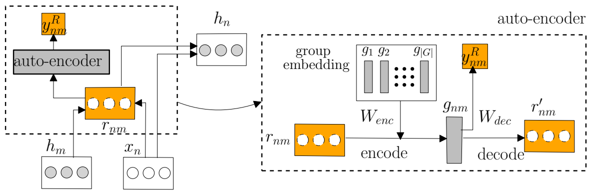


Figure 3: An autoencoder for relation grouping.

Here ℓ_{R_1} is the reconstruction loss with $\mathbf{W}_{dec}$ being the decoding matrix, ℓ_{R_2} follows (3) with $\hat{\mathbf{y}}^R_{nm} = \text{softmax}(\mathbf{W}_R\mathbf{g}_{nm} + \mathbf{b}_R)$ and ℓ_{R_3} is the regularization term on the correlations among latent groups with $\mathbf{I}$ being the identity matrix and $\bar{\mathbf{G}}$ being a normalized group embedding matrix that consists of normalized $\mathbf{g}_i$'s as column vectors. This regularization term enforces orthogonality between $\mathbf{g}_i$ and $\mathbf{g}_j$ for $i \neq j$. α and β are used to control the trade-off among different losses. With the auto-encoder, the auxiliary task of relation classification is conditioned on group assignment. The reconstruction loss further ensures the consistency between relation features and groupings, which is supposed to dominate classification loss when the observed labels are inaccurate. We denote RNSCN with auto-encoder by RNSCN$^+$.

4.3 Joint Models for Sequence Labeling

RNSCN or RNSCN$^+$ focuses on capturing and representing syntactic relations to build a bridge between domains and learn more powerful representations for tokens. However, it ignores the linear-chain correlations among tokens within a sentence, which is important for aspect and opinion terms extraction. Therefore, we propose a joint model, denoted by RNSCN-GRU (RNSCN$^+$-GRU), which integrates a GRU-based recurrent neural network on top of RNSCN (RNSCN$^+$), i.e., the input for GRU is the hidden representations $\mathbf{h}_n$ learned by RNSCN or RNSCN$^+$ for the n-th token in the sentence. For simplicity in presentation, we denote the computation of GRU by using the notation f_{GRU}. To be specific, by taking $\mathbf{h}_n$ as input, the final feature representation $\mathbf{h}'_n$ for each word is obtained through

$$\mathbf{h}'_n = f_{GRU}(\mathbf{h}'_{n-1}, \mathbf{h}_n; \mathbf{\Theta}), \quad (10)$$

where $\mathbf{\Theta}$ is the collection of the GRU parameters. The final token-level prediction is made through

$$\hat{\mathbf{y}}_n = \text{softmax}(\mathbf{W}_l \cdot \mathbf{h}'_n + \mathbf{b}_l), \quad (11)$$

where $\mathbf{W}_l \in \mathbb{R}^{5\times d'}$ transforms a d'-dimensional feature vector to class probabilities (note that we have 5 different classes as defined in Section 3).

The second joint model, namely RNSCN-CRF, combines a linear-chain CRF with RNSCN to learn the discriminative mapping from high-level features to labels. The advantage of CRF is to learn sequential interactions between each pair of adjacent words as well as labels and provide structural outputs. Formally, the joint model aims to output a sequence of labels with maximum conditional probability given its input. Denote by $\mathbf{y}$ a sequence of labels for a sentence and by $\mathbf{H}$ the embedding matrix for each sentence (each column denotes a hidden feature vector of a word in the sentence learned by RNSCN), the inference is computed as:

$$\begin{aligned}\hat{\mathbf{y}} &= \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{H}) \\ &= \arg\max_{\mathbf{y}} \frac{1}{Z(\mathbf{H})}\prod_{c\in C}\exp\langle \mathbf{W}_c, g(\mathbf{H}, \mathbf{y}_c)\rangle \quad (12)\end{aligned}$$

where C indicates the set of different cliques (unary and pairwise cliques in the context of linear-chain). $\mathbf{W}_c$ is tied for each different $\mathbf{y}_c$, which indicates the labels for clique c. The operator $\langle\cdot,\cdot\rangle$ is the element-wise multiplication, and $g(\cdot)$ produces the concatenation of $\{\mathbf{h}_n\}$'s in a context window of each word. The above two models both consider the sequential interaction of the words within each sentence, but the formalization and training are totally different. We will report the results for both joint models in the experiment section.

4.4 Training

Recall that in our cross-domain setting, the labels for terms extraction are only available in the source domain, but the auxiliary relation labels can be automatically produced for both domains via the dependency parser. Besides the source domain labeled data $\mathcal{D}_S = \{(\mathbf{x}_{S_i}, \mathbf{y}_{S_i})\}_{i=1}^{n_S}$, we denote by $\mathcal{D}_R = \{(\mathbf{r}_j, \mathbf{y}_j^R)\}_{j=1}^{n_R}$ the combined source and target domain data with auxiliary relation labels. For training, the total loss consists of token-prediction loss ℓ_S and relation-prediction loss ℓ_R:

$$\mathcal{L} = \sum_{\mathcal{D}_S} \ell_S(\mathbf{y}_{S_i}, \hat{\mathbf{y}}_{S_i}) + \gamma \sum_{\mathcal{D}_R} \ell_R(\mathbf{r}_j, \mathbf{y}_j^R), \quad (13)$$

where γ is the trade-off parameter, ℓ_S is the cross-entropy loss between the predicted extraction label in (11) and the ground-truth, and ℓ_R is defined in (6) for RNSCN$^+$ or (3) for RNSCN. For RNSCN-CRF, the loss becomes the negative log probability of the true label given the corresponding input:

$$\ell_S(\mathbf{y}_{S_i}, \hat{\mathbf{y}}_{S_i}) = -\log(\mathbf{y}_{S_i}|\mathbf{h}_{S_i}). \quad (14)$$

Dataset	Description	# Sentences	Training	Testing
R	Restaurant	5,841	4,381	1,460
L	Laptop	3,845	2,884	961
D	Device	3,836	2,877	959

Table 1: Data statistics with number of sentences.

The parameters for token-level predictions and relation-level predictions are updated jointly such that the information from the auxiliary task could be propagated to the target task to obtain better performance. This idea is in accordance with structural learning proposed by Ando and Zhang (2005), which shows that multiple related tasks are useful for finding the optimal hypothesis space. In our case, the set of multiple tasks includes the target terms extraction task and the auxiliary relation prediction task, which are closely related. The parameters are all shared across domains. The joint model is trained using back-propagation from the top layer of GRU or CRF to RNSCN until reaching to the input word embeddings in the bottom.

5 Experiments

5.1 Data & Experimental Setup

The data is taken from the benchmark customer reviews in three different domains, namely restaurant, laptop and digital devices. The restaurant domain contains a combination of restaurant reviews from SemEval 2014 task 4 subtask 1 (Pontiki et al., 2014) and SemEval 2015 task 12 subtask 1 (Pontiki et al., 2015). The laptop domain consists of laptop reviews from SemEval 2014 task 4 subtask 1. For digital device, we take reviews from (Hu and Liu, 2004) containing sentences from 5 digital devices. The statistics for each domain are shown in Table 1. In our experiments, we randomly split the data in each domain into training set and testing set with the proportion being 3:1. To obtain more rigorous result, we make three random splits for each domain and test the learned model on each split. The number of sentences for training and testing after each split is also shown in Table 1. Each sentence is labeled with aspect terms and opinion terms.

For each cross-domain task, we conduct both inductive and transductive experiments. Specifically, we train our model only on the training sets from both (labeled) source and (unlabeled) target domains. For testing, the inductive results are obtained using the test data from the target domain, and the transductive results are obtained using the (unlabeled) training data from the target domain.

The evaluation metric we used is F1 score. Following the setting from existing work, only exact match could be counted as correct.

For experimental setup, we use Stanford Dependency Parser (Klein and Manning, 2003) to generate dependency trees. There are in total 43 different dependency relations, i.e. 43 classes for the auxiliary task. We set the number of latent relation groups as 20. The input word features for RNSCN are pre-trained word embeddings using word2vec (Mikolov et al., 2013) which is trained on 3M reviews from the Yelp dataset[2] and electronics dataset in Amazon reviews[3] (McAuley et al., 2015). The dimension of word embeddings is 100. Because of the relatively small size of the training data compared with the number of parameters, we firstly pre-train RNSCN for 5 epochs with mini-batch size 30 and rmsprop initialized at 0.01. The joint model of RNSCN$^+$-GRU is then trained with rmsprop initialized at 0.001 and mini-batch size 30. The trade-off parameter α, β and γ are set to be 1, 0.001 and 0.1, respectively. The hidden-layer dimension for GRU is 50, and the context window size is 3 for input feature vectors of GRU. For the joint model of RNSCN-CRF, we implement SGD with a decaying learning rate initialized at 0.02. The context window size is also 3 in this case. Both joint models are trained for 10 epochs.

5.2 Comparison & Results

We compared our proposed model with several baselines and variants of the proposed model:

- **RNCRF**: A joint model of recursive neural network and CRF proposed by (Wang et al., 2016) for single-domain aspect and opinion terms extraction. We make all the parameters shared across domains for target prediction.
- **RNGRU**: A joint model of RNN and GRU. The hidden layer of RNN is taken as input for GRU. We share all the parameters across domains, similar to RNCRF.
- **CrossCRF**: A linear-chain CRF with hand-engineered features that are useful for cross-domain settings (Jakob and Gurevych, 2010), e.g., POS tags, dependency relations.
- **RAP**: The Relational Adaptive bootstraPping method proposed by (Li et al., 2012) that uses TrAdaBoost to expand lexicons.
- **Hier-Joint**: A recent deep model proposed by Ding et al. (2017) that achieves state-of-the-art performance on aspect terms extraction across domains.
- **RNSCN-GRU**: Our proposed joint model integrating auxiliary relation prediction task into RNN that is further combined with GRU.
- **RNSCN-CRF**: The second proposed model similar to RNSCN-GRU, which replace GRU with CRF.
- **RNSCN$^+$-GRU**: Our final joint model with auto-encoders to reduce auxiliary label noise.

Note that we do not implement other recent deep adaptation models for comparison (Chen et al., 2012; Yang and Hospedales, 2015), because Hier-Joint (Ding et al., 2017) has already demonstrated better performances than these models. The overall comparison results with the baselines are shown in Table 2 with average F1 scores and standard deviations over three random splits. Clearly, the results for aspect terms (AS) transfer are much lower than opinion terms (OP) transfer, which indicate that the aspect terms are usually quite different across domains, whereas the opinion terms could be more common and similar. Hence the ability to adapt the aspect extraction from the source domain to the target domain becomes more crucial. On this behalf, our proposed model shows clear advantage over other baselines for this more difficult transfer problem. Specifically, we achieve 6.77%, 5.88%, 10.55% improvement over the best-performing baselines for aspect extraction in R$\rightarrow$L, L$\rightarrow$D and D$\rightarrow$L, respectively. By comparing with RNCRF and RNGRU, we show that the structural correspondence network is indeed effective when integrated into RNN.

To show the effect of the integration of the autoencoder, we conduct experiments over different variants of the proposed model in Table 3. RNSCN-GRU represents the model without autoencoder, which achieves much better F1 scores on most experiments compared with the baselines in Table 2. RNSCN$^+$-GRU outperforms RNSCN-GRU in almost all experiments. This indicates the autoencoder automatically learns data-dependent groupings, which is able to reduce unnecessary label noise. To further verify that the autoencoder indeed reduces label noise when the parser is inaccurate, we generate new noisy parse trees by replacing some relations within each sentence with a random

[2]http://www.yelp.com/dataset_challenge

[3]http://jmcauley.ucsd.edu/data/amazon/links.html

Models	R→L		R→D		L→R		L→D		D→R		D→L	
	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP
CrossCRF	19.72 (1.82)	59.20 (1.34)	21.07 (0.44)	52.05 (1.67)	28.19 (0.58)	65.52 (0.89)	29.96 (1.69)	56.17 (1.49)	6.59 (0.49)	39.38 (3.06)	24.22 (2.54)	46.67 (2.43)
RAP	25.92 (2.75)	62.72 (0.49)	22.63 (0.52)	54.44 (2.20)	46.90 (1.64)	67.98 (1.05)	34.54 (0.64)	54.25 (1.65)	45.44 (1.61)	60.67 (2.15)	28.22 (2.42)	59.79 (4.18)
Hier-Joint	33.66 (1.47)	- -	33.20 (0.52)	- -	48.10 (1.45)	- -	31.25 (0.49)	- -	47.97 (0.46)	- -	34.74 (2.27)	-
RNCRF	24.26 (3.97)	60.86 (3.35)	24.31 (2.57)	51.28 (1.78)	40.88 (2.09)	66.50 (1.48)	31.52 (1.40)	55.85 (1.09)	34.59 (1.34)	63.89 (1.59)	40.59 (0.80)	60.17 (1.20)
RNGRU	24.23 (2.41)	60.65 (1.04)	20.49 (2.68)	52.28 (2.69)	39.78 (0.61)	62.99 (0.95)	32.51 (1.12)	52.24 (2.37)	38.15 (2.82)	64.21 (1.11)	39.44 (2.79)	60.85 (1.25)
RNSCN-CRF	35.26 (1.31)	61.67 (1.35)	32.00 (1.48)	52.81 (1.29)	**53.38** (1.49)	67.60 (0.99)	34.63 (1.38)	56.22 (1.10)	48.13 (0.71)	65.06 (0.66)	46.71 (1.16)	61.88 (1.52)
RNSCN-GRU	37.77 (0.45)	62.35 (1.85)	33.02 (0.58)	57.54 (1.27)	53.18 (0.75)	71.44 (0.97)	35.65 (0.77)	60.02 (0.80)	**49.62** (0.34)	69.42 (2.27)	45.92 (1.14)	63.85 (1.97)
RNSCN$^+$-GRU	**40.43** (0.96)	**65.85** (1.50)	**35.10** (0.62)	**60.17** (0.75)	52.91 (1.82)	**72.51** (1.03)	**40.42** (0.70)	**61.15** (0.60)	48.36 (1.14)	**73.75** (1.76)	**51.14** (1.68)	**71.18** (1.58)

Table 2: Comparisons with different baselines.

Models	R→L		R→D		L→R		L→D		D→R		D→L	
	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP
RNSCN-GRU	37.77	62.35	33.02	57.54	**53.18**	71.44	35.65	60.02	**49.62**	69.42	45.92	63.85
RNSCN-GRU (r)	32.97	50.18	26.21	53.58	35.88	65.73	32.87	57.57	40.03	67.34	40.06	59.18
RNSCN$^+$-GRU	**40.43**	**65.85**	**35.10**	**60.17**	52.91	**72.51**	**40.42**	**61.15**	48.36	**73.75**	**51.14**	**71.18**
RNSCN$^+$-GRU (r)	39.27	59.41	33.42	57.24	45.79	69.96	38.21	59.12	45.36	72.84	50.45	68.05

Table 3: Comparisons with different variants of the proposed model.

		R→L		R→D		L→R		L→D		D→R		D→L	
		AS	OP	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP
OUT	Hier-Joint	33.66	-	33.20	-	48.10	-	31.25	-	47.97	-	34.74	-
	RNSCN$^+$-GRU*	39.06	-	34.07	-	47.98	-	38.51	-	47.49	-	48.49	-
	RNSCN$^+$	31.60	**65.89**	24.37	60.01	39.58	71.03	34.40	60.47	41.02	71.23	45.54	69.00
	RNSCN$^+$-GRU	**40.43**	65.85	**35.10**	**60.17**	**52.91**	**72.51**	**40.42**	**61.15**	**48.36**	**73.75**	**51.14**	**71.18**
IN	Hier-Joint	32.41	-	29.79	-	47.04	-	31.26	-	**47.41**	-	33.80	-
	RNSCN$^+$-GRU*	40.34	-	30.75	-	48.69	-	37.40	-	46.49	-	48.50	-
	RNSCN$^+$	30.76	63.65	22.48	59.24	39.54	70.25	35.32	60.00	37.75	70.64	43.72	68.27
	RNSCN$^+$-GRU	**41.27**	**65.44**	**33.58**	**60.28**	**52.48**	**72.10**	**39.73**	**60.18**	47.10	**72.19**	**50.23**	**70.21**

Table 4: Comparisons with different transfer setting.

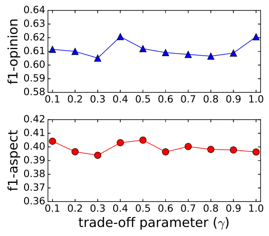
relation. Specifically, in each source domain, for each relation that connects to any aspect or opinion word, it has 0.5 probability of being replaced by any other relation. In Table 3, We denote the model with noisy relations with (r). Obviously, the performance of RNSCN-GRU without an autoencoder significantly deteriorates when the auxiliary labels are very noisy. On the contrary, RNSCN$^+$-GRU (r) achieves acceptable results compared to RNSCN$^+$-GRU. This proves that the autoencoder makes the model more robust to label noise and helps to adapt the information more accurately to the target data. Note that a large drop for $L \rightarrow R$ in aspect extraction might be caused by a large portion of noisy replacements for this particular data which makes it too hard to train a good classifier. This may not greatly influence opinion extraction, as shown, because the two domains usually share many common opinion terms. However, the significant difference in aspect terms makes the learning more dependent on common relations.

The above comparisons are made using the test data from target domains which are not available during training (i.e., the inductive setting). For more complete comparison, we also conduct experiments in the transductive setting. We pick our best model RNSCN$^+$-GRU, and show the effect of different components. To do that, we first remove the sequential structure on top, resulting in RNSCN$^+$. Moreover, we create another variant by removing opinion term labels to show the effect of the double propogation between aspect terms and opinion terms. The resulting model is named RNSCN$^+$-GRU*. As shown in Table 4, we denote by OUT and IN the inductive and transductive setting, respectively. The results shown are the average F1 scores among three splits[4]. In general, RNSCN$^+$-GRU shows similar performances for both inductive and transductive settings. This indicates the

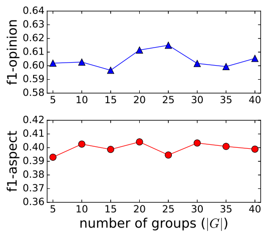
[4]We omit standard deviation here due to the limit of space.

G	Word
1	this, the, their, my, here, it, I, our, not
2	quality, jukebox, maitre-d, sauces, portions, volume, friend, noodles, calamari
3	in, slightly, often, overall, regularly, since, back, much, ago
4	handy, tastier, white, salty, right, vibrant, first, ok
5	get, went, impressed, had, try, said, recommended, call, love
6	is, are, feels, believes, seems, like, will, would

Table 5: Case studies on word clustering

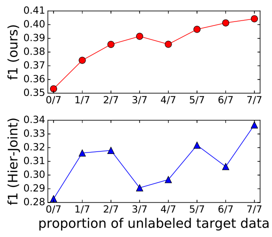


(a) On trade-off parameter.

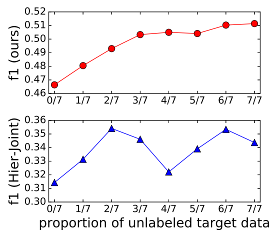


(b) On number of groups.

Figure 4: Sensitivity studies for L$\rightarrow$D.



(a) F1-aspect on R$\rightarrow$L



(b) F1-aspect on D$\rightarrow$L

Figure 5: F1 vs proportion of unlabeled target data.

robustness and the ability to learn well when test data is not presented during training. Without opinion labels, RNSCN$^+$-GRU* still achieves better results than Hier-Joint most of the time. Its lower performance compared to RNSCN$^+$-GRU also indicates that in the cross-domain setting, the dual information between aspects and opinions is beneficial to find appropriate and discriminative relation feature space. Finally, the results for RNSCN$^+$ by removing GRU are lower than the joint model, which proves the importance of combining syntactic tree structure with sequential modeling.

To qualitatively show the effect of the auxiliary task with auto-encoders for clustering syntactically similar words across domains, we provide some case studies on the predicted groups of some words in Table 5. Specifically, for each relation in the dependency tree, we use (4) to obtain the most probable group to assign the word in the child node. The left column shows the predicted group index with the right column showing the corresponding words. Clearly, the words in the same group have similar syntactic functionalities, whereas the word types vary across groups.

In the end, we verify the robustness and capability of the model by conducting sensitivity studies and experiments with varying number of unlabeled target data for training, respectively. Figure 4 shows the sensitivity test for L$\rightarrow$D, which indicates that changing of the trade-off parameter γ or the number of groups $|G|$ does not affect the model's performance greatly, i.e., less than 1% for aspect extraction and 2% for opinion extraction. This proves that our model is robust and stable against small variations. Figure 5 compares the results of RNSCN$^+$-GRU with Hier-Joint when increasing the proportion of unlabeled target training data from 0 to 1. Obviously, our model shows steady improvement with the increasing number of unlabeled target data. This pattern proves our model's capability of learning from target domain for adaptation.

6 Conclusion

We propose a novel dependency-tree-based RNN, namely RNSCN (or RNSCN$^+$), for domain adaptation. The model integrates an auxiliary task into representation learning of nodes in the dependency tree. The adaptation takes place in a common relation feature space, which builds the structural correspondences using syntactic relations among the words in each sentence. We further develop a joint model to combine RNSCN/RNSCN$^+$ with a sequential labeling model for terms extraction.

Acknowledgements

This work is supported by NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, MOE AcRF Tier-1 grant 2016-T1-001-159, and Fuji Xerox Corporation through joint research on Multilingual Semantic Analysis.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR* 6:1817–1853.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boomboxes and blenders:

Domain adaptation for sentiment classification. In *ACL*. pages 187–205.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*. pages 120–128.

Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *ACL*. pages 730–740.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *ICML*. pages 1627–1634.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for transfer learning. In *ICML*. pages 193–200.

Marie C. de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *CrossParser*. pages 1–8.

Ying Ding, Jianfei Yu, and Jing Jiang. 2017. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *AAAI*. pages 3436–3442.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*. pages 97–110.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*. pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *EMNLP*. pages 1035–1045.

Wei Jin and Hung Hay Ho. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *ICML*. pages 465–472.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*. pages 423–430.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *COLING*. pages 653–661.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *ACL*. pages 410–419.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*. pages 1433–1443.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *WWW*. pages 131–140.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. pages 43–52.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *WWW*. pages 751–760.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *SemEval 2015*. pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*. pages 27–35.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.* 37(1):9–27.

Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training deep neural networks on noisy labels with bootstrapping. In *ICLR 2015*.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks. In *NIPS Workshop*. pages 1–9.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *WWW*. pages 111–120.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *EMNLP*. pages 616–626.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer tensor network for co-extraction of aspect and opinion terms. In *AAAI*. pages 3316–3322.

Yongxin Yang and Timothy M. Hospedales. 2015. A unified perspective on multi-domain and multi-task learning. In *ICLR*.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *IJCAI*. pages 2979–2985.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP*. pages 236–246.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *COLING*. pages 1462–1470.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *EMNLP*.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016. Bi-transferring deep neural networks for domain adaptation. In *ACL*. pages 322–332.

Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning

Baolin Peng* **Xiujun Li**† **Jianfeng Gao**† **Jingjing Liu**† **Kam-Fai Wong***‡
†Microsoft Research, Redmond, WA, USA
*The Chinese University of Hong Kong, Hong Kong
‡MoE Key Lab of High Confidence Software Technologies, China
{blpeng, kfwong}@se.cuhk.edu.hk
{xiul,jfgao,jingjl}@microsoft.com

Abstract

Training a task-completion dialogue agent via reinforcement learning (RL) is costly because it requires many interactions with real users. One common alternative is to use a user simulator. However, a user simulator usually lacks the language complexity of human interlocutors and the biases in its design may tend to degrade the agent. To address these issues, we present Deep Dyna-Q, which to our knowledge is the first deep RL framework that integrates planning for task-completion dialogue policy learning. We incorporate into the dialogue agent a model of the environment, referred to as the *world model*, to mimic real user response and generate simulated experience. During dialogue policy learning, the world model is constantly updated with real user experience to approach real user behavior, and in turn, the dialogue agent is optimized using both real experience and simulated experience. The effectiveness of our approach is demonstrated on a movie-ticket booking task in both simulated and human-in-the-loop settings[1].

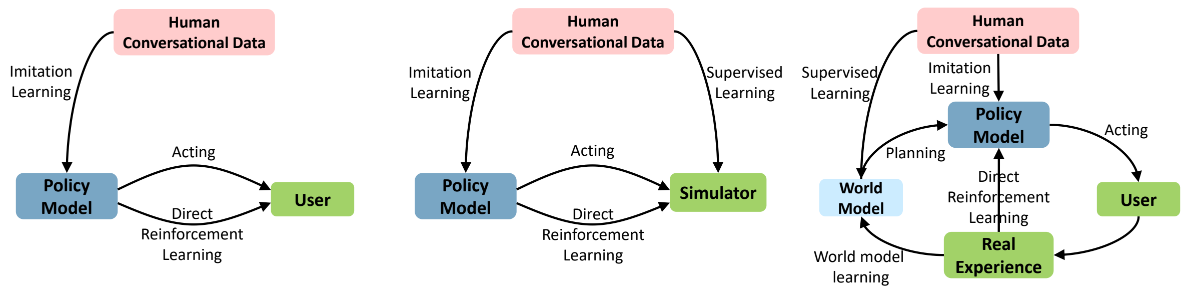
1 Introduction

Learning policies for task-completion dialogue is often formulated as a reinforcement learning (RL) problem (Young et al., 2013; Levin et al., 1997). However, applying RL to real-world dialogue systems can be challenging, due to the constraint that an RL learner needs an environment to operate in. In the dialogue setting, this requires a dialogue agent to interact with real users and adjust its policy in an online fashion, as illustrated in Figure 1(a). Unlike simulation-based games such as Atari games (Mnih et al., 2015) and AlphaGo (Silver et al., 2016a, 2017) where RL has made its greatest strides, task-completion dialogue systems may incur significant real-world cost in case of failure. Thus, except for very simple tasks (Singh et al., 2002; Gašić et al., 2010, 2011; Pietquin et al., 2011; Li et al., 2016a; Su et al., 2016b), RL is too expensive to be applied to real users to train dialogue agents from scratch.

One strategy is to convert human-interacting dialogue to a simulation problem (similar to Atari games), by building a user simulator using human conversational data (Schatzmann et al., 2007; Li et al., 2016b). In this way, the dialogue agent can learn its policy by interacting with the simulator instead of real users (Figure 1(b)). The simulator, in theory, does not incur any real-world cost and can provide unlimited simulated experience for reinforcement learning. The dialogue agent trained with such a user simulator can then be deployed to real users and further enhanced by only a small number of human interactions. Most of recent studies in this area have adopted this strategy (Su et al., 2016a; Lipton et al., 2016; Zhao and Eskenazi, 2016; Williams et al., 2017; Dhingra et al., 2017; Li et al., 2017; Liu and Lane, 2017; Peng et al., 2017b; Budzianowski et al., 2017; Peng et al., 2017a).

However, user simulators usually lack the conversational complexity of human interlocutors, and the trained agent is inevitably affected by biases in the design of the simulator. Dhingra et al. (2017) demonstrated a significant discrepancy in a simulator-trained dialogue agent when evaluated with simulators and with real users. Even more challenging is the fact that there is no universally accepted metric to evaluate a user simulator (Pietquin and Hastie, 2013). Thus, it remains

[1]The source code of this work is available at https://github.com/MiuLab/DDQ



(a) Learning with real users (b) Learning with user simulators (c) Learning with real users via DDQ

Figure 1: Three strategies of learning task-completion dialogue policies via RL.

controversial whether training task-completion dialogue agent via simulated users is a valid approach.

We propose a new strategy of learning dialogue policy by interacting with real users. Compared to previous works (Singh et al., 2002; Li et al., 2016a; Su et al., 2016b; Papangelis, 2012), our dialogue agent learns in a much more efficient way, using only a small number of real user interactions, which amounts to an affordable cost in many nontrivial dialogue tasks.

Our approach is based on the Dyna-Q framework (Sutton, 1990) where planning is integrated into policy learning for task-completion dialogue. Specifically, we incorporate a model of the environment, referred to as the *world model*, into the dialogue agent, which simulates the environment and generates simulated user experience. During the dialogue policy learning, real user experience plays two pivotal roles: first, it can be used to improve the world model and make it behave more like real users, via supervised learning; second, it can also be used to directly improve the dialogue policy via RL. The former is referred to as *world model learning*, and the latter *direct reinforcement learning*. Dialogue policy can be improved either using real experience directly (i.e., direct reinforcement learning) or via the world model indirectly (referred to as *planning* or *indirect reinforcement learning*). The interaction between world model learning, direct reinforcement learning and planning is illustrated in Figure 1(c), following the Dyna-Q framework (Sutton, 1990).

The original papers on Dyna-Q and most its early extensions used tabular methods for both planning and learning (Singh, 1992; Peng and Williams, 1993; Moore and Atkeson, 1993; Kuvayev and Sutton, 1996). This table-lookup representation limits its application to small problems only. Sutton et al. (2012) extends the Dyna architecture to linear function approximation, making it applicable to larger problems. In the dialogue setting, we are dealing with a much larger action-state space. Inspired by Mnih et al. (2015), we propose Deep Dyna-Q (DDQ) by combining Dyna-Q with deep learning approaches to representing the state-action space by neural networks (NN).

By employing the world model for planning, the DDQ method can be viewed as a model-based RL approach, which has drawn growing interest in the research community. However, most model-based RL methods (Tamar et al., 2016; Silver et al., 2016b; Gu et al., 2016; Racanière et al., 2017) are developed for simulation-based, synthetic problems (e.g., games), but not for human-in-the-loop, real-world problems. To these ends, our main contributions in this work are two-fold:

- We present Deep Dyna-Q, which to the best of our knowledge is the first deep RL framework that incorporates planning for task-completion dialogue policy learning.
- We demonstrate that a task-completion dialogue agent can efficiently adapt its policy on the fly, by interacting with real users via RL. This results in a significant improvement in success rate on a nontrivial task.

2 Dialogue Policy Learning via Deep Dyna-Q (DDQ)

Our DDQ dialogue agent is illustrated in Figure 2, consisting of five modules: (1) an LSTM-based natural language understanding (NLU) module (Hakkani-Tür et al., 2016) for identifying user intents and extracting associated slots; (2) a state tracker (Mrkšić et al., 2016) for tracking the dialogue states; (3) a dialogue policy which selects

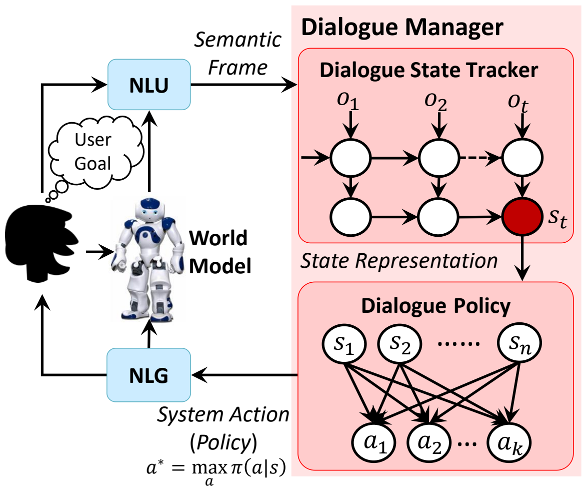


Figure 2: Illustration of the task-completion DDQ dialogue agent.

the next action[2] based on the current state; (4) a model-based natural language generation (NLG) module for converting dialogue actions to natural language response (Wen et al.); and (5) a world model for generating simulated user actions and simulated rewards.

As illustrated in Figure 1(c), starting with an initial dialogue policy and an initial world model (both trained with pre-collected human conversational data), the training of the DDQ agent consists of three processes: (1) *direct reinforcement learning*, where the agent interacts with a real user, collects real experience and improves the dialogue policy; (2) *world model learning*, where the world model is learned and refined using real experience; and (3) *planning*, where the agent improves the dialogue policy using simulated experience.

Although these three processes conceptually can occur simultaneously in the DDQ agent, we implement an iterative training procedure, as shown in Algorithm 1, where we specify the order in which they occur within each iteration. In what follows, we will describe these processes in details.

2.1 Direct Reinforcement Learning

In this process (lines 5-18 in Algorithm 1) we use the DQN method (Mnih et al., 2015) to improve the dialogue policy based on real experience. We consider task-completion dialogue as a Markov Decision Process (MDP), where the agent interacts with a user in a sequence of actions to accomplish a user goal. In each step, the agent observes the dialogue state s, and chooses the action a to execute, using an ϵ-greedy policy that selects a random action with probability ϵ or otherwise follows the greedy policy $a = \text{argmax}_{a'} Q(s, a'; \theta_Q)$. $Q(s, a; \theta_Q)$ which is the approximated value function, implemented as a Multi-Layer Perceptron (MLP) parameterized by θ_Q. The agent then receives reward[3] r, observes next user response a^u, and updates the state to s'. Finally, we store the experience (s, a, r, a^u, s') in the replay buffer D^u. The cycle continues until the dialogue terminates.

We improve the value function $Q(s, a; \theta_Q)$ by adjusting θ_Q to minimize the mean-squared loss function, defined as follows:

$$\begin{aligned}\mathcal{L}(\theta_Q) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}^u}[(y_i - Q(s, a; \theta_Q))^2] \\ y_i &= r + \gamma \max_{a'} Q'(s', a'; \theta_{Q'}) \qquad (1)\end{aligned}$$

where $\gamma \in [0, 1]$ is a discount factor, and $Q'(.)$ is the target value function that is only periodically updated (line 42 in Algorithm 1). By differentiating the loss function with respect to θ_Q, we arrive at the following gradient:

$$\begin{aligned}\nabla_{\theta_Q}\mathcal{L}(\theta_Q) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}^u}[(r+ \\ \gamma \max_{a'} Q'(s', a'; \theta_{Q'}) - Q(s, a; \theta_Q)) \qquad (2) \\ \nabla_{\theta_Q} Q(s, a; \theta_Q)]\end{aligned}$$

As shown in lines 16-17 in Algorithm 1, in each iteration, we improve $Q(.)$ using minibatch Deep Q-learning.

2.2 Planning

In the planning process (lines 23-41 in Algorithm 1), the world model is employed to generate simulated experience that can be used to improve dialogue policy. K in line 24 is the number of planning steps that the agent performs per step of direct reinforcement learning. If the world model is able to accurately simulate the environment, a big K can be used to speed up the policy learning. In DDQ, we use two replay buffers, D^u for storing real experience and D^s for simulated experience. Learning and planning are accomplished

[2]In the dialogue scenario, actions are dialogue-acts, consisting of a single act and a (possibly empty) collection of $(slot = value)$ pairs (Schatzmann et al., 2007).

[3]In the dialogue scenario, reward is defined to measure the degree of success of a dialogue. In our experiment, for example, success corresponds to a reward of 80, failure to a reward of -40, and the agent receives a reward of -1 at each turn so as to encourage shorter dialogues.

Algorithm 1 Deep Dyna-Q for Dialogue Policy Learning

```
Require: $N, \epsilon, K, L, C, Z$
Ensure: $Q(s,a;\theta_Q)$, $M(s,a;\theta_M)$
 1: initialize $Q(s,a;\theta_Q)$ and $M(s,a;\theta_M)$ via pre-training on human conversational data
 2: initialize $Q'(s,a;\theta_{Q'})$ with $\theta_{Q'} = \theta_Q$
 3: initialize real experience replay buffer $D^u$ using Reply Buffer Spiking (RBS), and simulated experience replay buffer $D^s$ as empty
 4: for $n$=1:$N$ do
 5:     # Direct Reinforcement Learning starts
 6:     user starts a dialogue with user action $a^u$
 7:     generate an initial dialogue state $s$
 8:     while $s$ is not a terminal state do
 9:         with probability $\epsilon$ select a random action $a$
10:         otherwise select $a = \text{argmax}_{a'} Q(s,a';\theta_Q)$
11:         execute $a$, and observe user response $a^u$ and reward $r$
12:         update dialogue state to $s'$
13:         store $(s,a,r,a^u,s')$ to $D^u$
14:         $s = s'$
15:     end while
16:     sample random minibatches of $(s,a,r,s')$ from $D^u$
17:     update $\theta_Q$ via $Z$-step minibatch Q-learning according to Equation (2)
18:     # Direct Reinforcement Learning ends
19:     # World Model Learning starts
20:     sample random minibatches of training samples $(s,a,r,a^u,s')$ from $D^u$
21:     update $\theta_M$ via $Z$-step minibatch SGD of multi-task learning
22:     # World Model Learning ends
23:     # Planning starts
24:     for $k$=1:$K$ do
25:         $t$ = FALSE, $l = 0$
26:         sample a user goal $G$
27:         sample user action $a^u$ from $G$
28:         generate an initial dialogue state $s$
29:         while $t$ is FALSE $\wedge\, l \leq L$ do
30:             with probability $\epsilon$ select a random action $a$
31:             otherwise select $a = \text{argmax}_{a'} Q(s,a';\theta_Q)$
32:             execute $a$
33:             world model responds with $a^u$, $r$ and $t$
34:             update dialogue state to $s'$
35:             store $(s,a,r,s')$ to $D^s$
36:             $l = l+1$, $s = s'$
37:         end while
38:         sample random minibatches of $(s,a,r,s')$ from $D^s$
39:         update $\theta_Q$ via $Z$-step minibatch Q-learning according to Equation (2)
40:     end for
41:     # Planning ends
42:     every $C$ steps reset $\theta_{Q'} = \theta_Q$
43: end for
```

by the same DQN algorithm, operating on real experience in D^u for learning and on simulated experience in D^s for planning. Thus, here we only describe the way the simulated experience is generated.

Similar to Schatzmann et al. (2007), at the beginning of each dialogue, we uniformly draw a user goal $G = (C, R)$, where C is a set of constraints and R is a set of requests (line 26 in Algorithm 1). For movie-ticket booking dialogues, constraints are typically the name and the date of the movie, the number of tickets to buy, etc. Requests can contain these slots as well as the location of the theater, its start time, etc. Table 3 presents some sampled user goals and dialogues generated by simulated and real users, respectively. The first user action a^u (line 27) can be either a request or an inform dialogue-act. A request, such as `request(theater; moviename=batman)`, consists of a request slot and multiple ($\geqslant$ 1) constraint slots, uniformly sampled from R and C, respectively. An inform contains constraint slots only. The user action can also be converted to natural language via NLG, e.g., `"which theater will show batman?"`

In each dialogue turn, the world model takes as input the current dialogue state s and the last agent action a (represented as an one-hot vector), and generates user response a^u, reward r, and a binary variable t, which indicates whether the dialogue terminates (line 33). The generation is accomplished using the world model $M(s,a;\theta_M)$, a MLP shown in Figure 3, as follows:

$$\begin{aligned} h &= \tanh(W_h(s,a) + b_h) \\ r &= W_r h + b_r \\ a^u &= \text{softmax}(W_a h + b_a) \\ t &= \text{sigmoid}(W_t h + b_t) \end{aligned}$$

where (s,a) is the concatenation of s and a, and W and b are parameter matrices and vectors, respectively.

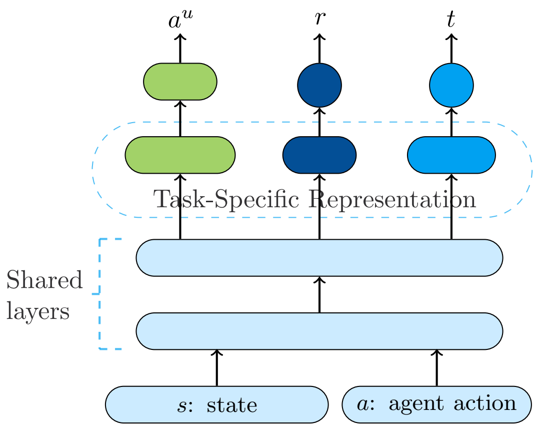


Figure 3: The world model architecture.

2.3 World Model Learning

In this process (lines 19-22 in Algorithm 1), $M(s, a; \theta_M)$ is refined via minibatch SGD using real experience in the replay buffer D^u. As shown in Figure 3, $M(s, a; \theta_M)$ is a multi-task neural network (Liu et al., 2015) that combines two classification tasks of simulating a^u and t, respectively, and one regression task of simulating r. The lower layers are shared across all tasks, while the top layers are task-specific.

3 Experiments and Results

We evaluate the DDQ method on a movie-ticket booking task in both simulation and human-in-the-loop settings.

3.1 Dataset

Raw conversational data in the movie-ticket booking scenario was collected via Amazon Mechanical Turk. The dataset has been manually labeled based on a schema defined by domain experts, as shown in Table 4, which consists of 11 dialogue acts and 16 slots. In total, the dataset contains 280 annotated dialogues, the average length of which is approximately 11 turns.

3.2 Dialogue Agents for Comparison

To benchmark the performance of DDQ, we have developed different versions of task-completion dialogue agents, using variations of Algorithm 1.

- A **DQN** agent is learned by standard DQN, implemented with direct reinforcement learning only (lines 5-18 in Algorithm 1) in each epoch.
- The **DDQ(**K**)** agents are learned by DDQ of Algorithm 1, with an initial world model pre-trained on human conversational data, as described in Section 3.1. K is the number of planning steps. We trained different versions of DDQ(K) with different K's.
- The **DDQ(**K**, rand-init** θ_M**)** agents are learned by the DDQ method with a randomly initialized world model.
- The **DDQ(**K**, fixed** θ_M**)** agents are learned by DDQ with an initial world model pre-trained on human conversational data. But the world model is not updated afterwards. That is, the *world model learning* part in Algorithm 1 (lines 19-22) is removed. The DDQ(K, fixed θ_M) agents are evaluated in the simulation setting only.
- The **DQN(**K**)** agents are learned by DQN, but with K times more real experiences than the DQN agent. DQN(K) is evaluated in the simulation setting only. Its performance can be viewed as the upper bound of its DDQ(K) counterpart, assuming that the world model in DDQ(K) perfectly matches real users.

Implementation Details All the models in these agents ($Q(s, a; \theta_Q)$, $M(s, a; \theta_M)$) are MLPs with `tanh` activations. Each policy network $Q(.)$ has one hidden layer with 80 hidden nodes. As shown in Figure 3, the world model $M(.)$ contains two shared hidden layers and three task-specific hidden layers, with 80 nodes in each. All the agents are trained by Algorithm 1 with the same set of hyper-parameters. ϵ-greedy is always applied for exploration. We set the discount factor γ = 0.95. The buffer sizes of both D^u and D^s are set to 5000. The target value function is updated at the end of each epoch. In each epoch, $Q(.)$ and $M(.)$ are refined using one-step ($Z = 1$) 16-tuple-minibatch update. [4] In planning, the maximum length of a simulated dialogue is 40 ($L = 40$). In addition, to make the dialogue training efficient, we also applied a variant of imitation learning, called Reply Buffer Spiking (RBS) (Lipton et al., 2016). We built a naive but occasionally successful rule-based agent based on human conversational dataset (line 1 in Algorithm 1), and pre-filled the real experience replay buffer D^u with 100 dialogues of experience (line 2) before training for all the variants of agents.

3.3 Simulated User Evaluation

In this setting the dialogue agents are optimized by interacting with user simulators, instead of real users. Thus, the world model is learned to mimic user simulators. Although the simulator-trained agents are sub-optimal when applied to real users due to the discrepancy between simulators and real users, the simulation setting allows us to perform a detailed analysis of DDQ without much cost and to reproduce the experimental results easily.

[4] We found in our experiments that setting $Z > 1$ improves the performance of all agents, but does not change the conclusion of this study: DDQ consistently outperforms DQN by a statistically significant margin. Conceptually, the optimal value of Z used in planning is different from that in direct reinforcement learning, and should vary according to the quality of the world model. The better the world model is, the more aggressive update (thus bigger Z) is being used in planning. We leave it to future work to investigate how to optimize Z for planning in DDQ.

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN	.4260	-3.84	31.93	.5308	10.78	22.72	.6480	27.66	22.21
DDQ(5)	.6056	20.35	26.65	.7128	36.76	19.55	.7372	39.97	18.99
DDQ(5, rand-init θ_M)	.5904	18.75	26.21	.6888	33.47	20.36	.7032	36.06	18.64
DDQ(5, fixed θ_M)	.5540	14.54	25.89	.6660	29.72	22.39	.6860	33.58	19.49
DQN(5)	*.6560*	*29.38*	*21.76*	*.7344*	*41.09*	*16.07*	*.7576*	*43.97*	*15.88*
DDQ(10)	**.6624**	28.18	24.62	**.7664**	42.46	21.01	**.7840**	45.11	19.94
DDQ(10, rand-init θ_M)	.6132	21.50	26.16	.6864	32.43	21.86	.7628	42.37	20.32
DDQ(10, fixed θ_M)	.5884	18.41	26.41	.6196	24.17	22.36	.6412	26.70	22.49
DQN(10)	*.7944*	*48.61*	*15.43*	*.8296*	*54.00*	*13.09*	*.8356*	*54.89*	*12.77*

Table 1: Results of different agents at training epoch = {100, 200, 300}. Each number is averaged over 5 runs, each run tested on 2000 dialogues. Excluding DQN(5) and DQN(10) which serve as the upper bounds, any two groups of success rate (except three groups: at epoch 100, DDQ(5, rand-init θ_M) and DDQ(10, fixed θ_M), at epoch 200, DDQ(5, rand-init θ_M) and DDQ(10, rand-init θ_M), at epoch 300, DQN and DDQ(10, fixed θ_M)) evaluated at the same epoch is statistically significant in mean with $p < 0.01$. (Success: success rate)

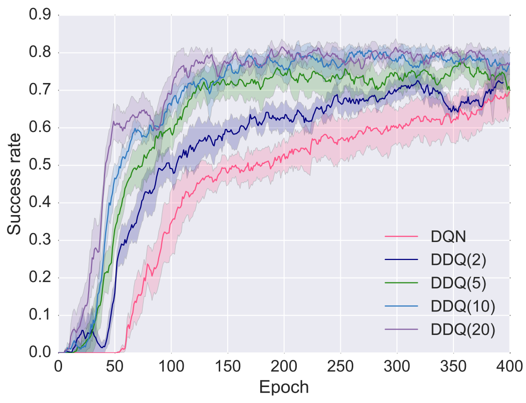


Figure 4: Learning curves of the DDQ(K) agents with $K = 2, 5, 10, 20$. The DQN agent is identical to a DDQ(K) agent with $K = 0$.

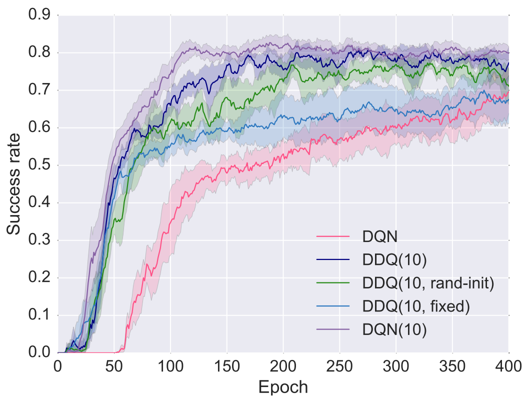


Figure 5: Learning curves of DQN, DDQ(10), DDQ(10, rand-init θ_M), DDQ(10, fixed θ_M), and DQN(10).

User Simulator We adapted a publicly available user simulator (Li et al., 2016b) to the task-completion dialogue setting. During training, the simulator provides the agent with a simulated user response in each dialogue turn and a reward signal at the end of the dialogue. A dialogue is considered successful only when a movie ticket is booked successfully and when the information provided by the agent satisfies all the user's constraints. At the end of each dialogue, the agent receives a positive reward of $2 * L$ for success, or a negative reward of $-L$ for failure, where L is the maximum number of turns in each dialogue, and is set to 40 in our experiments. Furthermore, in each turn, the agent receives a reward of -1, so that shorter dialogues are encouraged. Readers can refer to Appendix B for details on the user simulator.

Results The main simulation results are reported in Table 1 and Figures 4 and 5. For each agent, we report its results in terms of success rate, average reward, and average number of turns (averaged over 5 repetitions of the experiments). Results show that the DDQ agents consistently outperform DQN with a statistically significant margin. Figure 4 shows the learning curves of different DDQ agents trained using different planning steps. Since the training of all RL agents started with RBS using the same rule-based agent, their performance in the first few epochs is very close. After that, performance improved for all values of K, but much more rapidly for larger values. Recall that the DDQ(K) agent with K=0 is identical to the DQN agent, which does no planning but relies on direct reinforcement learning only. Without planning, the DQN agent took about 180 epochs (real dialogues) to reach the success rate of 50%,

Agent	Epoch = 100			Epoch = 150			Epoch = 200		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN	.0000	-58.69	39.38	.4080	-5.730	30.38	.4545	0.350	30.38
DDQ(5)	.4620	00.78	31.33	.5637	15.05	26.17	.6000	19.84	26.32
DDQ(5, rand-init θ_M)	.3600	-11.67	31.74	.5500	13.71	26.58	.5752	16.84	26.37
DDQ(10)	**.5555**	14.69	25.92	**.6416**	25.85	24.28	**.7332**	38.88	20.21
DDQ(10, rand-init θ_M)	.5010	6.27	29.70	.6055	22.11	23.11	.7023	36.90	21.20

Table 2: The performance of different agents at training epoch = $\{100, 150, 200\}$ in the human-in-the-loop experiments. The difference between the results of all agent pairs evaluated at the same epoch is statistically significant ($p < 0.01$). (Success: success rate)

and DDQ(10) took only 50 epochs.

Intuitively, the optimal value of K needs to be determined by seeking the best trade-off between the quality of the world model and the amount of simulated experience that is useful for improving the dialogue agent. This is a non-trivial optimization problem because both the dialogue agent and the world model are updated constantly during training and the optimal K needs to be adjusted accordingly. For example, we find in our experiments that at the early stages of training, it is fine to perform planning aggressively by using large amounts of simulated experience even though they are of low quality, but in the late stages of training where the dialogue agent has been significantly improved, low-quality simulated experience is likely to hurt the performance. Thus, in our implementation of Algorithm 1, we use a heuristic[5] to reduce the value of K in the late stages of training (e.g., after 150 epochs in Figure 4) to mitigate the negative impact of low-qualify simulated experience. We leave it to future work how to optimize the planning step size during DDQ training in a principled way.

Figure 5 shows that the quality of the world model has a significant impact on the agent's performance. The learning curve of DQN(10) indicates the best performance we can expect with a *perfect* world model. With a pre-trained world model, the performance of the DDQ agent improves more rapidly, although eventually, the DDQ and DDQ(rand-init θ_M) agents reach the same success rate after many epochs. The world model learning process is crucial to both the efficiency of dialogue policy learning and the final performance of the agent. For example, in the early stages (before 60 epochs), the performances of DDQ and DDQ(fixed θ_M) remain very close to each other, but DDQ reaches a success rate almost 10% better than DDQ(fixed θ_M) after 400 epochs.

[5]The heuristic is not presented in Algorithm 1. Readers can refer to the released source code for details.

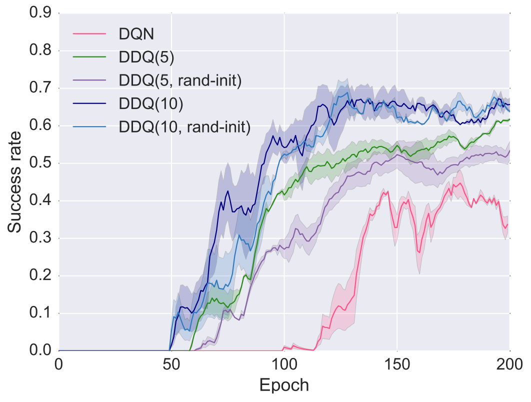


Figure 6: Human-in-the-loop dialogue policy learning curves in four different agents.

3.4 Human-in-the-Loop Evaluation

In this setting, five dialogue agents (i.e., DQN, DDQ(10), DDQ(10, rand-init θ_M), DDQ(5), and DDQ(5, rand-init θ_M)) are trained via RL by interacting with real human users. In each dialogue session, one of the agents was randomly picked to converse with a user. The user was presented with a user goal sampled from the corpus, and was instructed to converse with the agent to complete the task. The user had the choice of abandoning the task and ending the dialogue at any time, if she or he believed that the dialogue was unlikely to succeed or simply because the dialogue dragged on for too many turns. In such cases, the dialogue session is considered failed. At the end of each session, the user was asked to give explicit feedback whether the dialogue succeeded (i.e., whether the movie tickets were booked with all the user constraints satisfied). Each learning curve is trained with two runs, with each run generating 150 dialogues (and $K{*}150$ additional simulated dialogues when planning is applied). In total, we collected 1500 dialogue sessions for training all five agents.

The main results are presented in Table 2 and

	Simulation Sample	Real User Sample

	movie-ticket booking user goal:
	{
	"request_slots": {
	"ticket": "?"
	"theater": "?"
	"starttime": "?"
	"date": "?"
	}
	"constraint_slots": {
	"numberofpeople":"2"
	"moviename": "deadpool"
	"city": "seattle"
	}
	}
		movie-ticket booking user goal:
		{
		"request_slots": {
		"ticket": "?"
		"theater": "?"
		"starttime": "?"
		}
		"constraint_slots": {
		"date":"this weekend"
		"numberofpeople": "1"
		"moviename": "batman"
		}
		}

	DQN agent	
1	*usr*: Hi, what date can I book 2 tickets for deadpool?	*usr*: Hi, which theater will show batman? I want 1 ticket.
2	*agt*: Which city would you like?	*agt*: Which city would you like?
3	*usr*: I want to watch at seattle.	*usr*: Boston.
4	*agt*: What time would you like to see it?	*agt*: What time would you like to see it?
5	*usr*: What start time is available?	*usr*: What start time is available?
6	*agt*: Which theater would you like?	*agt*: What date would you like to see it?
7	*usr*: Which theater is available?	*usr*: This weekend.
8	*agt*: What time would you like to see it?	*agt*:Which theater would you like?
9	*usr*: What start time is available?	*usr*: Which theater is available?
10	*agt*: Which theater would you like?	*agt*: Which theater would you like?
11	*usr*: Which theater is available?	*usr*: Which theater is available?
12	*agt*: How many tickets do you need?	*agt*: What time would you like to see it?
13	*usr*: I want 2 tickets please!	*usr*: What start time is available?
14	*agt*: Which theater would you like?	*agt*: Which theater would you like?
15	*usr*: Which theater is available?	*usr*: Thanks, I do not need the ticket.
16	*agt*: What time would you like to see it?	*agt*: Thank you.
	⋮	
39	*agt*: Which theater would you like?	
40	*usr*: Which theater is available?	
	Failure: Agent does not book the right ticket within the max turns.	**Failure:** User terminates the dialogue before reaching the max turns because the agent repeatly asks the same question.
	DDQ(10) agent	
1	*usr*: Hi, when is deadpool playing at seattle?	*usr*: Hi, which theater will show batman? I want 1 ticket.
2	*agt*: Tonight.	*agt*: Century Eastport 16.
3	*usr*: What is the start time?	*usr*: What is the start time?
4	*agt*: Which theater would you like?	*agt*: 9PM.
5	*usr*: I want to watch it at amc pacific place 11.	*usr*: Can I buy a ticket here?
6	*agt*: 10pm is available.	*agt*: This weekend?
7	*usr*: I want 2 tickets please!	*usr*: Yes, one ticket please!
8	*agt*: Great, here are 2 deadpool tickets for you at amc pacific place 11 in seattle tonight 10 pm.	*agt*: Sure, here it is - batman at Century Eastport 16, 9PM this weekend.
9	*usr*: Thank you.	*usr*: Thank you.
10	*agt*: Thank you.	*agt*: Thank you.
	Success	**Success**

Table 3: Two sample dialogue sessions by DQN and DDQ(10) agents trained at epoch 100: Left: simulated user experiments; Right: human-in-the-loop experiments. (*agt*: agent, *usr*: user)

Figure 6, with each agent averaged over two independent runs. The results confirm what we observed in the simulation experiments. The conclusions are summarized as below:

- The DDQ agent significantly outperforms DQN, as demonstrated by the comparison between DDQ(10) and DQN. Table 3 presents four example dialogues produced by two dialogue agents interacting with simulated and human users, respectively. The DQN agent, after being trained with 100 dialogues, still behaved like a naive rule-based agent that requested information bit by bit in a fixed order. When the user did not answer the request explicitly (e.g., `usr: which theater is available?`), the agent failed to respond properly. On the other hand, with planning, the DDQ agent trained with 100 real dialogues is much more robust and can complete 50% of user tasks successfully.
- A larger K leads to more aggressive planning and better results, as shown by DDQ(10) vs. DDQ(5).
- Pre-training world model with human con-

versational data improves the learning efficiency and the agent's performance, as shown by DDQ(5) vs. DDQ(5, rand-init θ_M), and DDQ(10) vs. DDQ(10, rand-init θ_M).

4 Conclusion

We propose a new strategy for a task-completion dialogue agent to learn its policy by interacting with real users. Compared to previous work, our agent learns in a much more efficient way, using only a small number of real user interactions, which amounts to an affordable cost in many nontrivial domains. Our strategy is based on the Deep Dyna-Q (DDQ) framework where planning is integrated into dialogue policy learning. The effectiveness of DDQ is validated by human-in-the-loop experiments, demonstrating that a dialogue agent can efficiently adapt its policy on the fly by interacting with real users via deep RL.

One interesting topic for future research is *exploration in planning*. We need to deal with the challenge of adapting the world model in a changing environment, as exemplified by the domain extension problem (Lipton et al., 2016). As pointed out by Sutton and Barto (1998), the general problem here is a particular manifestation of the conflict between exploration and exploitation. In a planning context, exploration means trying actions that may improve the world model, whereas exploitation means trying to behave in the optimal way given the current model. To this end, we want the agent to explore in the environment, but not so much that the performance would be greatly degraded.

Additional Authors

Shang-Yu Su (National Taiwan University, Room 524, CSIE Bldg., No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan. email: shangyusu.tw@gmail.com)

Acknowledgments

We would like to thank Chris Brockett, Yun-Nung Chen, Michel Galley and Lihong Li for their insightful comments on the paper. We would like to acknowledge the volunteers from Microsoft Research for helping us with the human-in-the-loop experiments. This work was done when Baolin Peng and Shang-Yu Su were visiting Microsoft. Baolin Peng is in part supported by Innovation and Technology Fund (6904333), and General Research Fund of Hong Kong (12183516).

References

Pawel Budzianowski, Stefan Ultes, Pei-Hao Su, Nikola Mrksic, Tsung-Hsien Wen, Inigo Casanueva, Lina Rojas-Barahona, and Milica Gasic. 2017. Subdomain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint arXiv:1706.06210* .

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 484–495.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 201–204.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, pages 312–317.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*. pages 2829–2838.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Leonid Kuvayev and Richard S Sutton. 1996. Model-based reinforcement learning with an approximate, learned model. In *in Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*. Citeseer.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, pages 72–79.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016a.

Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823* .

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016b. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688* .

Xuijun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*. pages 733–743.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081* .

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *Proceedings of 2017 IEEE Workshop on Automatic Speech Recognition and Understanding*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval .

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Andrew W Moore and Christopher G Atkeson. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning* 13(1):103–130.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777* .

Alexandros Papangelis. 2012. A comparative study of reinforcement learning techniques on dialogue management. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 22–31.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Yun-Nung Chen, and Kam-Fai Wong. 2017a. Adversarial advantage actor-critic model for task-completion dialogue policy learning. *arXiv preprint arXiv:1710.11277* .

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017b. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2221–2230.

Jing Peng and Ronald J Williams. 1993. Efficient learning and planning within the dyna framework. *Adaptive Behavior* 1(4):437–454.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, et al. 2011. Sample efficient online learning of optimal dialogue policies with kalman temporal differences. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. volume 22, page 1878.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review* .

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. 2017. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*. pages 5694–5705.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL 2007; Companion Volume, Short Papers*. Association for Computational Linguistics, pages 149–152.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016a. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. 2016b. The predictron: End-to-end learning and planning. *arXiv preprint arXiv:1612.08810* .

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research* 16:105–133.

Satinder P Singh. 1992. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the National Conference on Artificial Intelligence*. JOHN WILEY & SONS LTD, 10, page 202.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016a. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689* .

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016b. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669* .

Richard S Sutton. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*. pages 216–224.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling. 2012. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285* .

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2016. Value iteration networks. In *Advances in Neural Information Processing Systems*. pages 2154–2162.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. ???? Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1711–1721.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560* .

A Dataset Annotation Schema

Table 4 lists all annotated dialogue acts and slots in details.

Annotations	
Intent	request, inform, deny, confirm_question, confirm_answer, greeting, closing, not_sure, multiple_choice, thanks, welcome
Slot	city, closing, date, distanceconstraints, greeting, moviename, numberofpeople, price, starttime, state, taskcomplete, theater, theater_chain, ticket, video_format, zip

Table 4: The data annotation schema

B User Simulator

In the task-completion dialogue setting, the entire conversation is around a user goal implicitly, but the agent knows nothing about the user goal explicitly and its objective is to help the user to accomplish this goal. Generally, the definition of user goal contains two parts:

- *inform_slots* contain a number of slot-value pairs which serve as constraints from the user.
- *request_slots* contain a set of slots that user has no information about the values, but wants to get the values from the agent during the conversation. ticket is a default slot which always appears in the *request_slots* part of user goal.

To make the user goal more realistic, we add some constraints in the user goal: slots are split into two groups. Some of slots must appear in the user goal, we called these elements as *Required slots*. In the movie-booking scenario, it includes moviename, theater, starttime, date, numberofpeople; the rest slots are *Optional slots*, for example, theater_chain, video_format etc.

We generated the user goals from the labeled dataset mentioned in Section 3.1, using two mechanisms. One mechanism is to extract all the slots (known and unknown) from the first user turns (excluding the greeting user turn) in the data, since usually the first turn contains some or all the required information from user. The other mechanism is to extract all the slots (known and unknown) that first appear in all the user turns, and then aggregate them into one user goal. We dump these user goals into a file as the user-goal database. Every time when running a dialogue, we randomly sample one user goal from this user goal database.

Learning to Ask Questions in Open-domain Conversational Systems with Typed Decoders

Yansen Wang[1,*], **Chenyi Liu**[1,*], **Minlie Huang**[1,†], **Liqiang Nie**[2]
[1]Conversational AI group, AI Lab., Department of Computer Science, Tsinghua University
[1]Beijing National Research Center for Information Science and Technology, China
[2]Shandong University
ys-wang15@mails.tsinghua.edu.cn;liucy15@mails.tsinghua.edu.cn;
aihuang@tsinghua.edu.cn;nieliqiang@gmail.com

Abstract

Asking good questions in large-scale, open-domain conversational systems is quite significant yet rather untouched. This task, substantially different from traditional question generation, requires to question not only with various patterns but also on diverse and relevant topics. We observe that a good question is a natural composition of *interrogatives*, *topic words*, and *ordinary words*. Interrogatives lexicalize the pattern of questioning, topic words address the key information for topic transition in dialogue, and ordinary words play syntactical and grammatical roles in making a natural sentence. We devise two typed decoders (*soft typed decoder* and *hard typed decoder*) in which a type distribution over the three types is estimated and used to modulate the final generation distribution. Extensive experiments show that the typed decoders outperform state-of-the-art baselines and can generate more meaningful questions.

1 Introduction

Learning to ask questions (or, question generation) aims to generate a question to a given input. Deciding what to ask and how is an indicator of machine understanding (Mostafazadeh et al., 2016), as demonstrated in machine comprehension (Du et al., 2017; Zhou et al., 2017b; Yuan et al., 2017) and question answering (Tang et al., 2017; Wang et al., 2017). Raising good questions is essential to conversational systems because a good system can well interact with users by asking and responding (Li et al., 2016). Furthermore, asking questions is one of the important proactive behaviors that can drive dialogues to go deeper and further (Yu et al., 2016).

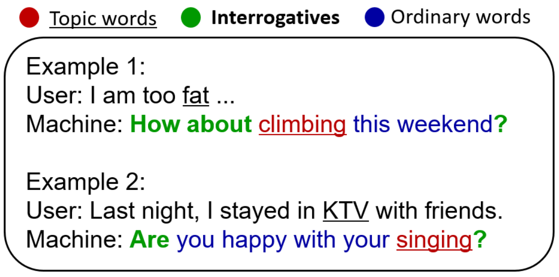
Question generation (QG) in open-domain conversational systems differs substantially from the traditional QG tasks. The ultimate goal of this task is to enhance the *interactiveness and persistence of human-machine interactions*, while for traditional QG tasks, seeking information through a generated question is the major purpose. The response to a generated question will be supplied in the following conversations, which may be novel but not necessarily occur in the input as that in traditional QG (Du et al., 2017; Yuan et al., 2017; Tang et al., 2017; Wang et al., 2017; Mostafazadeh et al., 2016). Thus, the purpose of this task is to spark novel yet related information to drive the interactions to continue.

Due to the different purposes, this task is unique in two aspects: it requires to question not only in various patterns but also about diverse yet relevant topics. **First**, there are various questioning patterns for the same input, such as Yes-no questions and Wh-questions with different interrogatives. Diversified questioning patterns make dialogue interactions richer and more flexible. Instead, traditional QG tasks can be roughly addressed by syntactic transformation (Andrenucci and Sneiders, 2005; Popowich and Winne, 2013), or implicitly modeled by neural models (Du et al., 2017). In such tasks, the information questioned on is pre-specified and usually determines the pattern of questioning. For instance, asking Who-question for a given person, or Where-question for a given location.

Second, this task requires to address much more transitional topics of a given input, which is the nature of conversational systems. For instance, for the input *"I went to dinner with my friends"*, we may question about topics such as *friend, cuisine,*

[*]Authors contributed equally to this work.
[†]Corresponding author: Minlie Huang.

price, place and *taste*. Thus, this task generally requires *scene understanding* to imagine and comprehend a scenario (e.g., *dining at a restaurant*) that can be interpreted by topics related to the input. However, in traditional QG tasks, the core information to be questioned on is pre-specified and rather static, and *paraphrasing* is more required.


● Topic words ● **Interrogatives** ● Ordinary words

Example 1:
User: I am too fat ...
Machine: **How about** climbing this weekend**?**

Example 2:
User: Last night, I stayed in KTV with friends.
Machine: **Are** you happy with your singing**?**

Figure 1: Good questions in conversational systems are a natural composition of interrogatives, topic words, and ordinary words.

Undoubtedly, asking good questions in conversational systems needs to address the above issues (*questioning with diversified patterns, and addressing transitional topics naturally in a generated question*). As shown in Figure 1, a good question is a natural composition of interrogatives, topic words, and ordinary words. Interrogatives indicate the pattern of questioning, topic words address the key information of topic transition, and ordinary words play syntactical and grammatical roles in making a natural sentence.

We thus classify the words in a question into three types: **interrogative**, **topic word**, and **ordinary word** automatically. We then devise two decoders, Soft Typed Decoder (STD) and Hard Typed Decoder (HTD), for question generation in conversational systems[1]. STD deals with word types in a latent and implicit manner, while HTD in a more explicit way. At each decoding position, we firstly estimate a type distribution over word types. STD applies a mixture of type-specific generation distributions where type probabilities are the coefficients. By contrast, HTD reshapes the type distribution by Gumbel-softmax and modulates the generation distribution by type probabilities. Our contributions are as follows:

- To the best of our knowledge, this is the first study on question generation in the setting of conversational systems. We analyze the key differences between this new task and other traditional question generation tasks.
- We devise soft and hard typed decoders to ask good questions by capturing different roles of different word types. Such typed decoders may be applicable to other generation tasks if word semantic types can be identified.

[1]To simplify the task, as a preliminary research, we consider the one-round conversational system.

2 Related Work

Traditional question generation can be seen in task-oriented dialogue system (Curto et al., 2012), sentence transformation (Vanderwende, 2008), machine comprehension (Du et al., 2017; Zhou et al., 2017b; Yuan et al., 2017; Subramanian et al., 2017), question answering (Qin, 2015; Tang et al., 2017; Wang et al., 2017; Song et al., 2017), and visual question answering (Mostafazadeh et al., 2016). In such tasks, the answer is known and is part of the input to the generated question. Meanwhile, the generation tasks are not required to predict additional topics since all the information has been provided in the input. They are applicable in scenarios such as designing questions for reading comprehension (Du et al., 2017; Zhou et al., 2017a; Yuan et al., 2017), and justifying the visual understanding by generating questions to a given image (video) (Mostafazadeh et al., 2016).

In general, traditional QG tasks can be addressed by the heuristic rule-based reordering methods (Andrenucci and Sneiders, 2005; Ali et al., 2010; Heilman and Smith, 2010), slot-filling with question templates (Popowich and Winne, 2013; Chali and Golestanirad, 2016; Labutov et al., 2015), or implicitly modeled by recent neural models(Du et al., 2017; Zhou et al., 2017b; Yuan et al., 2017; Song et al., 2017; Subramanian et al., 2017). These tasks generally do not require to generate a question with various patterns: for a given answer and a supporting text, the question type is usually decided by the input.

Question generation in large-scale, open-domain dialogue systems is relatively unexplored. Li et al. (2016) showed that asking questions in task-oriented dialogues can offer useful feedback to facilitate learning through interactions. Several questioning mechanisms were devised with hand-crafted templates, but unfortunately not applicable to open-domain conversational systems. Similar to our goal, a visual QG task is proposed to generate a question to interact with other people, given an image as input (Mostafazadeh et al., 2016).

3 Methodology

3.1 Overview

The task of question generation in conversational systems can be formalized as follows: given a user post $X = x_1 x_2 \cdots x_m$, the system should generate a natural and meaningful question $Y = y_1 y_2 \cdots y_n$ to interact with the user, formally as

$$Y^* = \underset{Y}{argmax}\, \mathcal{P}(Y|X).$$

As aforementioned, asking good questions in conversational systems requires to question with diversified patterns and address transitional topics naturally in a question. To this end, we classify the words in a sentence into three types: *interrogative, topic word*, and *ordinary word*, as shown in Figure 1. During training, the type of each word in a question is decided automatically[2]. We manually collected about 20 interrogatives. The verbs and nouns in a question are treated as topic words, and all the other words as ordinary words. During test, we resort to PMI (Church and Hanks, 1990) to predict a few topic words for a given post.

On top of an encoder-decoder framework, we propose two decoders to effectively use word types in question generation. The first model is *soft typed decoder (STD)*. It estimates a type distribution over word types and three type-specific generation distributions over the vocabulary, and then obtains a mixture of type-specific distributions for word generation.

The second one is a *hard* form of STD, *hard typed decoder (HTD)*, in which we can control the decoding process more explicitly by approximating the operation of *argmax* with Gumbel-softmax (Jang et al., 2016). In both decoders, the final generation probability of a word is modulated by its word type.

3.2 Encoder-Decoder Framework

Our model is based on the general encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014). Formally, the model encodes an input sequence $X = x_1 x_2 \cdots x_m$ into a sequence of hidden states $\mathbf{h}_i$, as follows,

$$\mathbf{h}_t = \mathbf{GRU}(\mathbf{h}_{t-1}, \boldsymbol{e}(x_t)),$$

[2]Though there may be errors in word type classification, we found it works well in response generation.

where GRU denotes gated recurrent units (Cho et al., 2014), and $\boldsymbol{e}(x)$ is the word vector of word x. The decoder generates a word sequence by sampling from the probability $\mathcal{P}(y_t|y_{<t}, X)$ ($y_{<t} = y_1 y_2 \cdots y_{t-1}$, the generated subsequence) which can be computed via

$$\mathcal{P}(y_t|y_{<t}, X) = \mathbf{MLP}(\mathbf{s}_t, \boldsymbol{e}(y_{t-1}), \mathbf{c}_t),$$
$$\mathbf{s}_t = \mathbf{GRU}(\mathbf{s}_{t-1}, \boldsymbol{e}(y_{t-1}), \mathbf{c}_t),$$

where $\mathbf{s}_t$ is the state of the decoder at the time step t, and this GRU has different parameters with the one of the encoder. The context vector $\mathbf{c}_t$ is an attentive read of the hidden states of the encoder as $\mathbf{c}_t = \sum_{i=1}^{T} \alpha_{t,i}\mathbf{h}_i$, where the weight $\alpha_{t,i}$ is scored by another $\mathbf{MLP}(\mathbf{s}_{t-1}, \mathbf{h}_i)$ network.

3.3 Soft Typed Decoder (STD)

In a general encoder-decoder model, the decoder tends to generate universal, meaningless questions like "*What's up?*" and "*So what?*". In order to generate more meaningful questions, we propose a soft typed decoder. It assumes that each word has a *latent type* among the set {*interrogative, topic word, ordinary word*}. The soft typed decoder firstly estimates a word type distribution over latent types in the given context, and then computes type-specific generation distributions over the entire vocabulary for different word types. The final probability of generating a word is a mixture of type-specific generation distributions where the coefficients are type probabilities.

The final generation distribution $\mathcal{P}(y_t|y_{<t}, X)$ from which a word can be sampled, is given by

$$\mathcal{P}(y_t|y_{<t}, X) = \sum_{i=1}^{k} \mathcal{P}(y_t|ty_t = c_i, y_{<t}, X) \cdot \mathcal{P}(ty_t = c_i|y_{<t}, X), \quad (1)$$

where ty_t denotes the word type at time step t and c_i is a word type. Apparently, this formulation states that the final generation probability is a mixture of the type-specific generation probabilities $\mathcal{P}(y_t|ty_t = c_i, y_{<t}, X)$, weighted by the probability of the type distribution $\mathcal{P}(ty_t = c_i|y_{<t}, X)$. We name this decoder as *soft typed decoder*. In this model, word type is latent because we do not need to specify the type of a word explicitly. In other words, each word can belong to any of the three types, but with different probabilities given the current context.

The probability distribution over word types $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$ ($k = 3$ in this paper) (termed

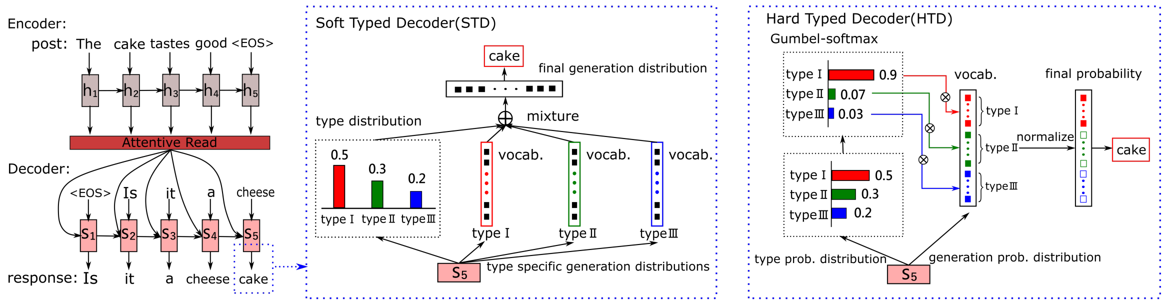


Figure 2: Illustration of STD and HTD. STD applies a mixture of type-specific generation distributions where type probabilities are the coefficients. In HTD, the type probability distribution is reshaped by Gumbel-softmax and then used to modulate the generation distribution. In STD, the generation distribution is over the same vocabulary whereas dynamic vocabularies are applied in HTD.

as *type distribution*) is given by

$$\mathcal{P}(ty_t|y_{<t}, X) = softmax(\mathbf{W}_0\mathbf{s}_t + \mathbf{b}_0), \qquad (2)$$

where s_t is the hidden state of the decoder at time step t, $\mathbf{W}_0 \in R^{k\times d}$, and d is the dimension of the hidden state.

The type-specific generation distribution is given by

$$\mathcal{P}(y_t|ty_t = c_i, y_{<t}, X) = softmax(\mathbf{W}_{c_i}\mathbf{s}_t + \mathbf{b}_{c_i}),$$

where $\mathbf{W}_{c_i} \in R^{|V|\times d}$ and $|V|$ is the size of the entire vocabulary. Note that the type-specific generation distribution is parameterized by $\mathbf{W}_{c_i}$, indicating that the distribution for each word type has its own parameters.

Instead of using a single distribution $\mathcal{P}(y_t|y_{<t}, X)$ as in a general Seq2Seq decoder, our soft typed decoder enriches the model by applying multiple type-specific generation distributions. This enables the model to express more information about the next word to be generated. Also note that the generation distribution is over the same vocabulary, and therefore there is no need to specify word types explicitly.

3.4 Hard Typed Decoder (HTD)

In the soft typed decoder, we assume that each word is a distribution over the word types. In this sense, the type of a word is **implicit**. We do not need to specify the type of each word **explicitly**. In the hard typed decoder, words in the entire vocabulary are dynamically classified into three types for each post, and the decoder first estimates a type distribution at each position and then generates a word with the highest type probability. This process can be formulated as follows:

$$c^* = \arg\max_{c_i} \mathcal{P}(ty_t = c_i|y_{<t}, X), \qquad (3)$$

$$\mathcal{P}(y_t|y_{<t}, X) = \mathcal{P}(y_t|ty_t = c^*, y_{<t}, X). \qquad (4)$$

This is essentially the *hard* form of Eq. 1, which just selects the type with the maximal probability. However, this *argmax* process may cause two problems. First, such a cascaded decision process (firstly selecting the most probable word type and secondly choosing a word from that type) may lead to severe grammatical errors if the first selection is wrong. Second, *argmax* is discrete and non-differentiable, and it breaks the back-propagation path during training.

To make best use of word types in *hard typed decoder*, we address the above issues by applying *Gumbel-Softmax* (Jang et al., 2016) to approximate the operation of *argmax*. There are several steps in the decoder (see Figure 2):

First, the type of each word (*interrogative, topic, or ordinary*) in a question is decided automatically during training, as aforementioned.

Second, the generation probability distribution is estimated as usual,

$$\mathcal{P}(y_t|y_{<t}, X) = softmax(\mathbf{W}_0\mathbf{s}_t + \mathbf{b}_0). \qquad (5)$$

Further, the type probability distribution at each decoding position is estimated as follows,

$$\mathcal{P}(ty_t|y_{<t}, X) = softmax(\mathbf{W}_1\mathbf{s}_t + \mathbf{b}_1). \qquad (6)$$

Third, the generation probability for each word is modulated by its corresponding type probability:

$$\mathcal{P}'(y_t|y_{<t},X) = \mathcal{P}(y_t|y_{<t},X)\cdot\boldsymbol{m}(y_t),$$
$$\boldsymbol{m}(y_t) = \begin{cases} 1 & ,c(y_t) = c^* \\ 0 & ,c(y_t) \neq c^* \end{cases} \tag{7}$$

where $c(y_t)$ looks up the word type of word y_t, and c^* is the type with the highest probability as defined in Eq. 3. This formulation has exactly the effect of *argmax*, where the decoder will only generate words of type with the highest probability.

To make $\mathcal{P}^*(y_t|y_{<t},X)$ a distribution, we normalize these values by a normalization factor Z:

$$Z = \frac{1}{\sum_{y_t\in\mathcal{V}}\mathcal{P}'(y_t|y_{<t},X)}$$

where $\mathcal{V}$ is the decoding vocabulary. Then, the final probability can be denoted by

$$\mathcal{P}^*(y_t|y_{<t},X) = Z\cdot\mathcal{P}'(y_t|y_{<t},X). \tag{8}$$

As mentioned, in order to have an effect of *argmax* but still maintain the differentiability, we resort to *Gumbel-Softmax* (Jang et al., 2016), which is a differentiable surrogate to the *argmax* function. The type probability distribution is then adjusted to the following form:

$$\boldsymbol{m}(y_t) = \mathbf{GS}(\mathcal{P}(ty_t = c(y_t)|y_{<t},X)),$$
$$\mathbf{GS}(\pi_i) = \frac{e^{(log(\pi_i)+g_i)/\tau}}{\sum_{j=1}^{k} e^{(log(\pi_j)+g_j)/\tau}}, \tag{9}$$

where $\pi_1, \pi_2, \cdots, \pi_k$ represents the probabilities of the original categorical distribution, g_j are i.i.d samples drawn from Gumbel(0,1)[3] and τ is a constant that controls the smoothness of the distribution. When $\tau \to 0$, Gumbel-Softmax performs like argmax, while if $\tau \to \infty$, Gumbel-Softmax performs like a uniform distribution. In our experiments, we set τ a constant between 0 and 1, making Gumbel-Softmax smoother than argmax, but sharper than normal softmax.

Note that in HTD, we apply dynamic vocabularies for different responses during training. The words in a response are classified into the three types dynamically. A specific type probability will only affect the words of that type. During test, for each post, topic words are predicted with PMI, interrogatives are picked from a small dictionary, and the rest of words in the vocabulary are treated as ordinary words.

[3]If $u \sim Uniform(0,1)$, then $g = -log(-log(u)) \sim Gumbel(0,1)$.

3.5 Loss Function

We adopt negative data likelihood (equivalent to cross entropy) as the loss function, and additionally, we apply supervision on the mixture weights of word types, formally as follows:

$$\Phi_1 = \sum_t -\log\mathcal{P}(y_t = \tilde{y}_t|y_{<t},X), \tag{10}$$
$$\Phi_2 = \sum_t -\log\mathcal{P}(ty_t = \widetilde{ty}_t|y_{<t},X), \tag{11}$$
$$\Phi = \Phi_1 + \lambda\Phi_2, \tag{12}$$

where $\widetilde{ty}_t$ represents the reference word type and $\tilde{y}_t$ represents the reference word at time t. λ is a factor to balance the two loss terms, and we set λ=0.8 in our experiments.

Note that for HTD, we substitute $\mathcal{P}^*(y_t = w_j|y_{<t},X)$ (as defined by Eq. 8) into Eq. 10.

3.6 Topic Word Prediction

The only difference between training and inference is the means of choosing topic words. During training, we identify the nouns and verbs in a response as topic words; whereas during inference, we adopt PMI (Church and Hanks, 1990) and $Rel(k_i, X)$ to predict a set of topic words k_i for an input post X, as defined below:

$$PMI(w_x, w_y) = log\frac{p(w_x, w_y)}{p_1(w_x) * p_2(w_y)},$$
$$Rel(k_i, X) = \sum_{w_x\in X} e^{PMI(w_x,k_i)},$$

where $p_1(w)/p_2(w)$ represent the probability of word w occurring in a post/response, respectively, and $p(w_x, w_y)$ is the probability of word w_x occurring in a post and w_y in a response.

During inference, we predict at most 20 topic words for an input post. Too few words will affect the grammaticality since the predicted set contains infrequent topic words, while too many words introduce more common topics leading to more general responses.

4 Experiment

4.1 Dataset

To estimate the probabilities in PMI, we collected about 9 million post-response pairs from Weibo. To train our question generation models, we distilled the pairs whereby the responses are in question form with the help of around 20 hand-crafted templates. The templates contain a list of interrogatives and other implicit questioning patterns. Such patterns detect sentences led by words like *what, how many, how about* or sentences ended with a *question mark*. After that, we removed the pairs whose responses are universal questions that can be used to reply many different posts. This is a simple yet effective way to avoid situations where the type probability distribution is dominated by interrogatives and ordinary words.

Ultimately, we obtained the dataset comprising about 491,000 post-response pairs. We randomly selected 5,000 pairs for testing and another 5,000 for validation. The average number of words in post/response is 8.3/9.3 respectively. The dataset contains 66,547 different words, and 18,717 words appear more than 10 times. The dataset is available at: `http://coai.cs.tsinghua.edu.cn/hml/dataset/`.

4.2 Baselines

We compared the proposed decoders with four state-of-the-art baselines.

Seq2Seq: A simple encoder-decoder with attention mechanisms (Luong et al., 2015).

MA: The mechanism-aware (MA) model applies multiple responding mechanisms represented by real-valued vectors (Zhou et al., 2017a). The number of mechanisms is set to 4 and we randomly picked one response from the generated responses for evaluation to avoid selection bias.

TA: The topic-aware (TA) model generates informative responses by incorporating topic words predicted from the input post (Xing et al., 2017).

ERM: Elastic responding machine (ERM) adaptively selects a subset of responding mechanisms using reinforcement learning (Zhou et al., 2018a). The settings are the same as the original paper.

4.3 Experiment Settings

Parameters were set as follows: we set the vocabulary size to $20,000$ and the dimension of word vectors as 100. The word vectors were pre-trained with around 9 million post-response pairs from Weibo and were being updated during the training of the decoders. We applied the 4-layer GRU units (hidden states have 512 dimensions). These settings were also applied to all the baselines. λ in Eq. 12 is 0.8. We set different values of τ in Gumbel-softmax at different stages of training. At the early stage, we set τ to a small value (0.6) to obtain a sharper reformed distribution (more like argmax). After several steps, we set τ to a larger value (0.8) to apply a more smoothing distribution. Our codes are available at: `https://github.com/victorywys/Learning2Ask_TypedDecoder`.

4.4 Automatic Evaluation

We conducted automatic evaluation over the $5,000$ test posts. For each post, we obtained responses from the six models, and there are $30,000$ post-response pairs in total.

4.4.1 Evaluation Metrics

We adopted *perplexity* to quantify how well a model fits the data. Smaller values indicate better performance. To evaluate the diversity of the responses, we employed *distinct-1* and *distinct-2* (Li et al., 2015). These two metrics calculates the proportion of the total number of distinct unigrams or bigrams to the total number of generated tokens in all the generated responses.

Further, we calculated the proportion of the responses containing at least one topic word in the list predicted by PMI. This is to evaluate the ability of addressing topic words in response. We term this metric as *topical response ratio (TRR)*. We predicted 20 topic words with PMI for each post.

4.4.2 Results

Comparative results are presented in Table 1. STD and HTD perform fairly well with lower perplexities, higher distinct-1 and distinct-2 scores, and remarkably better topical response ratio (TRR). Note that MA has the lowest perplexity because the model tends to generate more universal responses.

Model	Perplexity	Distinct-1	Distinct-2	TRR
Seq2Seq	63.71	0.0573	0.0836	6.6%
MA	**54.26**	0.0576	0.0644	4.5%
TA	58.89	0.1292	0.1781	8.7%
ERM	67.62	0.0355	0.0710	4.5%
STD	56.77	0.1325	0.2509	12.1%
HTD	56.10	**0.1875**	**0.3576**	**43.6%**

Table 1: Results of automatic evaluation.

Our decoders have better distinct-1 and distinct-2 scores than baselines do, and HTD performs much better than the strongest baseline TA. Noticeably, the means of using topic information in our models differs substantially from that in TA. Our decoders predict whether a topic word should be decoded at each position, whereas TA takes as

Models	Appropriateness			Richness			Willingness		
	Win (%)	Lose (%)	Tie (%)	Win (%)	Lose (%)	Tie (%)	Win (%)	Lose (%)	Tie (%)
STD vs. Seq2Seq	42.0	38.6	19.4	37.2**	15.2	47.6	45.4*	38.6	16.0
STD vs. MA	39.6*	31.2	29.2	32.6**	16.8	50.6	49.4**	27.0	23.6
STD vs. TA	42.2	40.0	17.8	49.0**	5.4	45.6	47.6*	40.2	12.2
STD vs. ERM	43.4*	34.4	22.2	60.6**	13.2	26.2	43.2*	36.8	20.0
HTD vs. Seq2Seq	50.6**	30.6	18.8	46.0**	10.2	43.8	58.4**	33.2	8.4
HTD vs. MA	54.8**	24.4	20.8	45.0**	17.0	38.0	67.0**	18.0	15.0
HTD vs. TA	52.0**	38.2	9.8	55.0**	5.4	39.6	62.6**	31.0	6.4
HTD vs. ERM	64.8**	23.2	12.0	72.2**	8.4	19.4	56.6**	36.6	6.8
HTD vs. STD	52.0**	33.0	15.0	38.0**	26.2	35.8	61.8**	30.6	7.6

Table 2: Annotation results. Win for "A vs. B" means A is better than B. Significance tests with Z-test were conducted. Values marked with * means *p-value* < 0.05, and ** for *p-value* < 0.01.

input topic word embeddings at all decoding positions.

Our decoders have remarkably better topic response ratios (TRR), indicating that they are more likely to include topic words in generation.

4.5 Manual Evaluation

We resorted to a crowdsourcing service for manual annotation. 500 posts were sampled for manual annotation[4]. We conducted pair-wise comparison between two responses generated by two models for the same post. In total, there are 4,500 pairs to be compared. For each response pair, five judges were hired to give a preference between the two responses, in terms of the following three metrics. Tie was allowed, and system identifiers were masked during annotation.

4.5.1 Evaluation Metrics

Each of the following metrics is evaluated independently on each pair-wise comparison:

Appropriateness: measures whether a question is reasonable in logic and content, and whether it is questioning on the key information. Inappropriate questions are either irrelevant to the post, or have grammatical errors, or universal questions.

Richness: measures whether a response contains topic words that are relevant to a given post.

Willingness to respond: measures whether a user will respond to a generated question. This metric is to justify how likely the generated questions can elicit further interactions. If people are willing to respond, the interactions can go further.

[4]During the sampling process, we removed those posts that are only interpretable with other context or background.

4.5.2 Results

The label of each pair-wise comparison is decided by majority voting from five annotators. Results shown in Table 2 indicate that STD and HTD outperform all the baselines in terms of all the metrics. This demonstrates that our decoders produce more appropriate questions, with richer topics. Particularly, our decoders have substantially better *willingness scores*, indicating that questions generated by our models are more likely to elicit further interactions. Noticeably, HTD outperforms STD significantly, indicating that it is beneficial to specify word types explicitly and apply dynamic vocabularies in generation.

We also observed that STD outperforms Seq2Seq and TA, but the differences are not significant in appropriateness. This is because STD generated about 7% non-question responses which were judged as inappropriate, while Seq2Seq and TA generated universal questions (inappropriate too but beat STD in annotation) to these posts.

4.5.3 Annotation Statistics

The proportion of the pair-wise annotations in which at least three of five annotators assign the same label to a record is 90.57%/93.11%/96.62% for appropriateness/ richness/willingness, respectively. The values show that we have fairly good agreements with majority voting.

4.6 Questioning Pattern Distribution

To analyze whether the model can question with various patterns, we manually annotated the questioning patterns of the responses to 100 sampled posts. The patterns are classified into 11 types including Yes-No, How-, Why-, What-, When-, and Who- questions. We then calculated the KL divergence between the pattern type distribution by a model and that by human (i.e., gold responses).

Results in Table 3 show that the pattern distribution by our model is closer to that in human-written responses, indicating that our decoders can better learn questioning patterns from human language. Further investigation reveals that the baselines tend to generate simple questions like *What?*(什么？) or *Really?*(真的吗), and constantly focus on using one or two question patterns whereas our decoders use more diversified patterns as appeared in the human language.

Model	Seq2Seq	TA	MA	ERM	STD	HTD
KLD	0.431	0.416	0.332	0.276	**0.072**	0.140

Table 3: KL divergence between the questioning pattern distribution by a model and that by human.

4.7 Examples of the Generated Questions

Table 4 presents some generated questions by our decoders, which are more appropriate. On the contrary, Seq2Seq, MA and ERM tend to generate more universal questions. These examples also clearly show that asking questions in open-domain conversational systems requires *scene understanding*, which is verified by this scene example of *singing at karaoke*(在卡拉ok唱歌).

Post:	我们昨天去唱歌了 We went to karaoke yesterday.
Seq2Seq:	**什么时候?** **When?**
MA:	你**怎么**知道我是在问 **How** did you know I am questioning you**?**
TA:	**什么?** **What?**
ERM:	**什么**情况**?** **What** happened **?**
STD:	去**哪里**唱歌**?** **Where** did you sing karaoke**?**
HTD:	你们**几**个人唱歌**?** **How many** people were singing with you**?**

Table 4: Generated question examples. Interrogative words are bolded and topic words underlined.

To further understand how to ask good questions in conversational systems, we presented more generated question examples by human and our typed decoders. The first example (Post-1) in Table 5 shows that a good question can be composed of only interrogatives and ordinary words. This shows the complexity of this task and also the necessity of modeling interrogatives. Post-2 indicates a typical questioning pattern which is ques-

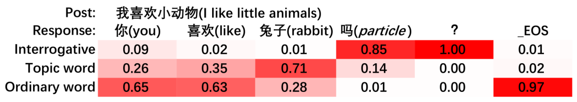
Post-1:	不好不好。哼！讨厌你啦。 No, no. Humph, I hate you!
Human:	我**怎么**了**?** **What** is wrong with me**?**
STD:	我**怎么**了**?** **What** is wrong with me**?**
HTD:	我**怎么**了**?** **What** is wrong with me**?**
Post-2:	刚用了杀虫剂，现在已经有效果了。 I just sprayed the insecticide, and now the effect appears.
Human:	这么快!效果**怎么样?** So quick! **What** is the effect**?**
STD:	效果**如何?** **What** is the effect**?**
HTD:	效果**如何?** **What** is the effect**?**
Post-3:	我在体育中心上班。 I work at the sports center.
Human:	你每天从家过去要**多久?** **How** long does it take you to go there from home**?**
STD:	你是**什么**部门的啊**?** **Which** department are you from**?**
HTD:	那你是体育学院的学生**吗?** **Are** you a student from the College of Physical Education**?**
Post-4:	今天的寿司真好吃。 The sushi today is so delicious!
Human:	我记得你不喜欢吃日式料理的不是**吗?** I remember you didn't like eating Japanese food, **did** you**?**
STD:	你**什么**时候来无锡，我请你。 **When** will you come to Wuxi**?** It is my treat.
HTD:	是**吗?**改天尝尝! Really**?** I will try sometime!

Table 5: Examples for typical questioning patterns. Interrogative words in response are bolded and topic words are underlined.

tioning on a particular topic word (效果-*effect*) of the input. While for Post-3, the questions are asking about transitional topics of the input (上班-*work* → 部门-*department;* 体育中心-*sports center* → 体育学院-*college of Physical Education*), indicating a typical case of **topic transition** in our task (also seen in Post-4, 寿司-sushi →日式料理-Japanese food). This example also demonstrates that for the same input, there are **various questioning patterns**: a How-question asked by human, a Which-question by STD, and a Yes-No question by HTD. As for Post-4, the gold question requires a background that is only shared between the poster and responder, while STD and HTD tend to raise more general questions due to the lack of such shared knowledge.

4.8 Visualization of Type Distribution

To gain more insights into how a word type influence the generation process, we visualized the type probability at each decoding position in HTD. This example (Figure 3) shows that the model can capture word types well at different positions. For instance, at the first and second positions, ordinary words have the highest probabilities for generating 你-*you* and 喜欢-*like*, and at the third position, a topic word 兔子-*rabbit* is predicted while the last two positions are for interrogatives (a particle and a question mark).



Post:	我喜欢小动物(I like little animals)					
Response:	你(you)	喜欢(like)	兔子(rabbit)	吗(*particle*)	?	_EOS
Interrogative	0.09	0.02	0.01	0.85	1.00	0.01
Topic word	0.26	0.35	0.71	0.14	0.00	0.02
Ordinary word	0.65	0.63	0.28	0.01	0.00	0.97

Figure 3: Type distribution examples from HTD. The generated question is "你喜欢兔子吗? *do you like rabbit?*". *_EOS* means end of sentence.

4.9 Error Analysis

We presented error type distribution by manually analyzing 100 bad responses sampled from STD and HTD respectively, where *bad* means the response by our model is worse than that by some baseline during the pair-wise annotation.

There are 4 typical error types: *no topic words (NoT)* in a response (mainly universal questions), *wrong topics (WrT)* where topic words are irrelevant, *type generation error (TGE)* where a wrong word type is predicted (See Eq. 2) and it causes grammatical errors, and *other errors*.

Error Type	NoT	WrT	TGE	Others
STD	34%	34%	29%	3%
HTD	29%	39%	29%	3%

Table 6: Error type distribution.

The error distribution is shown in Table 6. For STD, most of the errors are attributed to no topic or wrong topics, while for HTD, the majority of errors fall into *wrong topics*.

NoT	**Post-1:**	今天好开心啊! I am so happy today!
	STD:	你**怎么**知道**?** **How** do you know **?**
WrT	**Post-2:**	海报非常棒，期待若曦与我们男人的首度合作。 The poster is great and we look forward to our first cooperation with Ruoxi.
	HTD:	你海报**怎么**样啊**?** **How** about your poster **?**
TGE	**Post-3:**	又生病啦?吃点药就好了。 Got sick again? Just take some medicine and you'll be fine soon.
	STD:	我也不知道**怎么**回事。 I don't know **what** happened.
	HTD:	肠胃痛了，上火**吗?** Stomach aching, ulcer **?**

Table 7: Cases for the error types with interrogative words bolded and topic words underlined.

There are typical cases for these error types: (1) Posts such as *"I am so happy today!"* contains no topic words or rare topic words. In this case, our method is unable to predict the topic words so that the models tend to generate universal questions. This happens more frequently in STD because the topic words are not specified explicitly. (2) Posts contains multiple topic words, but the model sometimes focuses on an inappropriate one. For instance, for Post-2 in Table 7, HTD focused on 海报-*poster* but 合作-*cooperation* is a proper one to be focused on. (3) For complex posts, the models failed to predict the correct word type in response. For Post-3, STD generated a declarative sentence and HTD generated a question which, however, is not adequate within the context.

These cases show that controlling the questioning patterns and the informativeness of the content faces with the compatibility issue, which is challenging in language generation. These errors are also partially due to the imperfect ability of topic word prediction by PMI, which is challenging itself in open-domain conversational systems.

5 Conclusion and Future Work

We present two typed decoders to generate questions in open-domain conversational systems. The decoders firstly estimate a type distribution over word types, and then use the type distribution to modulate the final word generation distribution. Through modeling the word types in language generation, the proposed decoders are able to question with various patterns and address novel yet related transitional topics in a generated question. Results show that our models can generate more appropriate questions, with richer topics, thereby more likely to elicit further interactions.

The work can be extended to multi-turn conversation generation by including an additional detector predicting when to ask a question. The detector can be implemented by a classifier or some heuristics. Furthermore, the typed decoders are applicable to the settings where word types can be easily obtained, such as in emotional text generation (Ghosh et al., 2017; Zhou et al., 2018b).

Acknowledgements

This work was partly supported by the National Science Foundation of China under grant No.61272227/61332007 and the National Basic Research Program (973 Program) under grant No. 2013CB329403. We would like to thank Prof. Xiaoyan Zhu for her generous support.

References

Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*. pages 58–67.

A. Andrenucci and E. Sneiders. 2005. Automated question answering: review of the main approaches. In *ICITA*. pages 514–519.

Yllias Chali and Sina Golestanirad. 2016. Ranking automatically generated questions using common human queries. In *INLG*. pages 217–221.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*. pages 1724–1734.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16(1):22–29.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. 2012. Question generation based on lexico-syntactic patterns learned from the web. *Dialogue Discourse* .

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *ACL*. pages 1342–1352.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. In *ACL*. pages 634–642.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *NAACL HLT*. pages 609–617.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* .

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *ACL (1)*. pages 889–898.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*. pages 110–119.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016. Learning through dialogue interactions by asking questions. *arXiv preprint arXiv:1612.04936* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*. pages 1412–1421.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *ACL*. pages 1802–1813.

David Lindberg Fred Popowich and John Nesbit Phil Winne. 2013. Generating natural language questions to support learning on-line. *ENLG* pages 105–114.

Haocheng Qin. 2015. *Question Paraphrase Generation for Question Answering System*. Master's thesis, University of Waterloo.

Linfeng Song, Zhiguo Wang, and Wael Hamza. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058* .

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2017. Neural models for key phrase detection and question generation. *arXiv preprint arXiv:1706.04560* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. pages 3104–3112.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* .

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge*.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. *arXiv preprint arXiv:1706.01450* .

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*. pages 3351–3357.

Zhou Yu, Ziyu Xu, Alan W Black, and Alex I. Rudnicky. 2016. Strategy and policy learning for non-task-oriented conversational systems. In *SIGDIAL*. pages 404–412.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *The Workshop on Representation Learning for NLP*. pages 15–25.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017a. Mechanism-aware neural machine for dialogue response generation. In *AAAI*. pages 3400–3407.

Ganbin Zhou, Ping Luo, Yijun Xiao, Fen Lin, Bo Chen, and Qing He. 2018a. Elastic responding machine for dialog generation with dynamically mechanism selecting. In *AAAI Conference on Artificial Intelligence, AAAI*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018b. Emotional chatting machine: Emotional conversation generation with internal and external memory. *AAAI* .

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017b. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .

Personalizing Dialogue Agents: I have a dog, do you have pets too?

Saizheng Zhang[†,1], **Emily Dinan**[‡], **Jack Urbanek**[‡], **Arthur Szlam**[‡], **Douwe Kiela**[‡], **Jason Weston**[‡]
[†] Montreal Institute for Learning Algorithms, MILA
[‡] Facebook AI Research
saizheng.zhang@umontreal.ca, {edinan,jju,aszlam,dkiela,jase}@fb.com

Abstract

Chit-chat models are known to have several problems: they lack specificity, do not display a consistent personality and are often not very captivating. In this work we present the task of making chit-chat more engaging by conditioning on profile information. We collect data and train models to (i) condition on their given profile information; and (ii) information about the person they are talking to, resulting in improved dialogues, as measured by next utterance prediction. Since (ii) is initially unknown, our model is trained to engage its partner with personal topics, and we show the resulting dialogue can be used to predict profile information about the interlocutors.

1 Introduction

Despite much recent success in natural language processing and dialogue research, communication between a human and a machine is still in its infancy. It is only recently that neural models have had sufficient capacity and access to sufficiently large datasets that they appear to generate meaningful responses in a chit-chat setting. Still, conversing with such generic chit-chat models for even a short amount of time quickly exposes their weaknesses (Serban et al., 2016; Vinyals and Le, 2015).

Common issues with chit-chat models include: (i) the lack of a consistent personality (Li et al., 2016a) as they are typically trained over many dialogs each with different speakers, (ii) the lack of an explicit long-term memory as they are typically trained to produce an utterance given only the recent dialogue history (Vinyals and Le, 2015); and (iii) a tendency to produce non-specific answers like "I don't know" (Li et al., 2015). Those three problems combine to produce an unsatisfying overall experience for a human to engage with. We believe some of those problems are due to there being no good publicly available dataset for general chit-chat.

[1]Work done while at Facebook AI Research.

Because of the low quality of current conversational models, and because of the difficulty in evaluating these models, chit-chat is often ignored as an end-application. Instead, the research community has focused on task-oriented communication, such as airline or restaurant booking (Bordes and Weston, 2016), or else single-turn information seeking, i.e. question answering (Rajpurkar et al., 2016). Despite the success of the latter, simpler, domain, it is well-known that a large quantity of human dialogue centers on socialization, personal interests and chit-chat (Dunbar et al., 1997). For example, less than 5% of posts on Twitter are questions, whereas around 80% are about personal emotional state, thoughts or activities, authored by so called "Meformers" (Naaman et al., 2010).

In this work we make a step towards more engaging chit-chat dialogue agents by endowing them with a configurable, but persistent persona, encoded by multiple sentences of textual description, termed a profile. This profile can be stored in a memory-augmented neural network and then used to produce more personal, specific, consistent and engaging responses than a persona-free model, thus alleviating some of the common issues in chit-chat models. Using the same mechanism, any existing information about the persona of the dialogue partner can also be used in the same way. Our models are thus trained to both ask and answer questions about personal topics, and the resulting dialogue can be used to build a model of the persona of the speaking partner.

To support the training of such models, we present the PERSONA-CHAT dataset, a new dialogue dataset consisting of 164,356 utterances between crowdworkers who were randomly paired and each asked to act the part of a given provided persona (randomly assigned, and created by another set of crowdworkers). The paired workers were asked to chat naturally and to get to know each other during the conversation. This produces interesting and engaging conversations that our agents can try to learn to mimic.

Studying the next utterance prediction task during dialogue, we compare a range of models: both generative and ranking models, including Seq2Seq models and Memory Networks (Sukhbaatar et al., 2015) as well as other standard retrieval baselines. We show experimentally that in either the generative or ranking case conditioning the agent with persona information gives improved prediction of the next dialogue utterance. The PERSONA-CHAT dataset is designed to facilitate research into alleviating some of the issues that traditional chit-chat models face, and with the aim of making such models more consistent and engaging, by endowing them with a persona. By comparing against chit-chat models built using the OpenSubtitles and Twitter datasets, human evaluations show that our dataset provides more engaging models, that are simultaneously capable of being fluent and consistent via conditioning on a persistent, recognizable profile.

2 Related Work

Traditional dialogue systems consist of building blocks, such as dialogue state tracking components and response generators, and have typically been applied to tasks with labeled internal dialogue state and precisely defined user intent (i.e., goal-oriented dialogue), see e.g. (Young, 2000). The most successful goal-oriented dialogue systems model conversation as partially observable Markov decision processes (POMDPs) (Young et al., 2013). All those methods typically do not consider the chit-chat setting and are more concerned with achieving functional goals (e.g. booking an airline flight) than displaying a personality. In particular, many of the tasks and datasets available are constrained to narrow domains (Serban et al., 2015).

Non-goal driven dialogue systems go back to Weizenbaum's famous program ELIZA (Weizenbaum, 1966), and hand-coded systems have continued to be used in applications to this day. For example, modern solutions that build an open-ended dialogue system to the Alexa challenge combine hand-coded and machine-learned elements (Serban et al., 2017a). Amongst the simplest of statistical systems that can be used in this domain, that are based on data rather than hand-coding, are information retrieval models (Sordoni et al., 2015), which retrieve and rank responses based on their matching score with the recent dialogue history. We use IR systems as a baseline in this work.

End-to-end neural approaches are a class of models which have seen growing recent interest. A popular class of methods are generative recurrent systems like seq2seq applied to dialogue (Sutskever et al., 2014; Vinyals and Le, 2015; Sordoni et al., 2015; Li et al., 2016b; Serban et al., 2017b). Rooted in language modeling, they are able to produce syntactically coherent novel responses, but their memory-free approach means they lack long-term coherence and a persistent personality, as discussed before. A promising direction, that is still in its infancy, to fix this issue is to use a memory-augmented network instead (Sukhbaatar et al., 2015; Dodge et al., 2015) by providing or learning appropriate memories.

Serban et al. (2015) list available corpora for training dialogue systems. Perhaps the most relevant to learning chit-chat models are ones based on movie scripts such as OpenSubtitles and Cornell Movie-Dialogue Corpus, and dialogue from web platforms such as Reddit and Twitter, all of which have been used for training neural approaches (Vinyals and Le, 2015; Dodge et al., 2015; Li et al., 2016b; Serban et al., 2017b). Naively training on these datasets leads to models with the lack of a consistent personality as they will learn a model averaged over many different speakers. Moreover, the data does little to encourage the model to engage in understanding and maintaining knowledge of the dialogue partner's personality and topic interests.

According to Serban et al. (2015)'s survey, personalization of dialogue systems is "an important task, which so far has not received much attention". In the case of goal-oriented dialogue some work has focused on the agent being aware of the human's profile and adjusting the dialogue accordingly, but without a personality to the agent itself (Lucas et al., 2009; Joshi et al., 2017). For the chit-chat setting, the most relevant work is (Li et al., 2016a). For each user in the Twitter corpus, personas were captured via distributed embeddings (one per speaker) to encapsulate individual characteristics such as background information and speaking style, and they then showed using those vectors improved the output of their seq2seq model for the same speaker. Their work does not focus on attempting to engage the other speaker by getting to know them, as we do here. For that reason, our focus is on explicit profile information, not hard-to-interpret latent variables.

3 The PERSONA-CHAT Dataset

The aim of this work is to facilitate more engaging and more personal chit-chat dialogue. The PERSONA-CHAT dataset is a crowd-sourced dataset, collected via Amazon Mechanical Turk, where each of the pair of speakers condition their dialogue on a given profile, which is provided.

The data collection consists of three stages:

(i) Personas: we crowdsource a set of 1155 possible personas, each consisting of at least 5 profile sentences, setting aside 100 never seen before personas for validation, and 100 for test.

(ii) Revised personas: to avoid modeling that takes advantage of trivial word overlap, we crowdsource additional rewritten sets of the same 1155 personas, with related sentences that are rephrases, generalizations or specializations, rendering the task much more challenging.

(iii) Persona chat: we pair two Turkers and assign them each a random (original) persona from the pool, and ask them to chat. This resulted in a dataset of 164,356 utterances over 10,981 dialogs, 15,705 utterances (968 dialogs) of which are set aside for validation, and 15,119 utterances (1000 dialogs) for test.

The final dataset and its corresponding data collection source code, as well as models trained on the data, are all available open source in ParlAI[2].

In the following, we describe each data collection stage and the resulting tasks in more detail.

3.1 Personas

We asked the crowdsourced workers to create a character (persona) description using 5 sentences, providing them only a single example:

"I am a vegetarian. I like swimming. My father used to work for Ford. My favorite band is Maroon5. I got a new job last month, which is about advertising design."

Our aim was to create profiles that are natural and descriptive, and contain typical topics of human interest that the speaker can bring up in conversation. Because the personas are not the real profiles of the Turkers, the dataset does not contain personal information (and they are told specifically not to use any). We asked the workers to make each sentence short, with a maximum of 15 words per sentence. This is advantageous both for humans and machines: if they are too long, crowdsourced workers are likely to lose interest, and for machines the task could become more difficult.

Some examples of the personas collected are given in Table 1 (left).

3.2 Revised Personas

A difficulty when constructing dialogue datasets, or text datasets in general, is that in order to encourage research progress, the task must be carefully constructed so that is neither too easy nor too difficult for the current technology (Voorhees et al., 1999). One issue with conditioning on textual personas is that there is a danger that humans will, even if asked not to, unwittingly repeat profile information either verbatim or with significant word overlap. This may make any subsequent machine learning tasks less challenging, and the solutions will not generalize to more difficult tasks. This has been a problem in some recent datasets: for example, the dataset curation technique used for the well-known SQuAD dataset suffers from this word overlap problem to a certain extent (Chen et al., 2017).

To alleviate this problem, we presented the original personas we collected to a new set of crowdworkers and asked them to rewrite the sentences so that a new sentence is about *"a related characteristic that the same person may have"*, hence the revisions could be rephrases, generalizations or specializations. For example *"I like basketball"* can be revised as *"I am a big fan of Michael Jordan"* not because they mean the same thing but because the same persona could contain both.

In the revision task, workers are instructed not to trivially rephrase the sentence by copying the original words. However, during the entry stage if a non-stop word is copied we issue a warning,

[2]`https://github.com/facebookresearch/ParlAI/tree/master/projects/personachat`

Original Persona	**Revised Persona**
I love the beach.	To me, there is nothing like a day at the seashore.
My dad has a car dealership	My father sales vehicles for a living.
I just got my nails done	I love to pamper myself on a regular basis.
I am on a diet now	I need to lose weight.
Horses are my favorite animal.	I am into equestrian sports.
I play a lot of fantasy videogames.	RPGs are my favorite genre.
I have a computer science degree.	I also went to school to work with technology.
My mother is a medical doctor	The woman who gave birth to me is a physician.
I am very shy.	I am not a social person.
I like to build model spaceships.	I enjoy working with my hands.

Table 1: Example Personas (left) and their revised versions (right) from the PERSONA-CHAT dataset. The revised versions are designed to be characteristics that the same persona might have, which could be rephrases, generalizations or specializations.

Persona 1	**Persona 2**
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

Table 2: Example dialog from the PERSONA-CHAT dataset. Person 1 is given their own persona (top left) at the beginning of the chat, but does not know the persona of Person 2, and vice-versa. They have to get to know each other during the conversation.

and ask them to rephrase, guaranteeing that the instructions are followed. For example, *"My father worked for Ford."* can be revised to *"My dad worked in the car industry"*, but not *"My dad was employed by Ford."* due to word overlap.

Some examples of the revised personas collected are given in Table 1 (right).

3.3 Persona Chat

After collecting personas, we then collected the dialogues themselves, conditioned on the personas. For each dialogue, we paired two random crowdworkers, and gave them the instruction that they will chit-chat with another worker, while playing the part of a given character. We then provide them with a randomly chosen persona from our pool, different to their partners. The instructions are on purpose quite terse and simply ask them to "chat with the other person naturally and try to get to know each other". In an early study we noticed the crowdworkers tending to talk about themselves (their own persona) too much, so we also added the instructions "both ask questions and answer questions of your chat partner" which seemed to help. We also gave a bonus for high quality dialogs. The dialog is turn-based, with a maximum of 15 words per message. We again gave instructions to not trivially copy the character descriptions into the messages, but also wrote explicit code sending them an error if they tried to do so, using simple string matching. We define a minimum dialogue length which is randomly between 6 and 8 turns each for each dialogue. An example dialogue from the dataset is given in Table 2.

3.4 Evaluation

We focus on the standard dialogue task of predicting the next utterance given the dialogue history, but consider this task both with and without the profile information being given to the learning agent. Our goal is to enable interesting directions for future research, where chatbots can for instance have personalities, or imputed personas could be used to make dialogue more engaging to the user.

We consider this in four possible scenarios: conditioning on no persona, your own persona, their persona, or both. These scenarios can be tried using either the original personas, or the revised ones. We then evaluate the task using three metrics: (i) the log likelihood of the correct sequence, measured via perplexity, (ii) F1 score, and (iii) next utterance classification loss, following Lowe et al. (2015). The latter consists of choosing N random distractor responses from other dialogues (in our setting, N=19) and the model selecting the best response among them, resulting in a score of one if the model chooses the correct response, and zero otherwise (called hits@1 in the experiments).

4 Models

We consider two classes of model for next utterance prediction: ranking models and generative models. Ranking models produce a next utterance by considering any utterance in the training set as a possible candidate reply. Generative models generate novel sentences by conditioning on the dialogue history (and possibly, the persona), and then generating the response word-by-word. Note one can still evaluate the latter as ranking models by computing the probability of generating a given candidate, and ranking candidates by those scores.

4.1 Baseline ranking models

We first consider two baseline models, an IR baseline (Sordoni et al., 2015) and a supervised embedding model, Starspace (Wu et al., 2017)[3]. While there are many IR variants, we adopt the simplest one: find the most similar message in the (training) dataset and output the response from that exchange. Similarity is measured by the tf-idf weighted cosine similarity between the bags of words. Starspace is a recent model that also performs information retrieval but by learning the similarity between the dialog and the next utterance by optimizing the embeddings directly for that task using the margin ranking loss and k-negative sampling. The similarity function $sim(q, c')$ is the cosine similarity of the sum of word embeddings of the query q and candidate c'. Denoting the dictionary of $\mathcal{D}$ word embeddings as W which is a $\mathcal{D} \times d$ matrix, where W_i indexes the i^{th} word (row), yielding its d-dimensional embedding, it embeds the sequences q and c'.

[3]github.com/facebookresearch/StarSpace

In both methods, IR and StarSpace, to incorporate the profile we simply concatenate it to the query vector bag of words.

4.2 Ranking Profile Memory Network

Both the previous models use the profile information by combining it with the dialogue history, which means those models cannot differentiate between the two when deciding on the next utterance. In this model we instead use a memory network with the dialogue history as input, which then performs attention over the profile to find relevant lines from the profile to combine with the input, and then finally predicts the next utterance. We use the same representation and loss as in the Starspace model, so without the profile, the two models are identical. When the profile is available attention is performed by computing the similarity of the input q with the profile sentences p_i, computing the softmax, and taking the weighted sum:

$$q^+ = q + \sum s_i p_i, \qquad s_i = \text{Softmax}(sim(q, p_i))$$

where $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$. One can then rank the candidates c' using $sim(q^+, c')$. One can also perform multiple "hops" of attention over the profile rather than one, as shown here, although that did not bring significant gains in our parameter sweeps.

4.3 Key-Value Profile Memory Network

The key-value (KV) memory network (Miller et al., 2016) was proposed as an improvement to the memory network by performing attention over keys and outputting the values (instead of the same keys as in the original), which can outperform memory networks dependent on the task and definition of the key-value pairs. Here, we apply this model to dialogue, and consider the keys as dialog histories (from the training set), and the values as the next dialogue utterances, i.e., the replies from the speaking partner. This allows the model to have a memory of past dialogues that it can directly use to help influence its prediction for the current conversation. The model we choose is identical to the profile memory network just described in the first hop over profiles, while in the second hop, q^+ is used to attend over the keys and output a weighted sum of values as before, producing q^{++}. This is then used to rank the candidates c' using $sim(q^{++}, c')$ as before. As the set of (key-value) pairs is large this would make training very slow. In our experiments we simply trained the profile memory network and used the same weights from that model and applied this architecture at test time instead. Training the model directly would presumably give better results, however this heuristic already proved beneficial compared to the original network.

4.4 Seq2Seq

The input sequence x is encoded by applying $h_t^e = LSTM_{enc}(x_t \mid h_{t-1}^e)$. We use GloVe (Pennington et al., 2014) for our word embeddings. The final hidden state, h_t^e, is fed into the decoder $LSTM_{dec}$ as the initial state h_0^d. For each time step t, the decoder then produces the probability of a word j occurring in that place via the softmax, i.e.,

$$p(y_{t,j} = 1 \mid y_{t-1}, \dots, y_1) = \frac{\exp(w_j h_t^d)}{\sum_{j'=1}^{K} \exp(w_{j'} h_t^d)}.$$

The model is trained via negative log likelihood. The basic model can be extended to include persona information, in which case we simply prepend it to the input sequence x, i.e., $x = \forall p \in P \mid\mid x$, where $\mid\mid$ denotes concatenation. For the OpenSubtitles and Twitter datasets trained in Section 5.2 we found training a language model (LM), essentially just the decoder part of this model, worked better and we report that instead.

4.5 Generative Profile Memory Network

Finally, we introduce a generative model that encodes each of the profile entries as individual memory representations in a memory network. As before, the dialogue history is encoded via $LSTM_{enc}$, the final state of which is used as the initial hidden state of the decoder. Each entry $p_i = \langle p_{i,1}, \dots, p_{i,n} \rangle \in P$ is then encoded via $f(p_i) = \sum_j^{|p_i|} \alpha_i p_{i,j}$. That is, we weight words by their inverse term frequency: $\alpha_i = 1/(1 + log(1 + \text{tf}))$ where tf is computed from the GloVe index via Zipf's law[4]. Let F be the set of encoded memories. The decoder now attends over the encoded profile entries, i.e., we compute the mask a_t, context c_t and next input $\hat{x}_t$ as:

$$a_t = softmax(FW_a h_t^d),$$
$$c_t = a_t^\intercal F; \quad \hat{x}_t = tanh(W_c[c_{t-1}, x_t]).$$

If the model has no profile information, and hence no memory, it becomes equivalent to the Seq2Seq model.

5 Experiments

We first report results using automated evaluation metrics, and subsequently perform an extrinsic evaluation where crowdsourced workers perform a human evaluation of our models.

5.1 Automated metrics

The main results are reported in Table 3. Overall, the results show the following key points:

Persona Conditioning Most models improve significantly when conditioning prediction on their own persona at least for the original (non-revised) versions, which is an easier task than the revised ones which have no word overlap. For example, the Profile Memory generation model has improved perplexity and hits@1 compared to Seq2Seq, and all the ranking algorithms (IR baseline, Starspace and Profile Memory Networks) obtain improved hits@1.

Ranking vs. Generative. Ranking models are far better than generative models at ranking. This is perhaps obvious as that is the metric they are optimizing, but still the performance difference is quite stark. It may be that the word-based probability which generative models use works well, but is not calibrated well enough to give a sentence-based probability which ranking requires. Human evaluation is also used to compare these methods, which we perform in Sec. 5.2.

Ranking Models. For the ranking models, the IR baseline is outperformed by Starspace due to its learnt similarity metric, which in turn is outperformed by Profile Memory networks due to the attention mechanism over the profiles (as all other parts of the models are the same). Finally KV Profile Memory networks outperform Profile Memory Networks in the no persona case due to the ability to consider neighboring dialogue history and next

[4]tf $= 1e6 * 1/(idx^{1.07})$

Method	**No Persona**		**Original Persona**		**Revised Persona**	
	ppl	**hits@1**	**ppl**	**hits@1**	**ppl**	**hits@1**
Generative Models						
Seq2Seq	38.08	0.092	40.53	0.084	40.65	0.082
Profile Memory	38.08	0.092	34.54	0.125	38.21	0.108
Ranking Models						
IR baseline	-	0.214	-	0.410	-	0.207
Starspace	-	0.318	-	0.491	-	0.322
Profile Memory	-	0.318	-	0.509	-	0.354
KV Profile Memory	-	0.349	-	0.511	-	0.351

Table 3: **Evaluation of dialog utterance prediction with various models** in three settings: without conditioning on a persona, conditioned on the speakers given persona ("Original Persona"), or a revised persona that does not have word overlap.

Method					**Persona**
Model	Profile	**Fluency**	**Engagingness**	**Consistency**	**Detection**
Human	Self	4.31(1.07)	4.25(1.06)	4.36(0.92)	0.95(0.22)
Generative PersonaChat Models					
Seq2Seq	None	3.17(1.10)	3.18(1.41)	2.98(1.45)	0.51(0.50)
Profile Memory	Self	3.08(1.40)	3.13(1.39)	3.14(1.26)	0.72(0.45)
Ranking PersonaChat Models					
KV Memory	None	3.81(1.14)	3.88(0.98)	3.36(1.37)	0.59(0.49)
KV Profile Memory	Self	3.97(0.94)	3.50(1.17)	3.44(1.30)	0.81(0.39)
Twitter LM	None	3.21(1.54)	1.75(1.04)	1.95(1.22)	0.57(0.50)
OpenSubtitles 2018 LM	None	2.85(1.46)	2.13(1.07)	2.15(1.08)	0.35(0.48)
OpenSubtitles 2009 LM	None	2.25(1.37)	2.12(1.33)	1.96(1.22)	0.38(0.49)
OpenSubtitles 2009 KV Memory	None	2.14(1.20)	2.22(1.22)	2.06(1.29)	0.42(0.49)

Table 4: **Human Evaluation** of various PERSONA-CHAT models, along with a comparison to human performance, and Twitter and OpenSubtitles based models (last 4 rows), standard deviation in parenthesis.

utterance pairs in the training set that are similar to the current dialogue, however when using persona information the performance is similar.

Revised Personas. Revised personas are much harder to use. We do however still see some gain for the Profile Memory networks compared to none (0.354 vs. 0.318 hits@1). We also tried two variants of training: with the original personas in the training set or the revised ones, a comparison of which is shown in Table 6 of the Appendix. *Training* on revised personas helps, both for test examples that are in original form or revised form, likely due to the model be forced to learn more than simple word overlap, forcing the model to generalize more (i.e., learn semantic similarity of differing phrases).

Their Persona. We can also condition a model on the other speaker's persona, or both personas at once, the results of which are in Tables 5 and 6 in the Appendix. Using "Their persona" has less impact on this dataset. We believe this is because most speakers tend to focus on themselves when it comes to their interests. It would be interesting how often this is the case in other datasets. Certainly this is skewed by the particular instructions one could give to the crowdworkers. For example if we gave the instructions "try not to talk about yourself, but about the other's interests' likely these metrics would change.

5.2 Human Evaluation

As automated metrics are notoriously poor for evaluating dialogue (Liu et al., 2016) we also perform human evaluation using crowdsourced workers. The procedure is as follows. We perform almost exactly the same setup as in the dataset collection process itself as in Section 3.3. In that setup, we paired two Turkers and assigned them each a random (original) persona from the collected pool, and asked them to chat. Here, from the Turker's point of view everything looks the same except instead of being paired with a Turker they are paired with one of our models instead (they do not know this). In this setting, for both the Turker and the model, the personas come from the test set pool.

After the dialogue, we then ask the Turker some additional questions in order to evaluate the quality of the model. We ask them to evaluate fluency, engagingness and consistency (scored between 1-5). Finally, we measure the ability to detect the other speaker's profile by displaying two possible profiles, and ask which is more likely to be the profile of the person the Turker just spoke to. More details of these measures are given in the Appendix.

The results are reported in Table 4 for the best performing generative and ranking models, in both the No Persona and Self Persona categories, 100 dialogues each. We also evaluate the scores of human performance by replacing the chatbot with a human (another Turker). This effectively gives us upper bound scores which we can aim for with our models. Finally, and importantly, we compare our models trained on PERSONA-CHAT with chit-chat models trained with the Twitter and OpenSubtitles datasets (2009 and 2018 versions) instead, following Vinyals and Le (2015). Example chats from a few of the models are shown in the Appendix in Tables 7, 8, 9, 10, 11 and 12.

Firstly, we see a difference in fluency, engagingness and consistency between all PERSONA-CHAT models and the models trained on OpenSubtitles and Twitter. PERSONA-CHAT is a resource that is particularly strong at providing training data for the beginning of conversations, when the two speakers do not know each other, focusing on asking and answering questions, in contrast to other resources. We also see suggestions of more subtle differences between the models, although these differences are obscured by the high variance of the human raters' evaluations. For example, in both the generative and ranking model cases, models endowed with a persona can be detected by the human conversation partner, as evidenced by the persona detection accuracies, whilst maintaining fluency and consistency compared to their non-persona driven counterparts.

Finding the balance between fluency, engagement, consistency, and a persistent persona remains a strong challenge for future research.

5.3 Profile Prediction

Two tasks could naturally be considered using PERSONACHAT: (1) next utterance prediction during dialogue, and (2) profile prediction given dialogue history. The main study of this work has been Task 1, where we have shown the use of profile information. Task 2, however, can be used to extract such information. While a full study is beyond the scope of this paper, we conducted some preliminary experiments, the details of which are in Appendix D. They show (i) human speaker's profiles can be predicted from their dialogue with high accuracy (94.3%, similar to human performance in Table 4) or even from the model's dialogue (23% with KV Profile Memory) showing the model is paying attention to the human's interests. Further, the accuracies clearly improve with further dialogue, as shown in Table 14. Combining Task 1 and Task 2 into a full system is an exciting area of future research.

6 Conclusion & Discussion

In this work we have introduced the PERSONA-CHAT dataset, which consists of crowd-sourced dialogues where each participant plays the part of an assigned persona; and each (crowd-sourced) persona has a word-distinct paraphrase. We test various baseline models on this dataset, and show that models that have access to their own personas in addition to the state of the dialogue are scored as more consistent by annotators, although not more engaging. On the other hand, we show that models trained on PERSONA-CHAT (with or without personas) are more engaging than models trained on dialogue from other resources (movies, Twitter).

We believe PERSONA-CHAT will be a useful resource for training components of future dialogue systems. Because we have paired human generated profiles and conversations, the data aids the construction of agents that have consistent personalities and viewpoints. Furthermore, predicting the profiles from a conversation moves chit-chat tasks in the direction of goal-directed dialogue, which has metrics for success. Because we collect paraphrases of the profiles, they cannot be trivially matched; indeed, we believe the original and rephrased profiles are interesting as a semantic similarity dataset in their own right. We hope that the data will aid training agents that can ask questions about users' profiles, remember the answers, and use them naturally in conversation.

References

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*.

Robin IM Dunbar, Anna Marriott, and Neil DC Duncan. 1997. Human conversational behavior. *Human nature*, 8(3):231–246.

Chaitanya K Joshi, Fei Mi, and Boi Faltings. 2017. Personalization in goal-oriented dialog. *arXiv preprint arXiv:1706.07503*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

JM Lucas, F Fernández, J Salazar, J Ferreiros, and R San Segundo. 2009. Managing speaker identity and user profiles in a spoken dialogue system. *Procesamiento del Lenguaje Natural*, 43:77–84.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. 2010. Is it really about me?: message content in social awareness streams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 189–192. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017a. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016. Generative deep neural networks for dialogue: A short review. *arXiv preprint arXiv:1611.06216*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.

Joseph Weizenbaum. 1966. Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017. Starspace: Embed all the things! *arXiv preprint arXiv:1709.03856*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Steve J Young. 2000. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.

Efficient Large-Scale Neural Domain Classification with Personalized Attention

Young-Bum Kim **Dongchan Kim** **Anjishnu Kumar** **Ruhi Sarikaya**
Amazon Alexa
{youngbum,dongchan,anjikum,rsarikaya}@amazon.com

Abstract

In this paper, we explore the task of mapping spoken language utterances to one of thousands of natural language understanding domains in intelligent personal digital assistants (IPDAs). This scenario is observed in mainstream IPDAs in industry that allow third parties to develop thousands of new domains to augment built-in first party domains to rapidly increase domain coverage and overall IPDA capabilities. We propose a scalable neural model architecture with a shared encoder, a novel attention mechanism that incorporates personalization information and domain-specific classifiers that solves the problem efficiently. Our architecture is designed to efficiently accommodate incremental domain additions achieving two orders of magnitude speed up compared to full model retraining. We consider the practical constraints of real-time production systems, and design to minimize memory footprint and runtime latency. We demonstrate that incorporating personalization significantly improves domain classification accuracy in a setting with thousands of overlapping domains.

1 Introduction

Intelligent personal digital assistants (IPDAs) are one of the most advanced and successful artificial intelligence applications that have spoken language understanding (SLU). Many IPDAs have recently emerged in industry including Amazon Alexa, Google Assistant, Apple Siri, and Microsoft Cortana (Sarikaya, 2017). IPDAs have traditionally supported only dozens of well-separated domains, each defined in terms of a specific application or functionality such as calendar and local search (Tur and de Mori, 2011; Sarikaya et al., 2016). To rapidly increase domain coverage and extend capabilities, some IPDAs have released Software Development Toolkits (SDKs) to allow third-party developers to quickly build and integrate new domains, which we refer to as *skills* henceforth. Amazon's *Alexa Skills Kit* (Kumar et al., 2017a), Google's *Actions* and Microsoft's *Cortana Skills Kit* are all examples of such SDKs. Alexa Skills Kit is the largest of these services with over 40,000 skills.

For IPDAs, finding the most relevant skill to handle an utterance is an open problem for three reasons. First, the sheer number of skills makes the task difficult. Unlike traditional systems that have on the order of 10-20 built-in domains, large-scale IPDAs can have up to 40,000 skills. Second, the number of skills is rapidly expanding with 100+ new skills added per week. Large-scale IPDAs should be able to accommodate new skills efficiently without compromising performance. Third, unlike traditional built-in domains that are carefully designed to be disjoint by a central team, skills are built independently by different developers and can cover overlapping functionalities. For instance, there are over 50 recipe skills in Alexa that can handle recipe-related utterances.

One simple solution to this problem has been to require the user to explicitly mention a skill name and follow a strict invocation pattern as in *"Ask {Uber} to {get me a ride}."* However, this significantly limits the natural interaction with IPDAs. Users have to remember skill names and invocation patterns, and it places a cognitive burden on users who tend to forget both. Skill discovery is difficult with a pure voice user interface, it is hard for users to know the capabilities of thousands of skills a priori, which may leads to limited user engagement with skills and potentially with IPDAs.

In this paper, we propose a solution that addresses all three practical challenges without requiring skill names or invocation patterns. Our approach is based on a scalable neural model architecture with a shared encoder, a skill attention mechanism and skill-specific classification networks that can efficiently perform large-scale skill classification in IPDAs using a weakly supervised training dataset. We demonstrate that our model achieves a high accuracy on a manually transcribed test set after being trained with weak supervision. Moreover, our architecture is designed to efficiently integrate new skills that appear in-between full model retraining cycles into the model. Besides accuracy, we also keep practical constraints in mind and focus on minimizing memory footprint and runtime latency, while ensuring architecture is scalable to thousands of skills, all of which are important for real-time production systems. Furthermore, we investigate two different ways of incorporating user personalization information into the model, our naive baseline method adds the information as a 1-bit flag in the feature space of the skill-specific networks, the *personalized attention* technique computes a convex combination of skill embeddings for the user's enabled skills and significantly outperforms the naive personalization baseline. We show the effectiveness of our approach with extensive experiments using 1,500 skills from a deployed IPDA system.

2 Related Work

Traditional multi-domain SLU/NLU systems are designed hierarchically, starting with domain classification to classify an incoming utterance into one of many possible domains, followed by further semantic analysis with domain-specific intent classification and slot tagging (Tur and de Mori, 2011). Traditional systems have typically been limited to a small number of domains, designed by specialists to be well-separable. Therefore, domain classification has been considered a less complex task compared to other semantic analysis such as intent and slot predictions. Traditional domain classifiers are built using simple linear models such as Multinomial Logistic Regression or Support Vector Machines in a one-versus-all setting for multi-class prediction. The models typically use word n-gram features and also those based on static lexicon match, and there have been several recent studies applying deep learning techniques (Xu and Sarikaya, 2014).

There is also a line of prior work on enhancing sequential text classification or tagging. Hierarchical character-to-word level LSTM (Hochreiter and Schmidhuber, 1997) architectures similar to our models have been explored for the Named Entity Recognition task by Lample et al. (2016). Character-informed sequence models have also been explored for simple text classification with small sets of classes by Xiao and Cho (2016). Joulin et al. (2016) explored highly scalable text classification using a shared hierarchical encoder, but their hierarchical softmax-based output formulation is unsuitable for incremental model updates. Work on zero-shot domain classifier expansion by Kumar et al. (2017b) struggled to rank incoming domains higher than training domains. The attention-based approach of Kim et al. (2017d) does not require retraining from scratch, but it requires keeping all models stored in memory which is computationally expensive. Multi-Task learning was used in the context of SLU by Tur (2006) and has been further explored using neural networks for phoneme recognition (Seltzer and Droppo, 2013) and semantic parsing (Fan et al., 2017; Bapna et al., 2017). There have been many other pieces of prior work on improving NLU systems with pre-training (Kim et al., 2015b; Celikyilmaz et al., 2016; Kim et al., 2017e), multi-task learning (Zhang and Wang, 2016; Liu and Lane, 2016; Kim et al., 2017b), transfer learning (El-Kahky et al., 2014; Kim et al., 2015a,c; Chen et al., 2016a; Yang et al., 2017), domain adaptation (Kim et al., 2016; Jaech et al., 2016; Liu and Lane, 2017; Kim et al., 2017d,c) and contextual signals (Bhargava et al., 2013; Chen et al., 2016b; Hori et al., 2016; Kim et al., 2017a).

3 Weakly Supervised Training Data Generation

Our model addresses the domain classification task in SLU systems. In traditional IPDA systems, these domains are hand-crafted by experts to be well separable and can easily be annotated by humans because they are small in number. The emergence of self-service SLU results in a large number of potentially mutually overlapping SLU domains. This means that eliciting large volumes of high quality human annotations to train our model

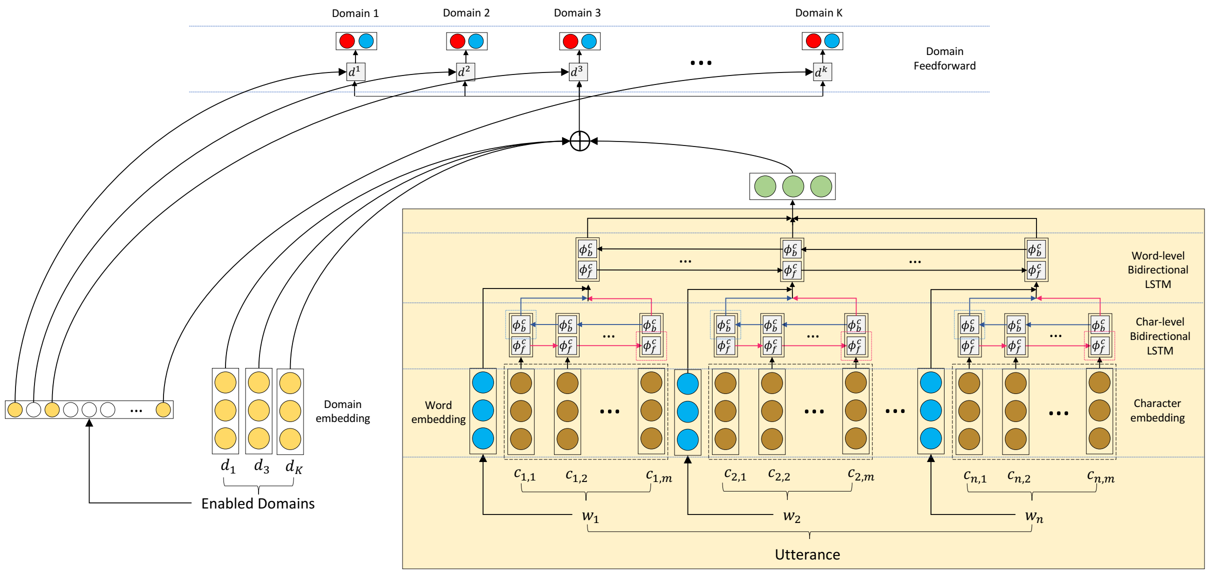


Figure 1: The overall architecture of the personalized dynamic domain classifier.

is no longer feasible, and we cannot assume that domains are designed to be well separable.

Instead we can generate training data by adopting the weak supervision paradigm introduced by (Hoffmann et al., 2011), which proposes using heuristic labeling functions generate large numbers of noisy data samples. Clean data generation with weak supervision is a challenging problem, so we address it by decomposing it into two simpler problems, of candidate generation and noise suppression, however it remains important for our model to be noise robust.

3.1 Data Programming

The key insight of the *Data Programming* approach is that $O(1)$ simple labeling functions can be used to approximate $O(n)$ human annotated data points with much less effort. We adopt the formalism used by (Ratner et al., 2016) to treat each of instance data generation rule as a rich generative model, defined by a labeling function λ and describe different families of labeling functions. Our data programming pipeline is analogous to the noisy channel model proposed for spelling correction by (Kernighan et al., 1990), and consists of a set of candidate generation and noise detection functions.

$$\arg\max_{\mu} P(\mu|s_i) = \arg\max_{\mu} P(s_i|\mu).\, P(\mu)$$

where μ and s_i represent utterances and the ith skill respectively. $P(s_i|\mu)$ the probability of a skill being valid for an utterance is approximated by simple functions that act as candidate data *generators* $\lambda_g \in \Lambda_g$ based on recognitions produced by a family of *query patterns* $\lambda_q \in \Lambda_q$. $P(\mu)$ is represented by a family of simple functions that act as *noise detectors* $\lambda_n \in \Lambda_n$, which mark utterances as likely being noise.

We apply the technique to the query logs of a popular IPDA, which has support for personalized third party domains. Looking at the structure of utterances that match query pattern λ_q, each utterance of form *"Ask $\{$Uber$\}$ to $\{$get me a car$\}$"* can be considered as being parametrized by the underlying latent command μ_z, that is *"Get me a car"*, a target domain corresponding to service s_t, which in this case is *Uber* and the query recognition pattern λ_q, in this case *"Ask $\{s_t\}$ to $\{\mu_z\}$"*. Next we assume that the distribution of latent commands over domains are independent of the query pattern.

$$P(\mu_z, s_t) \approx P(\mu, s_t, \lambda_q)$$

Making this simple distributional approximation allows us to generate a large number of noisy training samples. The family of generator functions $\lambda_g \in \Lambda_g$ is thus defined such that $u_z = \lambda_g^i(\mu, \lambda_q^i)$

3.2 Noise Reduction

The distribution defined above contains a large number of noisy positive samples. Related to $P(\mu)$ in the noisy channel in the spell correction context, we defined a small family of heuristic noise detection functions $\lambda_n \in \Lambda_n$ that discards training data instances that are not likely to be well formed. For instance,

- λ_n^1 requires u to contain a minimum threshold of information by removing those with μ_z that has token length fewer than 3. Utterances shorter than this mostly consist of non-actionable commands.
- λ_n^2 discards all data samples below a certain threshold of occurrences in live traffic, since utterances that are rarely observed are more likely to be ASR errors or unnatural.
- λ_n^3 discards the data samples for a domain if they come from an overly broad pattern with a catch-all behavior.
- λ_n^4 discards utterances that belong to shared intents provided by the SLU SDK.

The end result of this stage is to retain utterances such as 'call me a cab' from 'Ask Uber to call me a cab' but discard 'Boston' from 'Ask Accuweather for Boston'. One can easily imagine extending this framework with other high recall noise detectors, for example, using language models to discard candidates that are unlikely to be spoken sentences.

4 Model Architecture

Our model consists of a *shared encoder* network consisting of an orthography-sensitive hierarchical LSTM encoder that feeds into a set of domain specific classification layers trained to make a binary decision for each output label.

Our main novel contribution is the extension of this architecture with a *personalized attention* mechanism which uses the attention mechanism of (Bahdanau et al., 2014) to attend to memory locations corresponding to the specific domains enabled by a user, and allows the system to learn semantic representations of each domain via *domain embeddings*. As we will show, incorporating personalization features is key to disambiguating between multiple overlapping domains[1], and the personalized attention mechanism outperforms more naive forms of personalization. The personalized attention mechanism first computes an attention weight for each of enabled domains, performs a convex combination to compute a context vector and then concatenates this vector to the encoded utterance before the final domain classification. Figure 1 depicts the model in detail.

Our model can efficiently accommodate new domains not seen during initial training by keeping the shared encoder frozen, bootstrapping a domain embedding based on existing ones, then optimizing a small number of network parameters corresponding to domain-specific classifier, which is orders of magnitude faster and more data efficient than retraining the full classifier.

We make design decisions to ensure that our model has a low memory and latency footprint. We avoid expensive large vocabulary matrix multiplications on both the input and output stages, and instead use a combination of character embeddings and word embeddings in the input stage.[2] The output matrix is lightweight because each domain-specific classifier is a matrix of only 201×2 parameters. The inference task can be trivially parallelized across cores since there's no requirement to compute a partition function across a high-dimensional softmax layer, which is the slowest component of large label multiclass neural networks. Instead, we achieve comparability between the probability scores generated by individual models by using a customized loss formulation.[3]

4.1 Shared Encoder

First we describe our shared hierarchical utterance encoder, which is marked by the almond colored box in Figure 1. Our hierarchical character to word to utterance design is motivated by the need to make the model operate on an open vocabulary in terms of words and to make it robust to small changes in orthography resulting from fluctuations in the upstream ASR system, all while avoiding expensive large matrix multiplications associated with one-hot word encoding in large vocabulary systems.

We denote an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes a d dimensional input vector x and a d' dimensional state vector h to output a new d' dimensional state vector $h' =$

[1]We assume that users can customize their IPDA settings to enable certain domains.

[2]Using a one-hot representation of word vocabulary size 60,000 and hidden dimension 100 would require learning a matrix of size 60000 x 100 - using 100-dim word embeddings requires only a $\mathcal{O}(1)$ lookup followed by a 100 x 100 matrix, thus allowing our model to be significantly smaller and faster despite having what is effectively an open vocabulary

[3]Current inference consumes 50MB memory and the p99 latency is 15ms.

$\phi(x, h)$. Let $\mathcal{C}$ denote the set of characters and $\mathcal{W}$ the set of words in a given utterance. Let $\oplus$ denote the vector concatenation operation. We encode an utterance using BiLSTMs, and the model parameters Θ associated with this BiLSTM layer are

- Char embeddings $e_c \in \mathbb{R}^{25}$ for each $c \in \mathcal{C}$
- Char LSTMs $\phi_f^{\mathcal{C}}, \phi_b^{\mathcal{C}} : \mathbb{R}^{25} \times \mathbb{R}^{25} \rightarrow \mathbb{R}^{25}$
- Word embeddings $e_w \in \mathbb{R}^{50}$ for each $w \in \mathcal{W}$
- Word LSTMs $\phi_f^{\mathcal{W}}, \phi_b^{\mathcal{W}} : \mathbb{R}^{100} \times \mathbb{R}^{50} \rightarrow \mathbb{R}^{50}$

Let $w_1 \ldots w_n \in \mathcal{W}$ denote a word sequence where word w_i has character $w_i(j) \in \mathcal{C}$ at position j. First, the model computes a character-sensitive word representation $v_i \in \mathbb{R}^{100}$ as

$$\begin{aligned} f_j^{\mathcal{C}} &= \phi_f^{\mathcal{C}}\left(e_{w_i(j)}, f_{j-1}^{\mathcal{C}}\right) \qquad \forall j = 1 \ldots |w_i| \\ b_j^{\mathcal{C}} &= \phi_b^{\mathcal{C}}\left(e_{w_i(j)}, b_{j+1}^{\mathcal{C}}\right) \qquad \forall j = |w_i| \ldots 1 \\ v_i &= f_{|w_i|}^{\mathcal{C}} \oplus b_1^{\mathcal{C}} \oplus e_{w_i} \end{aligned}$$

for each $i = 1 \ldots n$.[4] These word representation vectors are encoded by forward and backward LSTMs for word $\phi_f^{\mathcal{W}}, \phi_b^{\mathcal{W}}$ as

$$\begin{aligned} f_i^{\mathcal{W}} &= \phi_f^{\mathcal{W}}\left(v_i, f_{i-1}^{\mathcal{W}}\right) \qquad \forall i = 1 \ldots n \\ b_i^{\mathcal{W}} &= \phi_b^{\mathcal{W}}\left(v_i, b_{i+1}^{\mathcal{W}}\right) \qquad \forall i = n \ldots 1 \end{aligned}$$

and induces a character and context-sensitive word representation $h_i \in \mathbb{R}^{100}$ as

$$h_i = f_i^{\mathcal{W}} \oplus b_i^{\mathcal{W}}$$

for each $i = 1 \ldots n$. For convenience, we write the entire operation as a mapping BiLSTM$_\Theta$:

$$(h_1 \ldots h_n) \leftarrow \text{BiLSTM}_\Theta(w_1 \ldots w_n)$$

$$\bar{h} = \sum_{i=1}^{n} h_i \tag{1}$$

4.2 Domain Classification

Our Multitask domain classification formulation is motivated by a desire to avoid computing the full partition function during test time, which tends to be the slowest component of a multiclass neural network classifer, as has been documented before by (Jozefowicz et al., 2016) and (Mikolov et al., 2013), amongst others.

[4]For simplicity, we assume some random initial state vectors such as $f_0^{\mathcal{C}}$ and $b_{|w_i|+1}^{\mathcal{C}}$ when we describe LSTMs.

However, we also want access to reliable probability estimates instead of raw scores - we accomplish this by constructing a custom loss function. During training, each domain classifier receives in-domain (IND) and out-of-domain (OOD) utterances, and we adapt the one-sided selection mechanism of (Kubat et al., 1997) to prevent OOD utterances from overpowering IND utterances, thus an utterance in a domain $d \in \mathcal{D}$ is considered as an IND utterance in the viewpoint of domain d and OOD for all other domains.

We first use the shared encoder to compute the utterance representation $\bar{h}$ as previously described. Then we define the probability of domain $\tilde{d}$ for the utterance by mapping $\bar{h}$ to a 2-dimensional output vector with a linear transformation for each domain $\tilde{d}$ as

$$z^{\tilde{d}} = \sigma(W^{\tilde{d}} \cdot \bar{h} + b^{\tilde{d}})$$

$$p(\tilde{d}|\bar{h}) \propto \begin{cases} \exp\left([z^{\tilde{d}}]_{IND}\right), & \text{if } \tilde{d} = d \\ \exp\left([z^{\tilde{d}}]_{OOD}\right), & \text{otherwise} \end{cases}$$

where σ is scaled exponential linear unit (SeLU) for normalized activation outputs (Klambauer et al., 2017) and $[z^{\tilde{d}}]_{IND}$ and $[z^{\tilde{d}}]_{OOD}$ denote the values in the IND and OOD position of vector $z^{\tilde{d}}$.

We define the joint domain classification loss $\mathcal{L}^{\mathcal{D}}$ as the summation of positive ($\mathcal{L}^P$) and negative ($\mathcal{L}^N$) class loss functions [5]:

$$\begin{aligned} \mathcal{L}^P\left(\Theta, \Theta^{\tilde{d}}\right) &= -\log p\left(\tilde{d}|\bar{h}\right) \\ \mathcal{L}^N\left(\Theta, \Theta^{\tilde{d}}\right) &= -\frac{1}{k-1}\left(\sum_{\bar{d} \in \mathcal{D}, \bar{d} \neq \tilde{d}} \log p\left(\bar{d}|\bar{h}\right)\right) \\ \mathcal{L}^{\mathcal{D}}\left(\Theta, \Theta^{\tilde{d}}\right) &= L^P\left(\Theta, \Theta^{\tilde{d}}\right) + L^N\left(\Theta, \Theta^{\tilde{d}}\right) \end{aligned}$$

where k is the total number of domains. We divide the second term by $k-1$ so that $\mathcal{L}^P$ and $\mathcal{L}^N$ are balanced in terms of the ratio of the training examples for a domain to those for other domains. While a softmax over the entire domains tends to highlight only the ground-truth domain while suppressing all the rest, the our joint domain classification with a softmax over two classes is designed to produce a more balanced confidence score per domain independent of other domains.

[5]$\Theta^{\tilde{d}}$ denotes the additional parameters in the classification layer for domain $\tilde{d}$.

4.3 Personalized Attention

We explore encoding a user's domain preferences in two ways. Our baseline method is a *1-bit flag* that is appended to the input features of each domain-specific classifier. Our novel *personalized attention* method induces domain embeddings by supervising an attention mechanism to attend to a user's enabled domains with different weights depending on their relevance. The domain embedding matrix in Figure 1 represents the embeddings of a user's enabled domains. We hypothesize that attention enables the network learn richer representations of user preferences and domain co-occurrence features.

Let $e_{\mathcal{D}}(\tilde{d}) \in R^{100}$ and $\bar{h} \in R^{100}$ denote the domain embeddings for domain $\tilde{d}$ and the utterance representation calculated by Eq. (1), respectively. The domain attention weights for a given user u who has a preferred domain list $d^{(u)} = \left(\tilde{d}_1^{(u)}, \ldots, \tilde{d}_k^{(u)}\right)$ are calculated by the dot-product operation,

$$a_i = \bar{h} \cdot e_{\mathcal{D}}\left(\tilde{d}_i^{(u)}\right) \qquad \forall i = 1 \ldots k$$

The final, normalized attention weights $\bar{a}$ are obtained after normalization via a softmax layer,

$$\bar{a}_i = \frac{\exp(a_i)}{\sum_{j=1}^{k} \exp(a_j)} \qquad \forall i = 1 \ldots k$$

The weighted combination of domain embeddings is

$$\bar{S}^{attended} = \sum_{i=1}^{k} \left(\bar{a}_i \cdot e_{\mathcal{D}}\left(\tilde{d}_i^{(u)}\right)\right)$$

Finally the two representations of enabled domains, namely the *attention* model and *1-bit flag* are then concatenated with the utterance representation and used to make per-domain predictions via domain-specific affine transformations:

$$\bar{z}_{att} = \bar{h} \oplus \bar{S}^{attended}$$
$$\bar{z}_{1bit} = \bar{h} \oplus \mathbb{I}(\tilde{d} \in enabled)$$

Here $\mathbb{I}(\bar{d} \in enabled)$ is a 1-bit indicator for whether the domain is enabled by the user or not. $\bar{z}_{att}$ and $\bar{z}_{1bit}$ represent the encoded hidden state of the `Attention` and `1-Bit Flag` configurations of the model from the experiment section. In our experiments we will compare these two ways of encoding personalization information, as well as evaluate a variant that combines the two. In this way we can ascertain whether the two personalization signals are complementary via an ablation study on the full model.

4.4 Domain Bootstrapping

Our model separates the responsibilities for utterance representation and domain classification between the shared encoder and the domain-specific classifiers. That is, the shared encoder needs to be retrained only if it cannot encode an utterance well (e.g., a new domain introduces completely new words) and the existing domain classifiers need to be retrained only when the shared encoder changes. For adding new domains efficiently without full retraining, the only two components in the architecture need to be updated for each new domain $\tilde{d}_{new}$, are the domain embeddings for the new domain and its domain-specific classifier.[6] We keep the weights of the encoder network frozen and use the hidden state vector $\bar{h}$, calculated by Eq. 1, as a feature vector to feed into the downstream classifiers. To initialize the m-dimensional domain embeddings $e_{\tilde{d}_{new}}$, we use the column-wise average of all utterance vectors in the training data $\bar{h}^{avg}$, and project it to the domain embedding space using a matrix $U \in R^{m \times m}$. Thus,

$$e_{\tilde{d}_{new}} = U^{*} \cdot \bar{h}^{avg}$$

The parameters of U^{*} are learned using the column-wise average utterance vectors $\bar{h}_j^{avg}$ and learned domain vectors for all existing domains d_j

$$U^{*} = \arg\min_{U} ||U \cdot \bar{h}_j^{avg} - e_{d_j}|| \quad \forall d_j \in \mathcal{D}$$

This is a write-to-memory operation that creates a new domain representation after attending to all existing domain representations. We then train the parameters of the domain-specific classifier with the new domain's data while keeping the encoder fixed. This mechanism allows us to efficiently support new domains that appear in-between full model deployment cycles without compromising performance on existing domains. A full model refresh would require us to fully retrain with the domains that have appeared in the intermediate period.

[6]We have assumed that the shared encoder covers most of the vocabulary of new domains; otherwise, the entire network may need to be retrained. Based on our observation of live usage data, this assumption is reasonable since the shared encoder after initial training is still shown to cover 95% of the vocabulary of new domains added in the subsequent week.

	WEAK			Mturk		
	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
Binary	78.29	87.90	88.92	73.79	85.35	86.45
MultiClass	78.58	87.12	88.11	73.78	84.54	85.55
MultiTask	80.46	89.27	90.16	75.66	86.48	87.66
1-Bit Flag	91.97	95.89	96.68	86.50	92.47	93.09
Attention*	94.83	97.11	98.35	89.64	95.39	96.70
1-Bit + Att	**95.19**	**97.32**	**98.64**	**89.65**	**95.79**	**96.98**

Table 1: The performance of different variants of our neural model in terms of top-N accuracy. `Binary` trains a separate binary classifier for each skill. `MultiClass` has a shared encoder followed by a softmax. `MultiTask` replaces the softmax with per-skill classifiers. `1-Bit Flag` adds a single bit for personalization to each skill classifier in MultiTask. `Attention` extends MultiTask with personalized attention. The last 3 models are personalized. **Best single encoding.*

5 Experiments

In this section we aim to demonstrate the effectiveness of our model architecture in two settings. First, we will demonstrate that attention based personalization significantly outperforms the baseline approach. Secondly, we will show that our model new domain bootstrapping procedure results in accuracies comparable to full retraining while requiring less than 1% of the orignal training time.

5.1 Experimental Setup

Weak: This is a weakly supervised dataset was generated by preprocessing utterances with strict invocation patterns according to the setup mentioned in Section 3. The dataset consists of 5.34M utterances from 637,975 users across 1,500 different skills. Since we are interested in capturing the temporal effects of the dataset as well as personalization effects, we partitioned the data based both on user and time. Our core training data for the experiments in this paper was drawn from one month of live usage, the validation data came from the next 15 days of usage, and the test data came from the subsequent 15 days. The training, validation and test sets are user-independent, and each user belongs to only one of the three sets to ensure no leakage of information.

MTurk: Since the Weak dataset is generated by weak supervision, we verified the performance of our approach with human generated utterances. A random sample of 12,428 utterances from the test partition of users were presented to 300 human judges, who were asked to produce two natural ways to issue the same command. This dataset is treated as a representative clean held out test set on which we can observe the generalization of our weakly supervised training and validation data to natural language.

New Skills: In order to simulate the scenario in which new skills appear within a week between model updates, we selected 250 new skills which do not overlap with the skills in the Weak dataset. The vocabulary size of 1,500 skills is 200K words, and on average, 5% of the vocabulary for new skills is not covered. We randomly sampled 4,000 unique utterances for each skill using the same weak supervision method, and split them into 3,000 utterances for training and 1,000 for testing.

5.2 Results and Discussion

Generalization from Weakly Supervised to Natural Utterances We first show the progression of model performance as we add more components to show their individual contribution. Secondly, we show that training our models on a weakly supervised dataset can generalize to natural speech by showing their test performance on the human-annotated test data. Finally, we compare two personalization strategies.

The full results are summarized in Table 1, which shows the top-N test results separately for the Weak dataset (weakly supervised) and MTurk dataset (human-annotated). We report top-N accuracy to show the potential for further re-ranking or disambiguation downstream. For top-1 results on the Weak dataset, using a separate binary classifier for each domain (Binary) shows a prediction accuracy at 78.29% and using a softmax layer on top of the shared encoder (MultiClass) shows a comparable accuracy at 78.58%. The performance shows a slight improvement when using the Multitask neural loss structure, but adding personalization signals to the Multitask structure showed a significant boost in performance. We noted the large difference between the 1-bit and attention architecture. At 94.83% accuracy, attention resulted in 35.6% relative error reduction over the 1-bit baseline 91.97% on the Weak validation set and 23.25% relative on the MTurk test set. We hypothesize that this may be because the attention mechanism allows the model to focus on complementary features in case of overlapping domains as well as learning domain co-occurrence statistics, both of which are not possible with the simple 1-bit flag.

When both personalization representations were combined, the performance peaked at 95.19% for the Weak dataset and a more modest

	Time	Accuracy
Binary	34.81	78.13
Expand	30.34	94.03
Refresh	5300.18	94.58

Table 2: Comparison of per-epoch training time (seconds) and top-1 accuracy (%) on an NVIDIA Tesla M40 GPU.

89.65% for the MTurk dataset. The improvement trend is extremely consistent across all top-N results for both of the Weak and MTurk datasets across all experiments. The disambiguation task is complex due to similar and overlapping skills, but the results suggest that incorporating personalization signals equip the models with much better discriminative power. The results also suggest that the two mechanisms for encoding personalization provide a small amount of complementary information since combining them together is better than using them individually. Although the performance on the Weak dataset tends to be more optimistic, the best performance on the human-annotated test data is still close to 90% for top-1 accuracy, which suggests that training our model with the samples derived from the invocation patterns can generalize well to natural utterances.

Rapid Bootstrapping of New Skills We show the results when new domains are added to an IPDA and the model needs to efficiently accommodate them with a limited number of data samples. We show the classification performance on the skills in the New Skills dataset while assuming we have access to the WEAK dataset to pre-train our model for transfer learning. In the `Binary` setting, a domain-specific binary classifier is trained for each domain. `Expand` describes the case in which we incrementally train on top of an existing model. `Refresh` is the setting in which the model is fully retrained from scratch with the new data - which would be ideal in case there were no time constraints.

We record the average training time for each epoch and the performance is measured with top-1 classification accuracy over new skills. The experiment results can be found in Table 2. Adapting a new skill is two orders of magnitude faster (30.34 seconds) than retraining the model (5300.18 seconds) while achieving 94.03% accuracy which is comparable to 94.58% accuracy of full retraining. The first two techniques can also be easily parallelized unlike the `Refresh` configuration.

	Top-1	Top-3	Top-5
Full	6.17	14.30	20.41
Enabled	85.62	96.15	98.06

Table 3: Top-N prediction accuracy (%) on the full skill set (Full) and only enabled skills (Enabled).

Behavior of Attention Mechanism Our expectation is that the model is able to learn to attend the relevant skills during the inference process. To study the behavior of the attention layer, we compute the top-N prediction accuracy based on the most relevant skills defined by the attention weights. In this experiment, we considered the subset of users who had enabled more than 20 domains to exclude trivial cases[7]. The results are shown in Table 3. When the model attends to the entire set of 1500 (*Full*), the top-5 prediction accuracy is 20.41%, which indicates that a large number of skills can process the utterance, and thus it is highly likely to miss the correct one in the top-5 predictions. This ambiguity issue can be significantly improved by users' enabled domain lists as proved by the accuracies (*Enabled*): 85.62% for top-1, 96.15% for top-3, and 98.06% for top-5.[8] Thus the attention mechanism can thus be viewed as an initial soft selection which is then followed by a fine-grained selection at the classification stage.

End-to-End User Evaluation All intermediate metrics on this task are proxies to a human customer's eventual evaluation. In order to assess the user experience, we need to measure its end-to-end performance. For a brief end-to-end system evaluation, 983 utterances from 283 domains were randomly sampled from the test set in the large-scale IPDA setting. 15 human judges (male=12, female=3) rated the system responses, 1 judge per utterance, on a 5-point Likert scale with 1 being *Terrible* and 5 being *Perfect*. The judgment score of 3 or above was taken as *SUCCESS* and 2 or below as *DEFECT*. The end-to-end *SUCCESS* rate,

[7]Thus, the random prediction accuracy on enabled domains is less than 5% and across the Full domain list is 0.066%

[8]Visual inspection of the embeddings confirms that meaningful clusters are learned. We see clusters related to home automation, commerce, cooking, trivia etc, we show some examples in Figure 2, 3 and 4. However there are still other clusters where the the relationships cannot be established as easily. An example of these is show in Figure 5. The personalized attention mechanism is learned using the semantic content as well as personalization signals, so we hypothesize clusters like this may be capturing user tendencies to enable these domains in a correlated manner.

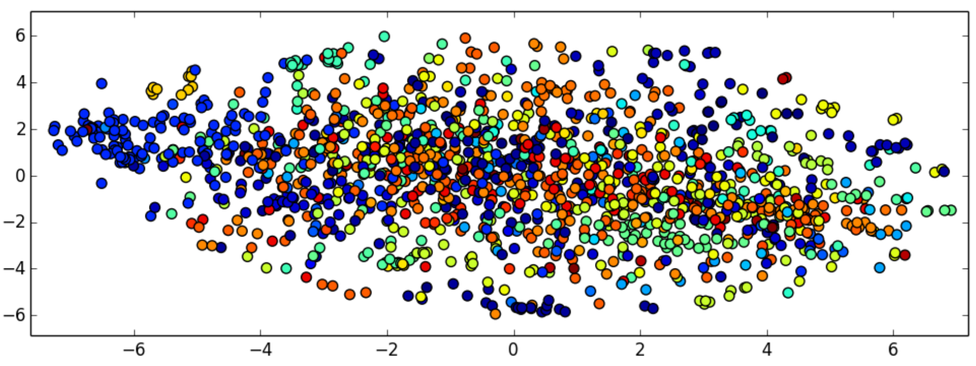


Figure 2: Embeddings of different domain categories visualized in 2D using TSNE (van der Maaten and Hinton, 2008). Different colors represent different categories, for e.g. the large blue cluster on the left is Home Automation.

thus user satisfaction, was shown to be 95.52%. The discrepancy between this score and the score produced on MTurk dataset indicates that even in cases in which the model makes classification mistakes, some of these interpretations remain perceptually meaningful to humans.

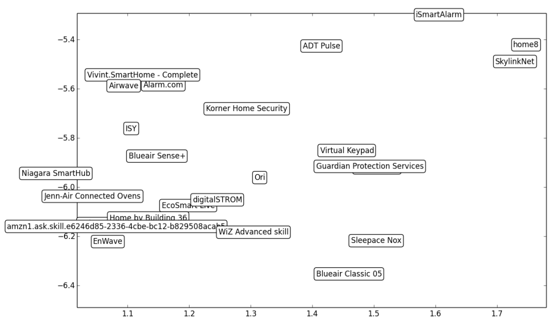


Figure 3: A large cluster of home automation domains.

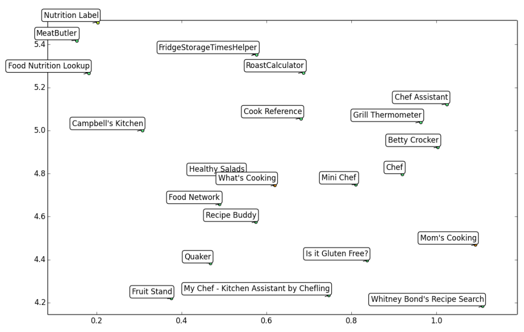


Figure 4: A cluster of domains related to cooking.

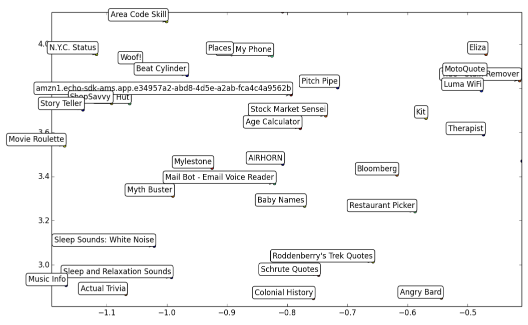


Figure 5: A mixed cluster with several different domain categories represented.

6 Conclusions

We have described a neural model architecture to address large-scale skill classification in an IPDA used by tens of millions of users every day. We have described how personalization features and an attention mechanism can be used for handling ambiguity between domains. We have also shown that the model can be extended efficiently and incrementally for new domains, saving multiple orders of magnitude in terms of training time. The model also addresses practical constraints of having a low memory footprint, low latency and being easily parallelized, all of which are important characteristics for real-time production systems. In future work, we plan to incorporate various types of context (e.g. anaphora, device-specific capabilities) and dialogue history into a large-scale NLU system.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero shot frame semantic parsing for domain scaling. In Interspeech 2017.

A. Bhargava, Asli Celikyilmaz, Dilek Z. Hakkani-Tur, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. IEEE International Conference on Acoustics, Speech and Signal Processing, pages 8337–8341.

Asli Celikyilmaz, Ruhi Sarikaya, Dilek Hakkani-Tür, Xiaohu Liu, Nikhil Ramesh, and Gökhan Tür. 2016. A new pre-training method for training deep learning models with application to spoken language understanding. In Interspeech, pages 3255–3259.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016a. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, pages 6045–6049.

Yun-Nung Chen, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016b. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In Interspeech.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4067–4071. IEEE.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. CoRR, abs/1706.04326.

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

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 541–550. Association for Computational Linguistics.

Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R Hershey. 2016. Context-sensitive and role-dependent spoken language understanding using bidirectional and attention lstms. Interspeech, pages 3236–3240.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. In Interspeech.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.

Mark D Kernighan, Kenneth W Church, and William A Gale. 1990. A spelling correction program based on a noisy channel model. In Proceedings of the 13th conference on Computational linguistics-Volume 2, pages 205–210. Association for Computational Linguistics.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 84–92.

Young-Bum Kim, Sungjin Lee, and Ruhi Sarikaya. 2017a. Speaker-sensitive dual memory networks for multi-turn slot tagging. In Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE, pages 547–553. IEEE.

Young-Bum Kim, Sungjin Lee, and Karl Stratos. 2017b. Onenet: Joint domain, intent, slot prediction for spoken language understanding. In Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE, pages 547–553. IEEE.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017c. Adversarial adaptation of synthetic or stale data. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1297–1307. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017d. Domain attention with an ensemble of experts. In Annual Meeting of the Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015b. Pre-training of hidden-unit crfs. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, volume 2, pages 192–198.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 387–396.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2017e. A framework for pre-training hidden-unit conditional random fields and its extension to long short term memory networks. Computer Speech & Language, 46:311–326.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015c. New transfer learning techniques for disparate label sets. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, volume 1, pages 473–482.

Gunter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. CoRR, abs/1706.02515.

Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In ICML, volume 97, pages 179–186. Nashville, USA.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, and Markus Dreyer. 2017a. Just ask: Building an architecture for extensible self-service spoken language understanding. arXiv preprint arXiv:1711.00549.

Anjishnu Kumar, Pavankumar Reddy Muddireddy, Markus Dreyer, and Björn Hoffmeister. 2017b. Zero-shot learning across heterogeneous overlapping domains. Proc. Interspeech 2017, pages 2914–2918.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In Proceedings of NAACL-HLT, pages 260–270.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In Interspeech, pages 685–689.

Bing Liu and Ian Lane. 2017. Multi-domain adversarial learning for slot filling in spoken language understanding. In NIPS Workshop on Conversational AI.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research, 9:2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In Advances in Neural Information Processing Systems, pages 3567–3575.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. IEEE Signal Processing Magazine, 34(1):67–81.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In Spoken Language Technology Workshop (SLT), 2016 IEEE, pages 391–397. IEEE.

Michael L Seltzer and Jasha Droppo. 2013. Multi-task learning in deep neural networks for improved phoneme recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6965–6969. IEEE.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 1, pages I–I. IEEE.

Gokhan Tur and Renato de Mori. 2011. Spoken Language Understanding: Systems for Extracting Semantic Information from Speech. New York, NY: John Wiley and Sons.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. arXiv preprint arXiv:1602.00367.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 136–140. IEEE.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. International Conference on Learning Representation (ICLR).

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In International Joint Conference on Artificial Intelligence (IJCAI), pages 2993–2999.

Multimodal Affective Analysis Using Hierarchical Attention Strategy with Word-Level Alignment

Yue Gu, Kangning Yang*, Shiyu Fu*, Shuhong Chen, Xinyu Li and Ivan Marsic
Multimedia Image Processing Lab
Electrical and Computer Engineering Department
Rutgers University, Piscataway, NJ, USA
{yue.guapp, ky189, sf568, sc1624, Xinyu.li1118, marsic}@rutgers.edu

Abstract

Multimodal affective computing, learning to recognize and interpret human affect and subjective information from multiple data sources, is still challenging because: (i) it is hard to extract informative features to represent human affects from heterogeneous inputs; (ii) current fusion strategies only fuse different modalities at abstract levels, ignoring time-dependent interactions between modalities. Addressing such issues, we introduce a hierarchical multimodal architecture with attention and word-level fusion to classify utterance-level sentiment and emotion from text and audio data. Our introduced model outperforms state-of-the-art approaches on published datasets, and we demonstrate that our model's synchronized attention over modalities offers visual interpretability.

1 Introduction

With the recent rapid advancements in social media technology, affective computing is now a popular task in human-computer interaction. Sentiment analysis and emotion recognition, both of which require applying subjective human concepts for detection, can be treated as two affective computing subtasks on different levels (Poria et al., 2017a). A variety of data sources, including voice, facial expression, gesture, and linguistic content have been employed in sentiment analysis and emotion recognition. In this paper, we focus on a multimodal structure to leverage the advantages of each data source. Specifically, given an utterance, we consider the linguistic content and acoustic characteristics together to recognize the opinion or emotion. Our work is important and useful because speech is the most basic and commonly used form of human expression.

A basic challenge in sentiment analysis and emotion recognition is filling the gap between extracted features and the actual affective states (Zhang et al., 2017). The lack of high-level feature associations is a limitation of traditional approaches using low-level handcrafted features as representations (Seppi et al., 2008; Rozgic et al., 2012). Recently, deep learning structures such as CNNs and LSTMs have been used to extract high-level features from text and audio (Eyben et al., 2010a; Poria et al., 2015). However, not all parts of the text and vocal signals contribute equally to the predictions. A specific word may change the entire sentimental state of text; a different vocal delivery may indicate inverse emotions despite having the same linguistic content. Recent approaches introduce attention mechanisms to focus the models on informative words (Yang et al., 2016) and attentive audio frames (Mirsamadi et al., 2017) for each individual modality. However, to our knowledge, there is no common multimodal structure with attention for utterance-level sentiment and emotion classification. To address such issue, we design a deep hierarchical multimodal architecture with an attention mechanism to classify utterance-level sentiments and emotions. It extracts high-level informative textual and acoustic features through individual bidirectional gated recurrent units (GRU) and uses a multi-level attention mechanism to select the informative features in both the text and audio module.

Another challenge is the fusion of cues from heterogeneous data. Most previous works focused on combining multimodal information at a holistic level, such as integrating independent predictions of each modality via algebraic rules (Wöllmer et al., 2013) or fusing the extracted modality-specific features from entire utterances

* Equally Contribution

(Poria et al., 2016). They extract word-level features in a text branch, but process audio at the frame-level or utterance-level. These methods fail to properly learn the time-dependent interactions across modalities and restrict feature integration at timestamps due to the different time scales and formats of features of diverse modalities (Poria et al., 2017a). However, to determine human meaning, it is critical to consider both the linguistic content of the word and how it is uttered. A loud pitch on different words may convey inverse emotions, such as the emphasis on "hell" for anger but indicating happy on "great". Synchronized attentive information across text and audio would then intuitively help recognize the sentiments and emotions. Therefore, we compute a forced alignment between text and audio for each word and propose three fusion approaches (horizontal, vertical, and fine-tuning attention fusion) to integrate both the feature representations and attention at the word-level.

We evaluated our model on four published sentiment and emotion datasets. Experimental results show that the proposed architecture outperforms state-of-the-art approaches. Our methods also allow for attention visualization, which can be used for interpreting the internal attention distribution for both single- and multi-modal systems. The contributions of this paper are: (i) a hierarchical multimodal structure with attention mechanism to learn informative features and high-level associations from both text and audio; (ii) three word-level fusion strategies to combine features and learn correlations in a common time scale across different modalities; (iii) word-level attention visualization to help human interpretation.

The paper is organized as follows: We list related work in section 2. Section 3 describes the proposed structure in detail. We present the experiments in section 4 and provide the result analysis in section 5. We discuss the limitations in section 6 and conclude with section 7.

2 Related Work

Despite the large body of research on audio-visual affective analysis, there is relatively little work on combining text data. Early work combined human transcribed lexical features and low-level hand-crafted acoustic features using feature-level fusion (Forbes-Riley and Litman, 2004; Litman and Forbes-Riley, 2004). Others used SVMs fed bag of words (BoW) and part of speech (POS) features in addition to low-level acoustic features (Seppi et al., 2008; Rozgic et al., 2012; Savran et al., 2012; Rosas et al., 2013; Jin et al., 2015). All of the above extracted low-level features from each modality separately. More recently, deep learning was used to extract higher-level multimodal features. Bidirectional LSTMs were used to learn long-range dependencies from low-level acoustic descriptors and derivations (LLDs) and visual features (Eyben et al., 2010a; Wöllmer et al., 2013). CNNs can extract both textual (Poria et al., 2015) and visual features (Poria et al., 2016) for multiple kernel learning of feature-fusion. Later, hierarchical LSTMs were used (Poria et al., 2017b). A deep neural network was used for feature-level fusion in (Gu et al., 2018) and (Zadeh et al., 2017) introduced a tensor fusion network to further improve the performance. A very recent work using word-level fusion was provided by (Chen et al., 2017). The key differences between this work and the proposed architecture are: (i) we design a fine-tunable hierarchical attention structure to extract word-level features for each individual modality, rather than simply using the initialized textual embedding and extracted LLDs from COVAREP (Degottex et al., 2014); (ii) we propose diverse representation fusion strategies to combine both the word-level representations and attention weights, instead of using only word-level fusion; (iii) our model allows visualizing the attention distribution at both the individual modality and at fusion to help model interpretability.

Our architecture is inspired by the document classification hierarchical attention structure that works at both the sentence and word level (Yang et al., 2016). For audio, an attention-based BLSTM and CNN were applied to discovering emotion from frames (Huang and Narayanan, 2016; Neumann and Vu, 2017). Frame-level weighted-pooling with local attention was shown to outperform frame-wise, final-frame, and frame-level mean-pooling for speech emotion recognition (Mirsamadi et al., 2017).

3 Method

We introduce a multimodal hierarchical attention structure with word-level alignment for sentiment analysis and emotion recognition (Figure 1). The model consists of three major parts: text attention module, audio attention module, and word-level fusion module. We first make a forced alignment between the text and audio during preprocessing. Then, the text attention module and audio attention module extract the features from the corresponding inputs (shown in Algorithm 1). The word-level fusion module fuses the extracted feature vectors and makes the final prediction via a shared representation (shown in Algorithm 2).

3.1 Forced Alignment and Preprocessing

The forced alignment between the audio and text on the word-level prepares the different data for feature extraction. We align the data at the word-level because words are the basic unit in English for human speech comprehension. We used *aeneas*[1] to determine the time interval for each word in the audio file based on the Sakoe-Chiba Band Dynamic Time Warping (DTW) algorithm (Sakoe and Chiba, 1978).

For the text input, we first embedded the words into 300-dimensional vectors by *word2vec* (Mikolov et al., 2013), which gives us the best result compared to GloVe and LexVec. Unknown words were randomly initialized. Given a sentence S with N words, let w_i represent the ith word. We embed the words through the *word2vec* embedding matrix W_e by:

$$T_i = W_e w_i, i \in [1, N] \tag{1}$$

where T_i is the embedded word vector.

For the audio input, we extracted Mel-frequency spectral coefficients (MFSCs) from raw audio signals as acoustic inputs for two reasons. Firstly, MFSCs maintain the locality of the data by preventing new bases of spectral energies resulting from discrete cosine transform in MFCCs extraction (Abdel-Hamid et al., 2014). Secondly, it has more dimensions in the frequency domain that aid learning in deep models (Gu et al., 2017). We used 64 filter banks to extract the MFSCs for each audio frame to form the MFSCs map. To facilitate training, we only used static coefficients. Each word's MFSCs can be represented as a matrix with $64 \times n$ dimensions, where n is the interval for the given word in frames. We zero-pad all intervals to the same length L, the maximum frame numbers of the word in the dataset. We did extract LLD features using OpenSmile (Eyben et al., 2010b) software and combined them with the MFSCs during our training stage. However, we did not find an obvious performance improvement, especially for the sentiment analysis. Considering the training cost of the proposed hierarchical acoustic architecture, we decided the extra features were not worth the tradeoff. The output is a 3D MFSCs map with dimensions $[N, 64, L]$.

[1]https://www.readbeyond.it/aeneas/

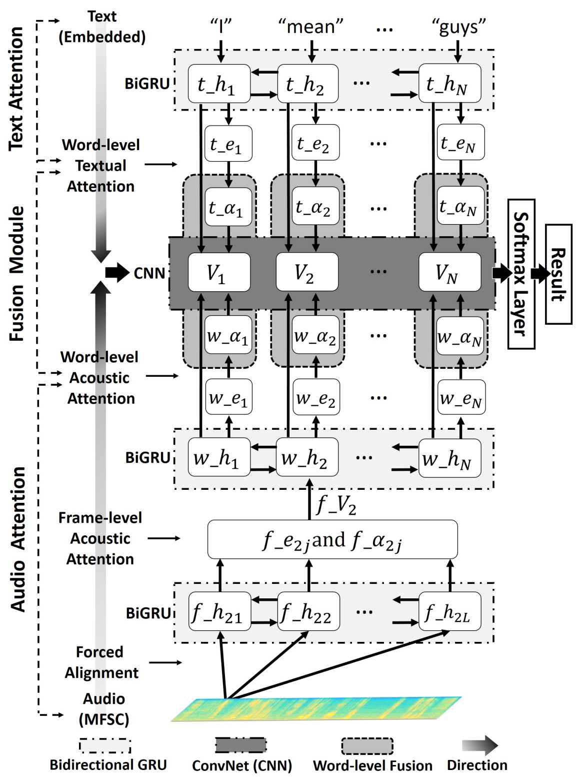


Figure 1: Overall Architecture

3.2 Text Attention Module

To extract features from embedded text input at the word level, we first used bidirectional GRUs, which are able to capture the contextual information between words. It can be represented as:

$$t\_h_i^{\rightarrow}, t\_h_i^{\leftarrow} = bi\_GRU(T_i), i \in [1, N] \tag{2}$$

where bi_GRU is the bidirectional GRU, $t\_h_i^{\rightarrow}$ and $t\_h_i^{\leftarrow}$ denote respectively the forward and backward contextual state of the input text. We combined $t\_h_i^{\rightarrow}$ and $t\_h_i^{\leftarrow}$ as t_h_i to represent the feature vector for the ith word. We choose GRUs instead of LSTMs because our experiments show that LSTMs lead to similar performance (0.07% higher accuracy) with around 25% more trainable parameters.

To create an informative word representation, we adopted a word-level attention strategy that generates a one-dimensional vector denoting the importance for each word in a sequence (Yang et al., 2016). As defined by (Bahdanau et al.,

Algorithm 1 FEATURE EXTRACTION

```
1: procedure FORCED ALIGNMENT
2:     Determine time interval of each word
3:     find $w_i \leftarrow \rightarrow [A_{ij}], j \in [1, L], i \in [1, N]$
4: end procedure
5: procedure TEXT BRANCH
6:     Text Attention Module
7:     for $i \in [1, N]$ do
8:         $T_i \leftarrow getEmbedded(w_i)$
9:         $t\_h_i \leftarrow bi\_GRU(T_i)$
10:        $t\_e_i \leftarrow getEnergies(t\_h_i)$
11:        $t\_\alpha_i \leftarrow getDistribution(t\_e_i)$
12:    end for
13:    return $t\_h_i, t\_\alpha_i$
14: end procedure
15: procedure AUDIO BRANCH
16:    for $i \in [1, N]$ do
17:        Frame-Level Attention Module
18:        for $j \in [1, L]$ do
19:            $f\_h_{ij} \leftarrow bi\_GRU(A_{ij})$
20:            $f\_e_{ij} \leftarrow getEnergies(f\_h_{ij})$
21:            $f\_\alpha_{ij} \leftarrow getDistribution(f\_e_{ij})$
22:        end for
23:        $f\_V_i \leftarrow weightedSum(f\_\alpha_{ij}, f\_h_{ij})$
24:        Word-Level Attention Module
25:        $w\_h_i \leftarrow bi\_GRU(f\_V_i)$
26:        $w\_e_i \leftarrow getEnergies(w\_h_i)$
27:        $w\_\alpha_i \leftarrow getDistribution(w\_e_i)$
28:    end for
29:    return $w\_h_i, w\_\alpha_i$
30: end procedure
```

2014), we compute the textual attentive energies t_e_i and textual attention distribution t_α_i by:

$$t\_e_i = tanh(W_t t\_h_i + b_t), i \in [1, N] \quad (3)$$

$$t\_\alpha_i = \frac{exp(t\_e_i{}^\top v_t)}{\sum_{k=1}^{N} exp(t\_e_k{}^\top v_t)} \quad (4)$$

where W_t and b_t are the trainable parameters and v_t is a randomly-initialized word-level weight vector in the text branch. To learn the word-level interactions across modalities, we directly use the textual attention distribution t_α_i and textual bidirectional contextual state t_h_i as the output to aid word-level fusion, which allows further computations between text and audio branch on both the contextual states and attention distributions.

3.3 Audio Attention Module

We designed a hierarchical attention model with frame-level acoustic attention and word-level attention for acoustic feature extraction.

Frame-level Attention captures the important MFSC frames from the given word to generate the word-level acoustic vector. Similar to the text attention module, we used a bidirectional GRU:

$$f\_h_{ij}^{\rightarrow}, f\_h_{ij}^{\leftarrow} = bi\_GRU(A_{ij}), j \in [1, L] \quad (5)$$

where $f\_h_{ij}^{\rightarrow}$ and $f\_h_{ij}^{\leftarrow}$ denote the forward and backward contextual states of acoustic frames. A_{ij} denotes the MFSCs of the jth frame from the ith word, $i \in [1, N]$. f_h_{ij} represents the hidden state of the jth frame of the ith word, which consists of $f\_h_{ij}^{\rightarrow}$ and $f\_h_{ij}^{\leftarrow}$. We apply the same attention mechanism used for textual attention module to extract the informative frames using equation 3 and 4. As shown in Figure 1, the input of equation 3 is f_h_{ij} and the output is the frame-level acoustic attentive energies f_e_{ij}. We calculate the frame-level attention distribution f_α_{ij} by using f_e_{ij} as the input for equation 4. We form the word-level acoustic vector f_V_i by taking a weighted sum of bidirectional contextual state f_h_{ij} of the frame and the corresponding frame-level attention distribution f_α_{ij} Specifically,

$$f\_V_i = \sum\nolimits_j f\_\alpha_{ij} f\_h_{ij} \quad (6)$$

Word-level Attention aims to capture the word-level acoustic attention distribution w_α_i based on formed word vector f_V_i. We first used equation 2 to generate the word-level acoustic contextual states w_h_i, where the input is f_V_i and $w\_h_i = (w\_h_i^{\rightarrow}, w\_h_i^{\leftarrow})$. Then, we compute the word-level acoustic attentive energies w_e_i via equation 3 as the input for equation 4. The final output is an acoustic attention distribution w_α_i from equation 4 and acoustic bidirectional contextual state w_h_i.

3.4 Word-level Fusion Module

Fusion is critical to leveraging multimodal features for decision-making. Simple feature concatenation without considering the time scales ignores the associations across modalities. We introduce word-level fusion capable of associating the text and audio at each word. We propose three fusion strategies (Figure 2 and Algorithm 2): horizontal fusion, vertical fusion, and fine-tuning attention fusion. These methods allow easy synchronization between modalities, taking advantage of the attentive associations across text and audio, creating a shared high-level representation.

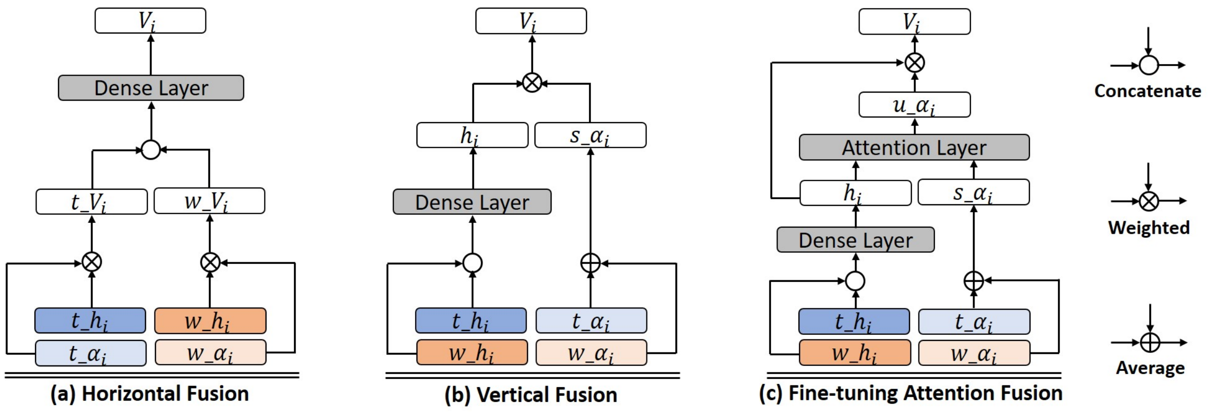


Figure 2: Fusion strategies. t_h_i: word-level textual bidirectional state. t_α_i: word-level textual attention distribution. w_h_i: word-level acoustic bidirectional state. w_α_i: word-level acoustic attention distribution. s_α_i: shared attention distribution. u_α_i: fine-tuning attention distribution. V_i: shared word-level representation.

Algorithm 2 FUSION

```
 1: procedure FUSION BRANCH
 2:     Horizontal Fusion (HF)
 3:     for $i \in [1, N]$ do
 4:         $t\_V_i \leftarrow weighted(t\_\alpha_i, t\_h_i)$
 5:         $w\_V_i \leftarrow weighted(w\_\alpha_i, w\_h_i)$
 6:         $V_i \leftarrow dense([t\_V_i, w\_V_i])$
 7:     end for
 8:     Vertical Fusion (VF)
 9:     for $i \in [1, N]$ do
10:         $h_i \leftarrow dense([t\_h_i, w\_h_i])$
11:         $s\_\alpha_i \leftarrow average([t\_\alpha_i, w\_\alpha_i])$
12:         $V_i \leftarrow weighted(h_i, s\_\alpha_i)$
13:     end for
14:     Fine-tuning Attention Fusion (FAF)
15:     for $i \in [1, N]$ do
16:         $u\_e_i \leftarrow getEnergies(h_i)$
17:         $u\_\alpha_i \leftarrow getDistribution(u\_e_i, s\_\alpha_i)$
18:         $V_i \leftarrow weighted(h_i, u\_\alpha_i)$
19:     end for
20:     Decision Making
21:     $E \leftarrow convNet(V_1, V_2, ..., V_N)$
22:     return E
23: end procedure
```

Horizontal Fusion (HF) provides the shared representation that contains both the textual and acoustic information for a given word (Figure 2 (a)). The HF has two steps: (i) combining the bidirectional contextual states (t_h_i and w_h_i in Figure 1) and attention distributions for each branch (t_α_i and w_α_i in Figure 1) independently to form the word-level textual and acoustic representations. As shown in Figure 2, given the input (t_α_i, t_h_i) and (w_α_i, w_h_i), we first weighed each input branch by:

$$t\_V_i = t\_\alpha_i t\_h_i \tag{7}$$

$$w\_V_i = w\_\alpha_i w\_h_i \tag{8}$$

where t_V_i and w_V_i are word-level representations for text and audio branches, respectively; (ii) concatenating them into a single space and further applying a dense layer to create the shared context vector V_i, and $V_i = (t\_V_i, w\_V_i)$. The HF combines the unimodal contextual states and attention weights; there is no attention interaction between the text modality and audio modality. The shared vectors retain the most significant characteristics from respective branches and encourages the decision making to focus on local informative features.

Vertical Fusion (VF) combines textual attentions and acoustic attentions at the word-level, using a shared attention distribution over both modalities instead of focusing on local informative representations (Figure 2 (b)). The VF is computed in three steps: (i) using a dense layer after the concatenation of the word-level textual (t_h_i) and acoustic (w_h_i) bidirectional contextual states to form the shared contextual state h_i; (ii) averaging the textual (t_α_i) and acoustic (w_α_i) attentions for each word as the shared attention distribution s_α_i; (iii) computing the weight of h_i and s_α_i as final shared context vectors V_i, where $V_i = h_i s\_\alpha_i$. Because the shared attention distribution (s_α_i) is based on averages of unimodal attentions, it is a joint attention of both textual and acoustic attentive information.

Fine-tuning Attention Fusion (FAF) preserves the original unimodal attentions and provides a fine-tuning attention for the final prediction (Figure2 (c)). The averaging of attention weights in vertical fusion potentially limits the representational power. Addressing such issue, we propose a trainable attention layer to tune the shared attention in three steps: (i) computing the shared attention distribution s_α_i and shared bidirectional contextual states h_i separately using the same approach as in vertical fusion; (ii) applying attention fine-tuning:

$$u\_e_i = tanh(W_u h_i + b_u) \tag{9}$$

$$u\_\alpha_i = \frac{exp(u\_e_i{}^\top v_u)}{\sum_{k=1}^{N} exp(u\_e_k{}^\top v_u)} + s\_\alpha_i \tag{10}$$

where W_u, b_u, and v_u are additional trainable parameters. The u_α_i can be understood as the sum of the fine-tuning score and the original shared attention distribution s_α_i; (iii) calculating the weight of u_α_i and h_i to form the final shared context vector V_i.

3.5 Decision Making

The output of the fusion layer V_i is the ith shared word-level vectors. To further make use of the combined features for classification, we applied a CNN structure with one convolutional layer and one max-pooling layer to extract the final representation from shared word-level vectors (Poria et al., 2016; Wang et al., 2016). We set up various widths for the convolutional filters (Kim, 2014) and generated a feature map c_k by:

$$f_i = tanh(W_c V_{i:i+k-1} + b_c) \tag{11}$$

$$c_k = max\{f_1, f_2, ..., f_N\} \tag{12}$$

where k is the width of the convolutional filters, f_i represents the features from window i to $i+k-1$. W_c and b_c are the trainable weights and biases. We get the final representation c by concatenating all the feature maps. A softmax function is used for the final classification.

4 Experiments

4.1 Datasets

We evaluated our model on four published datasets: two multimodal sentiment datasets (MOSI and YouTube) and two multimodal emotion recognition datasets (IEMOCAP and EmotiW).

MOSI dataset is a multimodal sentiment intensity and subjectivity dataset consisting of 93 reviews with 2199 utterance segments (Zadeh et al., 2016). Each segment was labeled by five individual annotators between -3 (strong negative) to +3 (strong positive). We used binary labels based on the sign of the annotations' average.

YouTube dataset is an English multimodal dataset that contains 262 positive, 212 negative, and 133 neutral utterance-level clips provided by (Morency et al., 2011). We only consider the positive and negative labels during our experiments.

IEMOCAP is a multimodal emotion dataset including visual, audio, and text data (Busso et al., 2008). For each sentence, we used the label agreed on by the majority (at least two of the three annotators). In this study, we evaluate both the 4-catgeory (*happy+excited*, *sad*, *anger*, and *neutral*) and 5-catgeory(*happy+excited*, *sad*, *anger*, *neutral*, and *frustration*) emotion classification problems. The final dataset consists of 586 *happy*, 1005 *excited*, 1054 *sad*, 1076 *anger*, 1677 *neutral*, and 1806 *frustration*.

EmotiW[2] is an audio-visual multimodal utterance-level emotion recognition dataset consist of video clips. To keep the consistency with the IEMOCAP dataset, we used four emotion categories as the final dataset including 150 *happy*, 117 *sad*, 133 *anger*, and 144 *neutral*. We used IBM Watson[3] speech to text software to transcribe the audio data into text.

4.2 Baselines

We compared the proposed architecture to published models. Because our model focuses on extracting sentiment and emotions from human speech, we only considered the audio and text branch applied in the previous studies.

4.2.1 Sentiment Analysis Baselines

BL-SVM extracts a bag-of-words as textual features and low-level descriptors as acoustic features. An SVM structure is used to classify the sentiments (Rosas et al., 2013).

LSTM-SVM uses LLDs as acoustic features and bag-of-n-grams (BoNGs) as textual features. The final estimate is based on decision-level fusion of text and audio predictions (Wöllmer et al., 2013).

[2]https://cs.anu.edu.au/few/ChallengeDetails.html

[3]https://www.ibm.com/watson/developercloud/speech-to-text/api/v1/

Sentiment Analysis (MOSI)					Emotion Recognition (IEMOCAP)				
Approach	Category	WA(%)	UA(%)	Weighted-F1	Approach	Category	WA(%)	UA(%)	Weighted-F1
BL-SVM*	2-class	70.4	70.6	0.668	SVM Trees	4-class	67.4	67.4	-
LSTM-SVM*	2-class	72.1	72.1	0.674	GSV-e Vector	4-class	63.2	62.3	-
C-MKL$_1$	2-class	73.6	-	0.752	C-MKL$_2$	4-class	65.5	65.0	-
TFN	2-class	75.2	-	0.760	H-DMS	5-class	60.4	60.2	0.594
LSTM(A)	2-class	73.5	-	0.703	UL-Fusion*	4-class	66.5	66.8	0.663
UL-Fusion*	2-class	72.5	72.5	0.730	DL-Fusion*	4-class	65.8	65.7	0.665
DL-Fusion*	2-class	71.8	71.8	0.720	Ours-HF	4-class	70.0	69.7	0.695
Ours-HF	2-class	74.1	74.4	0.744	Ours-VF	4-class	71.8	71.8	0.713
Ours-VF	2-class	75.3	75.3	0.755	Ours-FAF	4-class	**72.7**	**72.7**	**0.726**
Ours-FAF	2-class	**76.4**	**76.5**	**0.768**	Ours-FAF	5-class	**64.6**	**63.4**	**0.644**

Table 1: Comparison of models. *WA* = weighted accuracy. *UA* = unweighted accuracy. * denotes that we duplicated the method from cited research with the corresponding dataset in our experiment.

C-MKL$_1$ uses a CNN structure to capture the textual features and fuses them via multiple kernel learning for sentiment analysis (Poria et al., 2015).

TFN uses a tensor fusion network to extract interactions between different modality-specific features (Zadeh et al., 2017).

LSTM(A) introduces a word-level LSTM with temporal attention structure to predict sentiments on MOSI dataset (Chen et al., 2017).

4.2.2 Emotion Recognition Baselines

SVM Trees extracts LLDs and handcrafted bag-of-words as features. The model automatically generates an ensemble of SVM trees for emotion classification (Rozgic et al., 2012).

GSV-eVector generates new acoustic representations from selected LLDs using Gaussian Supervectors and extracts a set of weighed handcrafted textual features as an eVector. A linear kernel SVM is used as the final classifier (Jin et al., 2015).

C-MKL$_2$ extracts textual features using a CNN and uses openSMILE to extract 6373 acoustic features. Multiple kernel learning is used as the final classifier (Poria et al., 2016).

H-DMS uses a hybrid deep multimodal structure to extract both the text and audio emotional features. A deep neural network is used for feature-level fusion (Gu et al., 2018).

4.2.3 Fusion Baselines

Utterance-level Fusion (UL-Fusion) focuses on fusing text and audio features from an entire utterance (Gu et al., 2017). We simply concatenate the textual and acoustic representations into a joint feature representation. A softmax function is used for sentiment and emotion classification.

Decision-level Fusion (DL-Fusion) Inspired by (Wöllmer et al., 2013), we extract textual and acoustic sentence representations individually and infer the results via two softmax classifiers, respectively. As suggested by Wöllmer, we calculate a weighted sum of the text (1.2) result and audio (0.8) result as the final prediction.

4.3 Model Training

We implemented the model in Keras with Tensorflow as the backend. We set 100 as the dimension for each GRU, meaning the bidirectional GRU dimension is 200. For the decision making, we selected 2, 3, 4, and 5 as the filter width and apply 300 filters for each width. We used the rectified linear unit (ReLU) activation function and set 0.5 as the dropout rate. We also applied batch normalization functions between each layer to overcome internal covariate shift (Ioffe and Szegedy, 2015). We first trained the text attention module and audio attention module individually. Then, we tuned the fusion network based on the word-level representation outputs from each fine-tuning module. For all training procedures, we set the learning rate to 0.001 and used Adam optimization and categorical cross-entropy loss. For all datasets, we considered the speakers independent and used an 80-20 training-testing split. We further separated 20% from the training dataset for validation. We trained the model with 5-fold cross validation and used 8 as the mini batch size. We set the same amount of samples from each class to balance the training dataset during each iteration.

5 Result Analysis

5.1 Comparison with Baselines

The experimental results of different datasets show that our proposed architecture achieves state-of-the-art performance in both sentiment analysis and emotion recognition (Table 1). We re-implemented some published methods (Rosas et al., 2013; Wöllmer et al., 2013) on MOSI to get baselines.

For sentiment analysis, the proposed architecture with FAF strategy achieves 76.4% weighted accuracy, which outperforms all the five baselines (Table 1). The result demonstrates that the proposed hierarchical attention architecture and word-level fusion strategies indeed help improve the performance. There are several findings worth mentioning: (i) our model outperforms the baselines without using the low-level handcrafted acoustic features, indicating the sufficiency of MFSCs; (ii) the proposed approach achieves performance comparable to the model using text, audio, and visual data together (Zadeh et al., 2017). This demonstrates that the visual features do not contribute as much during the fusion and prediction on MOSI; (iii) we notice that (Poria et al., 2017b) reports better accuracy (79.3%) on MOSI, but their model uses a set of utterances instead of a single utterance as input.

For emotion recognition, our model with FAF achieves 72.7% accuracy, outperforming all the baselines. The result shows the proposed model brings a significant accuracy gain to emotion recognition, demonstrating the pros of the fine-tuning attention structure. It also shows that word-level attention indeed helps extract emotional features. Compared to C-MKL$_2$ and SVM Trees that require feature selection before fusion and prediction, our model does not need an additional architecture to select features. We further evaluated our models on 5 emotion categories, including frustration. Our model shows 4.2% performance improvement over H-DMS and achieves 0.644 weighted-F1. As H-DMS only achieves 0.594 F1 and also uses low-level handcrafted features, our model is more robust and efficient.

From Table 1, all the three proposed fusion strategies outperform UL-Fusion and DL-Fusion on both MOSI and IEMOCAP. Unlike utterance-level fusion that ignores the time-scale-sensitive associations across modalities, word-level fusion combines the modality-specific features for each word by aligning text and audio, allowing associative learning between the two modalities, similar to what humans do in natural conversation. The result indicates that the proposed methods improve the model performance by around 6% accuracy. We also notice that the structure with FAF outperforms the HF and VF on both MOSI and IEMOCAP dataset, which demonstrates the effectiveness and importance of the FAF strategy.

Modality	MOSI		IEMOCAP	
	WA	F1	WA	F1
T	75.0	0.748	61.8	0.620
A	60.2	0.604	62.5	0.614
T+A	**76.4**	**0.768**	**72.7**	**0.726**

Table 2: Accuracy (%) and F1 score on text only (T), audio only (A), and multi-modality using FAF (T+A).

Approach	MOSI ↓ YouTube		IEMOCAP ↓ EmotiW	
	WA	F1	WA	F1
Ours-HF	62.9	0.627	59.3	0.584
Ours-VF	64.7	0.643	60.8	0.591
Ours-FAF	**66.2**	**0.665**	**61.4**	**0.608**

Table 3: Accuracy (%) and F1 score for generalization testing.

5.2 Modality and Generalization Analysis

From Table 2, we see that textual information dominates the sentiment prediction on MOSI and there is an only 1.4% accuracy improvement from fusing text and audio. However, on IEMOCAP, audio-only outperforms text-only, but as expected, there is a significant performance improvement by combining textual and audio. The difference in modality performance might because of the more significant role vocal delivery plays in emotional expression than in sentimental expression.

We further tested the generalizability of the proposed model. For sentiment generalization testing, we trained the model on MOSI and tested on the YouTube dataset (Table 3), which achieves 66.2% accuracy and 0.665 F1 scores. For emotion recognition generalization testing, we tested the model (trained on IEMOCAP) on EmotiW and achieves 61.4% accuracy. The potential reasons that may influence the generalization are: (i) the biased labeling for different datasets (five annotators of MOSI vs one annotator of Youtube); (ii) incomplete utterance in YouTube dataset (such as "about", "he", etc.); (iii) without enough speech information (EmotiW is a wild audio-visual dataset that focuses on facial expression).

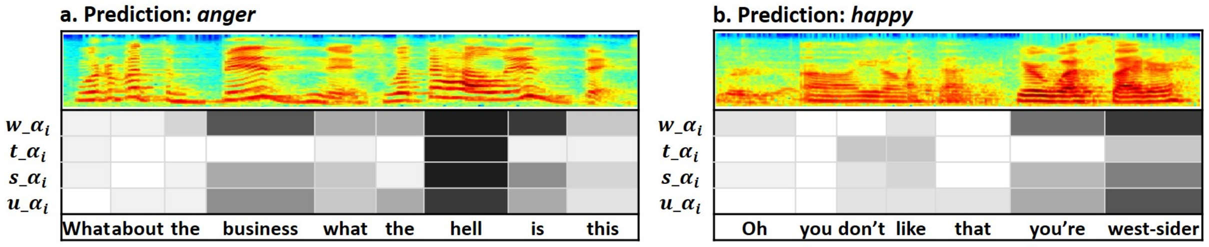


Figure 3: Attention visualization.

5.3 Visualize Attentions

Our model allows us to easily visualize the attention weights of text, audio, and fusion to better understand how the attention mechanism works. We introduce the emotional distribution visualizations for word-level acoustic attention (w_α_i), word-level textual attention (t_α_i), shared attention (s_α_i), and fine-tuning attention based on the FAF structure (u_α_i) for two example sentences (Figure 3). The color gradation represents the importance of the corresponding source data at the word-level.

Based on our visualization, the textual attention distribution (t_α_i) denotes the words that carry the most emotional significance, such as "hell" for *anger* (Figure 3 a). The textual attention shows that "don't", "like", and "west-sider" have similar weights in the *happy* example (Figure 3 b). It is hard to assign this sentence *happy* given only the text attention. However, the acoustic attention focuses on "you're" and "west-sider", removing emphasis from "don't" and "like". The shared attention (s_α_i) and fine-tuning attention (u_α_i) successfully combine both textual and acoustic attentions and assign joint attention to the correct words, which demonstrates that the proposed method can capture emphasis from both modalities at the word-level.

6 Discussion

There are several limitations and potential solutions worth mentioning: (i) the proposed architecture uses both the audio and text data to analyze the sentiments and emotions. However, not all the data sources contain or provide textual information. Many audio-visual emotion clips only have acoustic and visual information. The proposed architecture is more related to spoken language analysis than predicting the sentiments or emotions based on human speech. Automatic speech recognition provides a potential solution for generating the textual information from vocal signals. (ii) The word alignment can be easily applied to human speech. However, it is difficult to align the visual information with text, especially if the text only describes the video or audio. Incorporating visual information into an aligning model like ours would be an interesting research topic. (iii) The limited amount of multimodal sentiment analysis and emotion recognition data is a key issue for current research, especially for deep models that require a large number of samples. Compared large unimodal sentiment analysis and emotion recognition datasets, the MOSI dataset only consists of 2199 sentence-level samples. In our experiments, the EmotiW and MOUD datasets could only be used for generalization analysis due to their small size. Larger and more general datasets are necessary for multimodal sentiment analysis and emotion recognition in the future.

7 Conclusion

In this paper, we proposed a deep multimodal architecture with hierarchical attention for sentiment and emotion classification. Our model aligned the text and audio at the word-level and applied attention distributions on textual word vectors, acoustic frame vectors, and acoustic word vectors. We introduced three fusion strategies with a CNN structure to combine word-level features to classify emotions. Our model outperforms the state-of-the-art methods and provides effective visualization of modality-specific features and fusion feature interpretation.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and feedback. We thank the useful suggestions from Kaixiang Huang. This research was funded by the National Institutes of Health under Award Number R01LM011834.

References

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. 2014. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171. ACM.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarepa collaborative voice analysis repository for speech technologies. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 960–964. IEEE.

Florian Eyben, Martin Wöllmer, Alex Graves, Björn Schuller, Ellen Douglas-Cowie, and Roddy Cowie. 2010a. On-line emotion recognition in a 3-d activation-valence-time continuum using acoustic and linguistic cues. *Journal on Multimodal User Interfaces*, 3(1-2):7–19.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010b. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM.

Kate Forbes-Riley and Diane Litman. 2004. Predicting emotion in spoken dialogue from multiple knowledge sources. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

Yue Gu, Shuhong Chen, and Ivan Marsic. 2018. Deep multimodal learning for emotion recognition in spoken language. *arXiv preprint arXiv:1802.08332*.

Yue Gu, Xinyu Li, Shuhong Chen, Jianyu Zhang, and Ivan Marsic. 2017. Speech intention classification with multimodal deep learning. In *Canadian Conference on Artificial Intelligence*, pages 260–271. Springer.

Che-Wei Huang and Shrikanth S Narayanan. 2016. Attention assisted discovery of sub-utterance structure in speech emotion recognition. In *INTERSPEECH*, pages 1387–1391.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Qin Jin, Chengxin Li, Shizhe Chen, and Huimin Wu. 2015. Speech emotion recognition with acoustic and lexical features. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4749–4753. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diane J Litman and Kate Forbes-Riley. 2004. Predicting student emotions in computer-human tutoring dialogues. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 351. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Seyedmahdad Mirsamadi, Emad Barsoum, and Cha Zhang. 2017. Automatic speech emotion recognition using recurrent neural networks with local attention. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2227–2231. IEEE.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM.

Michael Neumann and Ngoc Thang Vu. 2017. Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech. *arXiv preprint arXiv:1706.00612*.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2539–2544.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE.

Verónica Pérez Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Multimodal sentiment analysis of spanish online videos. *IEEE Intelligent Systems*, 28(3):38–45.

Viktor Rozgic, Sankaranarayanan Ananthakrishnan, Shirin Saleem, Rohit Kumar, and Rohit Prasad. 2012. Ensemble of svm trees for multimodal emotion recognition. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–4. IEEE.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

Arman Savran, Houwei Cao, Miraj Shah, Ani Nenkova, and Ragini Verma. 2012. Combining video, audio and lexical indicators of affect in spontaneous conversation via particle filtering. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 485–492. ACM.

Dino Seppi, Anton Batliner, Björn Schuller, Stefan Steidl, Thurid Vogt, Johannes Wagner, Laurence Devillers, Laurence Vidrascu, Noam Amir, and Vered Aharonson. 2008. Patterns, prototypes, performance: classifying emotional user states. In *Ninth Annual Conference of the International Speech Communication Association*.

Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. 2016. Select-additive learning: Improving cross-individual generalization in multimodal sentiment analysis. *arXiv preprint arXiv:1609.05244*.

Martin Wöllmer, Felix Weninger, Tobias Knaup, Björn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems*, 28(3):46–53.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259*.

Shiqing Zhang, Shiliang Zhang, Tiejun Huang, Wen Gao, and Qi Tian. 2017. Learning affective features with a hybrid deep model for audio-visual emotion recognition. *IEEE Transactions on Circuits and Systems for Video Technology*.

Multimodal Language Analysis in the Wild: CMU-MOSEI Dataset and Interpretable Dynamic Fusion Graph

Amir Zadeh[1], **Paul Pu Liang**[2], **Jonathan Vanbriesen**[1], **Soujanya Poria**[3],
Edmund Tong[1], **Erik Cambria**[4], **Minghai Chen**[1], **Louis-Philippe Morency**[1]
{1- Language Technologies Institute, 2- Machine Learning Department}, CMU, USA
{3- A*STAR, 4- Nanyang Technological University}, Singapore
{abagherz,pliang,jvanbrie}@cs.cmu.edu, soujanya@sentic.net
edtong@cmu.edu, cambria@ntu.edu.sg, morency@cs.cmu.edu

Abstract

Analyzing human multimodal language is an emerging area of research in NLP. Intrinsically human communication is multimodal (heterogeneous), temporal and asynchronous; it consists of the language (words), visual (expressions), and acoustic (paralinguistic) modalities all in the form of asynchronous coordinated sequences. From a resource perspective, there is a genuine need for large scale datasets that allow for in-depth studies of multimodal language. In this paper we introduce CMU Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI), the largest dataset of sentiment analysis and emotion recognition to date. Using data from CMU-MOSEI and a novel multimodal fusion technique called the Dynamic Fusion Graph (DFG), we conduct experimentation to investigate how modalities interact with each other in human multimodal language. Unlike previously proposed fusion techniques, DFG is highly interpretable and achieves competitive performance compared to the current state of the art.

1 Introduction

Theories of language origin identify the combination of language and nonverbal behaviors (vision and acoustic modality) as the prime form of communication utilized by humans throughout evolution (Müller, 1866). In natural language processing, this form of language is regarded as human multimodal language. Modeling multimodal language has recently become a centric research direction in both NLP and multimodal machine learning (Hazarika et al., 2018; Zadeh et al., 2018a; Poria et al., 2017a; Baltrušaitis et al., 2017; Chen et al., 2017). Studies strive to model the dual dynamics of multimodal language: intra-modal dynamics (dynamics within each modality) and cross-modal dynamics (dynamics across different modalities). However, from a resource perspective, previous multimodal language datasets have severe shortcomings in the following aspects:

Diversity in the training samples: The diversity in training samples is crucial for comprehensive multimodal language studies due to the complexity of the underlying distribution. This complexity is rooted in variability of intra-modal and cross-modal dynamics for language, vision and acoustic modalities (Rajagopalan et al., 2016). Previously proposed datasets for multimodal language are generally small in size due to difficulties associated with data acquisition and costs of annotations.

Variety in the topics: Variety in topics opens the door to generalizable studies across different domains. Models trained on only few topics generalize poorly as language and nonverbal behaviors tend to change based on the impression of the topic on speakers' internal mental state.

Diversity of speakers: Much like writing styles, speaking styles are highly idiosyncratic. Training models on only few speakers can lead to degenerate solutions where models learn the identity of speakers as opposed to a generalizable model of multimodal language (Wang et al., 2016).

Variety in annotations Having multiple labels to predict allows for studying the relations between labels. Another positive aspect of having variety of labels is allowing for multi-task learning which has shown excellent performance in past research.

Our first contribution in this paper is to introduce the largest dataset of multimodal sentiment and emotion recognition called CMU Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI). CMU-MOSEI contains 23,453 annotated video segments from 1,000 distinct speakers and

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 2236–2246
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

250 topics. Each video segment contains manual transcription aligned with audio to phoneme level. All the videos are gathered from online video sharing websites [1]. The dataset is currently a part of the CMU Multimodal Data SDK and is freely available to the scientific community through Github [2].

Our second contribution is an interpretable fusion model called Dynamic Fusion Graph (DFG) to study the nature of cross-modal dynamics in multimodal language. DFG contains built-in efficacies that are directly related to how modalities interact. These efficacies are visualized and studied in detail in our experiments. Aside interpretability, DFG achieves superior performance compared to previously proposed models for multimodal sentiment and emotion recognition on CMU-MOSEI.

2 Background

In this section we compare the CMU-MOSEI dataset to previously proposed datasets for modeling multimodal language. We then describe the baselines and recent models for sentiment analysis and emotion recognition.

2.1 Comparison to other Datasets

We compare CMU-MOSEI to an extensive pool of datasets for sentiment analysis and emotion recognition. The following datasets include a combination of language, visual and acoustic modalities as their input data.

2.1.1 Multimodal Datasets

CMU-MOSI (Zadeh et al., 2016b) is a collection of 2199 opinion video clips each annotated with sentiment in the range [-3,3]. CMU-MOSEI is the next generation of CMU-MOSI. The **ICT-MMMO** (Wöllmer et al., 2013) consists of online social review videos annotated at the video level for sentiment. **YouTube** (Morency et al., 2011) contains videos from the social media web site YouTube that span a wide range of product reviews and opinion videos. **MOUD** (Perez-Rosas et al., 2013) consists of product review videos in Spanish. Each video consists of multiple segments labeled to display positive, negative or neutral sentiment. **IEMOCAP** (Busso et al., 2008) consists of 151 videos of recorded dialogues, with 2 speakers per session for a total of 302 videos across the dataset. Each segment is annotated for the presence of 9 emotions (angry, excited, fear, sad, surprised, frustrated, happy, disappointed and neutral) as well as valence, arousal and dominance.

[1]following creative commons license allows for personal unrestricted use and redistribution of the videos

[2]https://github.com/A2Zadeh/CMU-MultimodalDataSDK

Dataset	# S	# Sp	Mod	Sent	Emo	TL (hh:mm:ss)
CMU-MOSEI	**23,453**	**1,000**	$\{l, v, a\}$	✓	✓	**65:53:36**
CMU-MOSI	2,199	98	$\{l, v, a\}$	✓	✗	02:36:17
ICT-MMMO	340	200	$\{l, v, a\}$	✓	✗	13:58:29
YouTube	300	50	$\{l, v, a\}$	✓	✗	00:29:41
MOUD	400	101	$\{l, v, a\}$	✓	✗	00:59:00
SST	11,855	–	$\{l\}$	✓	✗	–
Cornell	2,000	–	$\{l\}$	✓	✗	–
Large Movie	25,000	–	$\{l\}$	✓	✗	–
STS	5,513	–	$\{l\}$	✓	✗	–
IEMOCAP	10,000	10	$\{l, v, a\}$	✗	✓	11:28:12
SAL	23	4	$\{v, a\}$	✗	✓	11:00:00
VAM	499	20	$\{v, a\}$	✗	✓	12:00:00
VAM-faces	1,867	20	$\{v\}$	✗	✓	–
HUMAINE	50	4	$\{v, a\}$	✗	✓	04:11:00
RECOLA	46	46	$\{v, a\}$	✗	✓	03:50:00
SEWA	538	408	$\{v, a\}$	✗	✓	04:39:00
SEMAINE	80	20	$\{v, a\}$	✗	✓	06:30:00
AFEW	1,645	330	$\{v, a\}$	✗	✓	02:28:03
AM-FED	242	242	$\{v\}$	✗	✓	03:20:25
Mimicry	48	48	$\{v, a\}$	✗	✓	11:00:00
AFEW-VA	600	240	$\{v, a\}$	✗	✓	00:40:00

Table 1: Comparison of the CMU-MOSEI dataset with previous sentiment analysis and emotion recognition datasets. #S denotes the number of annotated data points. #Sp is the number of distinct speakers. Mod indicates the subset of modalities present from $\{(l)anguage, (v)ision, (a)udio\}$. Sent and Emo columns indicate presence of sentiment and emotion labels. TL denotes the total number of video hours.

2.1.2 Language Datasets

Stanford Sentiment Treebank (SST) (Socher et al., 2013) includes fine grained sentiment labels for phrases in the parse trees of sentences collected from movie review data. While SST has larger pool of annotations, we only consider the root level annotations for comparison. **Cornell Movie Review** (Pang et al., 2002) is a collection of 2000 movie-review documents and sentences labeled with respect to their overall sentiment polarity or subjective rating. **Large Movie Review** dataset (Maas et al., 2011) contains text from highly polar movie reviews. **Sanders Tweets Sentiment (STS)** consists of 5513 hand-classified tweets each classified with respect to one of four topics of Microsoft, Apple, Twitter, and Google.

2.1.3 Visual and Acoustic Datasets

The **Vera am Mittag (VAM)** corpus consists of 12 hours of recordings of the German TV talk-show “Vera am Mittag” (Grimm et al., 2008). This audio-visual data is labeled for continuous-valued scale for three emotion primitives: valence, activation and dominance. VAM-Audio and VAM-Faces are subsets that contain on acoustic and visual inputs respectively. **RECOLA** (Ringeval et al., 2013) consists of 9.5 hours of audio, visual, and physiological (electrocardiogram, and electrodermal activity) recordings of online dyadic interactions. **Mimicry** (Bilakhia et al., 2015) consists of audiovisual recordings of human interactions in two situations: while discussing a political topic and while playing a role-playing game. **AFEW** (Dhall et al., 2012, 2015) is a dynamic temporal facial expressions data corpus consisting of close to real world environment extracted from movies.

Detailed comparison of CMU-MOSEI to the datasets in this section is presented in Table 1. CMU-MOSEI has longer total duration as well as larger number of data point in total. Furthermore, CMU-MOSEI has a larger variety in number of speakers and topics. It has all three modalities provided, as well as annotations for both sentiment and emotions.

2.2 Baseline Models

Modeling multimodal language has been the subject of studies in NLP and multimodal machine learning. Notable approaches are listed as follows and indicated with a symbol for reference in the Experiments and Discussion section (Section 5).

# **MFN:** (Memory Fusion Network) (Zadeh et al., 2018a) synchronizes multimodal sequences using a multi-view gated memory that stores intra-view and cross-view interactions through time. ■ **MARN:** (Multi-attention Recurrent Network) (Zadeh et al., 2018b) models intra-modal and multiple cross-modal interactions by assigning multiple attention coefficients. Intra-modal and cross-modal interactions are stored in a hybrid LSTM memory component. $*$ **TFN** (Tensor Fusion Network) (Zadeh et al., 2017) models inter and intra modal interactions by creating a multi-dimensional tensor that captures unimodal, bimodal and trimodal interactions. $\diamond$ **MV-LSTM** (Multi-View LSTM) (Rajagopalan et al., 2016) is a recurrent model that designates regions inside a LSTM to different views of the data. § **EF-LSTM** (Early Fusion LSTM) concatenates the inputs from different modalities at each time-step and uses that as the input to a single LSTM (Hochreiter and Schmidhuber, 1997; Graves et al., 2013; Schuster and Paliwal, 1997). In case of unimodal models EF-LSTM refers to a single LSTM.

We also compare to the following baseline models: † **BC-LSTM** (Poria et al., 2017b), ♣ **C-MKL** (Poria et al., 2016), $\flat$ **DF** (Nojavanasghari et al., 2016), ♡ **SVM** (Cortes and Vapnik, 1995; Zadeh et al., 2016b; Perez-Rosas et al., 2013; Park et al., 2014), $\bullet$ **RF** (Breiman, 2001), **THMM** (Morency et al., 2011), **SAL-CNN** (Wang et al., 2016), **3D-CNN** (Ji et al., 2013). For language only baseline models: $\cup$ **CNN-LSTM** (Zhou et al., 2015), **RNTN** (Socher et al., 2013), $\times$: **DynamicCNN** (Kalchbrenner et al., 2014), $\triangleright$ **DAN** (Iyyer et al., 2015), $\wr$ **DHN** (Srivastava et al., 2015), $\triangleleft$ **RHN** (Zilly et al., 2016). For acoustic only baseline models: **AdieuNet** (Trigeorgis et al., 2016), **SER-LSTM** (Lim et al., 2016).

3 CMU-MOSEI Dataset

Understanding expressed sentiment and emotions are two crucial factors in human multimodal language. We introduce a novel dataset for multimodal sentiment and emotion recognition called CMU Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI). In the following subsections, we first explain the details of the CMU-MOSEI data acquisition, followed by details of annotation and feature extraction.

3.1 Data Acquisition

Social multimedia presents a unique opportunity for acquiring large quantities of data from various speakers and topics. Users of these social multimedia websites often post their opinions in the forms of monologue videos; videos with only one person in front of camera discussing a certain topic of interest. Each video inherently contains three modalities: language in the form of spoken text, visual via perceived gestures and facial expressions, and acoustic through intonations and prosody.

During our automatic data acquisition process, videos from YouTube are analyzed for the presence of one speaker in the frame using face detection to ensure the video is a monologue. We limit the videos to setups where the speaker’s attention is exclusively towards the camera by rejecting videos that have moving cameras (such as camera on bikes or selfies recording while walking). We use a diverse set of 250 frequently used topics in online videos as the seed for acquisition. We restrict the

Figure 1: The diversity of topics of videos in CMU-MOSEI, displayed as a word cloud. Larger words indicate more videos from that topic. The most frequent 3 topics are reviews (16.2%), debate (2.9%) and consulting (1.8%) while the remaining topics are almost uniformly distributed.

number of videos acquired from each channel to a maximum of 10. This resulted in discovering 1,000 identities from YouTube. The definition of a identity is proxy to the number of channels since accurate identification requires quadratic manual annotations, which is infeasible for high number of speakers. Furthermore, we limited the videos to have manual and properly punctuated transcriptions provided by the uploader. The final pool of acquired videos included 5,000 videos which were then manually checked for quality of video, audio and transcript by 14 expert judges over three months. The judges also annotated each video for gender and confirmed that each video is an acceptable monologue. A set of 3228 videos remained after manual quality inspection. We also performed automatic checks on the quality of video and transcript which are discussed in Section 3.3 using facial feature extraction confidence and forced alignment confidence. Furthermore, we balance the gender in the dataset using the data provided by the judges (57% male to 43% female). This constitutes the final set of raw videos in CMU-MOSEI. The topics covered in the final set of videos are shown in Figure 1 as a Venn-style word cloud (Coppersmith and Kelly, 2014) with the size proportional to the number of videos gathered for that topic. The most frequent 3 topics are reviews (16.2%), debate (2.9%) and consulting (1.8%). The remaining topics are almost uniformly distributed [3].

The final set of videos are then tokenized into

[3]more detailed analysis such as exact percentages and number of videos per topic are available in the supplementary material

Total number of sentences	23453
Total number of videos	3228
Total number of distinct speakers	1000
Total number of distinct topics	250
Average number of sentences in a video	7.3
Average length of sentences in seconds	7.28
Total number of words in sentences	447143
Total of unique words in sentences	23026
Total number of words appearing at least 10 times in the dataset	3413
Total number of words appearing at least 20 times in the dataset	1971
Total number of words appearing at least 50 times in the dataset	888

Table 2: Summary of CMU-MOSEI dataset statistics.

sentences using punctuation markers manually provided by transcripts. Due to the high quality of the transcripts, using punctuation markers showed better sentence quality than using the Stanford CoreNLP tokenizer (Manning et al., 2014). This was verified on a set of 20 random videos by two experts. After tokenization, a set of 23,453 sentences were chosen as the final sentences in the dataset. This was achieved by restricting each identity to contribute at least 10 and at most 50 sentences to the dataset. Table 2 shows high-level summary statistics of the CMU-MOSEI dataset.

3.2 Annotation

Annotation of CMU-MOSEI follows closely the annotation of CMU-MOSI (Zadeh et al., 2016a) and Stanford Sentiment Treebank (Socher et al., 2013). Each sentence is annotated for sentiment on a [-3,3] Likert scale of: [-3: highly negative, -2 negative, -1 weakly negative, 0 neutral, +1 weakly positive, +2 positive, +3 highly positive]. Ekman emotions (Ekman et al., 1980) of {happiness, sadness, anger, fear, disgust, surprise} are annotated on a [0,3] Likert scale for presence of emotion x: [0: no evidence of x, 1: weakly x, 2: x, 3: highly x]. The annotation was carried out by 3 crowdsourced judges from Amazon Mechanical Turk platform. To avert implicitly biasing the judges and to capture the raw perception of the crowd, we avoided extreme annotation training and instead provided the judges with a 5 minutes training video on how to use the annotation system. All the annotations have been carried out by only master workers with higher than 98% approval rate to assure high quality annotations [4].

Figure 2 shows the distribution of sentiment and emotions in CMU-MOSEI dataset. The distribution

[4]Extensive statistics of the dataset including the crawling mechanism, the annotation UI, training procedure for the workers, agreement scores are available in submitted supplementary material available on arXiv.

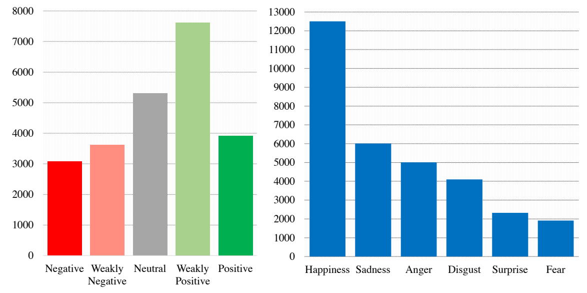


Figure 2: Distribution of sentiment and emotions in the CMU-MOSEI dataset. The distribution shows a natural skew towards more frequently used emotions. However, the least frequent emotion, fear, still has 1,900 data points which is an acceptable number for machine learning studies.

shows a slight shift in favor of positive sentiment which is similar to distribution of CMU-MOSI and SST. We believe that this is an implicit bias in online opinions being slightly shifted towards positive, since this is also present in CMU-MOSI. The emotion histogram shows different prevalence for different emotions. The most common category is happiness with more than 12,000 positive sample points. The least prevalent emotion is fear with almost 1900 positive sample points which is an acceptable number for machine learning studies.

3.3 Extracted Features

Data points in CMU-MOSEI come in video format with one speaker in front of the camera. The extracted features for each modality are as follows (for other benchmarks we extract the same features):

Language: All videos have manual transcription. Glove word embeddings (Pennington et al., 2014) were used to extract word vectors from transcripts. Words and audio are aligned at phoneme level using P2FA forced alignment model (Yuan and Liberman, 2008). Following this, the visual and acoustic modalities are aligned to the words by interpolation. Since the utterance duration of words in English is usually short, this interpolation does not lead to substantial information loss.

Visual: Frames are extracted from the full videos at 30Hz. The bounding box of the face is extracted using the MTCNN face detection algorithm (Zhang et al., 2016). We extract facial action units through Facial Action Coding System (FACS) (Ekman et al., 1980). Extracting these action units allows for accurate tracking and understanding of the facial expressions (Baltrušaitis et al., 2016). We also extract a set of six basic emotions purely from static faces using Emotient FACET (iMotions, 2017). MultiComp OpenFace (Baltrušaitis et al., 2016) is used to extract the set of 68 facial landmarks, 20 facial shape parameters, facial HoG features, head pose, head orientation and eye gaze (Baltrušaitis et al., 2016). Finally, we extract face embeddings from commonly used facial recognition models such as DeepFace (Taigman et al., 2014), FaceNet (Schroff et al., 2015) and SphereFace (Liu et al., 2017).

Acoustic: We use the COVAREP software (Degottex et al., 2014) to extract acoustic features including 12 Mel-frequency cepstral coefficients, pitch, voiced/unvoiced segmenting features (Drugman and Alwan, 2011), glottal source parameters (Drugman et al., 2012; Alku et al., 1997, 2002), peak slope parameters and maxima dispersion quotients (Kane and Gobl, 2013). All extracted features are related to emotions and tone of speech.

4 Multimodal Fusion Study

From the linguistics perspective, understanding the interactions between language, visual and audio modalities in multimodal language is a fundamental research problem. While previous works have been successful with respect to accuracy metrics, they have not created new insights on how the fusion is performed in terms of what modalities are related and how modalities engage in an interaction during fusion. Specifically, to understand the fusion process one must first understand the n-modal dynamics (Zadeh et al., 2017). n-modal dynamics state that there exists different combination of modalities and that all of these combinations must be captured to better understand the multimodal language. In this paper, we define building the n-modal dynamics as a hierarchical process and propose a new fusion model called the Dynamic Fusion Graph (DFG). DFG is easily interpretable through what is called efficacies in graph connections. To utilize this new fusion model in a multimodal language framework, we build upon Memory Fusion Network (MFN) by replacing the original fusion component in the MFN with our DFG. We call this resulting model the Graph Memory Fusion Network (Graph-MFN). Once the model is trained end to end, we analyze the efficacies in the DFG to study the fusion mechanism learned for modalities in multimodal language. In addition to being an interpretable fusion mechanism,

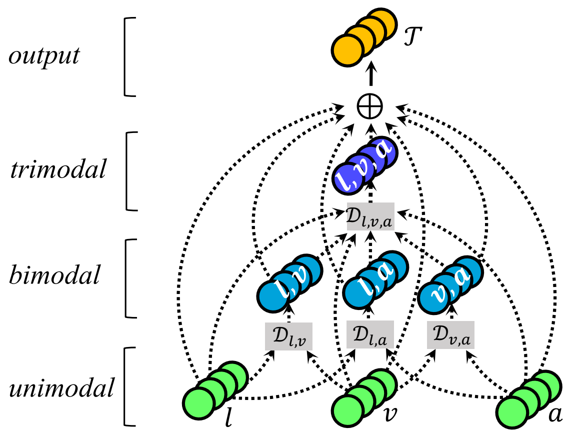


Figure 3: The structure of Dynamic Fusion Graph (DFG) for three modalities of $\{(l)anguage, (v)ision, (a)coustic\}$. Dashed lines in DFG show the dynamic connections between vertices controlled by the efficacies (α).

Graph-MFN also outperforms previously proposed state-of-the-art models for sentiment analysis and emotion recognition on the CMU-MOSEI.

4.1 Dynamic Fusion Graph

In this section we discuss the internal structure of the proposed Dynamic Fusion Graph (DFG) neural model (Figure 3. DFG has the following properties: 1) it explicitly models the n-modal interactions, 2) does so with an efficient number of parameters (as opposed to previous approaches such as Tensor Fusion (Zadeh et al., 2017)) and 3) can dynamically alter its structure and choose the proper fusion graph based on the importance of each n-modal dynamics during inference. We assume the set of modalities to be $M = \{(l)anguage, (v)ision, (a)coustic\}$. The unimodal dynamics are denoted as $\{l\}, \{v\}, \{a\}$, the bimodal dynamics as $\{l, v\}, \{v, a\}, \{l, a\}$ and trimodal dynamics as $\{l, v, a\}$. These dynamics are in the form of latent representations and are each considered as vertices inside a graph $G = (V, E)$ with V the set of vertices and E the set of edges. A directional neural connection is established between two vertices v_i and v_j only if $v_i \subset v_j$. For example, $\{l\} \subset \{l, v\}$ which results in a connection between $< language >$ and $< language, vision >$. This connection is denoted as an edge e_{ij}. $\mathcal{D}_j$ takes as input all v_i that satisfy the neural connection formula above for v_j.

We define an efficacy for each edge e_{ij} denoted as α_{ij}. v_i is multiplied by α_{ij} before being used as input to D_j. Each α is a sigmoid activated probability neuron which indicates how strong or weak the connection is between v_i and v_j. αs are the main source of interpretability in DFG. The vector of all αs is inferred using a deep neural network $\mathcal{D}_\alpha$ which takes as input singleton vertices in V (l, v, and a). We leave it to the supervised training objective to learn parameters of $\mathcal{D}_\alpha$ and make good use of efficacies, thus dynamically controlling the structure of the graph. The singleton vertices are chosen for this purpose since they have no incoming edges thus no efficacy associated with those edges (no efficacy is needed to infer the singleton vertices). The same singleton vertices l, v, and a are the inputs to the DFG. In the next section we discuss how these inputs are given to DFG. All vertices are connected to the output vertex $\mathcal{T}_t$ of the network via edges scaled by their respective efficacy. The overall structure of the vertices, edges and respective efficacies is shown in Figure 3. There are a total of 8 vertices (counting the output vertex), 19 edges and subsequently 19 efficacies.

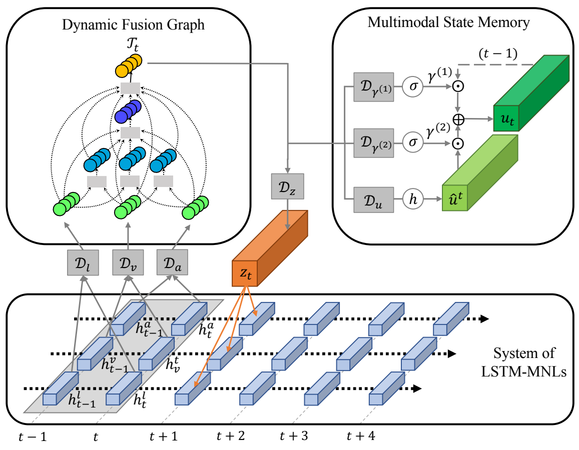


Figure 4: The overview of Graph Memory Fusion Network (Graph-MFN) pipeline. Graph-MFN replaces the fusion block in MFN with a Dynamic Fusion Graph (DFG). For description of variables and memory formulation please refer to the original Memory Fusion Network paper (Zadeh et al., 2018a).

4.2 Graph-MFN

To test the performance of DFG, we use a similar recurrent architecture to Memory Fusion Network (MFN). MFN is a recurrent neural model with three main components 1) System of LSTMs: a set of parallel LSTMs with each LSTM modeling a single modality. 2) Delta-memory Attention Network is the component that performs multimodal fusion

Dataset	**MOSEI Sentiment**						**MOSEI Emotions**											
Task	Sentiment						Anger		Disgust		Fear		Happy		Sad		Surprise	
Metric	A^2	F1	A^5	A^7	MAE	r	WA	F1	WA	F1	WA	F1	WA	F1	WA	F1	WA	F1
LANGUAGE																		
SOTA2	$74.1^{\S}$	$74.1^{\triangleright}$	$43.1^{\wr}$	$42.9^{\wr}$	$0.75^{\S}$	$0.46^{\wr}$	$56.0^{\cup}$	$71.0^{\times}$	$59.0^{\S}$	$67.1^{\triangleright}$	$56.2^{\S}$	$79.7^{\S}$	$53.0^{\triangleright}$	$44.1^{\triangleright}$	$53.8^{\wr}$	$49.9^{\wr}$	$53.2^{\times}$	$70.0^{\triangleright}$
SOTA1	$74.3^{\triangleright}$	$74.1^{\S}$	$43.2^{\S}$	$43.2^{\S}$	$0.74^{\triangleright}$	$0.47^{\S}$	$56.6^{\wr}$	$71.8^{\bullet}$	$64.0^{\triangleright}$	$72.6^{\bullet}$	$58.8^{\times}$	$89.8^{\bullet}$	$54.0^{\S}$	$47.0^{\S}$	$54.0^{\S}$	$61.2^{\bullet}$	**54.3**$^{\triangleright}$	$85.3^{\bullet}$
VISUAL																		
SOTA2	$73.8^{\S}$	$73.5^{\S}$	$42.5^{\triangleright}$	$42.5^{\triangleright}$	$0.78^{\wr}$	$0.41^{\heartsuit}$	$54.4^{\wr}$	$64.6^{\S}$	$54.4^{\heartsuit}$	$71.5^{\triangleleft}$	$51.3^{\S}$	$78.4^{\S}$	$53.4^{\wr}$	$40.8^{\S}$	$54.3^{\triangleright}$	$60.8^{\bullet}$	$51.3^{\triangleright}$	$84.2^{\S}$
SOTA1	$73.9^{\triangleright}$	$73.7^{\triangleright}$	$42.7^{\wr}$	$42.7^{\wr}$	$0.78^{\S}$	$0.43^{\wr}$	$60.0^{\S}$	$71.0^{\bullet}$	$60.3^{\wr}$	$72.4^{\bullet}$	**64.2**$^{\heartsuit}$	$89.8^{\bullet}$	$57.4^{\bullet}$	$49.3^{\bullet}$	$57.7^{\S}$	$61.5^{\triangleleft}$	$51.8^{\S}$	$85.4^{\bullet}$
ACOUSTIC																		
SOTA2	$74.2^{\wr}$	$73.8^{\triangle}$	$42.1^{\triangle}$	$42.1^{\triangle}$	$0.78^{\triangleright}$	$0.43^{\S}$	$55.5^{\triangleleft}$	$51.8^{\triangle}$	$58.9^{\triangleright}$	$72.4^{\bullet}$	$58.5^{\triangleright}$	$89.8^{\bullet}$	$57.2^{\cap}$	$55.5^{\cap}$	$58.9^{\triangleleft}$	$65.9^{\triangleleft}$	$52.2^{\heartsuit}$	$83.6^{\cap}$
SOTA1	$74.2^{\triangle}$	$73.9^{\wr}$	$42.4^{\cap}$	$42.4^{\cap}$	$0.74^{\cap}$	$0.43^{\triangleright}$	$56.4^{\triangle}$	$71.9^{\bullet}$	$60.9^{\S}$	$72.4^{\bullet}$	$62.7^{\S}$	$89.8^{\triangleleft}$	$61.5^{\S}$	$61.4^{\S}$	**62.0**$^{\cap}$	**69.2**$^{\cap}$	**54.3**$^{\triangleleft}$	$85.4^{\bullet}$
MULTIMODAL																		
SOTA2	$76.0^{\#}$	$76.0^{\#}$	$44.7^{\dagger}$	$44.6^{\dagger}$	0.72^{*}	0.52^{*}	$56.0\diamond$	$71.4^{\flat}$	$65.2^{\#}$	$71.4^{\#}$	$56.7^{\S}$	**89.9**$^{\#}$	$57.8^{\S}$	66.6^{*}	58.9^{*}	$60.8^{\#}$	52.2^{*}	$85.4^{\bullet}$
SOTA1	$76.4\diamond$	$76.4^{\diamond}$	44.8^{*}	44.7^{*}	$0.72^{\#}$	$0.52^{\#}$	60.5^{*}	$72.0^{\bullet}$	$67.0^{\flat}$	$73.2^{\bullet}$	$60.0^{\heartsuit}$	**89.9**$^{\bullet}$	**66.5***	**71.0**$^{\blacksquare}$	$59.2^{\S}$	$61.8^{\bullet}$	$53.3^{\#}$	$85.4^{\#}$
Graph-MFN	**76.9**	**77.0**	**45.1**	**45.0**	**0.71**	**0.54**	**62.6**	**72.8**	**69.1**	**76.6**	62.0	**89.9**	66.3	66.3	60.4	66.9	53.7	**85.5**

Table 3: Results for sentiment analysis and emotion recognition on the MOSEI dataset (reported results are as of 5/11/2018. please check the CMU Multimodal Data SDK github for current state of the art and new features for CMU-MOSEI and other datasets). SOTA1 and SOTA2 refer to the previous best and second best state-of-the-art models (from Section 2) respectively. Compared to the baselines Graph-MFN achieves superior performance in sentiment analysis and competitive performance in emotion recognition. For all metrics, higher values indicate better performance except for MAE where lower values indicate better performance.

by assigning coefficients to highlight cross-modal dynamics. 3) Multiview Gated Memory is a component that stores the output of multimodal fusion. We replace the Delta-memory Attention Network with DFG and refer to the modified model as Graph Memory Fusion Network (Graph-MFN). Figure 4 shows the overall architecture of the Graph-MFN.

Similar to MFN, Graph-MFN employs a system of LSTMs for modeling individual modalities. c_l, c_v, and c_a represent the memory of LSTMs for language, vision and acoustic modalities respectively. $D_m, m \in \{l, v, a\}$ is a fully connected deep neural network that takes in $h^m_{[t-1,t]}$ the LSTM representation across two consecutive timestamps, which allows the network to track changes in memory dimensions across time. The outputs of D_l, D_v and D_a are the singleton vertices for the DFG. The DFG models cross-modal interactions and encodes the cross-modal representations in its output vertex $\mathcal{T}_t$ for storage in the Multi-view Gated Memory u_t. The Multi-view Gated Memory functions using a network D_u that transforms $\mathcal{T}_t$ into a proposed memory update $\hat{u}_t$. γ_1 and γ_2 are the Multi-view Gated Memory's retain and update gates respectively and are learned using networks D_{γ_1} and D_{γ_2}. Finally, a network D_z transforms $\mathcal{T}_t$ into a multimodal representation z_t to update the system of LSTMs. The output of Graph-MFN in all the experiments is the output of each LSTM h^m_T as well as contents of the Multi-view Gated Memory at time T (last recurrence timestep), u_T. This output is subsequently connected to a classification or regression layer for final prediction (for sentiment and emotion recognition).

5 Experiments and Discussion

In our experiments, we seek to evaluate how modalities interact during multimodal fusion by studying the efficacies of DFG through time.

Table 3 shows the results on CMU-MOSEI. Accuracy is reported as A^x where x is the number of sentiment classes as well as F1 measure. For regression we report MAE and correlation (r). For emotion recognition due to the natural imbalances across various emotions, we use weighted accuracy (Tong et al., 2017) and F1 measure. Graph-MFN shows superior performance in sentiment analysis and competitive performance in emotion recognition. Therefore, DFG is both an effective and interpretable model for multimodal fusion.

To better understand the internal fusion mechanism between modalities, we visualize the behavior of the learned DFG efficacies in Figure 5 for various cases (deep red denotes high efficacy and deep blue denotes low efficacy).

Multimodal Fusion has a Volatile Nature: The first observation is that the structure of the DFG is changing case by case and for each case over time. As a result, the model seems to be selectively prioritizing certain dynamics over the others. For example, in case (I) where all modalities are informative, all efficacies seem to be high, imply-

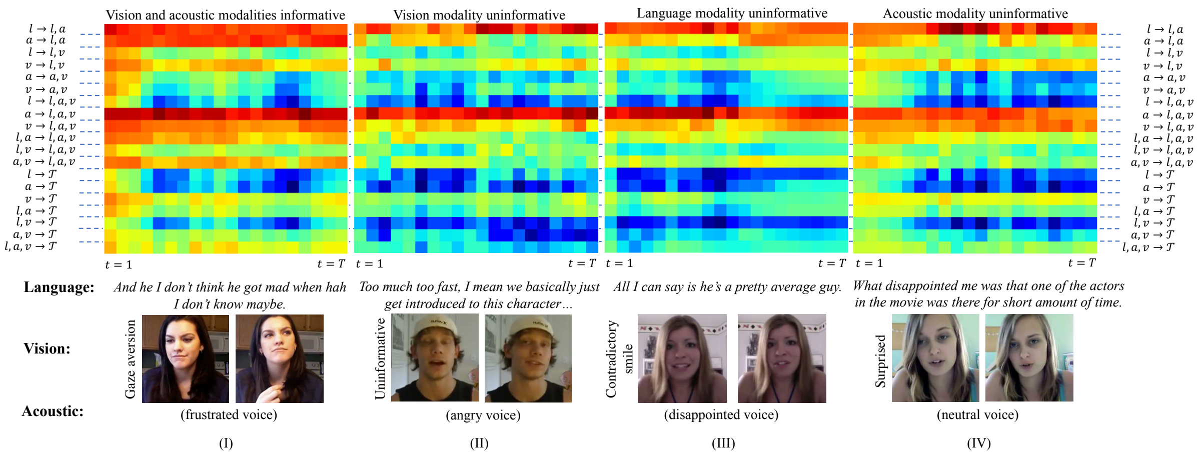


Figure 5: Visualization of DFG efficacies across time. The efficacies (thus the DFG structure) change over time as DFG is exposed to new information. DFG is able choose which n-modal dynamics to rely on. It also learns priors about human communication since certain efficacies (thus edges in DFG) remain unchanged across time and across data points.

ing that the DFG is able to find useful information in unimodal, bimodal and trimodal interactions. However, in cases (II) and (III) where the visual modality is either uninformative or contradictory, the efficacies of $v \to l, v$ and $v \to l, a, v$ and $l, a \to l, a, v$ are reduced since no meaningful interactions involve the visual modality.

Priors in Fusion: Certain efficacies remain unchanged across cases and across time. These are priors from Human Multimodal Language that DFG learns. For example the model always seems to prioritize fusion between language and audio in $(l \to l, a)$, and $(a \to l, a)$. Subsequently, DFG gives low values to efficacies that rely unilaterally on language or audio alone: the $(l \to \tau)$ and $(a \to \tau)$ efficacies seem to be consistently low. On the other hand, the visual modality appears to have a partially isolated behavior. In the presence of informative visual information, the model increases the efficacies of $(v \to \tau)$ although the values of other visual efficacies also increase.

Trace of Multimodal Fusion: We trace the dominant path that every modality undergoes during fusion: 1) *language* tends to first fuse with audio via $(l \to l, a)$ and the language and acoustic modalities together engage in higher level fusions such as $(l, a \to l, a, v)$. Intuitively, this is aligned with the close ties between language and audio through word intonations. 2) The *visual* modality seems to engage in fusion only if it contains meaningful information. In cases (I) and (IV), all the paths involving the visual modality are relatively active while in cases (II) and (III) the paths involving the visual modality have low efficacies. 3) The *acoustic* modality is mostly present in fusion with the language modality. However, unlike language, the acoustic modality also appears to fuse with the visual modality if both modalities are meaningful, such as in case (I).

An interesting observation is that in almost all cases the efficacies of unimodal connections to terminal $\mathcal{T}$ is low, implying that $\mathcal{T}$ prefers to not rely on just one modality. Also, DFG always prefers to perform fusion between language and audio as in most cases both $l \to l, a$ and $a \to l, a$ have high efficacies; intuitively in most natural scenarios language and acoustic modalities are highly aligned. Both of these cases show unchanging behaviors which we believe DFG has learned as natural priors of human communicative signal.

With these observations, we believe that DFG has successfully learned how to manage its internal structure to model human communication.

6 Conclusion

In this paper we presented the largest dataset of multimodal sentiment analysis and emotion recognition called CMU Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI). CMU-MOSEI consists of 23,453 annotated sentences from more than 1000 online speakers and 250 different topics. The dataset expands the horizons of Human Multimodal Language studies in NLP. One such study was presented in this paper where we analyzed the structure of multimodal fusion in sentiment analysis and emotion recognition. This was done using a novel interpretable fusion mechanism called Dynamic Fusion Graph (DFG). In our studies we investigated the behavior of modalities in interacting with each other using built-in efficacies of DFG. Aside analysis of fusion, DFG was trained in the Memory Fusion Network pipeline and showed superior performance in sentiment analysis and competitive performance in emotion recognition.

Acknowledgments

This material is based upon work partially supported by the National Science Foundation (Award #1833355) and Oculus VR. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation or Oculus VR, and no official endorsement should be inferred.

References

Paavo Alku, Tom Bäckström, and Erkki Vilkman. 2002. Normalized amplitude quotient for parametrization of the glottal flow. *the Journal of the Acoustical Society of America* 112(2):701–710.

Paavo Alku, Helmer Strik, and Erkki Vilkman. 1997. Parabolic spectral parameter—a new method for quantification of the glottal flow. *Speech Communication* 22(1):67–79.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *arXiv preprint arXiv:1705.09406* .

Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. 2016. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, pages 1–10.

Sanjay Bilakhia, Stavros Petridis, Anton Nijholt, and Maja Pantic. 2015. The mahnob mimicry database: A database of naturalistic human interactions. *Pattern Recognition Letters* 66(Supplement C):52 – 61. Pattern Recognition in Human Computer Interaction. https://doi.org/https://doi.org/10.1016/j.patrec.2015.03.005.

Leo Breiman. 2001. Random forests. *Mach. Learn.* 45(1):5–32. https://doi.org/10.1023/A:1010933404324.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette Chang, Sungbok Lee, and Shrikanth S. Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Journal of Language Resources and Evaluation* 42(4):335–359. https://doi.org/10.1007/s10579-008-9076-6.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI 2017, pages 163–171. https://doi.org/10.1145/3136755.3136801.

Glen Coppersmith and Erin Kelly. 2014. Dynamic wordclouds and vennclouds for exploratory data analysis. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 22–29.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20(3):273–297. https://doi.org/10.1023/A:1022627411411.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarep—a collaborative voice analysis repository for speech technologies. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pages 960–964.

A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. 2012. Collecting large, richly annotated facial-expression databases from movies. *IEEE MultiMedia* 19(3):34–41. https://doi.org/10.1109/MMUL.2012.26.

Abhinav Dhall, O.V. Ramana Murthy, Roland Goecke, Jyoti Joshi, and Tom Gedeon. 2015. Video and image based emotion recognition challenges in the wild: Emotiw 2015. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI '15, pages 423–426. https://doi.org/10.1145/2818346.2829994.

Thomas Drugman and Abeer Alwan. 2011. Joint robust voicing detection and pitch estimation based on residual harmonics. In *Interspeech*. pages 1973–1976.

Thomas Drugman, Mark Thomas, Jon Gudnason, Patrick Naylor, and Thierry Dutoit. 2012. Detection of glottal closure instants from speech signals: A quantitative review. *IEEE Transactions on Audio, Speech, and Language Processing* 20(3):994–1006.

Paul Ekman, Wallace V Freisen, and Sonia Ancoli. 1980. Facial signs of emotional experience. *Journal of personality and social psychology* 39(6):1125.

A. Graves, A. r. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. pages 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947.

Michael Grimm, Kristian Kroschel, and Shrikanth Narayanan. 2008. The vera am mittag german audio-visual emotional speech database. In *ICME*. IEEE, pages 865–868.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmerman. 2018. Memn: Multimodal emotional memory network for emotion recognition in dyadic conversational videos. In *NAACL*.

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

iMotions. 2017. Facial expression analysis. goo.gl/1rh1JN.

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL (1)*. pages 1681–1691.

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3d convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(1):221–231. https://doi.org/10.1109/TPAMI.2012.59.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* .

John Kane and Christer Gobl. 2013. Wavelet maxima dispersion for breathy to tense voice discrimination. *IEEE Transactions on Audio, Speech, and Language Processing* 21(6):1170–1179.

Wootaek Lim, Daeyoung Jang, and Taejin Lee. 2016. Speech emotion recognition using convolutional and recurrent neural networks. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2016 Asia-Pacific*. IEEE, pages 1–4.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 142–150. http://www.aclweb.org/anthology/P11-1015.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interactions*. ACM, pages 169–176.

Friedrich Max Müller. 1866. *Lectures on the science of language: Delivered at the Royal Institution of Great Britain in April, May, & June 1861*, volume 1. Longmans, Green.

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI 2016, pages 284–288. https://doi.org/10.1145/2993148.2993176.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*. pages 79–86.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI '14, pages 50–57. https://doi.org/10.1145/2663204.2663260.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Veronica Perez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Utterance-Level Multimodal Sentiment Analysis. In *Association for Computational Linguistics (ACL)*. Sofia, Bulgaria.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. 2017a. Context dependent sentiment analysis in user generated videos. In *Association for Computational Linguistics*.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *Association for Computational Linguistics*.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, pages 439–448.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrušaitis, and Roland Goecke. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision*.

Fabien Ringeval, Andreas Sonderegger, Jürgen S. Sauer, and Denis Lalanne. 2013. Introducing the recola multimodal corpus of remote collaborative and affective interactions. In *FG*. IEEE Computer Society, pages 1–8.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*. IEEE Computer Society, pages 815–823.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.* 45(11):2673–2681. https://doi.org/10.1109/78.650093.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, volume 1631, page 1642.

Rupesh K Srivastava, Klaus Greff, and Juergen Schmidhuber. 2015. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2377–2385. http://papers.nips.cc/paper/5850-training-very-deep-networks.pdf.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, USA, CVPR '14, pages 1701–1708. https://doi.org/10.1109/CVPR.2014.220.

Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating human trafficking with multimodal deep models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1547–1556.

George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou. 2016. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 5200–5204.

Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. 2016. Select-additive learning: Improving cross-individual generalization in multimodal sentiment analysis. *arXiv preprint arXiv:1609.05244* .

Martin Wöllmer, Felix Weninger, Tobias Knaup, Björn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems* 28(3):46–53.

Jiahong Yuan and Mark Liberman. 2008. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America* 123(5):3878.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Empirical Methods in Natural Language Processing, EMNLP*.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multi-view sequential learning. *arXiv preprint arXiv:1802.00927* .

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018b. Multi-attention recurrent network for human communication comprehension. *arXiv preprint arXiv:1802.00923* .

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016a. Mosi: Multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259* .

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016b. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems* 31(6):82–88.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters* 23(10):1499–1503.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A c-lstm neural network for text classification. *CoRR* abs/1511.08630.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent Highway Networks. *arXiv preprint arXiv:1607.03474* .

Efficient Low-rank Multimodal Fusion with Modality-Specific Factors

Zhun Liu[*], **Ying Shen**[*], **Varun Bharadhwaj Lakshminarasimhan**,
Paul Pu Liang, **Amir Zadeh**, **Louis-Philippe Morency**
School of Computer Science
Carnegie Mellon University
{zhunl,yshen2,vbl,pliang,abagherz,morency}@cs.cmu.edu

Abstract

Multimodal research is an emerging field of artificial intelligence, and one of the main research problems in this field is multimodal fusion. The fusion of multimodal data is the process of integrating multiple unimodal representations into one compact multimodal representation. Previous research in this field has exploited the expressiveness of tensors for multimodal representation. However, these methods often suffer from exponential increase in dimensions and in computational complexity introduced by transformation of input into tensor. In this paper, we propose the Low-rank Multimodal Fusion method, which performs multimodal fusion using low-rank tensors to improve efficiency. We evaluate our model on three different tasks: multimodal sentiment analysis, speaker trait analysis, and emotion recognition. Our model achieves competitive results on all these tasks while drastically reducing computational complexity. Additional experiments also show that our model can perform robustly for a wide range of low-rank settings, and is indeed much more efficient in both training and inference compared to other methods that utilize tensor representations.

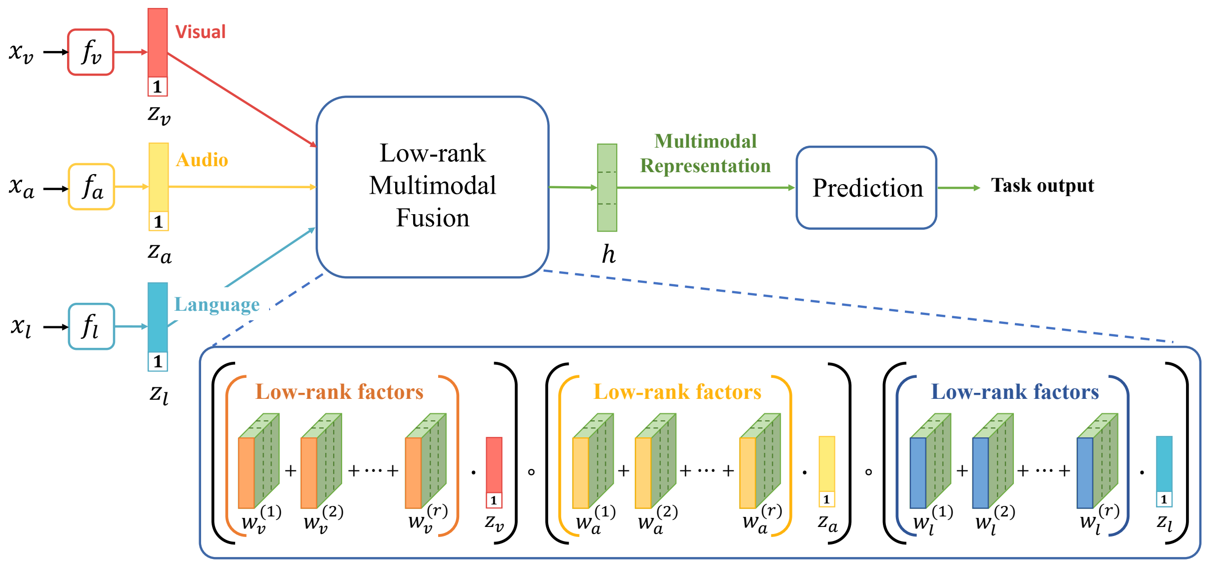
1 Introduction

Multimodal research has shown great progress in a variety of tasks as an emerging research field of artificial intelligence. Tasks such as speech recognition (Yuhas et al., 1989), emotion recognition, (De Silva et al., 1997), (Chen et al., 1998), (Wöllmer et al., 2013), sentiment analysis, (Morency et al., 2011) as well as speaker trait analysis and media description (Park et al., 2014a) have seen a great boost in performance with developments in multimodal research.

However, a core research challenge yet to be solved in this domain is multimodal fusion. The goal of fusion is to combine multiple modalities to leverage the complementarity of heterogeneous data and provide more robust predictions. In this regard, an important challenge has been on scaling up fusion to multiple modalities while maintaining reasonable model complexity. Some of the recent attempts (Fukui et al., 2016), (Zadeh et al., 2017) at multimodal fusion investigate the use of tensors for multimodal representation and show significant improvement in performance. Unfortunately, they are often constrained by the exponential increase of cost in computation and memory introduced by using tensor representations. This heavily restricts the applicability of these models, especially when we have more than two views of modalities in the dataset.

In this paper, we propose the Low-rank Multimodal Fusion, a method leveraging low-rank weight tensors to make multimodal fusion efficient without compromising on performance. The overall architecture is shown in Figure 1. We evaluated our approach with experiments on three multimodal tasks using public datasets and compare its performance with state-of-the-art models. We also study how different low-rank settings impact the performance of our model and show that our model performs robustly within a wide range of rank settings. Finally, we perform an analysis of the impact of our method on the number of parameters and run-time with comparison to other fusion methods. Through theoretical analysis, we show that our model can scale linearly in the number of modalities, and our experiments also show a corresponding speedup in training when compared with

[*] equal contributions



x_v
f_v
Visual
z_v
x_a
f_a
Audio
z_a
x_l
f_l
Language
z_l
Low-rank Multimodal Fusion
h
Multimodal Representation
Prediction
Task output
Low-rank factors
$w_v^{(1)}$ + $w_v^{(2)}$ + ⋯ + $w_v^{(r)}$ · z_v
∘
$w_a^{(1)}$ + $w_a^{(2)}$ + ⋯ + $w_a^{(r)}$ · z_a
$w_l^{(1)}$ + $w_l^{(2)}$ + ⋯ + $w_l^{(r)}$ · z_l
1

Figure 1: Overview of our Low-rank Multimodal Fusion model structure: LMF first obtains the unimodal representation z_a, z_v, z_l by passing the unimodal inputs x_a, x_v, x_l into three sub-embedding networks f_v, f_a, f_l respectively. LMF produces the multimodal output representation by performing low-rank multimodal fusion with modality-specific factors. The multimodal representation can be then used for generating prediction tasks.

other tensor-based models.

The main contributions of our paper are as follows:

- We propose the Low-rank Multimodal Fusion method for multimodal fusion that can scale linearly in the number of modalities.
- We show that our model compares to state-of-the-art models in performance on three multimodal tasks evaluated on public datasets.
- We show that our model is computationally efficient and has fewer parameters in comparison to previous tensor-based methods.

2 Related Work

Multimodal fusion enables us to leverage complementary information present in multimodal data, thus discovering the dependency of information on multiple modalities. Previous studies have shown that more effective fusion methods translate to better performance in models, and there's been a wide range of fusion methods.

Early fusion is a technique that uses feature concatenation as the method of fusion of different views. Several works that use this method of fusion (Poria et al., 2016) , (Wang et al., 2016) use input-level feature concatenation and use the concatenated features as input, sometimes even removing the temporal dependency present in the modalities (Morency et al., 2011). The drawback of this class of method is that although it achieves fusion at an early stage, intra-modal interactions are potentially suppressed, thus losing out on the context and temporal dependencies within each modality.

On the other hand, late fusion builds separate models for each modality and then integrates the outputs together using a method such as majority voting or weighted averaging (Wortwein and Scherer, 2017), (Nojavanasghari et al., 2016). Since separate models are built for each modality, inter-modal interactions are usually not modeled effectively.

Given these shortcomings, more recent work focuses on intermediate approaches that model both intra- and inter-modal dynamics. Fukui et al. (2016) proposes to use Compact Bilinear Pooling over the outer product of visual and linguistic representations to exploit the interactions between vision and language for visual question answering. Similar to the idea of exploiting interactions, Zadeh et al. (2017) proposes Tensor Fusion Network, which computes the outer product between unimodal representations from three different modalities to compute a tensor representation. These methods exploit tensor representations to model inter-modality interactions and have shown a great success. However, such methods suffer from exponentially increasing computational complexity, as the outer product over multiple modalities results in extremely high dimensional tensor representations.

For unimodal data, the method of low-rank tensor approximation has been used in a variety of applications to implement more efficient tensor operations. Razenshteyn et al. (2016) proposes a modified weighted version of low-rank approximation, and Koch and Lubich (2010) applies the method towards temporally dependent data to obtain low-rank approximations. As for applications, Lei et al. (2014) proposes a low-rank tensor technique for dependency parsing while Wang and Ahuja (2008) uses the method of low-rank approximation applied directly on multidimensional image data (Datum-as-is representation) to enhance computer vision applications. Hu et al. (2017) proposes a low-rank tensor-based fusion framework to improve the face recognition performance using the fusion of facial attribute information. However, none of these previous work aims to apply low-rank tensor techniques for multimodal fusion.

Our Low-rank Multimodal Fusion method provides a much more efficient method to compute tensor-based multimodal representations with much fewer parameters and computational complexity. The efficiency and performance of our approach are evaluated on different downstream tasks, namely sentiment analysis, speaker-trait recognition and emotion recognition.

3 Low-rank Multimodal Fusion

In this section, we start by formulating the problem of multimodal fusion and introducing fusion methods based on tensor representations. Tensors are powerful in their expressiveness but do not scale well to a large number of modalities. Our proposed model decomposes the weights into low-rank factors, which reduces the number of parameters in the model. This decomposition can be performed efficiently by exploiting the parallel decomposition of low-rank weight tensor and input tensor to compute tensor-based fusion. Our method is able to scale linearly with the number of modalities.

3.1 Multimodal Fusion using Tensor Representations

In this paper, we formulate multimodal fusion as a multilinear function $f : V_1 \times V_2 \times ... \times V_M \rightarrow H$ where $V_1, V_2, ..., V_M$ are the vector spaces of input modalities and H is the output vector space. Given a set of vector representations, $\{z_m\}_{m=1}^{M}$ which are encoding unimodal information of the M different modalities, the goal of multimodal fusion is to integrate the unimodal representations into one compact multimodal representation for downstream tasks.

Tensor representation is one successful approach for multimodal fusion. It first requires a transformation of the input representations into a high-dimensional tensor and then mapping it back to a lower-dimensional output vector space. Previous works have shown that this method is more effective than simple concatenation or pooling in terms of capturing multimodal interactions (Zadeh et al., 2017), (Fukui et al., 2016). Tensors are usually created by taking the outer product over the input modalities. In addition, in order to be able to model the interactions between any subset of modalities using one tensor, Zadeh et al. (2017) proposed a simple extension to append 1s to the unimodal representations before taking the outer product. The input tensor $\mathcal{Z}$ formed by the unimodal representation is computed by:

$$\mathcal{Z} = \bigotimes_{m=1}^{M} z_m, z_m \in \mathbb{R}^{d_m} \tag{1}$$

where $\bigotimes_{m=1}^{M}$ denotes the tensor outer product over a set of vectors indexed by m, and z_m is the input representation with appended 1s.

The input tensor $\mathcal{Z} \in \mathbb{R}^{d_1 \times d_2 \times ... d_M}$ is then passed through a linear layer $g(\cdot)$ to to produce a vector representation:

$$h = g(\mathcal{Z}; \mathcal{W}, b) = \mathcal{W} \cdot \mathcal{Z} + b, \ h, b \in \mathbb{R}^{d_y} \tag{2}$$

where $\mathcal{W}$ is the weight of this layer and b is the bias. With $\mathcal{Z}$ being an order-M tensor (where M is the number of input modalities), the weight $\mathcal{W}$ will naturally be a tensor of order-$(M+1)$ in $\mathbb{R}^{d_1 \times d_2 \times ... \times d_M \times d_h}$. The extra $(M+1)$-th dimension corresponds to the size of the output representation d_h. In the tensor dot product $\mathcal{W} \cdot \mathcal{Z}$, the weight tensor $\mathcal{W}$ can be then viewed as d_h order-M tensors. In other words, the weight $\mathcal{W}$ can be partitioned into $\widetilde{\mathcal{W}}_k \in \mathbb{R}^{d_1 \times ... \times d_M}$, $k = 1, ..., d_h$. Each $\widetilde{\mathcal{W}}_k$ contributes to one dimension in the output vector h, i.e. $h_k = \widetilde{\mathcal{W}}_k \cdot \mathcal{Z}$. This interpretation of tensor fusion is illustrated in Figure 2 for the bi-modal case.

One of the main drawbacks of tensor fusion is that we have to explicitly create the high-dimensional tensor $\mathcal{Z}$. The dimensionality of $\mathcal{Z}$

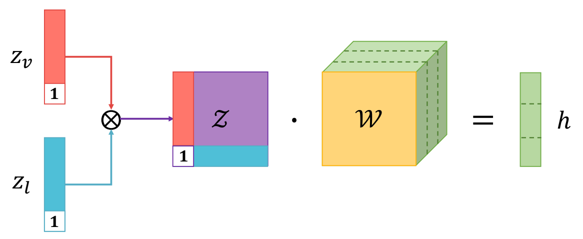


Figure 2: Tensor fusion via tensor outer product

will increase exponentially with the number of modalities as $\prod_{m=1}^{M} d_m$. The number of parameters to learn in the weight tensor $\mathcal{W}$ will also increase exponentially. This not only introduces a lot of computation but also exposes the model to risks of overfitting.

3.2 Low-rank Multimodal Fusion with Modality-Specific Factors

As a solution to the problems of tensor-based fusion, we propose Low-rank Multimodal Fusion (LMF). LMF parameterizes $g(\cdot)$ from Equation 2 with a set of modality-specific low-rank factors that can be used to recover a low-rank weight tensor, in contrast to the full tensor $\mathcal{W}$. Moreover, we show that by decomposing the weight into a set of low-rank factors, we can exploit the fact that the tensor $\mathcal{Z}$ actually decomposes into $\{z_m\}_{m=1}^{M}$, which allows us to directly compute the output h without explicitly tensorizing the unimodal representations. LMF reduces the number of parameters as well as the computation complexity involved in tensorization from being exponential in M to linear.

3.2.1 Low-rank Weight Decomposition

The idea of LMF is to decompose the weight tensor $\mathcal{W}$ into M sets of modality-specific factors. However, since $\mathcal{W}$ itself is an order-$(M+1)$ tensor, commonly used methods for decomposition will result in $M+1$ parts. Hence, we still adopt the view introduced in Section 3.1 that $\mathcal{W}$ is formed by d_h order-M tensors $\widetilde{\mathcal{W}}_k \in \mathbb{R}^{d_1 \times \ldots \times d_M}, k = 1, \ldots, d_h$ stacked together. We can then decompose each $\widetilde{\mathcal{W}}_k$ separately.

For an order-M tensor $\widetilde{\mathcal{W}}_k \in \mathbb{R}^{d_1 \times \ldots \times d_M}$, there always exists an exact decomposition into vectors in the form of:

$$\widetilde{\mathcal{W}}_k = \sum_{i=1}^{R} \bigotimes_{m=1}^{M} w_{m,k}^{(i)}, \quad w_{m,k}^{(i)} \in \mathbb{R}_m^{d} \tag{3}$$

The minimal R that makes the decomposition valid is called the **rank** of the tensor. The vector sets $\{\{w_{m,k}^{(i)}\}_{m=1}^{M}\}_{i=1}^{R}$ are called the rank R decomposition factors of the original tensor.

In LMF, we start with a fixed rank r, and parameterize the model with r decomposition factors $\{\{w_{m,k}^{(i)}\}_{m=1}^{M}\}_{i=1}^{r}, k = 1, \ldots, d_h$ that can be used to reconstruct a low-rank version of these $\widetilde{\mathcal{W}}_k$.

We can regroup and concatenate these vectors into M modality-specific low-rank factors. Let $\mathbf{w}_m^{(i)} = [w_{m,1}^{(i)}, w_{m,2}^{(i)}, \ldots, w_{m,d_h}^{(i)}]$, then for modality m, $\{\mathbf{w}_m^{(i)}\}_{i=1}^{r}$ is its corresponding low-rank factors. And we can recover a low-rank weight tensor by:

$$\mathcal{W} = \sum_{i=1}^{r} \bigotimes_{m=1}^{M} \mathbf{w}_m^{(i)} \tag{4}$$

Hence equation 2 can be computed by

$$h = \left(\sum_{i=1}^{r} \bigotimes_{m=1}^{M} \mathbf{w}_m^{(i)} \right) \cdot \mathcal{Z} \tag{5}$$

Note that for all m, $\mathbf{w}_m^{(i)} \in \mathbb{R}^{d_m \times d_h}$ shares the same size for the second dimension. We define their outer product to be over only the dimensions that are not shared: $\mathbf{w}_m^{(i)} \otimes \mathbf{w}_n^{(i)} \in \mathbb{R}^{d_m \times d_n \times d_h}$. A bimodal example of this procedure is illustrated in Figure 3.

Nevertheless, by introducing the low-rank factors, we now have to compute the reconstruction of $\mathcal{W} = \sum_{i=1}^{r} \bigotimes_{m=1}^{M} \mathbf{w}_m^{(i)}$ for the forward computation. Yet this introduces even more computation.

3.2.2 Efficient Low-rank Fusion Exploiting Parallel Decomposition

In this section, we will introduce an efficient procedure for computing h, exploiting the fact that tensor $\mathcal{Z}$ naturally decomposes into the original input $\{z_m\}_{m=1}^{M}$, which is parallel to the modality-specific low-rank factors. In fact, that is the main reason why we want to decompose the weight tensor into M modality-specific factors.

Using the fact that $\mathcal{Z} = \bigotimes_{m=1}^{M} z_m$, we can simplify equation 5:

$$\begin{aligned} h &= \left(\sum_{i=1}^{r} \bigotimes_{m=1}^{M} \mathbf{w}_m^{(i)} \right) \cdot \mathcal{Z} \\ &= \sum_{i=1}^{r} \left(\bigotimes_{m=1}^{M} \mathbf{w}_m^{(i)} \cdot \mathcal{Z} \right) \\ &= \sum_{i=1}^{r} \left(\bigotimes_{m=1}^{M} \mathbf{w}_m^{(i)} \cdot \bigotimes_{m=1}^{M} z_m \right) \\ &= \bigwedge_{m=1}^{M} \left[\sum_{i=1}^{r} \mathbf{w}_m^{(i)} \cdot z_m \right] \end{aligned} \tag{6}$$

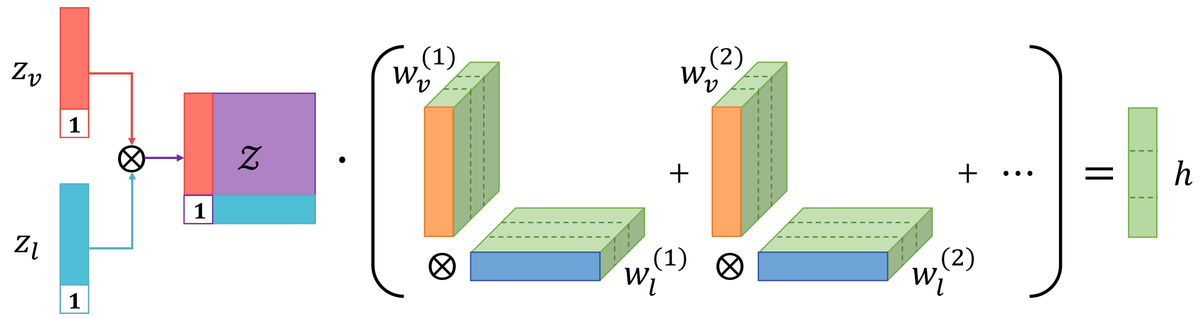


Figure 3: Decomposing weight tensor into low-rank factors (See Section 3.2.1 for details.)

where $\bigwedge_{m=1}^{M}$ denotes the element-wise product over a sequence of tensors: $\bigwedge_{t=1}^{3} x_t = x_1 \circ x_2 \circ x_3$.

An illustration of the trimodal case of equation 6 is shown in Figure 1. We can also derive equation 6 for a bimodal case to clarify what it does:

$$\begin{aligned} h &= \left(\sum_{i=1}^{r} \mathbf{w}_a^{(i)} \otimes \mathbf{w}_v^{(i)} \right) \cdot \mathcal{Z} \\ &= \left(\sum_{i=1}^{r} \mathbf{w}_a^{(i)} \cdot z_a \right) \circ \left(\sum_{i=1}^{r} \mathbf{w}_v^{(i)} \cdot z_v \right) \end{aligned} \tag{7}$$

An important aspect of this simplification is that it exploits the parallel decomposition of both $\mathcal{Z}$ and $\mathcal{W}$, so that we can compute h without actually creating the tensor $\mathcal{Z}$ from the input representations z_m. In addition, different modalities are decoupled in the simplified computation of h, which allows for easy generalization of our approach to an arbitrary number of modalities. Adding a new modality can be simply done by adding another set of modality-specific factors and extend Equation 7. Last but not least, Equation 6 consists of fully differentiable operations, which enables the parameters $\{\mathbf{w}_m^{(i)}\}_{i=1}^{r}\, m = 1, ..., M$ to be learned end-to-end via back-propagation.

Using Equation 6, we can compute h directly from input unimodal representations and their modal-specific decomposition factors, avoiding the weight-lifting of computing the large input tensor $\mathcal{Z}$ and $\mathcal{W}$, as well as the r linear transformation. Instead, the input tensor and subsequent linear projection are computed implicitly together in Equation 6, and this is far more efficient than the original method described in Section 3.1. Indeed, LMF reduces the computation complexity of tensorization and fusion from $O(d_y \prod_{m=1}^{M} d_m)$ to $O(d_y \times r \times \sum_{m=1}^{M} d_m)$.

In practice, we use a slightly different form of Equation 6, where we concatenate the low-rank factors into M order-3 tensors and swap the order in which we do the element-wise product and summation:

$$h = \sum_{i=1}^{r} \left[\bigwedge_{m=1}^{M} \left[\mathbf{w}_m^{(1)}, \mathbf{w}_m^{(2)}, ..., \mathbf{w}_m^{(r)} \right] \cdot \hat{z}_m \right]_{i,:} \tag{8}$$

and now the summation is done along the first dimension of the bracketed matrix. $[\cdot]_{i,:}$ indicates the i-th slice of a matrix. In this way, we can parameterize the model with M order-3 tensors, instead of parameterizing with sets of vectors.

4 Experimental Methodology

We compare LMF with previous state-of-the-art baselines, and we use the Tensor Fusion Networks (TFN) (Zadeh et al., 2017) as a baseline for tensor-based approaches, which has the most similar structure with us except that it explicitly forms the large multi-dimensional tensor for fusion across different modalities.

We design our experiments to better understand the characteristics of LMF. Our goal is to answer the following four research questions:

(1) Impact of Multimodal Low-rank Fusion: Direct comparison between our proposed LMF model and the previous TFN model.

(2) Comparison with the State-of-the-art: We evaluate the performance of LMF and state-of-the-art baselines on three different tasks and datasets.

(3) Complexity Analysis: We study the modal complexity of LMF and compare it with the TFN model.

(4) Rank Settings: We explore performance of LMF with different rank settings.

The results of these experiments are presented in Section 5.

4.1 Datasets

We perform our experiments on the following multimodal datasets, CMU-MOSI (Zadeh et al., 2016a),

Dataset	CMU-MOSI	IEMOCAP	POM
Level	Segment	Segment	Video
# Train	1284	6373	600
# Valid	229	1775	100
# Test	686	1807	203

Table 1: The speaker independent data splits for training, validation, and test sets.

POM (Park et al., 2014b), and IEMOCAP (Busso et al., 2008) for sentiment analysis, speaker traits recognition, and emotion recognition task, where the goal is to identify speakers emotions based on the speakers' verbal and nonverbal behaviors.

CMU-MOSI The CMU-MOSI dataset is a collection of 93 opinion videos from YouTube movie reviews. Each video consists of multiple opinion segments and each segment is annotated with the sentiment in the range [-3,3], where -3 indicates highly negative and 3 indicates highly positive.

POM The POM dataset is composed of 903 movie review videos. Each video is annotated with the following speaker traits: confident, passionate, voice pleasant, dominant, credible, vivid, expertise, entertaining, reserved, trusting, relaxed, outgoing, thorough, nervous, persuasive and humorous.

IEMOCAP The IEMOCAP dataset is a collection of 151 videos of recorded dialogues, with 2 speakers per session for a total of 302 videos across the dataset. Each segment is annotated for the presence of 9 emotions (angry, excited, fear, sad, surprised, frustrated, happy, disappointed and neutral).

To evaluate model generalization, all datasets are split into training, validation, and test sets such that the splits are speaker independent, i.e., no identical speakers from the training set are present in the test sets. Table 1 illustrates the data splits for all datasets in detail.

4.2 Features

Each dataset consists of three modalities, namely language, visual, and acoustic modalities. To reach the same time alignment across modalities, we perform word alignment using P2FA (Yuan and Liberman, 2008) which allows us to align the three modalities at the word granularity. We calculate the visual and acoustic features by taking the average of their feature values over the word time interval (Chen et al., 2017).

Language We use pre-trained 300-dimensional Glove word embeddings (Pennington et al., 2014) to encode a sequence of transcribed words into a sequence of word vectors.

Visual The library Facet[1] is used to extract a set of visual features for each frame (sampled at 30Hz) including 20 facial action units, 68 facial landmarks, head pose, gaze tracking and HOG features (Zhu et al., 2006).

Acoustic We use COVAREP acoustic analysis framework (Degottex et al., 2014) to extract a set of low-level acoustic features, including 12 Mel frequency cepstral coefficients (MFCCs), pitch, voiced/unvoiced segmentation, glottal source, peak slope, and maxima dispersion quotient features.

4.3 Model Architecture

In order to compare our fusion method with previous work, we adopt a simple and straightforward model architecture [2] for extracting unimodal representations. Since we have three modalities for each dataset, we simply designed three unimodal sub-embedding networks, denoted as f_a, f_v, f_l, to extract unimodal representations z_a, z_v, z_l from unimodal input features x_a, x_v, x_l. For acoustic and visual modality, the sub-embedding network is a simple 2-layer feed-forward neural network, and for language modality, we used an LSTM (Hochreiter and Schmidhuber, 1997) to extract representations. The model architecture is illustrated in Figure 1.

4.4 Baseline Models

We compare the performance of LMF to the following baselines and state-of-the-art models in multimodal sentiment analysis, speaker trait recognition, and emotion recognition.

Support Vector Machines Support Vector Machines (SVM) (Cortes and Vapnik, 1995) is a widely used non-neural classifier. This baseline is trained on the concatenated multimodal features for classification or regression task (Pérez-Rosas et al., 2013), (Park et al., 2014a), (Zadeh et al., 2016b).

Deep Fusion The Deep Fusion model (DF) (Nojavanasghari et al., 2016) trains one deep neural model for each modality and then combine the output of each modality network with a joint neural network.

Tensor Fusion Network The Tensor Fusion Network (TFN) (Zadeh et al., 2017) explicitly models view-specific and cross-view dynamics by creating a multi-dimensional tensor that captures uni-

[1]goo.gl/1rh1JN

[2]The source code of our model is available on Github at https://github.com/Justin1904/Low-rank-Multimodal-Fusion

modal, bimodal and trimodal interactions across three modalities.

Memory Fusion Network The Memory Fusion Network (MFN) (Zadeh et al., 2018a) accounts for view-specific and cross-view interactions and continuously models them through time with a special attention mechanism and summarized through time with a Multi-view Gated Memory.

Bidirectional Contextual LSTM The Bidirectional Contextual LSTM (BC-LSTM) (Zadeh et al., 2017), (Fukui et al., 2016) performs context-dependent fusion of multimodal data.

Multi-View LSTM The Multi-View LSTM (MV-LSTM) (Rajagopalan et al., 2016) aims to capture both modality-specific and cross-modality interactions from multiple modalities by partitioning the memory cell and the gates corresponding to multiple modalities.

Multi-attention Recurrent Network The Multi-attention Recurrent Network (MARN) (Zadeh et al., 2018b) explicitly models interactions between modalities through time using a neural component called the Multi-attention Block (MAB) and storing them in the hybrid memory called the Long-short Term Hybrid Memory (LSTHM).

4.5 Evaluation Metrics

Multiple evaluation tasks are performed during our evaluation: multi-class classification and regression. The multi-class classification task is applied to all three multimodal datasets, and the regression task is applied to the CMU-MOSI and the POM dataset. For binary classification and multi-class classification, we report F1 score and accuracy Acc$-k$ where k denotes the number of classes. Specifically, Acc-2 stands for the binary classification. For regression, we report Mean Absolute Error (MAE) and Pearson correlation (Corr). Higher values denote better performance for all metrics except for MAE.

5 Results and Discussion

In this section, we present and discuss the results from the experiments designed to study the research questions introduced in section 4.

5.1 Impact of Low-rank Multimodal Fusion

In this experiment, we compare our model directly with the TFN model since it has the most similar structure to our model, except that TFN explicitly forms the multimodal tensor fusion. The comparison reported in the last two rows of Table 2 demonstrates that our model significantly outperforms TFN across all datasets and metrics. This competitive performance of LMF compared to TFN emphasizes the advantage of Low-rank Multimodal Fusion.

5.2 Comparison with the State-of-the-art

We compare our model with the baselines and state-of-the-art models for sentiment analysis, speaker traits recognition and emotion recognition. Results are shown in Table 2. LMF is able to achieve competitive and consistent results across all datasets.

On the multimodal sentiment regression task, LMF outperforms the previous state-of-the-art model on MAE and Corr. Note the multiclass accuracy is calculated by mapping the range of continuous sentiment values into a set of intervals that are used as discrete classes.

On the multimodal speaker traits Recognition task, we report the average evaluation score over 16 speaker traits and shows that our model achieves the state-of-the-art performance over all three evaluation metrics on the POM dataset.

On the multimodal emotion recognition task, our model achieves better results compared to the state-of-the-art models across all emotions on the F1 score. F1-emotion in the evaluation metrics indicates the F1 score for a certain emotion class.

5.3 Complexity Analysis

Theoretically, the model complexity of our fusion method is $O(d_y \times r \times \sum_{m=1}^{M} d_m)$ compared to $O(d_y \prod_{m=1}^{M} d_m)$ of TFN from Section 3.1. In practice, we calculate the total number of parameters used in each model, where we choose $M = 3$, $d_1 = 32$, $d_2 = 32$, $d_3 = 64$, $r = 4$, $d_y = 1$. Under this hyper-parameter setting, our model contains about 1.1e6 parameters while TFN contains about 12.5e6 parameters, which is nearly 11 times more. Note that, the number of parameters above counts not only the parameters in the multimodal fusion stage but also the parameters in the subnetworks.

Furthermore, we evaluate the computational complexity of LMF by measuring the training and testing speeds between LMF and TFN. Table 3 illustrates the impact of Low-rank Multimodal Fusion on the training and testing speeds compared with TFN model. Here we set rank to be 4 since it can generally achieve fairly competent performance.

Dataset	CMU-MOSI					POM			IEMOCAP			
Metric	MAE	Corr	Acc-2	F1	Acc-7	MAE	Corr	Acc	F1-Happy	F1-Sad	F1-Angry	F1-Neutral
SVM	1.864	0.057	50.2	50.1	17.5	0.887	0.104	33.9	81.5	78.8	82.4	64.9
DF	1.143	0.518	72.3	72.1	26.8	0.869	0.144	34.1	81.0	81.2	65.4	44.0
BC-LSTM	1.079	0.581	73.9	73.9	28.7	0.840	0.278	34.8	81.7	81.7	84.2	64.1
MV-LSTM	1.019	0.601	73.9	74.0	33.2	0.891	0.270	34.6	81.3	74.0	84.3	66.7
MARN	0.968	0.625	77.1	77.0	34.7	-	-	39.4	83.6	81.2	84.2	65.9
MFN	0.965	0.632	**77.4**	**77.3**	**34.1**	0.805	0.349	41.7	84.0	82.1	83.7	69.2
TFN	0.970	0.633	73.9	73.4	32.1	0.886	0.093	31.6	83.6	82.8	84.2	65.4
LMF	**0.912**	**0.668**	76.4	75.7	32.8	**0.796**	**0.396**	**42.8**	**85.8**	**85.9**	**89.0**	**71.7**

Table 2: Results for sentiment analysis on CMU-MOSI, emotion recognition on IEMOCAP and personality trait recognition on POM. Best results are highlighted in bold.

Model	Training Speed (IPS)	Testing Speed (IPS)
TFN	340.74	1177.17
LMF	1134.82	2249.90

Table 3: Comparison of the training and testing speeds between TFN and LMF. The second and the third columns indicate the number of data point inferences per second (IPS) during training and testing time respectively. Both models are implemented in the same framework with equivalent running environment.

Based on these results, performing a low-rank multimodal fusion with modality-specific low-rank factors significantly reduces the amount of time needed for training and testing the model. On an NVIDIA Quadro K4200 GPU, LMF trains with an average frequency of 1134.82 IPS (data point inferences per second) while the TFN model trains at an average of 340.74 IPS.

5.4 Rank Settings

To evaluate the impact of different rank settings for our LMF model, we measure the change in performance on the CMU-MOSI dataset while varying the number of rank. The results are presented in Figure 4. We observed that as the rank increases, the training results become more and more unstable and that using a very low rank is enough to achieve fairly competent performance.

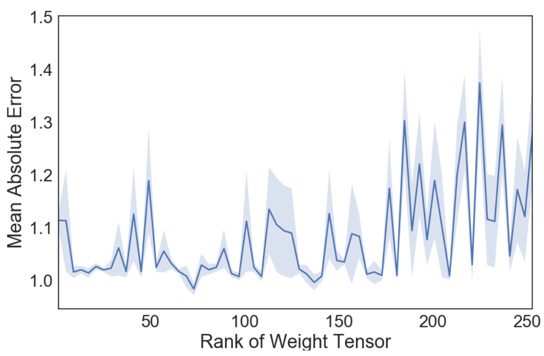


Figure 4: The Impact of different rank settings on Model Performance: As the rank increases, the results become unstable and low rank is enough in terms of the mean absolute error.

6 Conclusion

In this paper, we introduce a Low-rank Multimodal Fusion method that performs multimodal fusion with modality-specific low-rank factors. LMF scales linearly in the number of modalities. LMF achieves competitive results across different multimodal tasks. Furthermore, LMF demonstrates a significant decrease in computational complexity from exponential to linear time. In practice, LMF effectively improves the training and testing efficiency compared to TFN which performs multimodal fusion with tensor representations.

Future work on similar topics could explore the applications of using low-rank tensors for attention models over tensor representations, as they can be even more memory and computationally intensive.

Acknowledgements

This material is based upon work partially supported by the National Science Foundation (Award # 1833355) and Oculus VR. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation or Oculus VR, and no official endorsement should be inferred.

References

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette Chang, Sungbok Lee, and Shrikanth S. Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Journal of Lan-*

guage Resources and Evaluation 42(4):335–359. https://doi.org/10.1007/s10579-008-9076-6.

Lawrence S Chen, Thomas S Huang, Tsutomu Miyasato, and Ryohei Nakatsu. 1998. Multimodal human emotion/expression recognition. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. IEEE, pages 366–371.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI 2017, pages 163–171. https://doi.org/10.1145/3136755.3136801.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Liyanage C De Silva, Tsutomu Miyasato, and Ryohei Nakatsu. 1997. Facial emotion recognition using multi-modal information. In *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on*. IEEE, volume 1, pages 397–401.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarepa collaborative voice analysis repository for speech technologies. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pages 960–964.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Guosheng Hu, Yang Hua, Yang Yuan, Zhihong Zhang, Zheng Lu, Sankha S Mukherjee, Timothy M Hospedales, Neil M Robertson, and Yongxin Yang. 2017. Attribute-enhanced face recognition with neural tensor fusion networks.

Othmar Koch and Christian Lubich. 2010. Dynamical tensor approximation. *SIAM Journal on Matrix Analysis and Applications* 31(5):2360–2375.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1381–1391.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interactions*. ACM, pages 169–176.

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, pages 284–288.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014a. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, pages 50–57.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014b. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI '14, pages 50–57. https://doi.org/10.1145/2663204.2663260.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation.

Verónica Pérez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Utterance-level multimodal sentiment analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 973–982.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, pages 439–448.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrušaitis, and Roland Goecke. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision*.

Ilya Razenshteyn, Zhao Song, and David P Woodruff. 2016. Weighted low rank approximations with provable guarantees. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM, pages 250–263.

Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. 2016. Select-additive learning: Improving cross-individual generalization in multimodal sentiment analysis. *arXiv preprint arXiv:1609.05244* .

Hongcheng Wang and Narendra Ahuja. 2008. A tensor approximation approach to dimensionality reduction. *International Journal of Computer Vision* 76(3):217–229.

Martin Wöllmer, Felix Weninger, Tobias Knaup, Björn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems* 28(3):46–53.

Torsten Wortwein and Stefan Scherer. 2017. What really mattersan information gain analysis of questions and reactions in automated ptsd screenings. In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, pages 15–20.

Jiahong Yuan and Mark Liberman. 2008. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America* 123(5):3878.

Ben P Yuhas, Moise H Goldstein, and Terrence J Sejnowski. 1989. Integration of acoustic and visual speech signals using neural networks. *IEEE Communications Magazine* 27(11):65–71.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Empirical Methods in Natural Language Processing, EMNLP*.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018a. Memory fusion network for multi-view sequential learning. *arXiv preprint arXiv:1802.00927* .

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018b. Multi-attention recurrent network for human communication comprehension. *arXiv preprint arXiv:1802.00923* .

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016a. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259* .

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016b. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems* 31(6):82–88.

Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. 2006. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. IEEE, volume 2, pages 1491–1498.

Discourse Coherence: Concurrent Explicit and Implicit Relations

Hannah Rohde[1] **Alexander Johnson**[1] **Nathan Schneider**[2] **Bonnie Webber**[1]
[1] University of Edinburgh [2] Georgetown University
{Hannah.Rohde, Bonnie.Webber}@ed.ac.uk, s1668719@ed-alumni.net, nathan.schneider@georgetown.edu

Abstract

Theories of discourse coherence posit relations between discourse segments as a key feature of coherent text. Our prior work suggests that multiple discourse relations can be simultaneously operative between two segments for reasons not predicted by the literature. Here we test how this joint presence can lead participants to endorse seemingly divergent conjunctions (e.g., *but* and *so*) to express the link they see between two segments. These apparent divergences are not symptomatic of participant naïveté or bias, but arise reliably from the concurrent availability of multiple relations between segments – some available through explicit signals and some via inference. We believe that these new results can both inform future progress in theoretical work on discourse coherence and lead to higher levels of performance in discourse parsing.

1 Introduction

A question that remains unresolved in work on discourse coherence is the nature and number of relations that can hold between clauses in a coherent text (Halliday and Hasan, 1976; Stede, 2012).

Our earlier work (Rohde et al., 2015, 2016) showed that, in the presence of explicit discourse adverbials, people also infer additional discourse relations that they take to hold jointly with those associated with the adverbials. For example, in:

(1) It's too far to walk. Instead let's take the bus.

people infer a RESULT relation in the context of the adverbial *instead*, which itself signals that the bus stands in a SUBSTITUTION relation to walking. We showed this using crowdsourced conjunction-insertion experiments (Rohde et al., 2015, 2016), in which participants were asked to insert into the gap between two discourse segments, a *conjunction* that best expressed how they took the segments to be related. Rohde et al. (2017) also asked participants to select any other conjunctions that they took to convey the same sense as their "best" choice. (More details of these experiments are given in Section 3.)

All three studies showed participants selecting conjunctions whose sense differed from that of the explicit discourse adverbial. But Rohde et al. (2015, 2016) also showed participants often selecting conjunctions that signal different coherence relations than those selected by other participants. And Rohde et al. (2017) showed participants often identifying very different conjunctions as conveying the same meaning. For example, in passage (2), with the discourse adverbial *in other words*, one large fraction of participants chose to insert OR, while another large fraction inserted SO. Since the two are neither synonymous nor representative of the same relation, either the participants have come up with different analyses of the passages (Section 2) or something more surprising is at work.

(2) Unfortunately, nearly 75,000 acres of tropical forest are converted or deforested every day ______ in other words an area the size of Central Park disappears every 16 minutes. [SO$\sim$OR]

Rohde et al. (2017) noted other cases where different pairs of conjunctions (e.g., BECAUSE and BUT, BUT and OR, and BECAUSE and OR) appear systematically across participants and across passages for particular adverbials, and speculated on what these odd pairings may reveal, but did not provide any empirical evidence for why this happens. Here we present such evidence from an experiment on three discourse adverbials (*in other words*, *otherwise*, and *instead*).

After describing related work on multiple discourse relations (Section 2) and then our experimental methodology (Section 3), we step through results for these three adverbials. As a final piece of evidence, we manipulate the presence and absence of a fourth adverbial, *after all*, in order to demonstrate that inference of the relation(s) between segments in a passage is not always driven by the presence of such an adverbial.

2 Related Work

This is not the first work on discourse coherence to acknowledge the possibility of multiple relations holding between given discourse segments.

For example, the developers of Rhetorical Structure Theory acknowledged that even experienced RST analysts may interpret a text differently in terms of the relations they take to hold (Mann and Thompson, 1988, p. 265). But while RST allows for multiple **alternative analyses** of a text in terms of discourse relations, in practice, researchers working in the RST framework standardly produce a single analysis of a text, with a single relational labeling, selecting the analysis that is "most plausible in terms of the perceived goals of the writer" (Mann et al., 1989, pp. 34–35). If that single analysis is later mapped into a different structure to support further processing – e.g., a binary branching tree structure – the mapping does not change the chosen relational labeling.

Multiple relations may additionally hold in theories of discourse coherence that posit multiple *levels* of text analysis. For example, following Grosz and Sidner (1986), Moore and Pollack (1992) characterized text as having both an *informational structure* (relating information conveyed by discourse segments) and an *intentional structure* (relating the functions of those segments with respect to what the speaker is trying to accomplish through the text). The kinds of relations at the two levels are different, as can be seen in the following example from (Moore and Pollack, 1992, p. 540):

(3) a. George Bush supports big business.
b. He's sure to veto House Bill 1711.

At the level of intentions, (3a) aims to provide EVIDENCE for the claim in (3b), while at an informational level, (3a) serves as the CAUSE of the situation in (3b). RST would force annotators to choose only the analysis that best reflected the perceived goals of the writer.

Additionally, multiple relations can hold where there are **distinct explicit signals** for distinct discourse relations holding between a pair of segments (Cuenca and Marin, 2009; Fraser, 2013), as in:

(4) It's too far to walk. So instead let's take the bus.

where the conjunction *so* signals a RESULT relation and the adverbial *instead* signals that taking the bus stands in an SUBSTITUTION relation to walking.

Finally, a fourth way in which the previous literature has taken multiple discourse relations to hold is when a single phrase or lexico-syntactic construction **jointly signals** multiple discourse relations as holding over a text – for example, *since* as a subordinating conjunction may, in particular contexts, signal both a TEMPORAL relation and a CAUSAL relation, rather than just one or the other (Miltsakaki et al., 2005).

We are aware of only two resources that allow more than one discourse relation to be annotated between two segments – the Penn Discourse TreeBank (PDTB; Prasad et al., 2008, 2014) and, more recently, the BECauSE Corpus 2.0 (Dunietz et al., 2017). The PDTB allows multiple discourse relations of the third and fourth types noted above. It also allows them to be annotated if there is no explicit connective between a pair of segments but annotators see more than one sense relation as linking them, as in the following variant of (4):

(5) It's too far to walk. Let's take the bus.

Here a RESULT relation can be associated with an implicit token of *so* between the clauses, while a SUBSTITUTION relation can be associated with an implicit token of *instead*. The above are the main cases in which PDTB annotates multiple relations. Relevant to this paper, the PDTB does not annotate implicit conjunction relations where there is already an explicit discourse adverbial. Thus the PDTB would either ignore the implicit RESULT relation for (1) or (incorrectly) annotate *instead* in (1) as conveying both SUBSTITUTION and RESULT.

Moreover, while the PDTB has been used in training many (but not all) discourse parsers (Marcu, 2000; Lin et al., 2014; Feng and Hirst, 2012; Xue et al., 2015, 2016; Ji and Eisenstein, 2014), discourse parsing has for the most part ignored its annotations of multiple concurrent relations between clauses, except in the case of distinct explicit connectives expressing distinct relations. Instead, they have arbitrarily taken just a single relation to hold, even though the relations are simply recorded in an *a priori* canonical order. This practice is problematic because, for example, there may well be a difference in the properties of segments where two relations are jointly seen to hold, versus those segments in which only one or the other holds. This can result in unwanted noise in the data and lower the reliability of whatever is induced.

While our previous studies showed another source of multiple discourse relations holding concurrently between discourse segments, the work reported here explains how, in the context of multiple relations, participants can take very different conjunctions to be conveying the same relation, and what can change participants' selection of a conjunction to mark the relation they infer alongside that conveyed by an explicit discourse adverbial.

3 Methodology

A locally crowdsourced conjunction-insertion task provided a proxy for labelling relations between adjacent discourse segments within a passage.

Our materials consisted of passages containing an explicit discourse adverbial, preceded by a gap, which effectively separated the passage into two segments. The passages consisted of 16 with *in other words*, 16 with *instead*, 16 with *after all*, and 48 with *otherwise*. Participants were asked to read each passage and choose the conjunction(s) that best expressed how the two segments link together. The presentation of conjunction choices varied in order for each participant, but always consisted of AND, BECAUSE, BUT, OR, SO, NONE. While the task admittedly encourages participants to select one (or more) conjunctions, our prior work has shown that participants are very willing to use NONE if no conjunction is appropriate. We therefore take their insertion of a conjunction as their endorsement of the relation signaled by that conjunction. To further control data quality, we included 6 catch trials with an expected correct conjunction like "To be _______ not to be".

Three of the explicit discourse adverbials that we chose are *anaphoric*: *in other words*, *otherwise*, and *instead* (Webber et al., 2000). Unlike conjunctions such as AND, BECAUSE, BUT, OR and SO, they are not constrained by structure as to what they establish discourse relations with. So a conjunction-insertion task can be used to assess links between the segments (see also Scholman and Demberg 2017). Our three anaphoric adverbials share a core meaning of 'otherness' via their lexical semantics and flexibility in the relations they can participate in, making them a fruitful set to compare. The fourth adverbial, *after all*, allows us to test a hypothesis that the inferred connection between clauses is not driven by the adverbial alone.

These particular adverbials were selected because they had yielded unexpected combinations of conjunction insertions in our prior work (e.g., OR/SO with *in other words*). This is in contrast to adverbials like *therefore* and *nevertheless*, for which participants' conjunction combinations could be attributed to variation in the specificity of the conjunctions (SO/AND for *therefore*, BUT/AND for *nevertheless*). For our selection of a set of conjunctions to use as proxies for relation labels, we included all the coordinating conjunctions in English, as well as the subordinating conjunction BECAUSE as EXPLANATION relations are frequent.

All participants (N=28) were monolingual native English speakers who were selected following a pre-test to measure their ability to consistently insert conjunctions that captured the underlying coherence relations in a series of passages. All gave informed consent. They each received £50 for their time. Each participant saw one of two randomly ordered lists. Passages were presented in batches of 34, one batch per day for three days.

The materials were simplified variants of naturally occurring passages. Some were also manipulated systematically, in ways aimed at altering the availability of different coherence relations. Passages are available via the "dataset" link on the paper in the ACL anthology, and predictions about them are laid out in Sections 4.1–4.4.

4 Datasets

4.1 *In other words* Dataset

Rohde et al. (2016) report an OR∼SO response split for *in other words* when participants could insert only their top choice of conjunction. Figure 1 shows SO dominating participants' choice in all cases, but OR showing up among their choices in all but one passage (leftmost vertical bar). Additionally, several passages elicited BUT as the top choice of some participants.

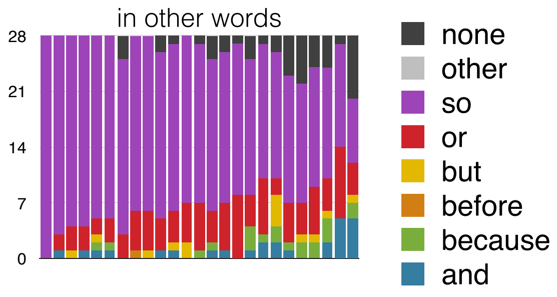


Figure 1: Stacked bar chart for conjunction insertions in passages with *in other words* (Rohde et al., 2016). Each vertical bar represents a passage with one response from each participant (N=28, no overlap with current participants).

The *in other words* passages of the current experiment tested two linked hypotheses: The first is that OR∼SO response splits arise from two components of the lexical semantics of the adverbial itself: its sense of an evoked alternative and its sense of a consequence via restatement, whereby the truth of the second segment holds because it provides a reformulated restatement of the first segment's content. For passage (2), this corresponds to the deforestation of 75,000 acres of tropical forest entailing the disappearance of an area the size of Central Park every 16 minutes.

The second hypothesis is that the prevalence of and substitutability between SO and OR in (2) depends on the immediately adjacency of the two segments. This was suggested by participant choices of BUT (cf. Figure 1), as well as the observation that *in other words* does not always license OR via its lexical semantics and SO via entailment, as shown in (6), where BUT has become more available. Note that none of the relations conveyed by these conjunctions (CONTRAST or CONCESSION for BUT, DISJUNCTION for OR, CONSEQUENCE for SO) are already conveyed by the adverbial itself, which for *in other words*) would be RESTATEMENT.

(6) Unfortunately, nearly 75,000 acres of tropical forest are converted or deforested every day. *I don't know where I heard that* ______ in other words an area the size of Central Park disappears every 16 minutes.

We tested these hypotheses by creating minimal pairs of 16 passages containing *in other words*. The pairs varied in the presence/absence of a metalinguistic comment intervening between the original description and its reformulation, as in (7)–(8).

(7) Typically, a cast-iron wood-burning stove is 60 percent efficient ______ in other words 40 percent of the wood ends up as ash, smoke or lost heat.

(8) Typically, a cast-iron wood-burning stove is 60 percent efficient. *How this is measured is unclear* ______ in other words 40 percent of the wood ends up as ash, smoke or lost heat.

For each passage, participants identified their preferred conjunction and then any others that they took to convey the same sense. Half the participants saw a given passage with no intervening metalinguistic comment, half with.

If our hypotheses are confirmed, it will show that manipulating the immediately preceding segment can shift participants' preference from relations associated with OR and SO (ALTERNATIVE and CONSEQUENCE) to relations of CONTRAST or CONCESSION. This would then be evidence that adjacency affects what coherence relations participants take to be available.

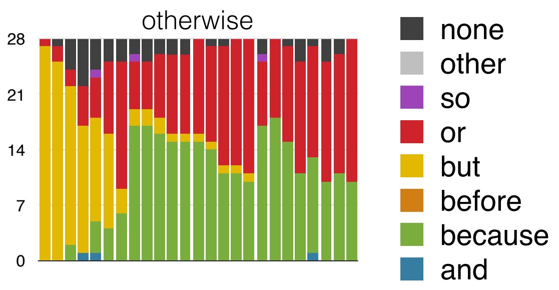


Figure 2: Stacked bar chart for participants' (N=28) conjunction insertions in *otherwise* passages (Rohde et al., 2016)

4.2 *Otherwise* Dataset

Rohde et al. (2016) report surprising response splits amongst BECAUSE∼BUT∼OR for *otherwise* in their conjunction-insertion data (Figure 2). Given that *otherwise* has several different functions (described below), we hypothesize that different response splits arise from the lexical semantics of *otherwise*, combined with inference as to the function of the *otherwise* clause in a given passage.

One function of *otherwise* is in ARGUMENTATION. Here, an *otherwise* clause provides a reason for a given claim, as in (9). Another function is in ENUMERATION, when the speaker first gives some preferred or more salient options, the *otherwise* clause introduces other alternative options, as in (10). A third use is in expressing an EXCEPTION to a generalization. Here, the main clause expresses a generalization, while *otherwise* clause specifies an exception (disjunctive alternative) to it, as in (11).

(9) Proper placement of the testing device is an important issue ______ otherwise the test results will be inaccurate.

(10) A baked potato, plonked on a side plate with sour cream flecked with chives, is the perfect accompaniment ______ otherwise you could serve a green salad and some good country bread.

(11) Mr. Lurie and Mr. Jarmusch actually catch a shark, a thrashing 10-footer ______ otherwise the action is light.

Results presented in (Rohde et al., 2017) for passages like (9) showed participant judgments of OR and BECAUSE, but not BUT. Passages like (10) yielded pairings of OR and BUT, but not BECAUSE. Lastly, passages like (11) yielded response splits between BUT and the less specific AND (Knott, 1996).

Note that due to overlaps in conjunction choice, some conjunctions cannot be unambiguously associated with a single use of *otherwise*: While BECAUSE may unambiguously signal that a participant has inferred ARGUMENTATION, OR might indicate inference of either ARGUMENTATION or

ENUMERATION. Thus we probe both participant choices of connectives and (via paraphrase) the use of *otherwise* that they take to hold.

We chose 16 passages for each use of *otherwise*, based on our own category judgments. For each passage, we asked participants to select the conjunction that best expressed how its two segments were related, and then any other connectives that they took to express the same thing.

A paraphrase task was then used as further evidence for the relation participants inferred in the *otherwise* passages. After completing a given session's batch of passages, participants were asked to select which of three options they took to be a valid paraphrase of the passage. Each use of *otherwise* was assigned a distinct paraphrase to link the left-hand and right-hand segments (LHS, RHS).

- ARGUMENTATION: "A reason for ⟨LHS⟩ is ⟨RHS⟩."
- EXCEPTION: "Generally ⟨RHS⟩. An exception is when ⟨LHS⟩."
- ENUMERATION: "There's more than one good option for ⟨goal⟩. They are: ⟨LHS⟩, ⟨RHS⟩."

We also allowed participants to choose a second paraphrase if they thought it appropriate.

4.3 *Instead* Dataset

Rohde et al. (2016) report a range of participant choices in conjunction-insertion passages involving *instead* (Figure 3). For passages on the left of the figure, participants uniformly chose BUT, while the passage on the far right yielded a strong preference for SO. Elsewhere, some chose BUT and some chose SO. (For the current experiment, we ignore the fact that AND can contingently substitute for either BUT or SO as a connective in text (Knott, 1996), focussing only on passages where participants explicitly choose BUT and/or SO.)

Rohde et al. (2017) report even more surprising participant responses to passages such as (12), where some participants selected both BUT and SO as equally expressing how the segments in the passage were related.

(12) There may not be a flight scheduled to Loja today ______ instead we can go to Cuenca. [BUT∼SO]

Neither the inter-participant split between BUT and SO in (Rohde et al., 2016) nor the intra-participant split between them (Rohde et al., 2017) can be explained in terms of *instead* itself, since

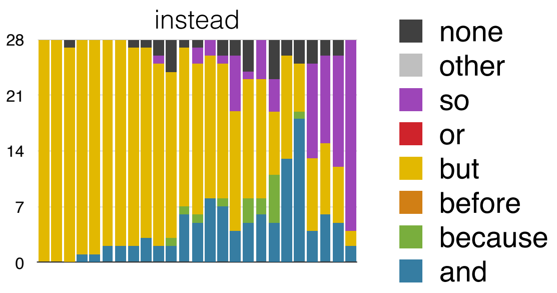


Figure 3: Stacked bar chart for participants' (N=28) conjunction insertions in *instead* passages (Rohde et al., 2016)

instead simply conveys that what follows is an alternative to an unrealised situation in the context (Prasad et al., 2008; Webber, 2013). The current experiment tests the hypothesis that this BUT∼SO split is a consequence of inference from properties of the segments themselves.

To test this hypothesis, we created 16 minimal pairs of passages containing *instead*, one of which emphasized the information structural parallelism between the clauses, as in (13a), and another variant (13b) that de-emphasized that parallelism in favor of a causal link implied by a downward-entailing construction such as *too X* (Webber, 2013). For each passage, half the participants saw the parallelism variant in the conjunction-insertion task, while half saw the causal variant.

(13) a. There was no flight scheduled to Loja yesterday ______ instead there were several to Cuenca.

b. There were too few flights scheduled to Loja yesterday ______ instead we went to Cuenca.

4.4 *After all* Dataset

In (Rohde et al., 2017), we reported a BECAUSE∼BUT response split for passages containing *after all*. We speculated that this may be because a passage such as (14) below presents an argument in which the second segment serves as a REASON (hence, BECAUSE) for the first segment, but also serves to CONTRAST with it (hence, BUT).

(14) Yes, I suppose there's a certain element of danger in it ______ (after all) there's a certain amount of danger in living, whatever you do.

We hypothesize that the BECAUSE∼BUT split cannot be a consequence of the adverbial *after all*, which the Cambridge Dictionary indicates is "used to add information that shows that what you have just said is true".[1] If REASON and/or CONTRAST

[1] https://dictionary.cambridge.org/us/dictionary/english/after-all

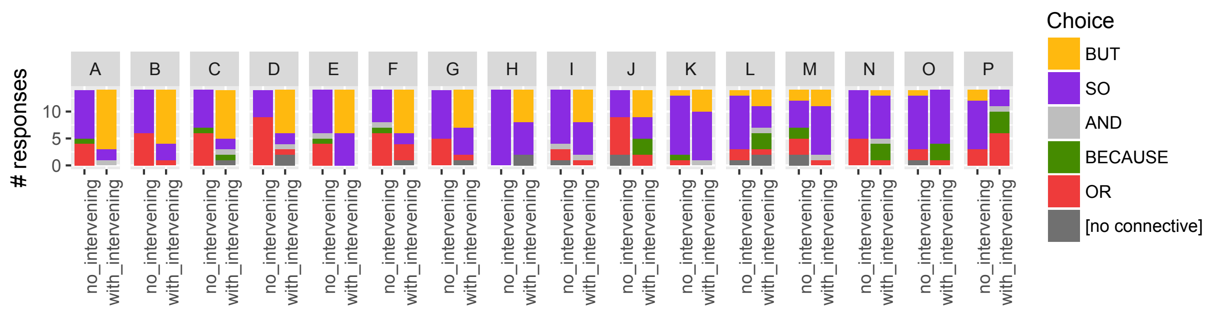


Figure 4: Distribution of participants' first choice of conjunction for passages with *in other words*. Each participant saw only one variant. Each vertical bar represents a passage with the responses from each participant, color-coded by conjunction.

are being conveyed, it can't be a consequence of *after all*. As such, this response split must depend on the reasoning that supports the inference of coherence between the two segments, separate from the adverbial itself.

We test the hypothesis that the response split is independent of the presence or absence of *after all*. Starting with 16 passages that originally contained *after all*, we created a variant of each passage without the adverbial. The conjunction insertion task was the same as with the other datasets.

5 Results

5.1 *In other words*: Inference and adjacency

Section 4.1 lays out the joint hypotheses that inferred relations in passages with *in other words* reflect two components of the lexical semantics of the adverbial (leading to the OR$\sim$SO split) and that the presence of intervening material before *in other words* reduces the availability of those relations, favoring BUT instead.

Figure 4 shows the predicted pattern: The no-intervening-content condition primarily yields OR/SO responses (with variation across passages on the OR-vs.-SO preference) with a relative increase in BUT responses in the intervening-content condition.[2] Passage B corresponds to the pair of examples (2)/(6), and passage C reflects (7)/(8).

For the analysis here and in Section 5.3, a relevant first-choice conjunction was chosen and the binary outcome of its insertion was modeled with a mixed-effect logistic regression. Here, the insertion of OR indeed varied with the presence/absence of intervening material ($\beta = -1.569$, $p < 0.005$).

We posit that increases in BUT associated with the intervening content indicate either an interruption of the meta-linguistic tangent or an intention to signal a contrast with the negative affect of the tangent itself (e.g., "I don't know where...", "frustrating way of putting it", "how this is measured is unclear"). We speculate that the presence of BECAUSE in passages with intervening content may arise when that content implies that the situation is somehow surprising, which in turn merits explanation (e.g., "it's an UNUSUAL role for her", "their ability to actually work sensitively is perhaps QUESTIONABLE", "it's STRANGE to think of a planet being born"). These hypotheses will themselves need to be tested.

[2]For Passage P in Figure 4, participants may have linked the *in other words* clause to the intervening material itself.

5.2 *Otherwise*: Inference from semantic features of segments

As noted in Section 4.2, passages containing *otherwise* were used to test how semantic properties of the segments themselves influenced conjunction choice. The categorization of passages by the researchers (16 ARGUMENTATION, 16 EXCEPTION, 16 ENUMERATION) predicts the conjunctions chosen by participants. In aggregate, ≈99% of responses to ARGUMENTATION passages were BECAUSE or OR or both. ≈92% of responses to EXCEPTION passages were BUT, AND, or both BUT and AND. And ≈98% of responses to ENUMERATION passages were BUT, AND, OR, or some subset thereof. For analysis, a mixed-effect logistic regression modeled the binary outcome of BUT insertion and showed significant variation across the three categories ($p < 0.001$). This measure captures the difference between pairs of categories: ARGUMENTATION permits BECAUSE and OR (hence BUT is rare) while ENUMERATION permits BUT and OR (hence BUT is present) and EXCEPTION favors BUT (hence BUT is very frequent). All pairwise comparisons yielded a main effect of category on this dependent measure (p's < 0.001).

Turning to individual passages, participant choices are shown in Figures 5–7. For ARGUMENTATION (Figure 5), the effect is uniformly strong, with all passages showing BECAUSE or OR as

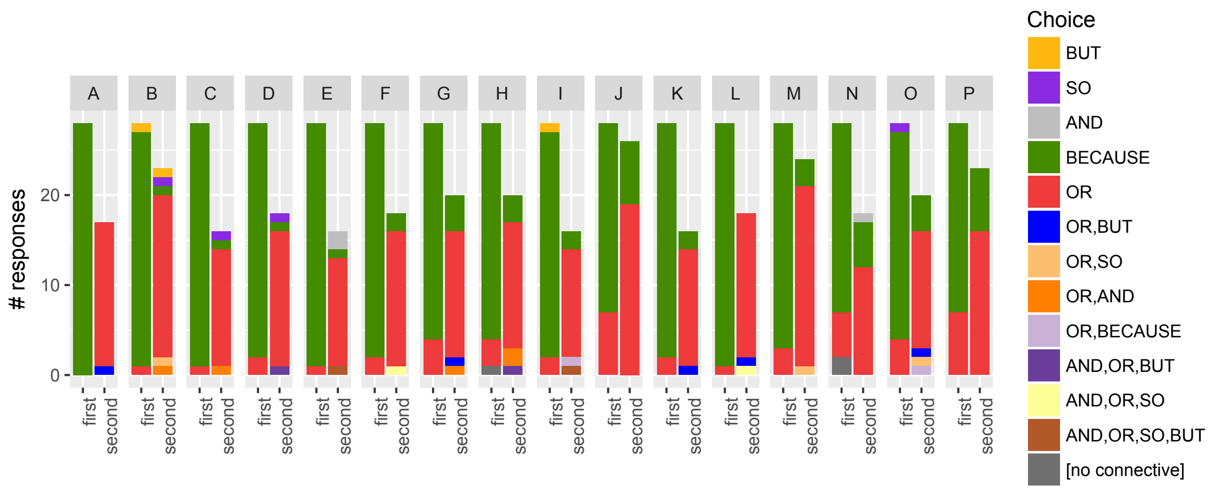


Figure 5: Distribution of first and second choice conjunctions for ARGUMENTATION *otherwise*. Labels such as "OR,BUT" are for multiple second choices. Each vertical bar represents a passage with the responses from each participant, color-coded by conjunction. Enlarged B/W versions of Figures 4–8 are available via the "notes" link on the paper in the ACL anthology.

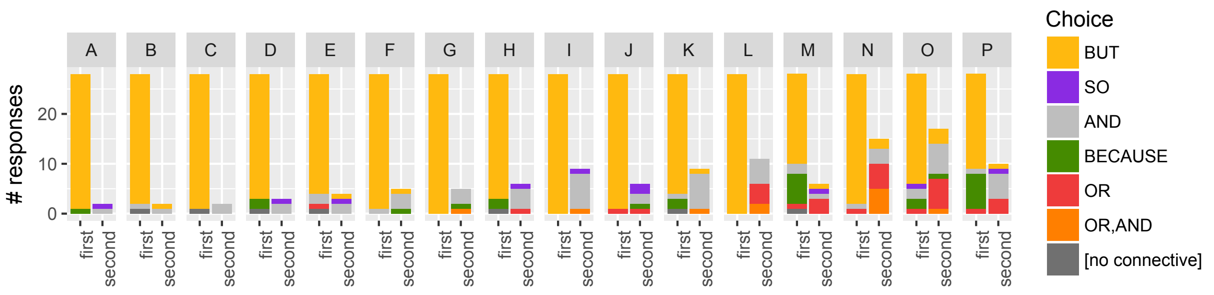


Figure 6: Distribution of first and second choice conjunctions for EXCEPTION *otherwise*. The label "OR,AND" in the legend implies both as second choices.

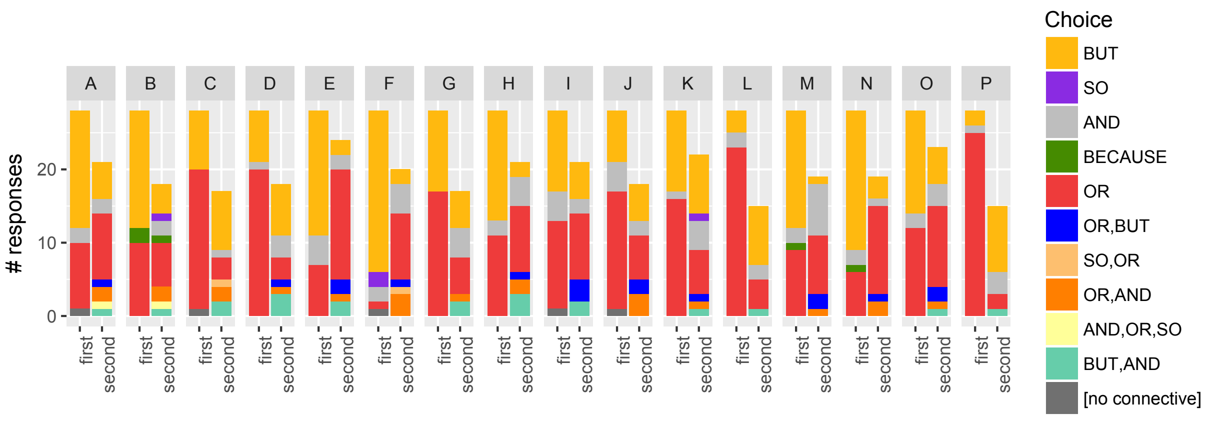


Figure 7: Distribution of first and second choice conjunctions for ENUMERATION *otherwise*. Labels in the legend such as "SO,OR" are for multiple second choices.

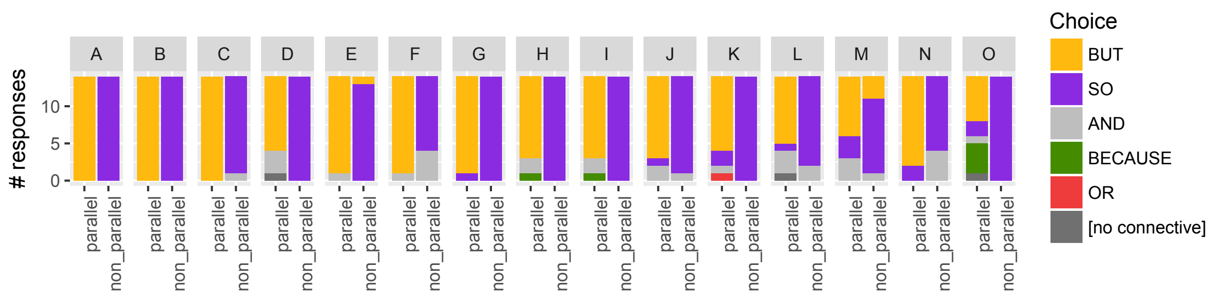


Figure 8: *Instead* passages, pairing a parallel variant and a causal variant. Each column shows the distribution of participants' first choice in the conjunction-insertion task. Each participant saw only one variant.

participants' top choice, with OR or BECAUSE chosen as equivalent (shown in the columns labelled "second"). For EXCEPTION (Figure 6), BUT is consistently the participants' top choice.

There are a few deviations from this near uniform endorsement of BUT for EXCEPTION (Figure 6, passages L–P). Any hypotheses, however, would require further experimentation to test. For example, in passage M (see (15)) and P (see (16)), participants rarely identified any conjunction as conveying the same sense as BUT. However, when their top choice was BECAUSE, they also selected OR as conveying the same sense. As noted above, BECAUSE and OR predominate with *otherwise* used in ARGUMENTATION. This raises the question of why passages M and P lead some participants to infer ARGUMENTATION and other participants, either EXCEPTION or ENUMERATION.

(15) Democrats insist that the poor should be the priority, and that tax relief should be directed at them _____ otherwise they lack a cogent vision of the needs of a new economy.

(16) He said that the proposed bill would give states more flexibility in deciding whether they wanted to use the Federal money for outright grants to municipalities or to set up loan programs _____ otherwise it left last fall's Congressional legislation unchanged.

Finally, though the pattern for ENUMERATION (Figure 7) is harder to see, combinations of BUT, OR and AND predominate as participants' top choices, with a few tokens of BECAUSE and SO, but too few to analyse as anything but noise.

The above results reflect researcher-assigned use labels. However, the confusion matrix in Table 1 shows that on the whole, participants agree with that assignment. The column labelled Multiple is for cases where participants offered two paraphrases. For ARGUMENTATION, at least one paraphrase always corresponded to EXCEPTION, while for ENUMERATION, it did so for most of these tokens (9/14). We comment on this below.

While there was less agreement when participants offered multiple paraphrases for researcher-assigned EXCEPTION, there may be too few tokens here to draw any kind of conclusion. In any case, the results for ARGUMENTATION and ENUMERATION agree both across participants (in what paraphrase they choose when they don't choose the researcher-assigned label) and within participants (in what pairs of paraphrases they gave for the original passage).

The above results support our hypothesis that variability in participants' choice of conjunctions follows from both the lexical semantics of *otherwise* and the relation that participants infer between the segments in the passage.

5.3 *Instead*: Inference from a single manipulated property

On aggregate, participants responded very differently to the parallel and causal variants of *instead* passages (cf. Section 4.3). Figure 8 shows that in all cases, the parallel variant yielded more BUT responses, whereas the non-parallel (causal) variant yielded significantly more SO responses (main effect of (non-)parallelism: β=−7.0008, p<0.001).[3]

Some of these results are very strong. For example, Passage A (17) drew all BUT responses for the

[3]We analyzed only 15 passages for *instead* and *after all*, due to a presentation error of the 16th for these adverbials.

	Participant			
Researcher	ARGUMENTATION	ENUMERATION	EXCEPTION	Multiple
ARGUMENTATION	401 (91.5%)	4	25	18
ENUMERATION	23	364 (81.4%)	46	14
EXCEPTION	21	29	393 (87.7%)	5

Table 1: Researcher labels assigned to *otherwise* passages vs. labels implied by participant paraphrases

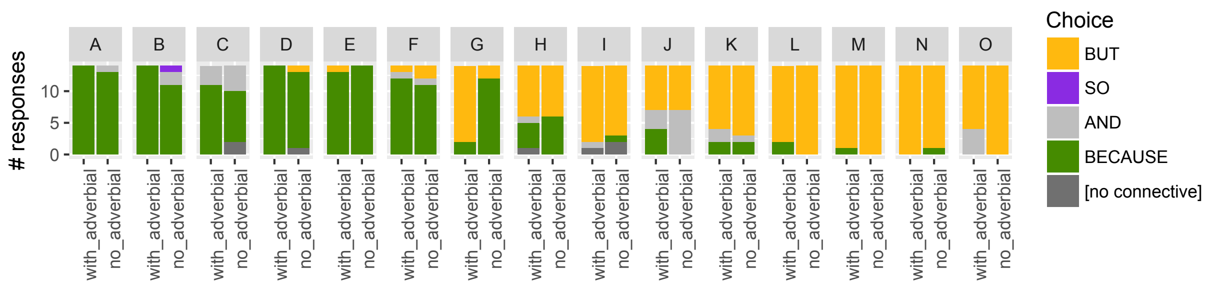


Figure 9: Distribution of first choice in conjunction selection task for passages with *after all*

parallel variant in (17a) and all SO responses for the causal variant in (17b), as did Passage B. In a few cases, however, the parallel variant drew variable responses, even while its causal variant drew strong SO responses. This is true of Passage O, with parallel and causal variants in (18a–b).

(17) a. They could have been playing football in the village green _____ instead they played in the street.

b. They didn't like playing football in the village green _____ instead they played in the street.

(18) a. Smugglers nowadays don't use overland passages _____ instead they use the seas to transport their goods.

b. Smugglers' overland passages nowadays are too visible _____ instead they use the seas to transport their goods.

One possible explanation is that participants varied in the role they assigned to the positive claim in the second segment of (18a) – either as a reason for the negative claim in the first segment (BECAUSE), as a contrast with that claim (BUT), or as its result (SO). Although manipulating the segment to enhance either parallelism or causality can change participant responses, it is clear that parallelism alone doesn't guarantee contrast.

5.4 *After all*: Adverb adds little to inference

Figure 9 shows participant choice of conjunction when *after all* is present and when it is absent. Their choice is largely the same for passages A–F and K–N, with and without the adverbial. As for passage O, since AND can contingently substitute for BUT (Knott, 1996), the response pattern can be considered the same as well. A by-passage correlation between the rate of BUT and BECAUSE responses across the two conditions confirms this similarity (R^2=.70, F(1,13)=30.98, p<0.001). The outlier is passage G:

(19) There was a testy moment driving over the George Washington Bridge when the toll-taker charged him $24 for his truck and trailer _____ after all it was New York.

With *after all*, the majority of participants chose BUT as best expressing how the two segments are connected, while without it, the majority chose BECAUSE. Whatever explanation we gave here would be pure speculation. We trust that the fact that the other 14 passages demonstrate the predicted effect provides sufficient evidence that splits in participant responses are not simply a result of the presence of a discourse adverbial.

6 Conclusion

While our previous work showed that multiple discourse relations can hold between two segments – relations at the same semantic level, simultaneously available to a reader – we provided no evidence as to what influences the particular relations that are taken to be available. Our current experiments have provided some such evidence. Specifically, we have shown that participant responses to systematically manipulated passages involving discourse adverbials can be explained in terms of both the lexical semantics of discourse adverbials and properties of the passages that contain them. As the conjunctions chosen by participants convey senses that differ from those of the discourse adverbials, we also provided evidence for the simultaneous availability of multiple coherence relations that arise from both explicit signals and inference. We hope the reader is now convinced that, in both psycholinguistic research on discourse coherence and computational work on discourse parsing, one needs to identify and examine evidence for coherence involving more than one discourse relation.

Acknowledgments

This project has been supported by a grant from the Nuance Foundation. In addition, a Leverhulme Trust Prize in Languages & Literatures to H. Rohde has enabled her to devote more time to research. We thank Amir Zeldes and the anonymous reviewers for their helpful feedback on the paper.

References

M.J. Cuenca and M.J. Marin. 2009. Co-occurrence of discourse markers in Catalan and Spanish oral narrative. *Journal of Pragmatics*, 41(5):899–914.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017. The BECauSE Corpus 2.0: Annotating causality and overlapping relations. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 95–104, Valencia, Spain.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 60–68, Jeju Island, Korea.

Bruce Fraser. 2013. Combinations of contrastive discourse markers in English. *International Review of Pragmatics*, 5:318–340.

Barbara Grosz and Candace Sidner. 1986. Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Michael Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 13–24, Baltimore, Maryland.

Alistair Knott. 1996. *A Data-driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. dissertation, Department of Artificial Intelligence, University of Edinburgh.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, pages 151–184.

William C. Mann, Christian M. I. M. Matthiessen, and Sandra A. Thompson. 1989. Rhetorical structure theory and text analysis. Technical Report ISI/RR-89-242, USC-ISI, Marina del Rey CA.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT Press.

Eleni Miltsakaki, Nikhil Dinesh, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Experiments on sense annotation and sense disambiguation of discourse connectives. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT'05)*, Barcelona, Spain.

Johanna Moore and Martha Pollack. 1992. A problem for RST: The need for multi-level discouse analysis. *Computational Linguistics*, 18(4):537–544.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 2961–2968, Marrakech, Morocco.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2014. Reflections on the Penn Discourse TreeBank, comparable corpora and complementary annotation. *Computational Linguistics*, 40(4):921–950.

Hannah Rohde, Anna Dickinson, Chris Clark, Annie Louis, and Bonnie Webber. 2015. Recovering discourse relations: Varying influence of discourse adverbials. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 22–31, Lisbon, Portugal.

Hannah Rohde, Anna Dickinson, Nathan Schneider, Christopher Clark, Annie Louis, and Bonnie Webber. 2016. Filling in the blanks in understanding discourse adverbials: Consistency, conflict, and context-dependence in a crowdsourced elicitation task. In *Proceedings of the Tenth Linguistic Annotation Workshop (LAW-X)*, pages 49–58, Berlin, Germany.

Hannah Rohde, Anna Dickinson, Nathan Schneider, Annie Louis, and Bonnie Webber. 2017. Exploring substitutability through discourse adverbials and multiple judgments. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, Montpellier, France.

Merel Scholman and Vera Demberg. 2017. Crowdsourcing discourse interpretations: On the influence of context and the reliability of a connective insertion task. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 24–33, Valencia, Spain.

Manfred Stede. 2012. *Discourse Processing*. Morgan & Claypool Publishers.

Bonnie Webber. 2013. What excludes an alternative in coherence relations? In *Proceedings of the 10th International Conference on Computational Semantics*, Potsdam, Germany.

Bonnie Webber, Aravind Joshi, and Alistair Knott. 2000. The anaphoric nature of certain discourse connectives. In *Making Sense: From Lexeme to Discourse*, Groningen, The Netherlands.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol T. Rutherford. 2015. The CoNLL-2015 Shared Task on Shallow Discourse Parsing. In *Proceedings of the Nineteenth Conference on Computational Language Learning: Shared Task*, pages 1–16, Beijing, China.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. CoNLL 2016 Shared Task on Multilingual Shallow Discourse Parsing. In *Proceedings of the 20th Conference on Computational Natural Language Learning – Shared Task*, pages 1–19, Berlin, Germany.

A Spatial Model for Extracting and Visualizing Latent Discourse Structure in Text

Shashank Srivastava[*]
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ssrivastava@cmu.edu

Nebojsa Jojic
Microsoft Research
Redmond, WA 98052, USA
jojic@microsoft.com

Abstract

We present a generative probabilistic model of documents as sequences of sentences, and show that inference in it can lead to extraction of long-range latent discourse structure from a collection of documents. The approach is based on embedding sequences of sentences from longer texts into a 2- or 3-D spatial grids, in which one or two coordinates model smooth topic transitions, while the third captures the sequential nature of the modeled text. A significant advantage of our approach is that the learned models are naturally visualizable and interpretable, as semantic similarity and sequential structure are modeled along orthogonal directions in the grid. We show that the method can capture discourse structures in narrative text across multiple genres, including biographies, stories, and newswire reports. In particular, our method can capture biographical templates from Wikipedia, and is competitive with state-of-the-art generative approaches on tasks such as predicting the outcome of a story, and sentence ordering.

1 Introduction

The ability to identify discourse patterns and narrative themes from language is useful in a wide range of applications and data analysis. From a perspective of language understanding, learning such latent structure from large corpora can provide background information that can aid machine reading. For example, computers can use such knowledge to predict what is likely to happen next in a narrative (Mostafazadeh et al., 2016), or reason about which narratives are coherent and which do not make sense (Barzilay and Lapata, 2008). Similarly, knowledge of discourse is increasingly important for language generation models. Modern neural generation models, while good at capturing surface properties of text – by fusing elements of syntax and style – are still poor at modeling long range dependencies that go across sentences (Li and Jurafsky, 2017; Wang et al., 2017). Models of long range flow in the text can thus be useful as additional input to such methods.

Previously, the question of modeling discourse structure in language has been explored through several lenses, including from perspectives of linguistics, cognitive science and information retrieval. Prominent among linguistic approaches are Discourse Representation Theory (Asher, 1986) and Rhetorical Structure Theory (Mann and Thompson, 1988); which formalize how discourse context can constrain the semantics of a sentence, and lay out ontologies of discourse relation types between parts of a document. This line of research has been largely constrained by the unavailability of corpora of discourse relations, which are expensive to annotate. Another line of research has focused on the task of automatic *script induction*, building on earlier work in the 1970's (Schank and Abelson, 1977). More recently, methods based on neural distributed representations have been explored (Li and Hovy, 2014; Kalchbrenner and Blunsom, 2013; Le and Mikolov, 2014) to model the flow of discourse. While these methods have had varying degrees of success, they are largely opaque and hard to interpret. In this work, we seek to provide a scalable model that can extract latent sequential structures from a collection of documents, and can be naturally visualized to provide a summary of the learned semantics and discourse trajectories.

In this work, we present an approach for extract-

[*]Work done while first author was an intern at Microsoft Research

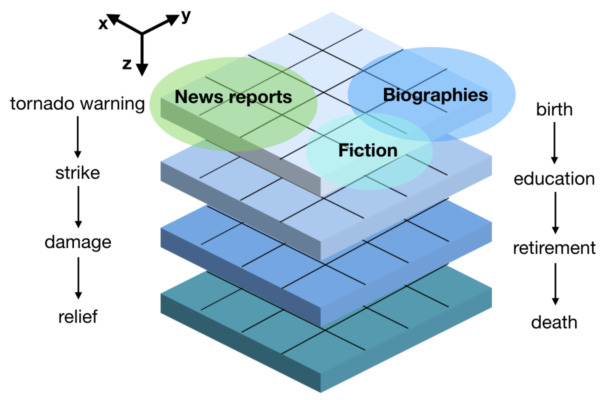


Figure 1: Modeling principle for Sequential Counting Grids. We design the method to capture semantic similarities between documents along XY planes (e.g., biographies might be more similar to literary fiction than news reports), as well extract sequential trajectories along the Z axes similar to those shown. The sequence of sentences in a document is latently aligned to positions in the grid, such that the model prefers alignments of contiguous sentences to grid cells that are spatially close.

ing and visualizing sequential structure from a collection of text documents. Our method is based on embedding sentences in a document in a 3-dimensional grid, such that contiguous sentences in the document are likely to be embedded in the same order in the grid. Further, sentences across documents that are semantically similar are also likely to be embedded in the same neighborhood in the grid. By leveraging the sequential order of sentences in a large document collection, the method can induce lexical semantics, as well as extract latent discourse trajectories in the documents. Figure 1 shows a conceptual schematic of our approach. The method can learn semantic similarity (across XY planes), as well as sequential discourse chains (along the Z-axis). The parameters and latent structure of the grid are learned by optimizing the likelihood of a collection of documents under a generative model. Our method outperforms state-of-the-art generative methods on two tasks: predicting the outcome of a story and coherence prediction; and is seen to yield a flexible range of interpretable visualizations in different domains of text. Our method is scalable, and can incorporate a broad range of features. In particular, the approach can work on simple tokenized text.

The remainder of this paper is organized as follows. In Section 2, we briefly summarize other related work. In Section 3, we describe our method in detail. We present experimental results in Section 4, and conclude with a brief discussion.

2 Related work

Building on linguistic theories of discourse and text coherence, several computational approaches have attempted to model discourse structure from multiple perspectives. Prominent among these are Narrative Event Chains (Chambers and Jurafsky, 2008) which learn chains of events that follow a pattern in a unsupervised framework, and the Entity grid model (Barzilay and Lapata, 2008), which represents sentences in a context in terms of discourse entities occurring in them and trains coherence classifiers over this representation. Other work extends these using better models of events and discourse entities (Lin et al., 2011; Pichotta and Mooney, 2015). Louis and Nenkova (2012) use manually provided syntactic patterns for sentence representation, and model transitions in text as Markov probabilities, which is related to our work. However, while they use simple HMMs over discrete topics, our method allows for a richer model that also captures smooth transition across them. Approaches such as Kalchbrenner and Blunsom (2013); Li et al. (2014); Li and Jurafsky (2017) model text through recurrent neural architectures, but are hard to interpret and visualize. Other approaches have explored applications related to modeling narrative discourse in context of limited tasks such as story cloze (Mostafazadeh et al., 2016) and identifying similar narratives (Chaturvedi et al., 2018).

From a large scale document-mining perspective, the question of extracting intra-document structure remains largely underexplored. While early models such as LDA completely ignore ordering and discourse elements of a documents, other methods that use distributed embeddings of documents are opaque (Le and Mikolov, 2014), even while they can in principle model sequential structure within a document. Methods such as HMM-LDA (Griffiths et al., 2005) and Topics-over-time (Wang and McCallum, 2006) address the related question of topic evolution in a stream of documents, but these approaches are too coarse to model intra-document sequential structure. In terms of our technical approach, we build on previous research on grid-based models (Jojic and Perina, 2011), which have previously been used for topic-modeling for images and text as unstructured bags-of-features.

3 Sequential CG model

In this section, we present our method, which we call **Sequential Counting Grids**, or **Sequential CG**. We first present our notation, model formulation and training approach. We discuss how the method is designed to incorporate smoothness and sequential structure, and how the method can be efficiently scaled to train on large document collections. In Section 3.2, we present a mixture model variant that combines Sequential CG with a unigram language model.

3.1 Model description

We represent a document as a sequence $\mathbf{s}$ of sentences, $\mathbf{s} = \{s_1, s_2 \dots s_D\}$, where D represents the number of sentences in the document. In general, we assume each sentence is represented as a multiset of features $s_i = \{c_z\}_i$, where c_z^i represents the count of the feature indexed by z in the ith sentence in the sequence.[1]

The Sequential CG consists of a 3-D grid G of size $E_x \times E_y \times E_z$, where E_x, E_y and E_z denote the extent of the grid along the X, Y and Z-axes (see Figure 1). Let us denote an index of a position in the grid by an integer-valued vector $\mathbf{i} = (i_x i_y i_z)$. The three components of the index together specify a XY location as well as a depth in the grid. The Sequential CG model is parametrized by two sets of parameters, $\pi_{\mathbf{i},z}$ and $\mathcal{P}_{\mathbf{ij}}$. Here, $\pi_{\mathbf{i},z}$ represents a multinomial distribution over the vocabulary of features z for each cell in the grid G, i.e. $\sum_z \pi_{\mathbf{i},z} = 1 \ \forall \ \mathbf{i} \in G$. To induce smoothness across XY planes, we further define histogram distributions $h_{\mathbf{i},z}$, which average the π distributions in a 2-D neighborhood $W_\mathbf{i}$ (of size specified by $W = [W_x, W_y]$) around the grid position $\mathbf{i}$. This notation follows Jojic and Perina (2011).

$$h_{\mathbf{i},z} = \frac{1}{W_x W_y} \sum_{\mathbf{i}' \in W_\mathbf{i}} \pi_{\mathbf{i}',z} \tag{1}$$

The generative model assumes that individual sentences in a document are generated by h distributions in the grid. Movements from one position $\mathbf{i}$ to another $\mathbf{j}$ in the grid are modeled as transition probabilities $\mathcal{P}_{\mathbf{ij}}$. The generative process consists of the following. We uniformly sample a starting location $\mathbf{i}_1$ in the grid. We sample words in the first sentence s_1 from $\pi_{\mathbf{i}1}$, and sample the next position $\mathbf{i}_2$ from the distribution $\mathcal{P}_{\mathbf{i}_1,:}$, and so on till we generate s_D. The alignments $\mathcal{I} = [\mathbf{i}_1, \mathbf{i}_2 \dots \mathbf{i}_D]$ of individual sentences in a document with positions in the grid are latent variables in our model.

Given the sequence of alignments $\mathcal{I}$ for a document, the conditional likelihood of generating s is given as a product of generating individual sentences:

$$p(s|\,\mathcal{I}) = \prod_d^D p(\{c_z^d\}|\,\mathbf{i}_d) = \prod_{d=1}^D \prod_z (h_{\mathbf{i}_d,z})^{c_z^d} \tag{2}$$

Since the alignments of sequences to their positions in the grids $\mathcal{I}$ are latent, we marginalize over these to maximize the likelihood of an observed collection of documents $\mathcal{S} := \{\mathbf{s}^t\}_{t=1}^T$. Here, T is the total number of documents, and t is an index over individual documents. Using Jensen's inequality, *any* distributions $q_\mathcal{I}^t$ over the hidden alignments $\mathcal{I}^t$ provide lower-bounds on the data log-likelihood.

$$\begin{aligned}\sum_t \log p(\mathbf{s}_t|\pi) &= \sum_t \log\Big(\sum_\mathcal{I} p(\mathbf{s}_t, \mathcal{I}|\pi)\Big) \\ &= \sum_t \log\Big(\sum_\mathcal{I} q_\mathcal{I}^t \frac{p(\mathbf{s}_t|\mathcal{I})p(\mathcal{I}))}{q_\mathcal{I}^t}\Big) \\ &\geq -\sum_t \sum_\mathcal{I} q_\mathcal{I}^t \log q_\mathcal{I}^t \\ &\quad + \sum_t \sum_\mathcal{I} q_\mathcal{I}^t \log\big(p(s|\mathcal{I}, \pi)p(\mathcal{I})\big)\end{aligned} \tag{3}$$

Here, $q_\mathcal{I}^t$ denotes a variational distribution for each of the data sequences $\mathbf{s}_t$. The learning algorithm consists of an iterative generalized EM procedure (which can be interpreted as a block-coordinate ascent in the latent variables $q_\mathcal{I}^t$ and the model parameters π and $\mathcal{P}$). We maximize the lower bound in Eqn 3 exactly by setting $q_\mathcal{I}^t$ to the posterior distribution of the data for the current values of the parameters π (standard E step). Thus, we have

$$\begin{aligned} q_\mathcal{I}^t &\propto p(\mathbf{s}|\mathcal{I})p(\mathcal{I}) \\ &= \Big[\prod_{d=1}^D \prod_z (h_{\mathbf{i}_d,z})^{c_z^d(t)}\Big]\Big[\prod_{d=2}^D \mathcal{P}_{\mathbf{i}_{d-1},\mathbf{i}_d}\Big]\end{aligned} \tag{4}$$

We do not need to explicitly compute the posterior distribution $q_\mathcal{I}^t = p(\mathcal{I}|\mathbf{s})$ at this point, but only use it to compute the relevant expectation statistics in the M-step. This can be done efficiently, as we

[1] These may simply consist of tokens (words, entities and MWEs) in the sentence, but can include additional information, such as sentiment or event annotations, or other discrete sentence-level representations

see next. In the M-step, we consider $q^t_{\mathcal{I}}$ as fixed, and maximize the objective in terms of the model parameters π. Substituting this in Eqn 3, and focusing on terms that depend on the model parameters (π and $\mathcal{P}$), we get

$$\begin{aligned}\mathcal{L}(\pi,\mathcal{P}) &\geq \sum_t \sum_{\mathcal{I}} q^t_{\mathcal{I}} \log\left(p(s|\mathcal{I},\pi)p(\mathcal{I})\right) + \mathcal{H}_q \\ &= \sum_t \sum_{\mathcal{I}} q^t_{\mathcal{I}} \Bigg(\sum_d \sum_z c^d_z(t) \log h_{\mathbf{i}_d,z} \\ &\qquad + \sum_d \log \mathcal{P}_{\mathbf{i}_{d-1},\mathbf{i}_d} \Bigg) \\ &= \sum_t \sum_{\mathcal{I}} \mathbb{E}_{q^t_{\mathcal{I}}} \Big[\sum_d \sum_z \mathbb{I}_{\mathbf{i}^t_d=\mathbf{i}} c^d_z(t) \log h_{\mathbf{i}_d,z} \Big] \\ &\quad + \sum_t \sum_{\mathcal{I}} \mathbb{E}_{q^t_{\mathcal{I}}} \Big[\sum_d \mathbb{I}_{\mathbf{i}^t_{d-1}=\mathbf{i},\mathbf{i}^t_d=\mathbf{j}} \log \mathcal{P}_{\mathbf{ij}} \Big]\end{aligned} \tag{5}$$

Maximizing the likelihood w.r.t. $\mathcal{P}$ leads to the following updates for the transition probabilities:[2]

$$\mathcal{P}_{\mathbf{ij}} = \frac{\sum_t \sum_d P(\mathbf{i}^t_{d-1} = \mathbf{i}, \mathbf{i}^t_d = \mathbf{j})}{\sum_t \sum_d P(\mathbf{i}^t_{d-1} = \mathbf{i})} \tag{6}$$

Here, the pairwise state-probabilities $P(\mathbf{i}^t_{d-1} = \mathbf{i}, \mathbf{i}^t_d = \mathbf{j})$ for adjacent sentences in a sequence can be efficiently calculated using the Forward-Backward algorithm. In Equation 5, rewriting the term containing h in terms of π using Eqn 1 (and ignoring constant terms W_xW_y), we get:

$$\begin{aligned}&\sum_t \sum_{\mathcal{I}} \mathbb{E}_{q^t_{\mathcal{I}}} \Big[\sum_d \sum_z \mathbb{I}_{\mathbf{i}^t_d=\mathbf{i}} c^d_z(t) \log \sum_{\mathbf{i}' \in W_{\mathbf{i}}} \pi_{\mathbf{i}',z} \Big] \\ &= \sum_t \sum_{\mathcal{I}} \sum_d P(\mathbf{i}^t_d = \mathbf{i}) \sum_z c^d_z(t) \log \sum_{\mathbf{i}' \in W_{\mathbf{i}}} \pi_{\mathbf{i}',z}\end{aligned} \tag{7}$$

The presence of a summation inside of a logarithm makes maximizing this objective for π harder. For this, we simply use Jensen's inequality introducing an additional variational distribution (for the latent grid positions within window $W_{\mathbf{i}}$), and maximize the lower bound. The final M-step update for π becomes:

$$\pi_{\mathbf{i},z} \propto \left(\sum_t \sum_d c^d_z(t) \sum_{\mathbf{k}|\mathbf{i} \in W_k} \frac{P(\mathbf{i}^t_d = \mathbf{k})}{h_{\mathbf{k},z}} \right) \pi_{\mathbf{i},z} \tag{8}$$

As before, the state-probabilities $P(\mathbf{i}^t_d = \mathbf{i})$ can be computed using the Forward Backward algorithm.

Intuitively, the expected alignments in the E-step are distributions over sequences of positions in the grid that best explain the structure of documents for the current value of Sequential CG parameters. In the M-step, we assume these distributions embedding documents into various parts of the grid as given, and update the multinomial parameters and transition probabilities. Modeling the transitions as having a Markov property allows us to use a dynamic programming approach (Forward Backward algorithm) to exactly compute the posterior probabilities required for parameter updates. We note that at the onset of the procedure, we need to initialize π randomly to break symmetries. Unless otherwise stated, in all experiments, we run EM to 200 iterations.

Correlating space with sequential structure: The use of histogram distributions h to generate data forces smoothness in the model along XY planes due to adjacent cells in the grid sharing a large number of parameters that contribute to their histograms (due to overlapping windows). On the other hand, in order to induce spatial proximity in the grid to mimic the sequential flow of discourse in documents, we constrain the transition matrix $\mathcal{P}$ (which specifies transition preferences from one position in the grid to another) to a sparse banded matrix. In particular, a position $\mathbf{i} = (i_x, i_y, i_z)$ in the grid can only transition to itself, its 4 neighbors in the same XY plane, and two cells in the succeeding two layers along the Z-axis ((i_x, i_y, i_{z+1}) and (i_x, i_y, i_{z+2})). This is enforced by fixing other elements in the transition matrix to 0, and only updating allowable transitions.

As an important note about implementation details, we observe here that the Forward-Backward procedure (which is repeatedly invoked during model training) can be naturally formulated in terms of matrix operations.[3] This allows training for the Sequential CG approach to be scalable for large document collections.

In our formulation, we have presented a Sequential CG model for a 3-D grid. This can be adapted to learn 2-D grids (trellis) by setting $E_y = 1$. In our experiments, we found 3-D grids to be better

[2]Since the optimal value for the concave problem $\sum_j y_j \log x_j$ s.t. $\sum_j x_j = 1$ occurs when $x^*_j \propto y_j$

[3]To explain, if $f^d_{1\times G}$ are forward probabilities for step d, and $O^{d+1}_{G\times G}$ are observation probabilities for step $d+1$, $f^{d+1} = f^d \times \mathcal{P} \times O^d$ computes forward probabilities for the next step in the sequence

in terms of task performance and visualization (for a comparable number of parameters).

3.2 Mixture model

The Sequential CG model described above can be combined with other generative models (e.g., language models) to get a mixture model. Here, we show how a unigram language model can be combined with Sequential CG. The rationale behind this is that since the Sequential CG is primarily designed to explain elements of documents that reflect sequential discourse structures, mixing with a context-agnostic distribution can allow it to focus specifically on elements that reflect sequential regularities. In experimental evaluation, we find that such a mixture model shows distinctly different behavior (see Section 4.1.1). Next, we briefly describe updates for this approach.

Let μ_z denote the multinomial distribution over features for the unigram model to be mixed with the CG. Let β_z be the mixing proportion for the feature z, i.e. an occurrence of z is presumed to come from the Sequential CG with probability β_z, and from the unigram distribution with probability $1 - \beta_z$. Further, let α_z^t be binary variable that denotes whether a particular instance of z comes from the Sequential CG, or the unigram model. Then, Equation 2 changes to:

$$p(s|\,\mathcal{I},\alpha) = \prod_{z,d}\Big((h_{\mathbf{i}_d,z})^{c_z^d}\beta_z\Big)^{\alpha_z^t}\big(\mu_z^{c_z^d}(1{-}\beta_z)\big)^{1-\alpha_z^t}$$

Since we do not observe α_z^t (i.e., which distribution generated a particular feature in a particular document), they are additional latent variables in the model. Thus, we need to introduce a Bernoulli variational distribution $q_{\alpha_{zt}}$. Doing this modifies relevant parts (containing $q_{\alpha_{zt}}$) of Equation 5 to:

$$\begin{aligned}&\sum_t\sum_{\mathcal{I}} q_{\mathcal{I}}^t\Bigg(\sum_z q_{\alpha_{zt}}\log\Big(\beta_z\prod_d h_{\mathbf{i}_d,z}^{c_z^d(t)}\Big)\\ &+(1-q_{\alpha_{zt}})\log\Big((1-\beta_z)\mu_z^{\sum_d c_z^d}\Big)\\ &+\sum_d\log\mathcal{P}_{\mathbf{i}_{d-1},\mathbf{i}_d}\Bigg)+\mathcal{H}_{q_{\alpha_{zt}}}\end{aligned} \tag{9}$$

This leads to the following additional updates for estimating $q_{\alpha_{zt}}$ (in the E-step)[4] and β_z (in the M-step).

[4]Since the optimal value for the concave problem $\sum_j x_j \log\frac{y_j}{x_j}$ s.t. $\sum_j x_j = 1$ occurs when $x_j^* \propto y_j$

$$q_{\alpha_{zt}} = \frac{\exp\Big(\sum_{\mathbf{i}}^{\mathcal{I}} P(\mathbf{i}_d^t{=}\mathbf{i})c_z^d(t)\log h_{\mathbf{i}_d,z}\Big)\beta_z}{\exp\Big(\sum_{\mathbf{i}}^{\mathcal{I}} P(\mathbf{i}_d^t{=}\mathbf{i})c_z^d(t)\log h_{\mathbf{i}_d,z}\Big)\beta_z{+}\mu_z^{\sum_d c_z^d}(1{-}\beta_z)}$$

In the M-step, β_z can be estimated simply as the fraction of times z is generated from the Sequential CG component.

$$\beta_z = \frac{\sum_t q_{\alpha_{zt}}}{\sum_t \mathbb{I}_z}$$

4 Evaluation

In this section, we analyze the performance of our approach on text collections from several domains (including short stories, newswire text and biographies). We first qualitatively evaluate our generative method on a dataset of biographical extracts from Wikipedia, which visually illustrates biographical trajectories learned by the model, operationalizing our model concept from Figure 1 in real data (see Figure 2). Next, we evaluate our method on two standard tasks requiring document understanding: story cloze evaluation and sentence ordering. Since our method is completely unsupervised and is not tailored to specific tasks, competitive performance on these tasks would indicate that the method learns helpful regularities in text structure, useful for general-purpose language understanding.

4.1 Visualizing Wikipedia biographies

We now qualitatively explore models learned by our method on a dataset of biographies from Wikipedia.[5] For this, we use the data previously collected and processed by Bamman and Smith (2014). In all, the original dataset consists of extracts from biographies of about 240,000 individuals. For ease of training, we trained our method on a subset of the 50,0000 shortest documents from this set. The original paper uses the numerical order of dates mentioned in the biographies to extract biographical templates, but we do not use this information. Figure 2 visualizes a Sequential CG model learned on this dataset for on a grid of dimensions $E = 8 \times 8 \times 5$, and a histogram window W of dimensions 3×3 . In general, we found that using larger grids leads to smoother transitions and learning more intricate patterns including hierarchies of trajectories, but here we show a model with a

[5]For all our experimental evaluation, we tokenize and lemmatize text using the Stanford CoreNLP pipeline, but retain entity-names and contiguous text-spans representing MWEs as single units

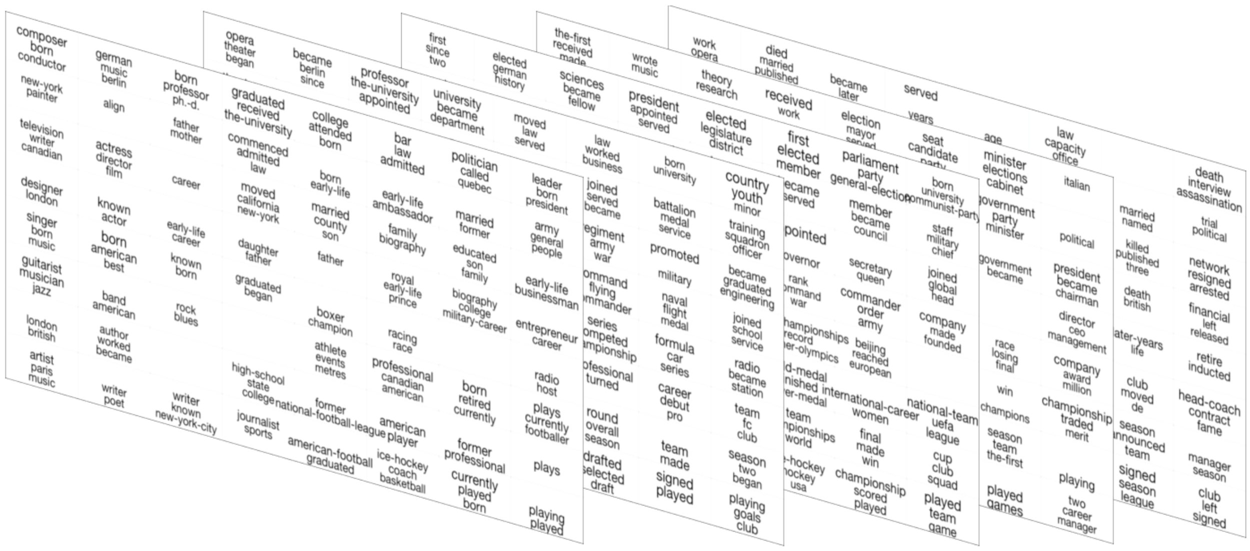


Figure 2: Visualization of a Sequential-CG model with grid size of $8 \times 8 \times 5$, trained on 50,000 documents from the Wikipedia biographies dataset. Cells in the grid show words with highest probabilities (empty cells may indicate that no word has a substantially higher probability than others).

smaller grid for ease of visualization. Here, the words in each cell in the grid denote the highest probability assignments in that cell. Larger fonts within a cell indicate higher probabilities.

We observe that the method successfully extracts various biographical trajectories, as well as capture a notion of similarity between them. To explain, we observe that the lower-right part of the learned grid largely models documents about sportpersons (with discernable regions focusing on sports like soccer, American football and ice-hockey). On the other hand, the left-half of the grid is dominated by biographies of people from the arts and humanities (inlcuding artists, writers, musicians, etc.). The top-center of the grid focuses on academicians and scientists, while the top-right represents biographies of political and military leaders. We note smooth transitions between different regions, which is precisely what we would expect from the use of the smoothing filter that incorporates parameter sharing across cells in the method. Further, as we go across the layers in the figure, we note the biographical trajectories learned by the model across the entire grid. For example, from the grid, the life trajectory of a football player can be visualized as being drafted, signing and playing for a team, and eventually becoming a head-coach or a hall-of-fame inductee.

4.1.1 Effects of mixing

The Sequential-CG method can be combined with other generative models in a mixture model, following the approach previously described in Section 3.2. A major reason to do this might be to allow the base model to handle general content, while allowing the Sequential-CG method to focus on modeling context-sensitive words only. Here, we empirically characterize the mixing behavior for different categories of words.

Figure 3 shows the mixing proportion of different words when the Sequential-CG model is combined with a unigram model. In the figure, the X-axis corresponds to words in the dataset with decreasing frequency of occurrence, whereas the Y-axis denotes the mixing proportions β_z learned by the mixture model. We note that the mixture model learns to explain frequent as well as the long-tail of rare words using the simple unigram model (as seen from low mixing proportion of Sequential-CG method). These regimes correspond to (1) stop-words and very common nouns, and (2) rare words respectively. In turn, this allows the Sequential-CG component to preserve more probability mass to explain the intermediate content words. Thus, the Sequential-CG component only needs to model words that reflect useful statistical sequential patterns, without expending modeling effort on background content (common words) or noise (rare words). For the long tail of infrequent words, we observe that Sequential CG is much more likely to generate verbs and adjectives, rather than nouns. This is as we would expect, since verbs and adjectives often denote events and sentiments, which can

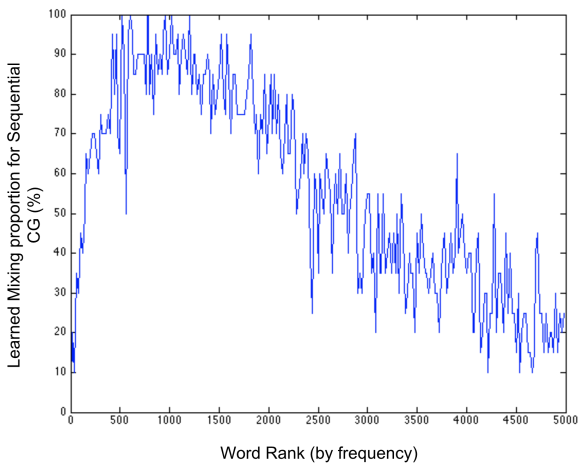


Figure 3: Learned mixing proportion (β_z) in the mixture model of Section 3.2 for words of different frequencies. β_z denotes the probability of a word being generated from the Sequential CG model (rather than from the Unigram model). The Sequential CG learns to model content words (with intermediate ranks), and conserves modeling effort by avoiding modeling both very common words (that occur across contexts), as well as rare words.

be important elements in discourse trajectories.

4.2 Story-cloze

We next evaluate our method on the story-cloze task presented by Mostafazadeh et al. (2016), which tests common-sense understanding in context of children stories. The task consists of identifying the correct ending to a four-sentence long story (called *context* in the original paper) and two possible ending options. The dataset for the task consists of a collection of around 45K unlabeled 5-sentence long stories as well as 3742 5-sentence stories with two provided endings, with one labeled as the correct ending. For this task, we train our method on grids of dimension $15 \times 15 \times 6$ (E), and histogram windows W of size 5×5 on the unlabeled collection of stories. At test time, for each story, we are provided two versions (a story-version v consists of the provided context c, followed by a possible ending e_1, i.e. $v = [c, e]$). For prediction, we need to define a goodness score S_v for a story-version.

In the simplest case, this score can simply be the log-likelihood $\log p_{SCG}(v)$ of the story-version, according to the Sequential-CG model. However, this is problematic since this is biased towards choosing shorter endings. To alleviate this, we define the goodness score by discounting the log-likelihood by the probability of the ending e itself, under a simple unigram model.

	Accuracy
Our Method variants	
Sequential CG + Unigram Mixture	0.602
Sequential CG + Brown clustering	0.593
Sequential CG + Sentiment	0.581
Sequential CG	0.589
Sequential CG (unnormalized)	0.531
DSSM	0.585
GenSim	0.539
Skip-thoughts	0.552
Narrative-Chain(Stories)	0.494
N-grams	0.494

Table 1: Performance of our approach on story-cloze task from Mostafazadeh et al. (2016) compared with other unsupervised approaches (accuracy numbers as reported in Mostafazadeh et al. (2016)).

$$S_v = \log p_{SCG}(c, e) - \log p_{uni}(e)$$

The predicted ending is the story-version with a higher score. Table 1 shows the performance of variants of our approach for the task. Our baselines include previous approaches for the same task: *DSSM* is a deep-learning based approach, which maps the context and ending to the same space, and is the best-performing method in Mostafazadeh et al. (2016). *GenSim* and *N-gram* return the ending that is more similar to the context based on *word2vec* embeddings (Mikolov et al., 2013) and n-grams, respectively. *Narrative-Chains* computes the probability of each alternative based on event-chains, following the approach of Chambers and Jurafsky (2008).

We note that our method improves on the previous best unsupervised methods for the task. This is quite surprising, since our Sequential-CG model in this case is trained on bag-of-lemma representations, and only needs sentence segmentation, tokenization and lemmatization for pre-processing. On the other hand, approaches such as *Narrative-Chains* require parsing and event-recognition, while approaches such as *GenSim* require learning word embeddings on large text corpora for training. Further, we note that predicting the ending without normalizing for the probability of the words in the ending results in significantly weaker performance, as expected. We train another

Mina lost her purse at a restaurant. She was so unhappy! She thought she would never get her things back. But then Mina got a wonderful surprise.

- A stranger had stolen her purse. **(-7.45)**
- A stranger had found her purse and returned it to the restaurant. **(-7.11)**

The Mills next door had a new car. It was stolen during the weekend. They came to my house and asked me if I knew anything. I told them I didn't, but for some reason they suspected me.

- They called the police to come to my house. **(6.69)**
- They liked me a lot after that. **(-0.94)**

Figure 4: Illustrative story-cloze examples where the model correctly identifies the appropriate ending (model score in parentheses).

	Accidents	*Earthquakes*
Sequential CG	0.813	0.946
VLV-GM (2017)	0.770	0.931
HMM (2012)	0.822	0.938
HMM+Entity (2012)	0.842	0.911
HMM+Content (2012)	0.742	0.953
Discriminative approaches		
DM (2017)	0.930	0.992
Recursive (2014)	0.864	0.976
Entity-Grid (2008)	0.904	0.872
Graph (2013)	0.846	0.635

Table 2: Performance of our approach on sentence ordering dataset from Barzilay and Lapata (2008).

variant of Sequential-CG with the sentence-level sentiment annotation (from Stanford CoreNLP) also added as a feature. This does not improve performance, consistent with findings in Mostafazadeh et al. (2016). We also experiment with a variant where we perform Brown clustering (Brown et al., 1992) of words in the unlabeled stories ($K = 500$ clusters), and include cluster-annotations as features for training the method. Doing this explicitly incorporates lexical similarity into the model, leading to a small improvement in performance. Finally, a mixture model consisting of the Sequential-CG and a unigram language model leads to a further improvement in performance. The performance of our unsupervised approach on this task indicates that it can learn discourse structures that are helpful for general language understanding.

The story-cloze task has recently also been addressed as a shared task at EACL (Mostafazadeh et al., 2017) with a significantly expanded dataset, and achieving much higher performance. However, we note that the proposed best-performing approaches (Chaturvedi et al., 2017; Schwartz et al., 2017) for this task are all supervised, and hence not included here for comparison.

Figure 4 shows examples where the model correctly identifies the ending. These show a mix of behavior such as sentiment coherence (identifying dissonance between 'wonderful surprise' and 'stolen') and modeling causation (police being called after being suspected).

4.3 Sentence Ordering

We next evaluate our method on the sentence ordering task, which requires distinguishing an original document from a version consisting of permutations of sentences of the original (Barzilay and Lapata, 2008; Louis and Nenkova, 2012). For this, we use two datasets of documents and their permutations from Barzilay and Lapata (2008), which are used as standard evaluation for coherence prediction tasks. These consist of (i) reports of accidents from the National Transportation Safety Bureau (we refer to this data as *accidents*), and (ii) newswire reports about earthquake events from the Associated press (we refer to this as *earthquakes*). Each dataset consists of 100 training documents, and about 100 documents for testing. Also provided are about 20 generated permutations for each document (resulting in 1986 test pairs for *accidents*, and 1955 test pairs for earthquakes). Documents in *accidents* consist of between 6 and 19 sentences each, with a median of 11 sentences. Documents in *earthquakes* consist of between 4 and 30 sentences each, with a median of 10 sentences.

Since the datasets for these tasks only have a relatively small number of training documents (100 each), we use Sequential-CG with smaller grids ($3 \times 3 \times 15$), and don't train a mixture model (which needs to learn a parameter β_z for each word in the vocabulary). Further, we train for a much smaller number of iterations to prevent overfitting ($K = 3$, chosen through cross-validation on the training set). During testing, since provided article pairs are simply permutations of each other and identical in content, we do not need to normalize as needed in Section 4.2. The score of a provided article is simply calculated as its log-likelihood. The article with higher likelihood is predicted to be the original.

Table 2 shows performance of the method compared with other approaches for coherence prediction. We note that Sequential-CG performs com-

TAIPEI, Taiwan (**AP**) An earthquake with a magnitude of 5.9 **jolted** Taiwan Friday.

The Central Weather Bureau recorded the quake at 11:17 a.m. (0317 GMT) and placed its **epicenter** in mountains 30 kilometers (18 miles) southeast of Taipei.

The quake was felt in part of northern and central Taiwan, and **shook** buildings in Taipei for a few seconds.

No damage or casualties were immediately reported.

The Central Weather Bureau said people in eastern Taiwan should prepare for **aftershocks.**

Figure 5: Example of newswire report about an earthquake event. Bold fonts represent words that align particularly well with the learned model at corresponding points in the narrative.

petitively with the state-of-the-art for generative approaches for the task, while needing no other annotation. In comparison, the HMM based approaches use significant annotation and syntactic features. Sequential-CG also outperforms several discriminative approaches for the task. In Figure 5 we illustrate the learned discourse trajectories in terms of the most salient features in each sentence. Words in bold are those identified by the model to be most context-appropriate at the corresponding point in the narrative. This is done by ranking words by the ratio between their probabilities ($\pi_{:,z}$) in the grid weighted by alignment locations of the document ($q^t_{\mathcal{I}}$), and unigram probabilities.

5 Conclusion

We have presented a simple model for extracting and visualizing latent discourse structure from unlabeled documents. The approach is coarse, and does not have explicit models for important elements such as entities and events in a discourse. However, the method outperforms some previous approaches on document understanding tasks, even while ignoring syntactic structure within sentences. The ability to visualize learning is a key component of our method, which can find significant applications in data mining and data-discovery in large text collections. More generally, similar approaches can explore a wider range of scenarios involving sequences of text. While here our focus was on learning discourse structures at the document level, similar methods can also be used at other scales, such as for syntactic or morphological analysis.

References

Nicholas Asher. 1986. Belief in discourse representation theory. *Journal of Philosophical Logic*, 15(2):127–189.

David Bamman and Noah Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics*, 2:363–376.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*, pages 789–797. The Association for Computer Linguistics.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614.

Snigdha Chaturvedi, Shashank Srivastava, and Dan Roth. 2018. 'Where have I heard this story before?' : Identifying narrative similarity in movie remakes. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thomas L Griffiths, Mark Steyvers, David M Blei, and Joshua B Tenenbaum. 2005. Integrating topics and syntax. In *Advances in neural information processing systems*, pages 537–544.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 93–103.

Nebojsa Jojic and Alessandro Perina. 2011. Multidimensional counting grids: Inferring word order from disordered bags of words. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 547–556. AUAI Press.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *ACL 2013*, page 119.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 198–209.

Jiwei Li, Rumeng Li, and Eduard H Hovy. 2014. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1157–1168. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT*, pages 839–849.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain.

Karl Pichotta and Raymond J Mooney. 2015. Learning statistical scripts with LSTM recurrent neural networks. In *AAAI*.

Roger C Schank and Robert P Abelson. 1977. Scripts, plans, goals and understanding: an inquiry into human knowledge structures. *Erlbaum*.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. *arXiv preprint arXiv:1702.01841*.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150, Copenhagen, Denmark. Association for Computational Linguistics.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 424–433, New York, NY, USA. ACM.

Joint Reasoning for Temporal and Causal Relations

Qiang Ning,[1] **Zhili Feng,**[2] **Hao Wu,**[3] **Dan Roth**[1,3]
Department of Computer Science
[1]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[2]University of Wisconsin-Madison, Madison, WI 53706, USA
[3]University of Pennsylvania, Philadelphia, PA 19104, USA
`qning2@illinois.edu, zfeng49@cs.wisc.edu, haowu4,danroth@seas.upenn.edu`

Abstract

Understanding temporal and causal relations between events is a fundamental natural language understanding task. Because a cause must occur earlier than its effect, temporal and causal relations are closely related and one relation often dictates the value of the other. However, limited attention has been paid to studying these two relations jointly. This paper presents a joint inference framework for them using constrained conditional models (CCMs). Specifically, we formulate the joint problem as an integer linear programming (ILP) problem, enforcing constraints that are inherent in the nature of time and causality. We show that the joint inference framework results in statistically significant improvement in the extraction of both temporal and causal relations from text.[1]

1 Introduction

Understanding events is an important component of natural language understanding. An essential step in this process is identifying relations between events, which are needed in order to support applications such as story completion, summarization, and timeline construction.

Among the many relation types that could exist between events, this paper focuses on the joint extraction of temporal and causal relations. It is well known that temporal and causal relations interact with each other and in many cases, the decision of one relation is made primarily based on evidence from the other. In Example 1, identifying the temporal relation between ***e1:died*** and ***e2:exploded*** is in fact a very hard case: There are no explicit temporal markers (e.g., "before", "after", or "since"); the events are in separate sentences so their syntactic connection is weak; although the occurrence time of ***e2:exploded*** is given (i.e., Friday) in text, it is not given for ***e1:died***. However, given the causal relation, ***e2:exploded*** caused ***e1:died***,it is clear that ***e2:exploded*** happened before ***e1:died***. The temporal relation is dictated by the causal relation.

Ex 1: Temporal relation dictated by causal relation.
More than 10 people ***(e1:died)*** on their way to the nearest hospital, police said. A suicide car bomb ***(e2:exploded)*** on Friday in the middle of a group of men playing volleyball in northwest Pakistan.
*Since **e2:exploded** is the reason of **e1:died**, the temporal relation is thus e2 being before e1.*

Ex 2: Causal relation dictated by temporal relation.
Mir-Hossein Moussavi ***(e3:raged)*** after government's efforts to ***(e4:stifle)*** protesters.
*Since **e3:raged** is temporally after **e4:stifle**, e4 should be the cause of e3.*

On the other hand, causal relation extraction can also benefit from knowing temporal relations. In Example 2, it is unclear whether the government stifled people because people raged, or people raged because the government stifled people: both situations are logically reasonable. However, if we account for the temporal relation (that is, ***e4:stifle*** happened before ***e3:raged***), it is clear that ***e4:stifle*** is the cause and ***e3:raged*** is the effect. In this case, the causal relation is dictated by the temporal relation.

The first contribution of this work is proposing a joint framework for **T**emporal and **C**ausal **R**easoning (TCR), inspired by these examples. Assuming the availability of a temporal extraction system and a causal extraction system, the proposed joint framework combines these two using a constrained conditional model (CCM) (Chang et al., 2012) framework, with an integer linear pro-

[1]The dataset and code used in this paper are available at `http://cogcomp.org/page/publication_view/835`

gramming (ILP) objective (Roth and Yih, 2004) that enforces declarative constraints during the inference phase. Specifically, these constraints include: (1) A cause must temporally precede its effect. (2) Symmetry constraints, i.e., when a pair of events, (A, B), has a temporal relation r (e.g., *before*), then (B, A) must have the reverse relation of r (e.g., *after*). (3) Transitivity constraints, i.e., the relation between (A, C) must be temporally consistent with the relation derived from (A, B) and (B, C). These constraints originate from the one-dimensional nature of time and the physical nature of causality and build connections between temporal and causal relations, making CCM a natural choice for this problem. As far as we know, very limited work has been done in joint extraction of both relations. Formulating the joint problem in the CCM framework is novel and thus the first contribution of this work.

A key obstacle in jointly studying temporal and causal relations lies in the absence of jointly annotated data. The second contribution of this work is the development of such a jointly annotated dataset which we did by augmenting the Event-Causality dataset (Do et al., 2011) with dense temporal annotations. This dataset allows us to show statistically significant improvements on both relations via the proposed joint framework.

This paper also presents an empirical result of improving the temporal extraction component. Specifically, we incorporate explicit time expressions present in the text and high-precision knowledge-based rules into the ILP objective. These sources of information have been successfully adopted by existing methods (Chambers et al., 2014; Mirza and Tonelli, 2016), but were never used within a global ILP-based inference method. Results on TimeBank-Dense (Cassidy et al., 2014), a benchmark dataset with temporal relations only, show that these modifications can also be helpful within ILP-based methods.

2 Related Work

Temporal and causal relations can both be represented by directed acyclic graphs, where the nodes are events and the edges are labeled with either *before, after*, etc. (in temporal graphs), or *causes* and *caused by* (in causal graphs). Existing work on *temporal* relation extraction was initiated by (Mani et al., 2006; Chambers et al., 2007; Bethard et al., 2007; Verhagen and Pustejovsky, 2008),

Ex 3: Global considerations are needed when making local decisions.

The FAA on Friday ***(e5:announced)*** it will close 149 regional airport control towers because of forced spending cuts. Before Friday's ***(e6:announcement)***, it ***(e7:said)*** it would consider keeping a tower open if the airport convinces the agency it is in the "national interest" to do so.

which formulated the problem as that of learning a classification model for determining the label of each edge locally (i.e., *local* methods). A disadvantage of these early methods is that the resulting graph may break the symmetric and transitive constraints. There are conceptually two ways to enforce such graph constraints (i.e., *global* reasoning). CAEVO (Chambers et al., 2014) grows the temporal graph in a multi-sieve manner, where predictions are added sieve-by-sieve. A graph closure operation had to be performed after each sieve to enforce constraints. This is solving the global inference problem greedily. A second way is to perform exact inference via ILP and the symmetry and transitivity requirements can be enforced as ILP constraints (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Denis and Muller, 2011; Do et al., 2012; Ning et al., 2017).

We adopt the ILP approach in the temporal component of this work for two reasons. First, as we show later, it is straightforward to build a joint framework with both temporal and causal relations as an extension of it. Second, the relation between a pair of events is often determined by the relations among other events. In Ex 3, if a system is unaware of $(e5, e6)$=*simultaneously* when trying to make a decision for $(e5, e7)$, it is likely to predict that $e5$ is *before* $e7$[2]; but, in fact, $(e5, e7)$=*after* given the existence of $e6$. Using global considerations is thus beneficial in this context not only for generating globally consistent temporal graphs, but also for making more reliable pairwise decisions.

Prior work on *causal* relations in *natural language text* was relatively sparse. Many causal extraction work in other domains assumes the existence of ground truth timestamps (e.g., (Sun et al., 2007; Güler et al., 2016)), but gold timestamps rarely exist in natural language text. In NLP, people have focused on causal relation identification using lexical features or discourse relations. For

[2]Consider the case that "The FAA ***e5:announced***…it ***e7:said*** it would…". Even humans may predict that $e5$ is *before* $e7$.

example, based on a set of explicit causal discourse markers (e.g., "because", "due to", and "as a result"), Hidey and McKeown (2016) built parallel Wikipedia articles and constructed an open set of implicit markers called AltLex. A classifier was then applied to identify causality. Dunietz et al. (2017) used the concept of construction grammar to tag causally related clauses or phrases. Do et al. (2011) considered global statistics over a large corpora, the cause-effect association (CEA) scores, and combined it with discourse relations using ILP to identify causal relations. These work only focused on the causality task and did not address the temporal aspect.

However, as illustrated by Examples 1-2, temporal and causal relations are closely related, as assumed by many existing works (Bethard and Martin, 2008; Rink et al., 2010). Here we argue that being able to capture both aspects in a joint framework provides a more complete understanding of events in natural language documents. Researchers have started paying attention to this direction recently. For example, Mostafazadeh et al. (2016b) proposed an annotation framework, CaTeRs, which captured both temporal and causal aspects of event relations in common sense stories. CATENA (Mirza and Tonelli, 2016) extended the multi-sieve framework of CAEVO to extracting both temporal and causal relations and exploited their interaction through post-editing temporal relations based on causal predictions. In this paper, we push this idea forward and tackle the problem in a joint and more principled way, as shown next.

3 Temporal and Causal Reasoning

In this section, we explain the proposed joint inference framework, **T**emporal and **C**ausal **R**easoning (TCR). To start with, we focus on introducing the temporal component, and clarify how to design the transitivity constraints and how to enforce other readily available prior knowledge to improve its performance. With this temporal component already explained, we further incorporate causal relations and complete the TCR joint inference framework. Finally, we transform the joint problem into an ILP so that it can be solved using off-the-shelf packages.

3.1 Temporal Component

Let $\mathcal{R}_T$ be the label set of temporal relations and $\mathcal{E}$ and $\mathcal{T}$ be the set of all events and the set of all time expressions (a.k.a. timex) in a document. For notation convenience, we use $\mathcal{EE}$ to represent the set of all event-event pairs; then $\mathcal{ET}$ and $\mathcal{TT}$ have obvious definitions. Given a pair in $\mathcal{EE}$ or $\mathcal{ET}$, assume for now that we have corresponding classifiers producing confidence scores for every temporal relation in $\mathcal{R}_T$. Let them be $s^{ee}(\cdot)$ and $s^{et}(\cdot)$, respectively. Then the inference formulation for all the temporal relations within this document is:

$$\hat{Y} = \arg\max_{Y\in\mathcal{Y}} \sum_{i\in\mathcal{EE}} s^{ee}\{i \mapsto Y_i\} + \sum_{j\in\mathcal{ET}} s^{et}\{j \mapsto Y_j\} \quad (1)$$

where $Y_k \in \mathcal{R}_T$ is the temporal label of pair $k \in \mathcal{MM}$ (Let $\mathcal{M} = \mathcal{E} \cup \mathcal{T}$ be the set of all temporal nodes), "$k \mapsto Y_k$" represents the case where the label of pair k is predicted to be Y_k, Y is a vectorization of all these Y_k's in one document, and $\mathcal{Y}$ is the constrained space that Y lies in.

We do not include the scores for $\mathcal{TT}$ because the temporal relationship between timexes can be trivially determined using the normalized dates of these timexes, as was done in (Do et al., 2012; Chambers et al., 2014; Mirza and Tonelli, 2016). We impose these relations via equality constraints denoted as $\mathcal{Y}_0$. In addition, we add symmetry and transitivity constraints dictated by the nature of time (denoted by $\mathcal{Y}_1$), and other prior knowledge derived from linguistic rules (denoted by $\mathcal{Y}_2$), which will be explained subsequently. Finally, we set $\mathcal{Y} = \cap_{i=0}^{2}\mathcal{Y}_i$ in Eq. (1).

Transitivity Constraints. Let the dimension of Y be n. Then a standard way to construct the symmetry and transitivity constraints is shown in (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Denis and Muller, 2011; Do et al., 2012; Ning et al., 2017)

$$\mathcal{Y}_1 = \big\{Y \in \mathcal{R}_T^n | \forall m_{1,2,3} \in \mathcal{M}, Y_{(m_1,m_2)} = \bar{Y}_{(m_2,m_1)}, \\ Y_{(m_1,m_3)} \in \text{Trans}(Y_{(m_1,m_2)}, Y_{(m_2,m_3)})\big\}$$

where the bar sign is used to represent the reverse relation hereafter, and Trans(r_1, r_2) is a set comprised of all the temporal relations from $\mathcal{R}_T$ that do not conflict with r_1 and r_2.

The construction of Trans(r_1, r_2) necessitates a clearer definition of $\mathcal{R}_T$, the importance of which is often overlooked by existing methods. Existing approaches all followed the interval representation of events (Allen, 1984), which yields 13 temporal relations (denoted by $\tilde{\mathcal{R}}_T$ here) as shown in Fig. 1. Most systems used a reduced set, for ex-

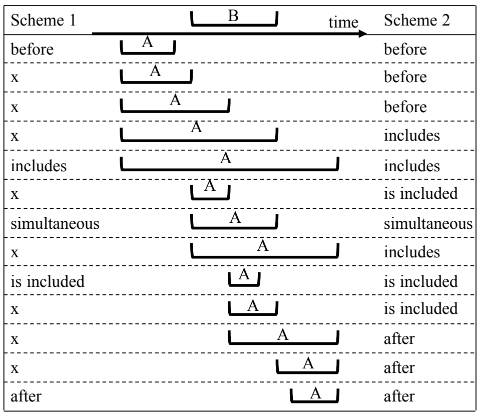


Figure 1: **Two possible interpretations to the label set of** $\mathcal{R}_T = \{b, a, i, ii, s, v\}$ for the temporal relations between (A, B). "x" means that the label is ignored. Brackets represent time intervals along the time axis. Scheme 2 is adopted consistently in this work.

No.	r_1	r_2	Trans(r_1, r_2)
1	r	r	r
2	r	**s**	r
3	r_1	r_2	$\overline{\text{Trans}(\bar{r}_2, \bar{r}_1)}$
4	**b**	**i**	**b**, **i**, **v**
5	**b**	**ii**	**b**, **ii**, **v**
6	**b**	**v**	**b**, **i**, **ii**, **v**
7	**a**	**i**	**a**, **i**, **v**
8	**a**	**ii**	**a**, **ii**, **v**
9	**a**	**v**	**a**, **i**, **ii** ,**v**
10	**i**	**v**	**b**, **a**, **i**, **v**
11	**ii**	**v**	**b**, **a**, **ii**, **v**

Table 1: **Transitivity relations** based on the label set reduction scheme 2 in Fig. 1. If $(m_1, m_2) \mapsto r_1$ and $(m_2, m_3) \mapsto r_2$, then the relation of (m_1, m_3) must be chosen from Trans(r_1, r_2), $\forall m_1, m_2, m_3 \in \mathcal{M}$. The top part of the table uses r to represent generic rules compactly. Notations: before (**b**), after (**a**), includes (**i**), is included (**ii**), simultaneously (**s**), vague (**v**), and $\bar{r}$ represents the reverse relation of r.

ample, {*before*, *after*, *includes*, *is included*, *simultaneously*, *vague*}. For notation convenience, we denote them $\mathcal{R}_T = \{b, a, i, ii, s, v\}$. Using a reduced set is more convenient in data annotation and leads to better performance in practice.

However, there has been limited discussion in the literature on how to interpret the reduced relation types. For example, is the "*before*" in $\mathcal{R}_T$ exactly the same as the "*before*" in the original set ($\tilde{\mathcal{R}}_T$) (as shown on the left-hand-side of Fig. 1), or is it a combination of multiple relations in $\tilde{\mathcal{R}}_T$ (the right-hand-side of Fig. 1)? We compare two reduction schemes in Fig. 1, where scheme 1 ignores low frequency labels directly and scheme 2 absorbs low frequency ones into their temporally closest labels. The two schemes barely have differences when a system only looks at a single pair of mentions at a time (this might explain the lack of discussion over this issue in the literature), but they lead to different Trans(r_1, r_2) sets and this difference can be magnified when the problem is solved jointly and when the label distribution changes across domains. To completely cover the 13 relations, we adopt scheme 2 in this work.

The resulting transitivity relations are shown in Table 1. The top part of Table 1 is a compact representation of three generic rules; for instance, Line 1 means that the labels themselves are transitive. Note that during human annotation, if an annotator looks at a pair of events and decides that multiple well-defined relations can exist, he/she labels it *vague*; also, when aggregating the labels from multiple annotators, a label will be changed to *vague* if the annotators disagree with each other. In either case, *vague* is chosen to be the label when a single well-defined relation cannot be uniquely determined by the contextual information. This explains why a *vague* relation (v) is always added in Table 1 if more than one label in Trans(r_1, r_2) is possible. As for Lines 6, 9-11 in Table 1 (where *vague* appears in Column r_2), Column Trans(r_1,r_2) was designed in such a way that r_2 cannot be uniquely determined through r_1 and Trans(r_1,r_2). For instance, r_1 is *after* on Line 9, if we further put *before* into Trans(r_1,r_2), then r_2 would be uniquely determined to be *before*, conflicting with r_2 being *vague*, so *before* should not be in Trans(r_1,r_2).

Enforcing Linguistic Rules. Besides the transitivity constraints represented by $\mathcal{Y}_1$ above, we also propose to enforce prior knowledge to further constrain the search space for Y. Specifically, linguistic insight has resulted in rules for predicting the temporal relations with special syntactic or semantic patterns, as was done in CAEVO (a state-of-the-art method). Since these rule predictions often have high-precision, it is worthwhile incorporating them in global reasoning methods as well.

In the CCM framework, these rules can be represented as hard constraints in the search space for Y. Specifically,

$$\mathcal{Y}_2 = \left\{Y_j = rule(j), \forall j \in \mathcal{J}^{(rule)}\right\}, \tag{2}$$

where $\mathcal{J}^{(rule)} \subseteq \mathcal{MM}$ is the set of pairs that can be determined by linguistic rules, and $rule(j) \in$

$\mathcal{R}_T$ is the corresponding decision for pair j according to these rules. In this work, we used the same set of rules designed by CAEVO for fair comparison.

3.2 Full Model with Causal Relations

Now we have presented the joint inference framework for temporal relations in Eq. (1). It is easier to explain our complete TCR framework on top of it. Let W be the vectorization of all causal relations and add the scores from the scoring function for causality $s^c(\cdot)$ to Eq. (1). Specifically, the full inference formulation is now:

$$\hat{Y}, \hat{W} = \arg \max_{Y \in \mathcal{Y}, W \in \mathcal{W}_Y} \sum_{i \in \mathcal{EE}} s^{ee}\{i \mapsto Y_i\} \tag{3}$$
$$+ \sum_{j \in \mathcal{ET}} s^{et}\{j \mapsto Y_j\} + \sum_{k \in \mathcal{EE}} s^c\{k \mapsto W_k\}$$

where $\mathcal{W}_Y$ is the search space for W. $\mathcal{W}_Y$ depends on the temporal labels Y in the sense that

$$\mathcal{W}_Y = \{W \in \mathcal{R}_C^m | \forall i, j \in \mathcal{E}, \text{if } W_{(i,j)} = c, \text{ then } W_{(j,i)} = \bar{c}, \text{ and } Y_{(i,j)} = b\} \tag{4}$$

where m is the dimension of W (i.e., the total number of causal pairs), $\mathcal{R}_C = \{c, \bar{c}, null\}$ is the label set for causal relations (i.e., "causes", "caused by", and "no relation"), and $W_{(i,j)}$ is the causal label for pair (i, j). The constraint represented by $\mathcal{W}_Y$ means that if a pair of events i and j are labeled to be "causes", then the causal relation between j and i must be "caused by", and the temporal relation between i and j must be "before".

3.3 Scoring Functions

In the above, we have built the joint framework on top of scoring functions $s^{ee}(\cdot)$, $s^{et}(\cdot)$ and $s^c(\cdot)$. To get $s^{ee}(\cdot)$ and $s^{et}(\cdot)$, we trained classifiers using the averaged perceptron algorithm (Freund and Schapire, 1998) and the same set of features used in (Do et al., 2012; Ning et al., 2017), and then used the soft-max scores in those scoring functions. For example, that means

$$s^{ee}\{i \mapsto r\} = \frac{w_r^T \phi(i)}{\sum_{r' \in \mathcal{R}_T} w_{r'}^T \phi(i)}, \; i \in \mathcal{EE}, \; r \in \mathcal{R}_T,$$

where $\{w_r\}$ is the learned weight vector for relation $r \in \mathcal{R}_T$ and $\phi(i)$ is the feature vector for pair $i \in \mathcal{EE}$.

Given a pair of ordered events, we need $s^c(\cdot)$ to estimate the scores of them being "causes" or "caused by". Since this scoring function has the same nature as $s^{ee}(\cdot)$, we can reuse the features from $s^{ee}(\cdot)$ and learn an averaged perceptron for $s^c(\cdot)$. In addition to these existing features, we also use prior statistics retrieved using our temporal system from a large corpus[3], so as to know *probabilistically* which event happens before another event. For example, in Example 1, we have a pair of events, ***e1:died*** and ***e2:exploded***. The prior knowledge we retrieved from that large corpus is that ***die*** happens before ***explode*** with probability 15% and happens after ***explode*** with probability 85%. We think this prior distribution is correlated with causal directionality, so it was also added as features when training $s^c(\cdot)$.

Note that the scoring functions here are implementation choice. The TCR joint framework is fully extensible to other scoring functions.

3.4 Convert the Joint Inference into an ILP

Conveniently, the joint inference formulation in Eq. (3) can be rewritten into an ILP and solved using off-the-shelf optimization packages, e.g., (Gurobi Optimization, Inc., 2012). First, we define indicator variables $y_i^r = \mathbb{I}\{Y_i = r\}$, where $\mathbb{I}\{\cdot\}$ is the indicator function, $\forall i \in \mathcal{MM}$, $\forall r \in \mathcal{R}_T$. Then let $p_i^r = s^{ee}\{i \mapsto r\}$ if $i \in \mathcal{EE}$, or $p_i^r = s^{et}\{i \mapsto r\}$ if $i \in \mathcal{ET}$; similarly, let $w_j^r = \mathbb{I}\{W_i = r\}$ be the indicator variables for W_j and q_j^r be the score for $W_j = r \in \mathcal{R}_C$. Therefore, without constraints $\mathcal{Y}$ and $\mathcal{W}_Y$ for now, Eq. (3) can be written as:

$$\hat{y}, \hat{w} = \arg\max \sum_{i \in \mathcal{EE} \cup \mathcal{ET}} \sum_{r \in \mathcal{R}_T} p_i^r y_i^r + \sum_{j \in \mathcal{EE}} \sum_{r \in \mathcal{R}_C} q_j^r w_j^r$$
$$\text{s.t.} \quad y_i^r, w_j^r \in \{0, 1\}, \sum_{r \in \mathcal{R}_T} y_i^r = \sum_{r \in \mathcal{R}_C} w_j^r = 1$$

The prior knowledge represented as $\mathcal{Y}$ and $\mathcal{W}_Y$ can be conveniently converted into constraints for this optimization problem. Specifically, $\mathcal{Y}_1$ has two components, symmetry and transitivity:

$$\mathcal{Y}_1: \quad \forall i, j, k \in \mathcal{M}, \; y_{i,j}^r = y_{j,i}^{\bar{r}}, \text{ (symmetry)}$$
$$y_{i,j}^{r_1} + y_{j,k}^{r_2} - \sum_{r_3 \in \text{Trans}(r_1, r_2)} y_{i,k}^{r_3} \leq 1 \text{ (transitivity)}$$

where $\bar{r}$ is the reverse relation of r (i.e., $\bar{b} = a$, $\bar{i} = ii$, $\bar{s} = s$, and $\bar{v} = v$), and Trans(r_1, r_2) is defined in Table 1. As for the transitivity constraints,

[3]https://catalog.ldc.upenn.edu/LDC2008T19, which is disjoint to the test set used here. Please refer to (Ning et al., 2018a) for more analysis on using this corpus to acquire prior knowledge that aids temporal relation classification.

if both $y_{i,j}^{r_1}$ and $y_{j,k}^{r_2}$ are 1, then the constraint requires at least one of $y_{i,k}^{r_3}, r_3 \in \text{Trans}(r_1, r_2)$ to be 1, which means the relation between i and k has to be chosen from $\text{Trans}(r_1, r_2)$, which is exactly what $\mathcal{Y}_1$ is intended to do.

The rules in $\mathcal{Y}_2$ is written as

$$\mathcal{Y}_2 : y_j^r = \mathbb{I}_{\{rule(j)=r\}}, \forall j \in \mathcal{J}^{(rule)} \text{ (linguistic rules)}$$

where $rule(j)$ and $\mathcal{J}^{(rule)}$ have been defined in Eq. (2). Converting the $\mathcal{TT}$ constraints, i.e., $\mathcal{Y}_0$, into constraints is as straightforward as $\mathcal{Y}_2$, so we omit it due to limited space.

Last, converting the constraints $\mathcal{W}_Y$ defined in Eq. (4) can be done as following:

$$\mathcal{W}_Y : w_{i,j}^c = w_{j,i}^{\bar{c}} \leq y_{i,j}^b, \forall i, j \in \mathcal{E}.$$

The equality part, $w_{i,j}^c = w_{j,i}^{\bar{c}}$ represents the symmetry constraint of causal relations; the inequality part, $w_{i,j}^c \leq y_{i,j}^b$ represents that if event i causes event j, then i must be before j.

4 Experiments

In this section, we first show on TimeBank-Dense (TB-Dense) (Cassidy et al., 2014), that the proposed framework improves temporal relation identification. We then explain how our new dataset with both temporal and causal relations was collected, based on which the proposed method improves for both relations.

4.1 Temporal Performance on TB-Dense

Multiple datasets with temporal annotations are available thanks to the TempEval (TE) workshops (Verhagen et al., 2007, 2010; UzZaman et al., 2013). The dataset we used here to demonstrate our improved temporal component was TB-Dense, which was annotated on top of 36 documents out of the classic TimeBank dataset (Pustejovsky et al., 2003). The main purpose of TB-Dense was to alleviate the known issue of sparse annotations in the evaluation dataset provided with TE3 (UzZaman et al., 2013), as pointed out in many previous work (Chambers, 2013; Cassidy et al., 2014; Chambers et al., 2014; Ning et al., 2017). Annotators of TB-Dense were forced to look at each pair of events or timexes within the same sentence or contiguous sentences, so that much fewer links were missed. Since causal link annotation is not available on TB-Dense, we only show our improvement in terms of temporal performance on TB-Dense. The standard train/dev/test split of TB-Dense was used and parameters were tuned to optimize the F_1 performance on dev. Gold events and time expressions were also used as in existing systems.

#	System	P	R	F_1
	Ablation Study			
1	Baseline	39.1	56.8	46.3
2	+Transitivity[†]	42.9	54.9	48.2
3	+$\mathcal{ET}$	44.3	54.8	49.0
4	+Rules	45.4	58.7	51.2
5	+Causal	**45.8**	**60.5**	**52.1**
	Existing Systems[‡]			
6	ClearTK	53.0	26.4	35.2
7	CAEVO	**56.0**	41.6	47.8
8	Ning et al. (2017)	47.1	**53.3**	**50.0**

[†]This is technically the same with Do et al. (2012), or Ning et al. (2017) without its structured learning component.
[‡]We added gold $\mathcal{TT}$ to both gold and system prediction. Without this, Systems 6-8 had F_1=28.7, 45.7, and 48.5, respectively, same with the reported values in Ning et al. (2017).

Table 2: **Ablation study of the proposed system in terms of the standard temporal awareness metric**. The baseline system is to make inference locally for each event pair without looking at the decisions from others. The "+" signs on lines 2-5 refer to adding a new source of information on top of its preceding system, with which the inference has to be global and done via ILP. All systems are significantly different to its preceding one with p<0.05 (McNemar's test).

The contributions of each proposed information sources are analyzed in the ablation study shown in Table 2, where we can see the F_1 score was improved step-by-step as new sources of information were added. Recall that $\mathcal{Y}_1$ represents transitivity constraints, $\mathcal{ET}$ represents taking event-timex pairs into consideration, and $\mathcal{Y}_2$ represents rules from CAEVO (Chambers et al., 2014). System 1 is the baseline we are comparing to, which is a local method predicting temporal relations one at a time. System 2 only applied $\mathcal{Y}_1$ via ILP on top of all $\mathcal{EE}$ pairs by removing the 2nd term in Eq. (1); for fair comparison with System 1, we added the same $\mathcal{ET}$ predictions from System 1. System 3 incorporated $\mathcal{ET}$ into the ILP and mainly contributed to an increase in precision (from 42.9 to 44.3); we think that there could be more gain if more time expressions existed in the testset. With the help of additional high-precision rules ($\mathcal{Y}_2$), the temporal performance can further be improved, as shown by System 4. Finally, using the causal extraction obtained via (Do et al., 2011) in the joint framework, the proposed method achieved the best precision, recall, and F_1 scores in our ablation study (Systems 1-5). According to the McNemar's test (Everitt, 1992; Dietterich, 1998), all Systems 2-5 were significantly different to its preceding system with $p<0.05$.

The second part of Table 2 compares several state-of-the-art systems on the same test set. ClearTK (Bethard, 2013) was the top performing system in TE3 in temporal relation extraction. Since it was designed for TE3 (not TB-Dense), it expectedly achieved a moderate recall on the test set of TB-Dense. CAEVO (Chambers et al., 2014) and Ning et al. (2017) were more recent methods and achieved better scores on TB-Dense. Compared with these state-of-the-art methods, the proposed joint system (System 5) achieved the best F_1 score with a major improvement in recall. We think the low precision compared to System 8 is due to the lack of structured learning, and the low precision compared to System 7 is propagated from the baseline (System 1), which was tuned to maximize its F_1 score. However, the effectiveness of the proposed information sources is already justified in Systems 1-5.

4.2 Joint Performance on Our New Dataset

4.2.1 Data Preparation

TB-Dense only has temporal relation annotations, so in the evaluations above, we only evaluated our temporal performance. One existing dataset with both temporal and causal annotations available is the Causal-TimeBank dataset (Causal-TB) (Mirza and Tonelli, 2014). However, Causal-TB is sparse in temporal annotations and is even sparser in causal annotations: In Table 3, we can see that with four times more documents, Causal-TB still has fewer temporal relations (denoted as T-Links therein), compared to TB-Dense; as for causal relations (C-Links), it has less than two causal relations in each document on average. Note that the T-Link sparsity of Causal-TB originates from TimeBank, which is known to have missing links (Cassidy et al., 2014; Ning et al., 2017). The C-Link sparsity was a design choice of Causal-TB in which C-Links were annotated based on only explicit causal markers (e.g., "A happened *because* of B").

Another dataset with parallel annotations is CaTeRs (Mostafazadeh et al., 2016b), which was primarily designed for the Story Cloze Test (Mostafazadeh et al., 2016a) based on common sense stories. It is different to the newswire domain that we are working on. Therefore, we decided to augment the EventCausality dataset provided in Do et al. (2011) with a modified version of the dense temporal annotation scheme proposed in Cassidy et al. (2014) and use this new dataset to showcase the proposed joint approach.

	Doc	Event	T-Link	C-Link
TB-Dense	36	1.6k	5.7k	-
EventCausality	25	0.8k	-	580
Causal-TB	183	6.8k	5.1k	318
New Dataset	25	1.3k	3.4k	172

Table 3: **Statistics of our new dataset with both temporal and causal relations annotated**, compared with existing datasets. T-Link: Temporal relation. C-Link: Causal relation. The new dataset is much denser than Causal-TB in both T-Links and C-Links.

The EventCausality dataset provides relatively dense causal annotations on 25 newswire articles collected from CNN in 2010. As shown in Table 3, it has more than 20 C-Links annotated per document on average (10 times denser than Causal-TB). However, one issue is that its notion for events is slightly different to that in the temporal relation extraction regime. To construct parallel annotations of both temporal and causal relations, we preprocessed all the articles in EventCausality using ClearTK to extract events and then manually removed some obvious errors in them. To annotate temporal relations among these events, we adopted the annotation scheme from TB-Dense given its success in mitigating the issue of missing annotations with the following modifications. First, we used a crowdsourcing platform, CrowdFlower, to collect temporal relation annotations. For each decision of temporal relation, we asked 5 workers to annotate and chose the majority label as our final annotation. Second, we discovered that comparisons involving ending points of events tend to be ambiguous and suffer from low inter-annotator agreement (IAA), so we asked the annotators to label relations based on the starting points of each event. This simplification does not change the nature of temporal relation extraction but leads to better annotation quality. For more details about this data collection scheme, please refer to (Ning et al., 2018b) for more details.

4.2.2 Results

Result on our new dataset jointly annotated with both temporal and causal relations is shown in Ta-

	Temporal			Causal
	P	R	F_1	Accuracy
1. Temporal Only	67.2	72.3	69.7	-
2. Causal Only	-	-	-	70.5
3. Joint System	**68.6**	**73.8**	**71.1**	**77.3**
Enforcing Gold Relations in Joint System				
4. Gold Temporal	100	100	100	*91.9*
5. Gold Causal	*69.3*	*74.4*	*71.8*	100

Table 4: **Comparison between the proposed method and existing ones, in terms of both temporal and causal performances**. See Sec. 4.2.1 for description of our new dataset. Per the McNemar's test, the joint system is significantly better than both baselines with $p<0.05$. Lines 4-5 provide the best possible performance the joint system could achieve if gold temporal/causal relations were given.

ble 4. We split the new dataset into 20 documents for training and 5 documents for testing. In the training phase, the training parameters were tuned via 5-fold cross validation on the training set.

Table 4 demonstrates the improvement of the joint framework over individual components. The "temporal only" baseline is the improved temporal extraction system for which the joint inference with causal links has NOT been applied. The "causal only" baseline is to use $s^c(\cdot)$ alone for the prediction of each pair. That is, for a pair i, if $s^c\{i \mapsto \text{causes}\} > s^c\{i \mapsto \text{caused by}\}$, we then assign "causes" to pair i; otherwise, we assign "caused by" to pair i. Note that the "causal accuracy" column in Table 4 was evaluated only on gold causal pairs.

In the proposed joint system, the temporal and causal scores were added up for all event pairs. The temporal performance got strictly better in precision, recall, and F_1, and the causal performance also got improved by a large margin from 70.5% to 77.3%, indicating that temporal signals and causal signals are helpful to each other. According to the McNemar's test, both improvements are significant with $p<0.05$.

The second part of Table 4 shows that if gold relations were used, how well each component would possibly perform. Technically, these gold temporal/causal relations were enforced via adding extra constraints to ILP in Eq. (3) (imagine these gold relations as a special rule, and convert them into constraints like what we did in Eq. (2)). When using gold temporal relations, causal accuracy went up to 91.9%. That is, 91.9% of the C-Links satisfied the assumption that the cause is temporally before the effect. First, this number is much higher than the 77.3% on line 3, so there is still room for improvement. Second, it means in this dataset, there were 8.1% of the C-Links in which the cause is temporally *after* its effect. We will discuss this seemingly counter-intuitive phenomenon in the Discussion section. When gold causal relations were used (line 5), the temporal performance was slightly better than line 3 in terms of both precision and recall. The small difference means that the temporal performance on line 3 was already very close to its best. Compared with the first line, we can see that gold causal relations led to approximately 2% improvement in precision and recall in temporal performance, which is a reasonable margin given the fact that C-Links are often much sparser than T-Links in practice.

Note that the temporal performance in Table 4 is consistently better than those in Table 2 because of the higher IAA in the new dataset. However, the improvement brought by joint reasoning with causal relations is the same, which further confirms the capability of the proposed approach.

5 Discussion

We have consistently observed that on the TB-Dense dataset, if automatically tuned to optimize its F_1 score, a system is very likely to have low precisions and high recall (e.g., Table 2). We notice that our system often predicts non-vague relations when the TB-Dense gold is vague, resulting in lower precision. However, on our new dataset, the same algorithm can achieve a more balanced precision and recall. This is an interesting phenomenon, possibly due to the annotation scheme difference which needs further investigation.

The temporal improvements in both Table 2 and Table 4 are relatively small (although statistically significant). This is actually not surprising because C-Links are much fewer than T-Links in newswires which focus more on the temporal development of stories. As a result, many T-Links are not accompanied with C-Links and the improvements are diluted. But for those event pairs having both T-Links and C-Links, the proposed joint framework is an important scheme to synthesize both signals and improve both. The comparison between Line 5 and Line 3 in Table 4 is a showcase of the effectiveness. We think that a deeper reason for the improvement achieved via a joint framework is that causality often encodes

Ex 4: Cause happened after effect.
The shares fell to a record low of ¥60 and ***(e8:finished)*** at ¥67 before the market ***(e9:closed)*** for the New Year holidays.
As she ***(e10:prepares)*** to ***(e11:host)*** her first show, Crowley writes on what viewers should expect.

humans prior knowledge as global information (e.g., "death" is *caused by* "explosion" rather than *causes* "explosion", regardless of the local context), while temporality often focuses more on the local context. From this standpoint, temporal information and causal information are complementary and helpful to each other.

When doing error analysis for the fourth line of Table 4, we noticed some examples that break the commonly accepted temporal precedence assumption. It turns out that they are not annotation mistakes: In Example 4, ***e8:finished*** is obviously *before* ***e9:closed***, but ***e9*** is a cause of ***e8*** since if the market did not close, the shares would not finish. In the other sentence of Example 4, she prepares *before* hosting her show, but ***e11:host*** is the cause of ***e10:prepares*** since if not for hosting, no preparation would be needed. In both cases, the cause is temporally after the effect because people are inclined to make projections for the future and change their behaviors before the future comes. The proposed system is currently unable to handle these examples and we believe that a better definition of what can be considered as events is needed, as part of further investigating how causality is expressed in natural language.

Finally, the constraints connecting causal relations to temporal relations are designed in this paper as "if A is the cause of B, then A must be *before* B". People have suggested other possibilities that involve the *includes* and *simultaneously* relations. While these other relations are simply different interpretations of temporal precedence (and can be easily incorporated in our framework), we find that they rarely happen in our dataset.

6 Conclusion

We presented a novel joint framework, **T**emporal and **C**ausal **R**easoning (TCR), using CCMs and ILP to the extraction problem of temporal and causal relations between events. To show the benefit of TCR, we have developed a new dataset that jointly annotates temporal and causal annotations, and then exhibited that TCR can improve both temporal and causal components. We hope that this notable improvement can foster more interest in jointly studying multiple aspects of events (e.g., event sequencing, coreference, parent-child relations) towards the goal of understanding events in natural language.

Acknowledgements

We thank all the reviewers for providing insightful comments and critiques. This research is supported in part by a grant from the Allen Institute for Artificial Intelligence (allenai.org); the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network; by DARPA under agreement number FA8750-13-2-0008; and by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053 (the ARL Network Science CTA).

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, of the Army Research Laboratory or the U.S. Government. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the ARL.

References

James F Allen. 1984. Towards a general theory of action and time. *Artificial intelligence* 23(2):123–154.

Steven Bethard. 2013. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *SemEval*. volume 2, pages 10–14.

Steven Bethard and James H Martin. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*. Association for Computational Linguistics, pages 177–180.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *IEEE International Conference on Semantic Computing (ICSC)*. pages 11–18.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. pages 189–198.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 501–506.

Nathanael Chambers. 2013. NavyTime: Event and time ordering from raw text. In *SemEval*. volume 2, pages 73–77.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics* 2:273–284.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. pages 173–176.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine Learning* 88(3):399–431.

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. volume 22, page 1788.

Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation* 10(7):1895–1923.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Edinburgh, Scotland.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017. Automatically tagging constructions of causation and their slot-fillers. *Transactions of the Association for Computational Linguistics* 5:117–133.

Brian S Everitt. 1992. *The analysis of contingency tables*. CRC Press.

Yoav Freund and Robert E. Schapire. 1998. Large margin classification using the Perceptron algorithm. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory (COLT)*. pages 209–217.

Başak Güler, Aylin Yener, and Ananthram Swami. 2016. Learning causal information flow structures in multi-layer networks. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. pages 1340–1344.

Gurobi Optimization, Inc. 2012. Gurobi optimizer reference manual. `http://www.gurobi.com`.

Christopher Hidey and Kathy McKeown. 2016. Identifying causal relations using parallel wikipedia articles. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 753–760.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *Proceedings the International Conference on Computational Linguistics (COLING)*. pages 2097–2106.

Paramita Mirza and Sara Tonelli. 2016. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *The 26th International Conference on Computational Linguistics*. pages 64–75.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016a. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*. pages 839–849.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016b. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*. pages 51–61.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. Copenhagen, Denmark.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018a. Improving temporal relation extraction with a globally acquired statistical resource. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Qiang Ning, Hao Wu, and Dan Roth. 2018b. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The TIMEBANK corpus. In *Corpus linguistics*. volume 2003, page 40.

Bryan Rink, Cosmin Adrian Bejan, and Sanda M Harabagiu. 2010. Learning textual graph patterns to detect causal event relations. In *FLAIRS Conference*.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*. pages 1–8.

Yizhou Sun, Kunqing Xie, Ning Liu, Shuicheng Yan, Benyu Zhang, and Zheng Chen. 2007. Causal relation of queries from temporal logs. In *The International World Wide Web Conference*. pages 1141–1142.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics*. volume 2, pages 1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval temporal relation identification. In *SemEval*. pages 75–80.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*. pages 189–192.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *SemEval*. pages 57–62.

Modeling Naive Psychology of Characters in Simple Commonsense Stories

Hannah Rashkin[†], **Antoine Bosselut**[†], **Maarten Sap**[†], **Kevin Knight**[‡] and **Yejin Choi**[†§]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[§]Allen Institute for Artificial Intelligence
{hrashkin,msap,antoineb,yejin}@cs.washington.edu
[‡] Information Sciences Institute & Computer Science, University of Southern California
knight@isi.edu

Abstract

Understanding a narrative requires reading between the lines and reasoning about the unspoken but obvious implications about events and people's mental states — a capability that is trivial for humans but remarkably hard for machines. To facilitate research addressing this challenge, we introduce a new annotation framework to explain naive psychology of story characters as fully-specified chains of mental states with respect to *motivations* and *emotional reactions*. Our work presents a new large-scale dataset with rich low-level annotations and establishes baseline performance on several new tasks, suggesting avenues for future research.

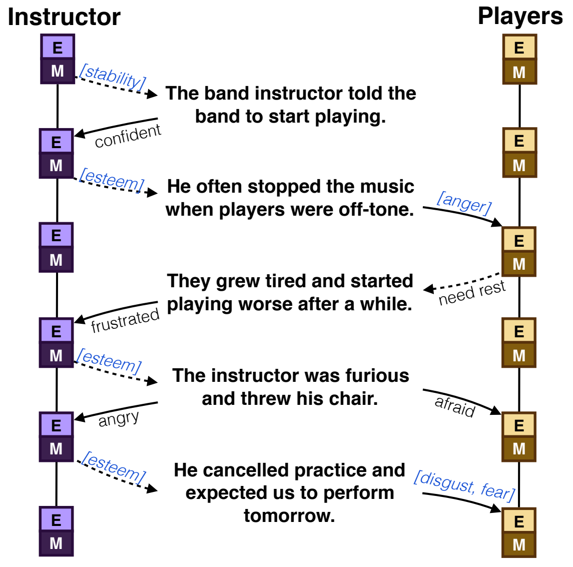

Instructor
Players
E
M
[stability]
The band instructor told the band to start playing.
confident
[esteem]
He often stopped the music when players were off-tone.
[anger]
They grew tired and started playing worse after a while.
need rest
frustrated
[esteem]
The instructor was furious and threw his chair.
afraid
angry
[esteem]
He cancelled practice and expected us to perform tomorrow.
[disgust, fear]

Figure 1: A story example with partial annotations for motivations (dashed) and emotional reactions (solid). Open text explanations are in black (e.g., "frustrated") and formal theory labels are in blue with brackets (e.g., "[esteem]").

1 Introduction

Understanding a story requires reasoning about the causal links between the events in the story and the mental states of the characters, even when those relationships are not explicitly stated. As shown by the commonsense story cloze shared task (Mostafazadeh et al., 2017), this reasoning is remarkably hard for both statistical and neural machine readers – despite being trivial for humans. This stark performance gap between humans and machines is not surprising as most powerful language models have been designed to effectively learn local fluency patterns. Consequently, they generally lack the ability to abstract away from surface patterns in text to model more complex implied dynamics, such as intuiting characters' mental states or predicting their plausible next actions.

In this paper, we construct a new annotation formalism to densely label commonsense short stories (Mostafazadeh et al., 2016) in terms of the mental states of the characters. The resulting dataset offers three unique properties. First, as highlighted in Figure 1, the dataset provides a fully-specified chain of *motivations* and *emotional reactions* for each story character as pre- and post-conditions of events. Second, the annotations include state changes for entities even when they are not mentioned directly in a sentence (e.g., in the fourth sentence in Figure 1, players would feel *afraid* as a result of the instructor throwing a chair), thereby capturing implied effects unstated in the story. Finally, the annotations encompass both formal labels from multiple theories of psychology (Maslow, 1943; Reiss, 2004; Plutchik, 1980) as well as open text descriptions of motivations and emotions, providing a comprehensive mapping between open text explanations and label categories (e.g., "to spend time with her son"

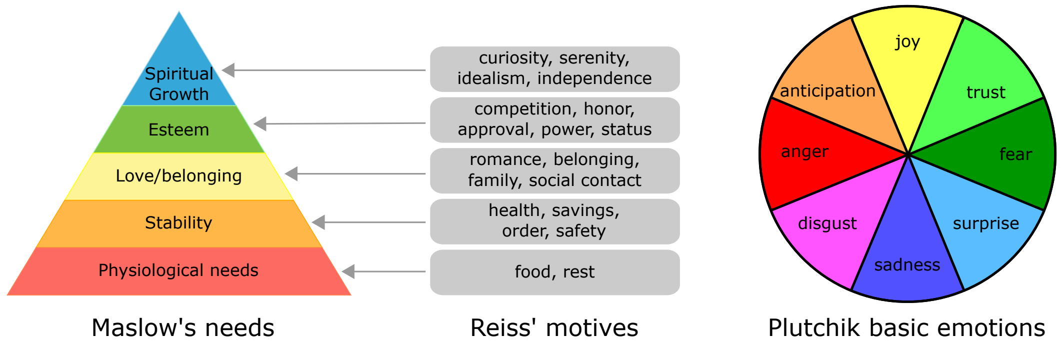


Figure 2: Theories of Motivation (Maslow and Reiss) and Emotional Reaction (Plutchik).

→ Maslow's category *love*). Our corpus[1] spans across 15k stories, amounting to 300k low-level annotations for around 150k character-line pairs.

Using our new corpus, we present baseline performance on two new tasks focusing on mental state tracking of story characters: *categorizing* motivations and emotional reactions using theory labels, as well as *describing* motivations and emotional reactions using open text. Empirical results demonstrate that existing neural network models including those with explicit or latent entity representations achieve promising results.

2 Mental State Representations

Understanding people's actions, motivations, and emotions has been a recurring research focus across several disciplines including philosophy and psychology (Schachter and Singer, 1962; Burke, 1969; Lazarus, 1991; Goldman, 2015). We draw from these prior works to derive a set of categorical labels for annotating the step-by-step causal dynamics between the mental states of story characters and the events they experience.

2.1 Motivation Theories

We use two popular theories of motivation: the "hierarchy of needs" of Maslow (1943) and the "basic motives" of Reiss (2004) to compile 5 coarse-grained and 19 fine-grained motivation categories, shown in Figure 2. Maslow's "hierarchy of needs" are comprised of five categories, ranging from *physiological needs* to *spiritual growth*, which we use as coarse-level categories. Reiss (2004) proposes 19 more fine-grained categories that provide a more informative range of motivations. For example, even though they both relate to the *physiological needs* Maslow category, the *food* and *rest* motives from Reiss (2004) are very different. While the Reiss theory allows for finer-grained annotations of motivation, the larger set of abstract concepts can be overwhelming for annotators. Motivated by Straker (2013), we design a hybrid approach, where Reiss labels are annotated as sub-categories of Maslow categories.

2.2 Emotion Theory

Among several theories of emotion, we work with the "wheel of emotions" of Plutchik (1980), as it has been a common choice in prior literature on emotion categorization (Mohammad and Turney, 2013; Zhou et al., 2016). We use the eight basic emotional dimensions as illustrated in Figure 2.

2.3 Mental State Explanations

In addition to the motivation and emotion categories derived from psychology theories, we also obtain open text descriptions of character mental states. These open text descriptions allow learning computational models that can *explain* the mental states of characters in natural language, which is likely to be more accessible and informative to end users than having theory categories alone. Collecting both theory categories and open text also allows us to learn the automatic mappings between the two, which generalizes the previous work of Mohammad and Turney (2013) on emotion category mappings.

3 Annotation Framework

In this study, we choose to annotate the simple commonsense stories introduced by Mostafazadeh et al. (2016). Despite their simplicity, these stories pose a significant challenge to natural language understanding models (Mostafazadeh et al., 2017).

[1]We make our dataset publicly available at https://uwnlp.github.io/storycommonsense/

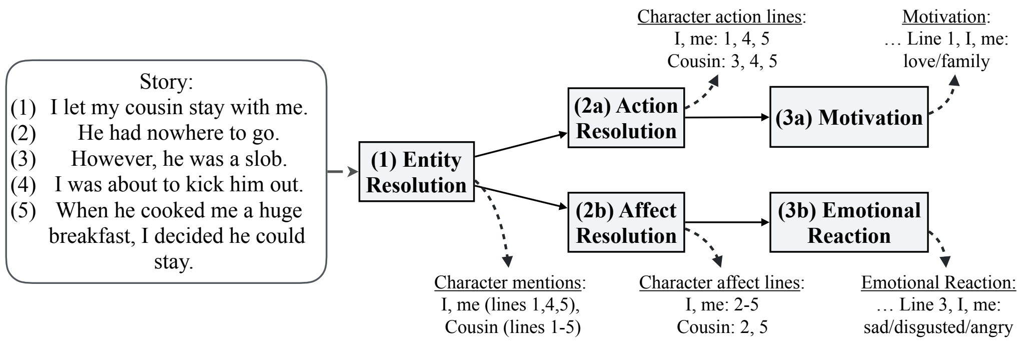

Story:
(1) I let my cousin stay with me.
(2) He had nowhere to go.
(3) However, he was a slob.
(4) I was about to kick him out.
(5) When he cooked me a huge breakfast, I decided he could stay.
(1) Entity Resolution
(2a) Action Resolution
(2b) Affect Resolution
(3a) Motivation
(3b) Emotional Reaction
Character action lines: I, me: 1, 4, 5 Cousin: 3, 4, 5
Motivation: … Line 1, I, me: love/family
Character mentions: I, me (lines 1,4,5), Cousin (lines 1-5)
Character affect lines: I, me: 2-5 Cousin: 2, 5
Emotional Reaction: … Line 3, I, me: sad/disgusted/angry

Figure 3: The annotation pipeline for the fine-grained annotations with an example story.

In addition, they depict multiple interactions between story characters, presenting rich opportunities to reason about character motivations and reactions. Furthermore, there are more than 98k such stories currently available covering a wide range of everyday scenarios.

Unique Challenges While there have been a variety of annotated resources developed on the related topics of sentiment analysis (Mohammad and Turney, 2013; Deng and Wiebe, 2015), entity tracking (Hoffart et al., 2011; Weston et al., 2015), and story understanding (Goyal et al., 2010; Ouyang and McKeown, 2015; Lukin et al., 2016), our study is the first to annotate the full chains of mental state effects for story characters. This poses several unique challenges as annotations require (1) interpreting discourse (2) understanding implicit causal effects, and (3) understanding formal psychology theory categories. In prior literature, annotations of this complexity have typically been performed by experts (Deng and Wiebe, 2015; Ouyang and McKeown, 2015). While reliable, these annotations are prohibitively expensive to scale up. Therefore, we introduce a new annotation framework that pipelines a set of smaller isolated tasks as illustrated in Figure 3. All annotations were collected using crowdsourced workers from Amazon Mechanical Turk.

3.1 Annotation Pipeline

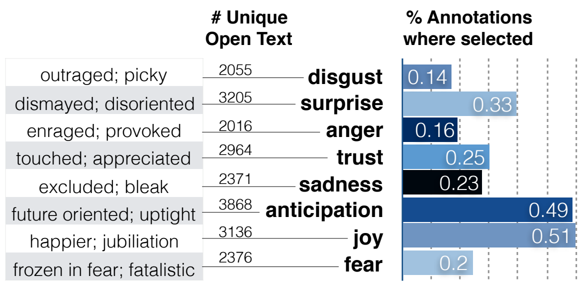
We describe the components and workflow of the full annotation pipeline shown in Figure 3 below. The example story in the figure is used to illustrate the output of various steps in the pipeline (full annotations for this example are in the appendix).

(1) Entity Resolution The first task in the pipeline aims to discover (1) the set of characters E_i in each story i and (2) the set of sentences S_{ij} in which a specific character $j \in E_i$ is explicitly mentioned. For example, in the story in Figure 3, the characters identified by annotators are "I/me" and "My cousin", whom appear in sentences $\{1, 4, 5\}$ and $\{1, 2, 3, 4, 5\}$, respectively.

We use S_{ij} to control the workflow of later parts of the pipeline by pruning future tasks for sentences that are not tied to characters. Because S_{ij} is used to prune follow-up tasks, we take a high recall strategy to include all sentences that at least one annotator selected.

(2a) Action Resolution The next task identifies whether a character j appearing in a sentence k is taking any action to which a motivation can be attributed. We perform action resolution only for sentences $k \in S_{ij}$. In the running example, we would want to know that the cousin in line 2 is not doing anything intentional, allowing us to omit this line in the next pipeline stage (3a) where a character's motives are annotated. Description of state (e.g., "Alex is feeling blue") or passive event participation (e.g., "Alex trips") are not considered volitional acts for which the character may have an underlying motive. For each line and story character pair, we obtain 4 annotations. Because pairs can still be filtered out in the next stage of annotation, we select a generous threshold where only 2 annotators must vote that an intentional action took place for the sentence to be used as an input to the motivation annotation task (3a).

(2b) Affect Resolution This task aims to identify all of the lines where a story character j has an emotional reaction. Importantly, it is often possible to infer the emotional reaction of a character j even when the character does not explicitly appear in a sentence k. For instance, in Figure 3, we want to annotate the narrator's reaction to line 2 even though they are not mentioned because their emotional response is inferrable. We obtain 4 an-



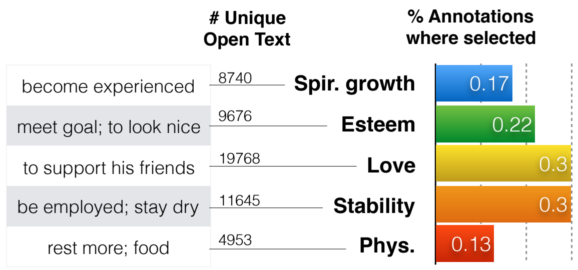


Figure 4: Examples of open-text explanations that annotators provided corresponding with the categories they selected. The bars on the right of the categories represent the percentage of lines where annotators selected that category (out of those character-line pairs with positive motivation/emotional reaction).

notations per character per line. The lines with at least 2 annotators voting are used as input for the next task: (3b) emotional reaction.

(3a) Motivation We use the output from the action resolution stage (2a) to ask workers to annotate character motives in lines where they intentionally initiate an event. We provide 3 annotators a line from a story, the preceding lines, and a specific character. They are asked to produce a free response sentence describing what causes the character's behavior in that line and to select the most related Maslow categories and Reiss subcategories. In Figure 3, an annotator described the motivation of the narrator in line 1 as wanting "to have company" and then selected the *love* (Maslow) and *family* (Reiss) as categorical labels. Because many annotators are not familiar with motivational theories, we require them to complete a tutorial the first time they attempt the task.

(3b) Emotional Reaction Simultaneously, we use the output from the affect resolution stage (2b) to ask workers what the emotional response of a character is immediately following a line in which they are affected. As with the motives, we give 3 annotators a line from a story, its previous context, and a specific character. We ask them to describe in open text how the character will feel following the event in the sentence (up to three emotions). As a follow-up, we ask workers to compare their free responses against Plutchik categories by using 3-point likert ratings. In Figure 3, we include a response for the emotional reaction of the narrator in line 1. Even though the narrator was not mentioned directly in that line, an annotator recorded that they will react to their cousin being a slob by feeling "annoyed" and selected the Plutchik categories for *sadness*, *disgust* and *anger*.

	Fine-grained		
	train	dev	test
# annotated stories	10000	2500	2500
# characters / story	2.03	2.02	1.82
# char-lines w/ motiv	40154	8762	6831
# char-lines w/ emot	76613	14532	13785

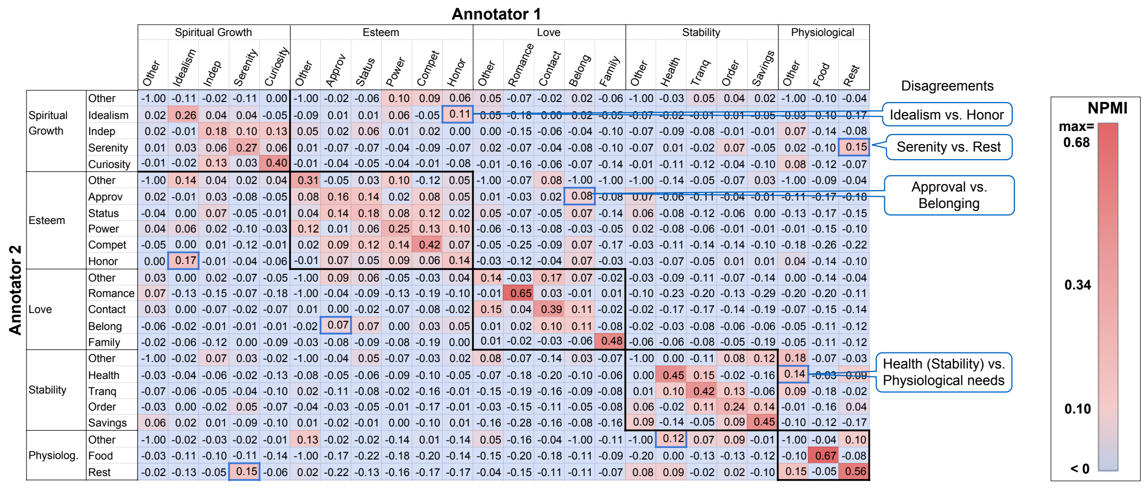
Table 1: Annotated data statistics for each dataset

3.2 Dataset Statistics and Insights

Cost The tasks corresponding to the theory category assignments are the hardest and most expensive in the pipeline (∼$4 per story). Therefore, we obtain theory category labels only for a third of our annotated stories, which we assign to the development and test sets. The training data is annotated with a shortened pipeline with only open text descriptions of motivations and emotional reactions from two workers (∼$1 per story).

Scale Our dataset to date includes a total of 300k low-level annotations for motivation and emotion across 15,000 stories (randomly selected from the ROC story training set). It covers over 150,000 character-line pairs, in which 56k character-line pairs have an annotated motivation and 105k have an annotated change in emotion (i.e. a label other than `none`). Table 1 shows the break down across training, development, and test splits. Figure 4 shows the frequency of different labels being selected for motivational and emotional categories in cases with positive change.

Agreements For quality control, we removed workers who consistently produced low-quality work, as discussed in the Appendix. In the categorization sets (Maslow, Reiss and Plutchik), we compare the performance of annotators by treating each individual category as a binary label (1



		Spiritual Growth					Esteem						Love					Stability					Physiological		
		Other	Idealism	Indep	Serenity	Curiosity	Other	Approv	Status	Power	Compet	Honor	Other	Romance	Contact	Belong	Family	Other	Health	Tranq	Order	Savings	Other	Food	Rest
Spiritual Growth	Other	-1.00	-0.11	-0.02	-0.11	0.00	-1.00	-0.02	-0.06	0.10	0.09	0.06	0.05	-0.07	-0.02	0.02	-0.06	-1.00	-0.03	0.05	0.04	0.02	-1.00	-0.10	-0.04
	Idealism	0.02	0.26	0.04	0.04	-0.05	-0.09	0.01	0.01	0.06	-0.05	0.11	-0.05	-0.18	0.00	-0.02	-0.05	-0.07	-0.02	-0.01	0.01	-0.05	-0.03	-0.10	-0.17
	Indep	0.02	-0.01	0.18	0.10	0.13	0.05	0.02	0.06	0.01	0.02	0.00	0.00	-0.15	-0.06	-0.04	-0.10	-0.07	-0.09	-0.08	-0.01	-0.01	0.07	-0.14	-0.08
	Serenity	0.01	0.03	0.06	0.27	0.06	0.01	-0.07	-0.07	-0.04	-0.09	-0.07	0.02	-0.07	-0.04	-0.08	-0.10	-0.07	0.01	-0.02	0.07	-0.05	0.02	-0.10	0.15
	Curiosity	-0.01	-0.02	0.13	0.03	0.40	-0.01	-0.04	-0.05	-0.04	-0.01	-0.08	-0.01	-0.16	-0.06	-0.07	-0.14	-0.01	-0.11	-0.12	-0.04	-0.10	0.08	-0.12	-0.07
Esteem	Other	-1.00	0.14	0.04	0.02	0.04	0.31	-0.05	0.03	0.10	-0.12	0.05	-1.00	-0.07	0.08	-1.00	-1.00	-1.00	-0.14	-0.05	-0.07	0.03	-1.00	-0.09	-0.04
	Approv	0.02	-0.01	0.03	-0.08	-0.05	0.08	0.16	0.14	0.02	0.08	0.05	0.01	-0.03	0.02	0.08	-0.08	0.07	-0.06	-0.11	-0.04	-0.01	-0.11	-0.17	-0.18
	Status	-0.04	0.00	0.07	-0.05	-0.01	0.04	0.14	0.18	0.08	0.12	0.02	0.05	-0.07	-0.05	0.07	-0.14	0.06	-0.08	-0.12	-0.06	0.00	-0.13	-0.17	-0.15
	Power	0.04	0.06	0.02	-0.10	-0.03	0.12	0.01	0.06	0.25	0.13	0.10	-0.06	-0.13	-0.08	-0.03	-0.05	0.02	-0.08	-0.06	-0.01	-0.01	-0.01	-0.15	-0.10
	Compet	-0.05	0.00	0.01	-0.12	-0.01	0.02	0.09	0.12	0.14	0.42	0.07	-0.05	-0.25	-0.09	0.07	-0.17	-0.03	-0.11	-0.14	-0.14	-0.10	-0.18	-0.26	-0.22
	Honor	0.00	0.17	-0.01	-0.04	-0.06	-0.01	0.07	0.05	0.09	0.06	0.14	-0.03	-0.12	-0.04	0.07	-0.03	-0.03	-0.07	-0.05	0.01	0.01	0.04	-0.14	-0.10
Love	Other	0.03	0.00	0.02	-0.07	-0.05	-1.00	0.09	0.06	-0.05	-0.03	0.04	0.14	-0.03	0.17	0.07	-0.02	-0.03	-0.09	-0.11	-0.07	-0.14	0.00	-0.14	-0.04
	Romance	0.07	-0.13	-0.15	-0.07	-0.18	-1.00	-0.04	-0.09	-0.13	-0.19	-0.10	-0.01	0.65	0.03	-0.01	0.01	-0.10	-0.23	-0.20	-0.13	-0.29	-0.20	-0.20	-0.11
	Contact	0.03	0.00	-0.07	-0.02	-0.07	0.01	0.00	-0.02	-0.07	-0.08	-0.02	0.15	0.04	0.39	0.11	-0.02	-0.02	-0.17	-0.17	-0.14	-0.19	-0.07	-0.15	-0.14
	Belong	-0.06	-0.02	-0.01	-0.01	-0.01	-0.02	0.07	0.07	0.00	0.03	0.05	0.01	0.02	0.10	0.11	-0.08	-0.02	-0.03	-0.08	-0.06	-0.06	-0.05	-0.11	-0.12
	Family	-0.02	-0.06	-0.12	0.00	-0.09	-0.03	-0.08	-0.09	-0.08	-0.19	0.00	0.01	-0.02	-0.03	-0.06	0.48	-0.06	-0.06	-0.08	-0.05	-0.19	-0.05	-0.11	-0.12
Stability	Other	-1.00	-0.02	0.07	0.03	-0.02	-1.00	-0.04	0.05	-0.07	-0.03	0.02	0.08	-0.07	-0.14	0.03	-0.07	-1.00	0.00	-0.11	0.08	0.12	0.18	-0.07	-0.03
	Health	-0.03	-0.04	-0.06	-0.02	-0.13	-0.08	-0.05	-0.06	-0.09	-0.10	-0.05	-0.07	-0.18	-0.20	-0.10	-0.06	0.00	0.45	0.15	-0.02	-0.16	0.14	-0.03	0.09
	Tranq	-0.07	-0.06	-0.05	-0.04	-0.10	0.02	-0.11	-0.08	-0.02	-0.16	-0.01	-0.15	-0.19	-0.16	-0.09	-0.08	0.01	0.10	0.42	0.13	-0.06	0.09	-0.18	-0.02
	Order	-0.03	0.00	-0.02	0.05	-0.07	-0.04	-0.03	-0.05	-0.01	-0.17	-0.01	-0.03	-0.15	-0.11	-0.05	-0.08	0.06	-0.02	0.11	0.24	0.14	-0.01	-0.16	0.04
	Savings	0.06	0.02	0.01	-0.09	-0.10	0.01	-0.02	-0.01	0.00	-0.10	0.01	-0.16	-0.28	-0.16	-0.08	-0.16	0.09	-0.14	-0.05	0.09	0.45	-0.10	-0.12	-0.17
Physiolog.	Other	-1.00	-0.02	-0.03	-0.02	-0.01	0.13	-0.02	-0.02	-0.14	0.01	-0.14	0.05	-0.16	-0.04	-1.00	-0.11	-1.00	0.12	0.07	0.09	-0.01	-1.00	-0.04	0.10
	Food	-0.03	-0.11	-0.10	-0.11	-0.14	-1.00	-0.17	-0.22	-0.18	-0.20	-0.14	-0.15	-0.20	-0.18	-0.11	-0.09	-0.20	0.00	-0.13	-0.13	-0.12	-0.10	0.67	-0.08
	Rest	-0.02	-0.13	-0.05	0.15	-0.06	0.02	-0.22	-0.13	-0.16	-0.17	-0.17	-0.04	-0.15	-0.11	-0.11	-0.07	0.08	0.09	-0.02	0.02	-0.10	0.15	-0.05	0.56

Figure 5: NPMI confusion matrix on motivational categories for all annotator pairs with color scaling for legibility. The highest values are generally along diagonal or within Maslow categories (outlined in black). We highlight a few common points of disagreement between thematically similar categories.

Label Type		**PPA**	**KA**	**% Agree w/ Maj. Lbl**
Maslow	Dev	.77	.30	0.88
	Test	.77	.31	0.89
Reiss	Dev	.91	.24	0.95
	Test	.91	.24	0.95
Plutchik	Dev	.71	.32	0.84
	Test	.70	.29	0.83

Table 2: Agreement Statistics (PPA = Pairwise percent agreement of worker responses per binary category, KA= Krippendorff's Alpha)

if they included the category in their set of responses) and averaging the agreement per category. For Plutchik scores, we count 'moderately associated' ratings as agreeing with 'highly' associated' ratings. The percent agreement and Krippendorff's alpha are shown in Table 2. We also compute the percent agreement between the individual annotations and the majority labels.[2]

These scores are difficult to interpret by themselves, however, as annotator agreement in our categorization system has a number of properties that are not accounted for by these metrics (disagreement preferences – joy and trust are closer than joy and anger – that are difficult to quantify in a principled way, hierarchical categories mapping Reiss subcategories from Maslow categories, skewed category distributions that inflate PPA and deflate KA scores, and annotators that could select multiple labels for the same examples).

To provide a clearer understanding of agreement within this dataset, we create aggregated confusion matrices for annotator pairs. First, we sum the counts of combinations of answers between all paired annotations (excluding `none` labels). If an annotator selected multiple categories, we split the count uniformly among the selected categories. We compute NPMI over the total confusion matrix. In Figure 5, we show the NPMI confusion matrix for motivational categories.

In the motivation annotations, we find the highest scores on the diagonal (i.e., Reiss agreement), with most confusions occurring between Reiss motives in the same Maslow category (outlined black in Figure 5). Other disagreements generally involve Reiss subcategories that are thematically similar, such as *serenity* (mental relaxation) and *rest* (physical relaxation). We provide this analysis for Plutchik categories in the appendix, finding high scores along the diagonal with disagreements typically occurring between categories in a "positive emotion" cluster (*joy*, *trust*) or a "negative emotion" cluster (*anger*, *disgust*,*sadness*).

4 Tasks

The multiple modes covered by the annotations in this new dataset allow for multiple new tasks to be explored. We outline three task types below, covering a total of eight tasks on which to evaluate.

[2]Majority label for the motivation categories is what was agreed upon by at least two annotators per category. For emotion categories, we averaged the point-wise ratings and counted a category if the average rating was ≥ 2.

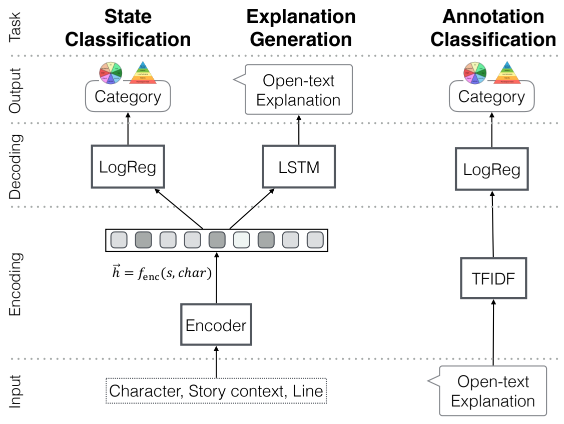


Figure 6: General model architectures for three new task types

Differences between task type inputs and outputs are summarized in Figure 6.

State Classification The three primary tasks involve categorizing the psychological states of story characters for each of the label sets (Maslow, Reiss, Plutchik) collected for the dev and test splits of our dataset. In each classification task, a model is given a line of the story (along with optional preceding context lines) and a character and predicts the motivation (or emotional reaction). A binary label is predicted for each of the Maslow needs, Reiss motives or Plutchik categories.

Annotation Classification Because the dev and test sets contain paired classification labels and free text explanations, we propose three tasks where a model must predict the correct Maslow/Reiss/Plutchik label given an emotional reaction or motivation explanation.

Explanation Generation Finally, we can use the free text explanations to train models to describe the psychological state of a character in free text (examples in Figure 4). These explanations allow for two conditional generation tasks where the model must generate the words describing the emotional reaction or motivation of the character.

5 Baseline Models

The general model architectures for the three tasks are shown in Figure 6. We describe each model component below. The state classification and explanation generation models could be trained separately or in a multi-task set-up.

In the state classification and explanation generation tasks, a model is given a line from a story $\mathbf{x}^s$ containing N words $\{w_0^s, w_1^s, \ldots, w_N^s\}$ from vocabulary V, a character in that story $e_j \in E$ where E is the set of characters in the story, and (optionally) the preceding sentences in the story $\mathbf{C} = \{\mathbf{x}^0 \ldots, \mathbf{x}^{s-1}\}$ containing words from vocabulary V. A representation for a character's psychological state is encoded as:

$$\mathbf{h}^e = \text{Encoder}(\mathbf{x}^s, \mathbf{C}[e_j]) \tag{1}$$

where $\mathbf{C}[e_j]$ corresponds to the concatenated subset of sentences in $\mathbf{C}$ where e_j appears.

5.1 Encoders

While the end classifier or decoder is different for each task, we use the same set of encoders based on word embeddings, common neural network architectures, or memory networks to formulate a representation of the sentence and character, $\mathbf{h}^e$. Unless specified, $\mathbf{h}^e$ is computed by encoding separate vector representations for the sentence ($\mathbf{x}^s \rightarrow \mathbf{h}^s$) and character-specific context ($\mathbf{C}[e_j] \rightarrow \mathbf{h}^c$) and concatenating these encodings ($\mathbf{h}^e = [\mathbf{h}^c; \mathbf{h}^s]$). We describe the encoders below:

TF-IDF We learn a TD-IDF model on the full training corpus of Mostafazadeh et al. (2016) (excluding the stories in our dev/test sets). To encode the sentence, we extract TF-IDF features for its words, yielding $v^s \in \mathcal{R}^V$. A projection and non-linearity is applied to these features to yield $\mathbf{h}^s$:

$$\mathbf{h}^s = \phi(W_s v^s + b_s) \tag{2}$$

where $W_s \in \mathcal{R}^{d \times H}$. The character vector $\mathbf{h}^c$ is encoded in the same way on sentences in the context pertaining to the character.

GloVe We extract pretrained Glove vectors (Pennington et al., 2014) for each word in V. The word embeddings are max-pooled, yielding embedding $v^s \in \mathcal{R}^H$, where H is the dimensionality of the Glove vectors. Using this max-pooled representation, $\mathbf{h}^s$ and $\mathbf{h}^c$ are extracted in the same manner as for TF-IDF features (Equation 2).

CNN We implement a CNN text categorization model using the same configuration as Kim (2014) to encode the sentence words. A sentence is represented as a matrix, $v^s \in \mathcal{R}^{M \times d}$ where each row is a word embedding x_n^s for a word $w_n^s \in \mathbf{x}^s$.

$$v^s = [x_0^s, x_1^s, \ldots, x_N^s] \tag{3}$$

$$\mathbf{h}^s = \text{CNN}(v^s) \tag{4}$$

where CNN represents the categorization model from (Kim, 2014). The character vector $\mathbf{h}^c$ is encoded in the same way with a separate CNN. Implementation details are provided in the appendix.

LSTM A two-layer bi-LSTM encodes the sentence words and concatenates the final time step hidden states from both directions to yield $\mathbf{h}^s$. The character vector $\mathbf{h}^c$ is encoded the same way.

REN We use the "tied" recurrent entity network from Henaff et al. (2017). A memory cell m is initialized for each of the J characters in the story, $E = \{e_0, \ldots, e_J\}$. The REN reads documents one sentence at a time and updates m_j for $e_j \in E$ after reading each sentence. Unlike the previous encoders, all sentences of the context $\mathbf{C}$ are given to the REN along with the sentence $\mathbf{x}^s$. The model learns to distribute encoded information to the correct memory cells. The representation passed to the downstream model is:

$$\mathbf{h}^e = \{m_j\}^s \tag{5}$$

where $\{m_j\}^s$ is the memory vector in the cell corresponding to e_j after reading $\mathbf{x}^s$. Implementation details are provided in the appendix.

NPN We also include the neural process network from Bosselut et al. (2018) with "tied" entities, but "untied" actions that are not grounded to particular concepts. The memory is initialized and accessed similarly as the REN. Exact implementation details are provided in the appendix.

5.2 State Classifier

Once the sentence-character encoding $\mathbf{h}^e$ is extracted, the state classifier predicts a binary label $\hat{y}_z$ for every category $z \in \mathcal{Z}$ where $\mathcal{Z}$ is the set of category labels for a particular psychological theory (e.g., disgust, surprise, etc. in the Plutchik wheel). We use logistic regression as a classifier:

$$\hat{y}_z = \sigma(W_z \mathbf{h}^e + b_z) \tag{6}$$

where W_z and b_z are a label-specific set of weights and biases for classifying each label $z \in \mathcal{Z}$.

5.3 Explanation Generator

The explanation generator is a single-layer LSTM (Hochreiter and Schmidhuber, 1997) that receives the encoded sentence-character representation $\mathbf{h}^e$ and predicts each word y_t in the explanation using the same method from Sutskever et al. (2014). Implementation details are provided in the appendix.

5.4 Annotation Classifier

For annotation classification tasks, words from open-text explanations are encoded with TF-IDF features. The same classifier architecture from Section 5.2 is used to predict the labels.

6 Experimental Setup

6.1 Training

State Classification The dev set D is split into two portions of 80% (D_1) and 20% (D_2). D_1 is used to train the classifier and encoder. D_2 is used to tune hyperparameters. The model is trained to minimize the weighted binary cross entropy of predicting a class label y_z for each class z:

$$\mathcal{L} = \sum_{z=1}^{Z} \gamma_z y_z \log \hat{y}_z + (1-\gamma_z)(1-y_z)\log(1-\hat{y}_z) \tag{7}$$

where Z is the number of labels in each of the three classifications tasks and γ_z is defined as:

$$\gamma_z = 1 - e^{-\sqrt{P(y_z)}} \tag{8}$$

where $P(y_z)$ is the marginal class probability of a positive label for z in the training set.

Annotation Classification The dev set is split in the same manner as for state classification. The TF-IDF features are trained on the set of training annotations D_t coupled with those from D_1. The model must minimize the same loss as in Equation 7. Details are provided in the appendix.

Explanation Generation We use the training set of open annotations to train a model to predict explanations. The decoder is trained to minimize the negative loglikelihood of predicting each word in the explanation of a character's state:

$$L_{gen} = -\sum_{t=1}^{T} \log P(y_t | y_0, ..., y_{t-1}, \mathbf{h}^e) \tag{9}$$

where $\mathbf{h}^e$ is the sentence-character representation produced by an encoder from Section 5.1.

6.2 Metrics

Classification For the state and annotation classification task, we report the micro-averaged precision (P), recall (R), and F1 score of the Plutchik, Maslow, and Reiss prediction tasks. We report the results of selecting a label at random in the top two rows of Table 3. Note that random is low because the distribution of positive instances for each

Model	Maslow			Reiss			Plutchik		
	P	R	F1	P	R	F1	P	R	F1
Random	7.45	49.99	12.96	1.76	50.02	3.40	10.35	50.00	17.15
Random (Weighted)	8.10	8.89	8.48	2.25	2.40	2.32	12.28	11.79	12.03
TF-IDF	30.10	21.21	24.88	18.40	20.67	19.46	20.05	24.11	21.90
+ Entity Context	29.79	34.56	32.00	20.55	24.81	22.48	22.71	25.24	23.91
GloVe	25.15	29.70	27.24	16.65	18.83	17.67	15.19	30.56	20.29
+ Entity Context	27.02	37.00	31.23	16.99	26.08	20.58	19.47	**46.65**	27.48
LSTM	24.64	35.30	29.02	19.91	19.76	19.84	20.27	30.37	24.31
+ Entity Context	**31.29**	33.85	32.52	18.35	27.61	22.05	23.98	31.41	27.20
+ Explanation Training	30.34	40.12	34.55	**21.38**	28.70	**24.51**	25.31	33.44	28.81
CNN (Kim, 2014)	26.21	31.09	28.44	20.30	23.24	21.67	21.15	23.36	22.20
+ Entity Context	27.47	41.01	32.09	18.89	31.22	23.54	24.32	30.76	27.16
+ Explanation Training	29.30	44.18	**35.23**	17.87	**37.52**	24.21	24.47	38.87	30.04
REN (Henaff et al., 2017)	26.24	42.14	32.34	16.79	22.20	19.12	**26.22**	33.26	29.32
+ Explanation Training	26.85	**44.78**	33.57	16.73	26.55	20.53	25.30	37.30	30.15
NPN (Bosselut et al., 2018)	24.27	44.16	31.33	13.13	26.44	17.55	21.98	37.31	27.66
+ Explanation Training	26.60	39.17	31.69	15.75	20.34	17.75	24.33	40.10	**30.29**

Table 3: State Classification Results

category is very uneven: macro-averaged positive class probabilities of 8.2, 1.7, and 9.9% per category for Maslow, Reiss, and Plutchik respectively.

Generation Because explanations tend to be short sequences (Figure 4) with high levels of synonymy, traditional metrics such as BLEU are inadequate for evaluating generation quality. We use the vector average and vector extrema metrics from Liu et al. (2016) computed using the Glove vectors of generated and reference words. We report results in Table 5 on the dev set and compare to a baseline that randomly samples an example from the dev set as a generated sequence.

6.3 Ablations

Story Context vs. No Context Our dataset is motivated by the importance of interpreting story context to categorize emotional reactions and motivations of characters. To test this importance, we ablate $\mathbf{h}^c$, the representation of the context sentences pertaining to the character, as an input to the state classifier for each encoder (except the REN and NPN). In Table 3, this ablation is the first row for each encoder presented.

Explanation Pretraining Because the state classification and explanation generation tasks use the same models to encode the story, we explore initializing a classification encoder with parameters trained on the generation task. For the CNN, LSTM, and REN encoders, we pretrain a generator to produce emotion or motivation explanations. We use the parameters from the emotion or motivation explanation generators to initialize the Plutchik or Maslow/Reiss classifiers respectively.

7 Experimental Results

State Classification We show results on the test set for categorizing Maslow, Reiss, and Plutchik states in Table 3. Despite the difficulty of the task, all models outperform the random baseline. Interestingly, the performance boost from adding entity-specific contextual information (i.e., not ablating $\mathbf{h}^c$) indicates that the models learn to condition on a character's previous experience to classify its mental state at the current time step. This effect can be seen in a story about a man whose flight is cancelled. The model without context predicts the same emotional reactions for the man, his wife and the pilot, but with context correctly predicts that the pilot will not have a reaction while predicting that the man and his wife will feel sad.

For the CNN, LSTM, REN, and NPN models, we also report results from pretraining encoder parameters using the free response annotations from the training set. This pretraining offers a clear performance boost for all models on all three prediction tasks, showing that the parameters of the encoder can be pretrained on auxiliary tasks providing emotional and motivational state signal.

The best performing models in each task are most effective at predicting Maslow *physiological* needs, Reiss *food* motives, and Plutchik reactions of *joy*. The relative ease of predicting motivations

	Maslow	Reiss	Plutchik
TFIDF	64.81	48.60	53.44

Table 4: F1 scores of predicting correct category labels from free response annotations

Model	Motivation		Emotion	
	Avg	VE	Avg	VE
Random	56.02	45.75	40.23	39.98
LSTM	58.48	51.07	52.47	52.30
CNN	57.83	50.75	52.49	52.31
REN	**58.83**	**51.79**	53.95	53.79
NPN	57.77	51.77	**54.02**	**53.85**

Table 5: Vector average and extrema scores for generation of annotation explanations

related to food (and physiological needs generally) may be because they involve a more limited and concrete set of actions such as eating or cooking.

Annotation Classification Table 4 shows that a simple model can learn to map open text responses to categorical labels. This further supports our hypothesis that pretraining a classification model on the free-response annotations could be helpful in boosting performance on the category prediction.

Explanation Generation Finally, we provide results for the task of generating explanations of motivations and emotions in Table 5. Because the explanations are closely tied to emotional and motivation states, the randomly selected explanation can often be close in embedding space to the reference explanations, making the random baseline fairly competitive. However, all models outperform the strong baseline on both metrics, indicating that the generated short explanations are closer semantically to the reference annotation.

8 Related work

Mental State Annotations Incorporating emotion theories into NLP tasks has been explored in previous projects. Ghosh et al. (2017) modulate language model distributions by increasing the probability of words that express certain affective LIWC (Tausczik and Pennebaker, 2016) categories. More generally, various projects tackle the problem of generating text from a set of attributes like sentiment or generic-ness (Ficler and Goldberg, 2017; Dong et al., 2017). Similarly, there is also a body of research in reasoning about commonsense stories and discourse (Li and Jurafsky, 2017; Mostafazadeh et al., 2016) or detecting emotional stimuli in stories (Gui et al., 2017). Previous work in plot units (Lehnert, 1981) developed formalisms for affect and mental state in story narratives that included motivations and reactions. In our work, we collect mental state annotations for stories to used as a new resource in this space.

Modeling Entity State Recently, novel works in language modeling (Ji et al., 2017; Yang et al., 2016), question answering (Henaff et al., 2017), and text generation (Kiddon et al., 2016; Bosselut et al., 2018) have shown that modeling entity state explicitly can boost performance while providing a preliminary interface for interpreting a model's prediction. Entity modeling in these works, however, was limited to tracking entity reference (Kiddon et al., 2016; Yang et al., 2016; Ji et al., 2017), recognizing entity state similarity (Henaff et al., 2017) or predicting simple attributes from entity states (Bosselut et al., 2018). Our work provides a new dataset for tracking emotional reactions and motivations of characters in stories.

9 Conclusion

We present a large scale dataset as a resource for training and evaluating mental state tracking of characters in short commonsense stories. This dataset contains over 300k low-level annotations for character *motivations* and *emotional reactions*. We provide benchmark results on this new resource. Importantly, we show that modeling character-specific context and pretraining on free-response data can boost labeling performance. While our work only use information present in our dataset, we view our dataset as a future testbed for evaluating models trained on any number of resources for learning common sense about emotional reactions and motivations.

Acknowledgments

We thank the reviewers for their insightful comments. We also thank Bowen Wang, xlab members, Martha Palmer, Tim O'Gorman, Susan W. Brown, and Ghazaleh Kazeminejad for helpful discussions on inter-annotator agreement and the annotation pipeline. This work was supported in part by NSF GRFP DGE-1256082, NSF IIS-1714566, IIS-1524371, Samsung AI, and DARPA CwC (W911NF-15-1-0543).

References

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *Proceedings of the 6th International Conference on Learning Representations*.

Kenneth Burke. 1969. *A Grammar of Motives*. Univ of California Press.

Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1323–1328.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *EACL*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *EMNLP Workshop on Stylistic Variation*.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-LM: A neural language model for customizable affective text generation. In *ACL*.

Alvin I Goldman. 2015. *Theory of Human Action*. Princeton University Press.

Amit Goyal, Ellen Riloff, and Hal Daumé. 2010. Automatically producing plot unit representations for narrative text. In *EMNLP*.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Lu Qin, and Jiachen Du. 2017. A question answering approach for emotion cause extraction. In *EMNLP*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1594–1603.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

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

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *EMNLP*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1831–1840.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *EMNLP*. pages 329–339.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Richard S Lazarus. 1991. Progress on a cognitive-motivational-relational theory of emotion. *American Psychologist* 46(8):819.

Wendy G. Lehnert. 1981. Plot units and narrative summarization. *Cognitive Science* 5:293–331.

Jiwei Li and Dan Jurafsky. 2017. Neural net models for Open-Domain discourse coherence. In *EMNLP*.

Chia-Wei Liu, Ryan Joseph Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Stephanie M. Lukin, Kevin Bowden, Casey Barackman, and Marilyn A. Walker. 2016. Personabank: A corpus of personal narratives and their story intention graphs. *CoRR* abs/1708.09082.

Abraham H Maslow. 1943. A theory of human motivation. *Psychol. Rev.* 50(4):370.

Saif Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence* 29:436–465.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL*.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*. pages 46–51.

Jessica Ouyang and Kathleen McKeown. 2015. Modeling reportable events as turning points in narrative. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*. pages 1532–1543.

Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. *Theories of emotion* 1(3-31):4.

Steven Reiss. 2004. Multifaceted nature of intrinsic motivation: The theory of 16 basic desires. *Rev. Gen. Psychol.* 8(3):179.

Stanley Schachter and Jerome Singer. 1962. Cognitive, social, and physiological determinants of emotional state. *Psychological review* 69(5):379.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

David Straker. 2013. Reiss' 16 human needs. `http://changingminds.org/explanations/needs/reiss_16_needs.htm`. Accessed: 2018-02-21.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Yla R Tausczik and James W Pennebaker. 2016. The psychological meaning of words: LIWC and computerized text analysis methods. *J. Lang. Soc. Psychol.* .

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *CoRR* abs/1611.01628. http://arxiv.org/abs/1611.01628.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Conference on Computer Vision and Pattern Recognition*.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quanming Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *EMNLP*.

A Deep Relevance Model for Zero-Shot Document Filtering

Chenliang Li[1], **Wei Zhou**[2], **Feng Ji**[2], **Yu Duan**[1], **Haiqing Chen**[2]
[1]School of Cyber Science and Engineering, Wuhan University, China
{cllee,duanyu}@whu.edu.cn
[2]Alibaba Group, Hangzhou, China
{fayi.zw,zhongxiu.jf,haiqing.chenhq}@alibaba-inc.com

Abstract

In the era of big data, focused analysis for diverse topics with a short response time becomes an urgent demand. As a fundamental task, information filtering therefore becomes a critical necessity. In this paper, we propose a novel deep relevance model for zero-shot document filtering, named DAZER. DAZER estimates the relevance between a document and a category by taking a small set of seed words relevant to the category. With pre-trained word embeddings from a large external corpus, DAZER is devised to extract the relevance signals by modeling the hidden feature interactions in the word embedding space. The relevance signals are extracted through a gated convolutional process. The gate mechanism controls which convolution filters output the relevance signals in a category dependent manner. Experiments on two document collections of two different tasks (*i.e.,* topic categorization and sentiment analysis) demonstrate that DAZER significantly outperforms the existing alternative solutions, including the state-of-the-art deep relevance ranking models.

1 Introduction

Filtering irrelevant information and organizing relevant information into meaningful topical categories is indispensable and ubiquitous. For example, a data analyst tracking an emerging event would like to retrieve the documents relevant to a specific topic (category) from a large document collection in a short response time. In the era of big data, the potentially possible categories covered by documents would be limitless. It is unrealistic to manually identify a lot of positive examples for each possible category. However, new information needs indeed emerge everywhere in many real-world scenarios. Recent studies on dataless text classification show promising results on reducing labeling effort (Liu et al., 2004; Druck et al., 2008; Chang et al., 2008; Song and Roth, 2014; Hingmire et al., 2013; Hingmire and Chakraborti, 2014; Chen et al., 2015; Li et al., 2016). Without any labeled document, a dataless classifier performs text classification by using a small set of relevant words for each category (called "seed words"). However, existing dataless classifiers do not consider *document filtering*. We need to provide the seed words for each category covered by the document collection, which is often infeasible in the real world.

To this end, we are particularly interested in the task of zero-shot document filtering. Here, *zero-shot* means that the instances of the targeted categories are unseen during the training phase. To facilitate zero-shot filtering, we take a small set of seed words to represent a category of interest. This is extremely useful when the information need (*i.e.,* the categories of interest) is dynamic and the text collection is large and temporally updated (*e.g.,* the possible categories are hard to know). Specifically, we propose a novel **d**eep relev**a**nce model for **z**ero-shot document filt**er**ing, named DAZER. In DAZER, we use the word embeddings learnt from an external large text corpus to represent each word. A category can then be well represented also in the embedding space (called *category embedding*) through some composition with the word embeddings of the provided seed words. Given a small number of seed words provided for a category as input, DAZER is devised to produce a score indicating the relevance between a document and the category. It is intuitive to connect zero-shot document filtering with the task of ad-hoc retrieval. Indeed, by treating the seed words of each category as a query, the zero-shot document filtering is equivalent to ranking documents based on their relevance to the query. The relevance ranking is a core task in information retrieval, and has been studied for many years. Although they share the same formulation, these two tasks diverge fundamentally. For ad-hoc retrieval, a user constructs a query with a specific information need. The relevant documents are assumed to contain these query words. This is confirmed by the existing works that exact keyword match is still the most important signal of relevance in ad-hoc retrieval (Fang and Zhai, 2006; Wu et al., 2007; Eickhoff et al., 2015; Guo et al., 2016a,b).

For document filtering, the seed words for a category are expected to convey the conceptual meaning of the latter. It is impossible to list all the words to fully cover the relevant documents of a category. Therefore, it is essential to capture the conceptual relevance for zero-shot document filtering. The classical retrieval models simply estimate the relevance based on the query keyword matching, which is far from capturing the conceptual relevance. The existing deep relevance models for ad-hoc retrieval utilize the statistics of the hard/soft-match signals in terms of cosine similarity between two word embeddings (Guo et al., 2016a; Xiong et al., 2017). However, the scalar information like cosine similarity between two embedding vectors is too coarse or limited to reflect the conceptual relevance. On the contrary, we believe that the embedding features could provide rich knowledge towards the conceptual relevance. A key challenge is to endow DAZER a strong generalization ability to also successfully extract the relevance signals for unseen categories. To achieve this purpose, we extract the relevance signals based on the hidden feature interactions between the category and each word in the embedding space. Specifically, two element-wise operations are utilized in DAZER: element-wise subtraction and element-wise product. Since these two kinds of interactions represent the relative information encoded in hidden embedding space, we expect that the relevance signal extraction process could generalize well to unseen categories. Firstly, DAZER utilizes a gated convolutional operation with k-max pooling to extract the relevance signals. Then, DAZER abstracts higher-level relevance features through a multi-layer perceptron, which can be considered as a relevance aggregation procedure. At last, DAZER calculates an overall score indicating the relevance between a document and the category. Without further constraints, it is possible for DAZER to encode the bias towards the category-specific features seen during the training (*i.e.,* model overfitting). Therefore, we further introduce an adversarial learning over the output of the relevance aggregation procedure. The purpose is to ensure that the higher-level relevance features contain no category-dependent information, leading to a better zero-shot filtering performance.

To the best of our knowledge, DAZER is the first deep model to conduct zero-shot document filtering. We conduct extensive experiments on two real-world document collections from two different domains (*i.e.,* 20-Newsgroup for topic categorization, and Movie Review for sentiment analysis). Our experimetnal results suggest that DAZER achieves promising filtering performance and performs significantly better than the existing alternative solutions, including state-of-the-art deep relevance ranking methods.

2 Deep Zero-Shot Document Filtering

Figure 1 illustrates the network structure of the proposed DAZER model. It consists of two main components: *relevance signal extraction* and *relevance aggregation*. In the following, we present each component in detail.

2.1 Relevance Signal Extraction

Given a document $d = (w_1, w_2, ..., w_{|d|})$ and a set of seed words $\mathbb{S}_c = \{s_{c,i}\}$ for category c, we first map each word w into its dense word embedding representation $\mathbf{e}_w \in \mathbb{R}^{l_e}$ where l_e denotes the dimension number. The embedding representation is pre-trained by using a representation learning method from an external large text corpus. Since our aim is to capture the conceptual relevance, we simply take the averaged embedding of the seed words to represent a category in the embedding space: $\mathbf{c}_c = 1/|\mathbb{S}_c| \sum_{s \in \mathbb{S}_c} \mathbf{e}_s$.

Interaction-based Representation. It is widely recognized that word embeddings are useful because both syntactic and semantic information of words are well encoded (Mikolov et al., 2013; Pennington et al., 2014). The element-wise hidden feature difference is a kind of relative infor-

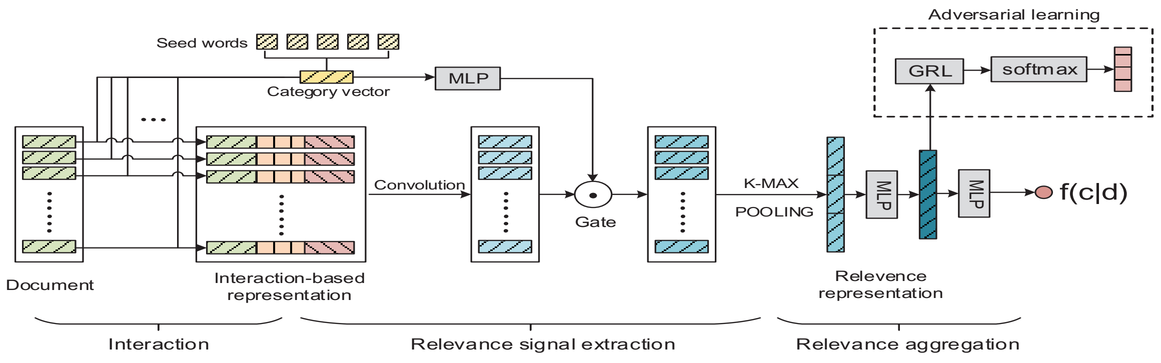


Figure 1: The architecture of DAZER

Examples
$\mathbf{c}_{atheism} - \mathbf{e}_{atheist} \approx \mathbf{c}_{baseball} - \mathbf{e}_{hitter}$
$\mathbf{c}_{autos} - \mathbf{e}_{toyata} \approx \mathbf{c}_{motorcycles} - \mathbf{e}_{yamaha}$
$\mathbf{c}_{baseball} - \mathbf{e}_{stadium} \approx \mathbf{c}_{med} - \mathbf{e}_{hosptial}$
$\mathbf{c}_{religion.misc} - \mathbf{e}_{faith} \approx \mathbf{c}_{med} - \mathbf{e}_{patient}$

Table 1: Examples by using embedding offset.

Examples
$\text{sign}(\mathbf{c}_{mideast} \odot \mathbf{e}_{muslim}) \approx \text{sign}(\mathbf{c}_{med} \odot \mathbf{e}_{doctor})$
$\text{sign}(\mathbf{c}_{space} \odot \mathbf{e}_{orbit}) \approx \text{sign}(\mathbf{c}_{hockey} \odot \mathbf{e}_{espn})$
$\text{sign}(\mathbf{c}_{electronics} \odot \mathbf{e}_{circuit}) \approx \text{sign}(\mathbf{c}_{pc} \odot \mathbf{e}_{controller})$
$\text{sign}(\mathbf{c}_{crypt} \odot \mathbf{e}_{algorithm}) \approx \text{sign}(\mathbf{c}_{space} \odot \mathbf{e}_{burning})$

Table 2: Examples by using element-wise product.

mation that captures the offset bettwen a word and a category in the embedding space. These embedding offsets contain more intricate relationships for a word pair. A well known example is: $\mathbf{e}_{king} - \mathbf{e}_{queen} \approx \mathbf{e}_{man} - \mathbf{e}_{woman}$ (Mikolov et al., 2013). Similar observations are made when we calculate the embedding offset between words and categories. Table 1 lists several interesting patterns observed for the embedding offsets between a category and a word in 20-Newsgroup dataset (ref. Section 3.2 for more details). We can see that the embedding offsets are somehow consistent with a particular relation between the two category-word pairs.

An effective way to measure the relatedness for two words is the inner product or cosine similarity between two corresponding word embeddings. This can be considered as a particular linear combination of corresponding feature products for the two embeddings: $rel(\mathbf{e}_1, \mathbf{e}_2) = \sum_i g(\mathbf{e}_1, \mathbf{e}_2, i)\mathbf{e}_{1,i} \cdot \mathbf{e}_{2,i} = \mathbf{g}(\mathbf{e}_1, \mathbf{e}_2)^T(\mathbf{e}_1 \odot \mathbf{e}_2)$ where $g(\mathbf{e}_1, \mathbf{e}_2, i)$ refers to the weight calculated for i-th dimension, and $\mathbf{g}(\mathbf{e}_1, \mathbf{e}_2) = [g(\mathbf{e}_1, \mathbf{e}_2, 1); ...; g(\mathbf{e}_1, \mathbf{e}_2, l_e)]$, $\odot$ is the element-wise product operation. The element-wise product between two embeddings is also a kind of relative information. The sign of a product of two embeddings in a specific dimension indicates whether the two embeddings share the same polarity in this dimension. And the resultant value manifests to what extent that this agreement/disagreement reaches. It is intuitive that the element-wise product offers some kinds of semantic relations. We conduct the element-wise product for each category-word pair in 20-Newsgroup dataset. Table 2 lists some interesting patterns we observe. The sign(x) function returns 1 when $x \geq 0$, otherwise return -1. Shown in the table, the sign pattern of the element-wise product encodes the relevance information between a category and its related words.

Inspired by these observations, we use these two kinds of element-wise interactions to complement the representation of a word in a document. Specifically, for each word w in document d, we derive its interaction-based representation $\mathbf{e}_w^c$ towards category c as follows:

$$\mathbf{e}_{c,w}^{diff} = \mathbf{c}_c - \mathbf{e}_w \tag{1}$$

$$\mathbf{e}_{c,w}^{prod} = \mathbf{c}_c \odot \mathbf{e}_w \tag{2}$$

$$\mathbf{e}_w^c = [\mathbf{e}_w \oplus \mathbf{e}_{c,w}^{diff} \oplus \mathbf{e}_{c,w}^{prod}] \tag{3}$$

where $\oplus$ is the vector concatenation operation. Note that these two kinds of feature interactions are mainly overlooked by the existing literature. The embedding offsets are used in deriving word semantic hierarchies in (Fu et al., 2014). However, there is no existing work incorporating these two kinds of feature interactions for relevance estimation. Here, we expect that these two kinds of feature interactions can magnify the relevance information regarding the category.

Convolution with k-max Pooling. We utilize

m convolution filters to extract the relevance signals for each word based on its local window of size l in the document. Specifically, after calculating the interaction-based representation $d = (\mathbf{e}_1^c, \mathbf{e}_2^c, ..., \mathbf{e}_{|d|}^c)$ for document d and category c, we apply the convolution operation as follows:

$$\mathbf{r}_i = \mathbf{W}_1 \mathbf{e}^c_{i-l:i+l} + \mathbf{b}_1 \tag{4}$$

where $\mathbf{r}_i \in \mathbb{R}^m$ is the hidden features regarding the relevance signal extracted for i-th word, $\mathbf{W}_1 \in \mathbb{R}^{m \times 3l_e(2l+1)}$ and $\mathbf{b}_1 \in \mathbb{R}^m$ are the weight matrix and the corresponding bias vector respectively, $\mathbf{e}^c_{i-l:i+l}$ refers to the concatenation from $\mathbf{e}^c_{i-l}$ to $\mathbf{e}^c_{i+l}$. Both l zero vectors are padded to the begining and the end of the document. With a local window of size l, the convolution operation can extract more accurate relevance information by taking the consecutive words (*e.g.,* phrases) into account. We then apply k-max pooling strategy to obtain the k most active features for each filter. Let $\mathbf{r}^j_{k-max}$ denote the k largest values for filter j, we form the overall relevance signals $\mathbf{r}_d$ extracted by all m filters through the concatenation: $\mathbf{r}_{c,d} = [\mathbf{r}^1_{k-max} \oplus \mathbf{r}^2_{k-max} \cdots \oplus \mathbf{r}^m_{k-max}]$.

Category-specific Gating Mechanism. Given a specific word w, the interaction-based representation $\mathbf{e}^c_w$ for each category c could be very different. Therefore, for a specific local context, the extracted relevance signal from a particular convolution filter could be also distinct for different categories. It is then reasonable to assume that the relevance signals for a specific category are captured by a subset of filters. We propose to identify which filters are relevant to a category through a category-specific gating mechanism. Given category c, category-specific gates $\mathbf{a}_c \in \mathbb{R}^m$ are calculated as follows:

$$\mathbf{a}_c = \sigma(\mathbf{W}_2 \mathbf{e}_c + \mathbf{b}_2) \tag{5}$$

where $\mathbf{W}_2 \in \mathbb{R}^{m \times 3l_e}$ and $\mathbf{b}_2 \in \mathbb{R}^m$ are the weight matrix and bias vectors respectively, $\sigma(\cdot)$ is the *sigmoid* function. With category-specific gating mechanism, Equation 4 can be rewritten as follows: $\mathbf{r}_i = \mathbf{a}_c \odot (\mathbf{W}_1 \mathbf{e}^c_{i-l:i+l} + \mathbf{b}_1)$

Here, $\mathbf{a}_c$ works as *on-off* switches for m filters. While $\mathbf{a}_{c,j} \rightarrow 1$ indicates that j-th filter should be turned on to capture the relevance singals under category c to its fullness, $\mathbf{a}_{c,j} \rightarrow 0$ indicates that the filter is turned off due to its irrelevance.

This collaboration of the convolution operation and gating mechanism is similar to the Gated Linear Units (GLU) recently proposed in (Dauphin et al., 2017). Given an input $\mathbf{X}$, GLU calculates the output as follow: $\mathbf{h}(\mathbf{X}) = (\mathbf{X}\mathbf{W} + \mathbf{b}) \odot \sigma(\mathbf{X}\mathbf{V} + \mathbf{c})$ where the first term in the right side refers to the convolution operation and the second term in the right side refers to the gating mechanism. In GLU, both the convolution operation and the gates share the same input $\mathbf{X}$. In contrast, in this work, we aim to identify which filters capture the relevance signals in a category-dependent manner. The experimental results validate that this category-dependent setting brings significant benefit for zero-shot filtering performance (ref. Section 3).

2.2 Relevance Aggregation

The raw relevance signals $\mathbf{r}_{c,d}$ are somehow category-dependent, since the relevant filters are category-dependent. The hidden features regarding the relevance are distilled through a fully-connected hidden layer with nonlinearity:

$$\mathbf{h}_{c,d} = g_a(\mathbf{W}_3 \mathbf{r}_{c,d} + \mathbf{b}_3) \tag{6}$$

where $\mathbf{W}_3 \in \mathbb{R}^{l_a \times 3km}$ and $\mathbf{b}_3 \in \mathbb{R}^{l_a}$ are the weight matrix and bias vector respectively, $g_a(\cdot)$ is the *tanh* function. This procedure can be considered as a relevance aggregation process. Then, the overall relevance score is then estimated as follow:

$$f(c|d) = tanh(\mathbf{w}^T \mathbf{h}_{c,d} + b) \tag{7}$$

where $\mathbf{w} \in \mathbb{R}^{l_a}$ and b are the parameters and bias respective.

2.3 Model Training

Adversarial Learning The hidden features $\mathbf{h}_{c,d}$ are expected to be category-independent. However, there is no guarantee that the category-specific information is not mixed with the relevance information extracted in $\mathbf{h}_{c,d}$. Here, we introduce an adversarial learning mechanism to ensure that no category-specific information can be memorized during the training. Otherwise, the proposed DAZER may not generalize well to unseen categories. Specifically, we introduce an category classifier over $\mathbf{h}_{c,d}$ to calculate the probability that $\mathbf{h}_{c,d}$ belongs to each category seen during the training: $p_{cat}(\cdot|\mathbf{h}_{c,d}) = softmax(\mathbf{W}_4 \mathbf{h}_{c,d} +$

$\mathbf{b}_4$) where $\mathbf{W}_4 \in \mathbb{R}^{C \times l_a}$ and $\mathbf{b}_4 \in \mathbb{R}^C$ are the weight matrix and bias vector for the classifier, C is the number of categories covered by the training set. We aim to optimize parameters $\phi = \{\mathbf{W}_4, \mathbf{b}_4\}$ to successfully classify $\mathbf{h}_{c,d}$ to its true category. Let θ denote the parameters regarding the calculation of $\mathbf{h}_{c,d}$, *i.e.,* $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$, ϕ is optimized to minimize the negative log-likelihood:

$$L_{cat}(\theta, \phi) = \frac{1}{|\mathbb{T}|} \sum_{(d,y) \in \mathbb{T}} -p_{cat}(y|\mathbf{h}_{y,d}) \tag{8}$$

where $\mathbb{T}$ denotes the training set $\{(d, y)\}$ such that document d is relevant to category y. On the other hand, we expect that $\mathbf{h}_{c,d}$ carries no category specific information, such that the classifier can not perform the category classification precisely. Hence, we add the Gradient Reversal Layer (GRL) (Ganin and Lempitsky, 2015; Ganin et al., 2016) between $\mathbf{h}_{c,d}$ and the category classifier. We can consider GRL as a pseudo-function $R_\lambda(\mathbf{x})$:

$$R_\lambda(\mathbf{x}) = \mathbf{x}; \quad \frac{\partial R_\lambda}{\partial \mathbf{x}} = -\lambda \mathbf{I} \tag{9}$$

It means that θ is optimized to make $\mathbf{h}_{c,d}$ indistinguishable by the classifier. In Equation 9, parameter λ controls the importance of the adversarial learning. DAZER is devised to return a relevance score, we utilize the pairwise margin loss for model training:

$$\begin{aligned} L_{hinge}(\theta, \delta) =& \frac{1}{|\mathbb{T}|} \sum_{(d,y) \in \mathbb{T}} \max(0, \Delta - f(y|d) \\ &+ f(y|d_y^-)) \end{aligned} \tag{10}$$

where document d_y^- is the negative sample for category y, Δ is the margin and set to be 1 in this work, and $\delta = \{\mathbf{w}, b\}$. Overall, the proposed DAZER is an end-to-end neural network model. The parameters $\Theta = \{\theta, \phi, \delta\}$ are optimized via back propagation and stochastic gradient descent. Specifically, we utilize Adam (Kingma and Ba, 2014) algorithm for parameter update over minibatches. The final objective loss used in the training is as follow:

$$L(\Theta) = L_{hinge}(\theta, \delta) + L_{cat}(\theta, \phi) + \lambda_\Theta \|\Theta\|_2 \tag{11}$$

where λ_Θ controls the importance of the regularizaton term.

Label	Seed Words
very negative	bad, horrible, negative, disgusting
negative	bad, confused, unexpected, useless, negative
neutral	normal, moderate, neutral, objective, impersonal
positive	good, positive, outstanding, satisfied, pleased
very positive	positive, impressive, unbelievable, awesome

Table 3: Seed words selected for Movie Review.

3 Experiment

In this section, we conduct experiments on two real-world document collections to evaluate the effectiveness of the proposed DAZER[1].

3.1 Existing Alternative Methods

Here, we compare the proposed DAZER against the following alternative solutions.

BM25 Model: BM25 is a widely known retrieval model based on keyword matching (Robertson and Walker, 1994). The default parameter setting is used in the experiments.

DSSM: DSSM utilizes a multi-layer perceptron to extract hidden representations for both the document and the query (Huang et al., 2013). Then, cosine similarity is calculated as the relevance score based on the representation vectors. Since we use pre-trained word embeddings from a large text corpus, we choose to replace the letter-tri-grams representation with the word embedding representation instead. We use the recommended network setting by its authors.

DRMM: DRMM calculates the relevance based on the histogram information of the semantic relatedness scores between each word in the document and each query word (Guo et al., 2016a). The recommended network setting (*i.e.,* LCH$\times$IDF) and parameter setting are used.

K-NRM: K-NRM is a kernel based neural model for relevance ranking based on word-level hard/soft matching signals (Xiong et al., 2017). We use the recommended setting as in their paper.

DeepRank: DeepRank is a neural relevance ranking model based on the query-centric context (Pang et al., 2017). The recommended setting is used for evaluation.

Seed-based Support Vector Machines (SSVM): We build a seed-driven training set by labeling a training document with a category if the document

[1] The implementation is available at `https://github.com/WHUIR/DAZER`

contains any seed word of that category. Then, we adopt a one-class SVM implemented by sklearn[2] for document filtering[3]. The optimal performance is reported by tuning the hyper-parameter.

3.2 Datasets and Experimental Setup

20-Newsgroup (20NG)[4] is a widely used benchmark for document classification research (Li et al., 2016). It consists of approximately 20K newsgroup articles from 20 different categories. The *bydate* version with $18,846$ documents is used here. As provided, the training set and test set contain 60% and 40% documents respectively.

Movie Review[5] is a collection of movie reviews in English (Pang and Lee, 2005). The scale dataset v1.0 is used in the experiments. Based on the numerical ratings, we split these reviews into five sentiment labels: *very negative, negative, neutral, positive* and *very positive*, which contains 167, 1030, 1786, 1682, 341 reviews respectively. For each sentiment label, we randomly split the reviews into a training set (80%) and a test set (20%).

Since our work targets at zero-shot document filtering for unseen categories, the word embeddings pre-trained by Glove over a large text corpus with total 840 billion tokens[6] are used across all the methods and the two datasets. The dimension of the word embeddings is $l_e = 300$. No further word embedding fine-tuning is applied. For both datasets, the stop words are removed firstly. Then, all the words are converted into their lowercased forms. We further remove the words whose word embeddings are not supported by Glove.

Evaluation Protocol. With the specified unseen categories, we take all the training documents of the other categories to train a model. Then, all documents in the test set are used for evaluation. For each unseen category, the task is to rank the documents of that category higher than the others. Here, we choose to report mean average precision (MAP) for performance evaluation. MAP is a widely used metric to evaluate the ranking quality. The higher the relevant documents are ranked, the larger the MAP value is, which means a better filtering performance. For all neural networks based models, the training documents from one randomly sampled training category work as the validation set for early stop. We report the averaged results over 5 runs for all the methods (excluding SSVM and BM25). The statistical significance is conducted by applying the student t-test.

Seed Word Selection. For 20NG dataset, we directly use the seed words[7] manually compiled in (Song and Roth, 2014). These seed words are selected from the category descriptions and widely used in the works of dataless text classification (Song and Roth, 2014; Chen et al., 2015; Li et al., 2016). For Movie Review, following the seed word selection process (*i.e.,* assisted by standard LDA) proposed in (Chen et al., 2015), we manually select the seed words for each sentiment label. Table 3 lists the seed words selected for each sentiment label for Movie Review dataset. There are on average 5.2 and 4.6 seed words for each category over 20NG and Movie Review respectively. It is worthwhile to highlight that no category information is exploited within the seed word selection process.

Parameter Setting. For DAZER, the number of convolution filters is $m = 50$ and $k = 3$ is used for k-max pooling. The dimension size for relevance aggregation is $l_a = 75$. The local window size l is set to be 2. The learning rate is 0.00001. The models are trained with a batch size of 16 and $\lambda_\Theta = 0.0001$, $\lambda = 0.1$.

3.3 Performance Comparison

For 20NG dataset, we randomly create 9 document filtering tasks which cover 10 out of 20 categories. For Movie Review, we take each sentiment label as an unseen category for evaluation. Table 4 lists the performance of 7 methods in terms of MAP for these filtering tasks. Here, we make the following observations.

First, the proposed DAZER significantly achieves much better filtering performance on all 14 tasks across the two datasets. The averaged MAP of DAZER over these 14 filtering tasks is 0.671. Note that only 5.2 and 4.6 seed words are used on average for each task. The second best performer is K-NRM, which achieves the second

[2] http://scikit-learn.org

[3] Signed distance to the separating hyperplan is used for ranking documents.

[4] http://qwone.com/~jason/20Newsgroups/

[5] The Movie Review dataset is available at http://www.cs.cornell.edu/people/pabo/movie-review-data/

[6] https://nlp.stanford.edu/projects/glove/

[7] The seed words are available at https://github.com/WHUIR/STM

Dataset	Category	DAZER	DRMM	K-NRM	DeepRank	DSSM	SSVM	BM25
20NG	pc	**0.535**	$\underline{0.382}^{\dagger}$	$0.369^{\dagger}$	$0.144^{\dagger}$	$0.222^{\dagger}$	0.117	0.313
	med	**0.826**	$\underline{0.662}^{\dagger}$	$0.645^{\dagger}$	$0.033^{\dagger}$	$0.192^{\dagger}$	0.104	0.403
	baseball	**0.764**	$0.731^{\dagger}$	$\underline{0.735}^{\dagger}$	$0.294^{\dagger}$	$0.373^{\dagger}$	0.291	0.414
	space	**0.780**	$0.593^{\dagger}$	$\underline{0.671}^{\dagger}$	$0.285^{\dagger}$	$0.142^{\dagger}$	0.140	0.641
	med-space	**0.805**	$0.640^{\dagger}$	$\underline{0.666}^{\dagger}$	$0.101^{\dagger}$	$0.174^{\dagger}$	0.122	0.522
	atheism-electronics	**0.464**	$0.242^{\dagger}$	$0.346^{\dagger}$	$\underline{0.418}^{\dagger}$	$0.219^{\dagger}$	0.132	0.263
	christian-mideast	**0.712**	$\underline{0.662}^{\dagger}$	$0.657^{\dagger}$	$0.298^{\dagger}$	$0.327^{\dagger}$	0.161	0.579
	baseball-hockey	**0.782**	$0.642^{\dagger}$	$\underline{0.736}^{\dagger}$	$0.332^{\dagger}$	$0.135^{\dagger}$	0.438	0.444
	pc-windowx-electronics	**0.489**	$0.274^{\dagger}$	$\underline{0.379}^{\dagger}$	$0.183^{\dagger}$	$0.278^{\dagger}$	0.120	0.314
Movie Review	very negative	**0.290**	$0.119^{\dagger}$	$0.114^{\dagger}$	$0.097^{\dagger}$	$\underline{0.216}^{\dagger}$	0.080	0.134
	negative	**0.807**	$0.528^{\dagger}$	$\underline{0.557}^{\dagger}$	$0.423^{\dagger}$	$0.478^{\dagger}$	0.236	0.090
	neutral	**0.798**	$\underline{0.764}^{\dagger}$	$0.749^{\dagger}$	$0.686^{\dagger}$	$0.678^{\dagger}$	0.365	0.007
	positive	**0.862**	$0.696^{\dagger}$	$0.706^{\dagger}$	$0.655^{\dagger}$	$\underline{0.753}^{\dagger}$	0.300	0.090
	very positive	**0.479**	$0.250^{\dagger}$	$\underline{0.339}^{\dagger}$	$0.217^{\dagger}$	$0.271^{\dagger}$	0.063	0.066
Avg		**0.671**	0.513	$\underline{0.548}$	0.298	0.318	0.191	0.306

Table 4: Performance of the 7 methods for zero-shot document filtering in terms of MAP. The best and second best results are highlighted in boldface and underlined respectively, on each task. † indicates that the difference to the best result is statistically significant at 0.05 level. Avg: averaged MAP over all tasks.

best on 7 tasks. Overall, the averaged performance gain for DAZER over K-NRM is about 30.8%.

Second, We observe that DSSM performs signficantly better for sentiment analysis than for topic categorization. As discussed in Section 4, DSSM is designed to perform semantic matching. Compared with topic categorization, sentiment analysis is more like a semantic matching task. SSVM delivers the worst performance on both datasets. This illustrates that the quality of the labeled documents is essential for supervised learning techniques. Apparently, recruiting training documents with the provided seed words in a simple fashion is error-prone. We also note that BM25 achieves inconsistent performance over the two kinds of tasks. It performs especially worse for sentiment analysis. This is reasonable because there are more diverse ways to express a specific sentiment. It is hard to cover a reasonable proportion of documents with limited number of sentimental seed words. In comparison, the proposed DAZER obtains a consistent performance for both topic categorization and sentiment analysis.

3.4 Analysis of DAZER

Component Setting. Here, we further discuss the impact of different component settings of DAZER on both 20NG and Movie Review datasets. Table 5 and 6 report the impacts of each component setting via an ablation test on the two datasets respectively. We can see that each component brings significantly positive benefit for document filtering. First, we can see that either element-wise subtraction or product contributes signifcantly to the performance improvement. Specifically, from Table 6, we can see that both the element-wise subtraction and element-wise product play equally on Movie Review dataset. On the other hand, it is observed that DAZER experiences significantly a much larger performance degradation on 20NG dataset. For example, a MAP of only 0.154 is achieved when $\mathbf{e}_{c,w}^{prod}$ is excluded from DAZER for the filtering task *space*. A much severer case is for the filtering task *baseball-hockey*. By excluding $\mathbf{e}_{c,w}^{prod}$, the MAP performance of DAZER is reduced from 0.782 to 0.045. That is, the element-wise product is more critical for extracting relevance signals for topical categorization. We also observe that these two hidden feature interactions together play a more important role for DAZER. For example, without both $\mathbf{e}_{c,w}^{diff}$ and $\mathbf{e}_{c,w}^{prod}$, DAZER only achieves a MAP of 0.126 for filtering task *space*. The large performance deterioration is also observed for other filtering tasks on 20NG dataset.

Either adversarial learning or category-specific gate mechanism enhances the filtering performance of DAZER, which validates the effectiveness of the two components for enhancing con-

Setting	pc	med	baseball	space	med-space	atheism-electronics	christian-mideast	baseball-hockey	pc-windowx-electronics
DAZER	**0.535**	**0.826**	**0.764**	**0.780**	**0.805**	**0.464**	**0.712**	0.782	**0.489**
- $\mathbf{e}^{diff}_{c,w}$	0.524	0.810	0.755	0.785	0.802	0.454	0.705	**0.788**	0.462
- $\mathbf{e}^{prod}_{c,w}$	0.219	0.043	0.200	0.154	0.139	0.217	0.244	0.045	0.141
- Gate	0.518	0.819	0.715	**0.780**	0.803	0.443	0.695	0.784	**0.489**
- Adv	0.531	0.819	0.749	0.775	0.795	0.458	0.701	0.779	0.485

Table 5: Impact of different settings for DAZER on 20NG. The best results are highlighted in boldface. - $\mathbf{e}^{diff}_{c,w}$: no element-wise subtraction; - $\mathbf{e}^{prod}_{c,w}$: no element-wise product; - Gate: no category-specific gate mechanism; - Adv: no adversarial learning.

Setting	very negative	negative	neutral	positive	very positive
DAZER	**0.290**	**0.807**	**0.798**	**0.862**	**0.479**
- $\mathbf{e}^{diff}_{c,w}$	0.246	0.773	0.776	0.847	0.453
- $\mathbf{e}^{prod}_{c,w}$	0.258	0.779	0.785	0.847	0.430
- Gate	0.278	0.755	0.785	0.848	0.429
- Adv	0.261	0.779	0.776	0.827	0.444

Table 6: Impact of different settings for DAZER on Movie Review. The best results are highlighted in boldface. - $\mathbf{e}^{diff}_{c,w}$: no element-wise subtraction; - $\mathbf{e}^{prod}_{c,w}$: no element-wise product; - Gate: no category-specific gate mechanism; - Adv: no adversarial learning.

ceptual relevance extraction. Also, without using adversarial learning, DAZER still achieves much better filtering performance than the existing baseline methods compared in Section 3.3. This observation is also held on 20NG dataset. This further validates that the two kinds of hidden feature interactions indeed encode rich knowledge towards the conceptual relevance.

Impact of Seed Words. It has been recognized that the less seed words incur worse document classification performance in the existing dataless document classification techniques (Song and Roth, 2014; Chen et al., 2015; Li et al., 2016). Following these works, we also use the words appearing in the category name of 20NG dataset as the corresponding seed words[8]. There are on average 2.75 seed words for a category of 20NG. Table 7 reports the MAP performace of each method on 20NG dataset. The experimental results show that all methods investigated in Section 3.3 experience signficant performance degradation for most filtering tasks. We plan to incorporate the pseudo-relevance feedback into DAZER to tackle the scarcity of the seed words. One possible solution is to enrich the architecture of DAZER to allow few-shot document filtering. That is, the filtering decisions of high-confidence are utilized to derive more seed words for better filtering performance.

[8]The seed words based on the category name are available at https://github.com/WHUIR/STM

4 Related Work

Document filtering is the task to separate relevant documents from the irrelevant ones for a specific topic (Robertson and Soboroff, 2002; Nanas et al., 2010; Gao et al., 2013, 2015; Proskurnia et al., 2017). Both ranking and classification based solutions have been developed (Harman, 1994; Robertson and Soboroff, 2002; Soboroff and Robertson, 2003). In earlier days, a filtering system is mainly devised to facilitate the document retrieval for the long-term information needs (Mostafa et al., 1997). The term-based pattern mining techniques are widely developed to perform document filtering. A network-based topic profile is built to exploit the term correlation patterns for document filtering (Nanas et al., 2010). Frequent term patterns in terms of fine-grained hidden topics are proposed in (Gao et al., 2013, 2015) for doucment filtering. Very recently, frequent term patterns are also utilized to perform event-based microblog filtering (Proskurnia et al., 2017). However, these approaches are all based on supervised-learning, which requires a significant amount of positive documents for each topic. In the era of big data, the information space and new information needs are continuously growing. Retrieval of the relevance information in a short response time becomes a fundamental need. Recently, many works have been proposed to conduct document filtering in an entity-centric manner (Frank et al., 2012; Balog and Ramampiaro, 2013; Zhou and Chang, 2013; Reinanda et al., 2016). The task is to identify the documents relevant to a specific entity that is well defined in an

Dataset	Category	DEZA	DRMM	KNRM	DeepRank	DSSM	SSVM	BM25
20NG	pc	**0.316**	0.170	0.144	0.104	**0.316**	0.057	0.092
	med	**0.831**	0.369	0.267	0.183	0.089	0.040	0.000
	baseball	**0.519**	0.315	0.301	0.299	0.419	0.066	0.161
	space	**0.641**	0.337	0.326	0.414	0.212	0.049	0.329
	med-space	**0.670**	0.348	0.331	0.279	0.076	0.044	0.165
	atheism-electronics	0.359	0.266	0.253	**0.499**	0.141	0.042	0.091
	christian-mideast	0.564	**0.582**	0.492	0.196	0.418	0.061	0.093
	baseball-hockey	**0.577**	0.409	0.391	0.336	0.154	0.061	0.194
	pc-windowx-electronics	**0.346**	0.176	0.194	0.185	0.227	0.067	0.124

Table 7: Performance of the 7 methods for zero-shot document filtering in terms of MAP. The words appearing in the category name are used as the seed words. The best and second best results are highlighted in boldface and underlined respectively, on each task.

external knowledge base. Specifically, Balog and Ramampiaro (2013) examine the choice of classification against ranking approaches. They found that ranking approach is more suitable for the filtering task. Following this conclusion, we formulate the zero-shot document filtering as a relevance ranking task. Many information needs may not be well represented by a specific entity. Hence, these entity-centric solutions are restricted to knowledge base related tasks.

Many ad-hoc retrieval models can be used to perform zero-shot document filtering. Indeed, traditional term-based document filtering approaches utilize many term-weighting schemes developed for ad-hoc retrieval. Traditional ad-hoc retrieval models mainly estimate the relevance based on keyword matching. BM25 (Robertson and Walker, 1994) can be considered as the optimal practice in this line of literature. The recent advances in word embedding offer effective learning of word semantic relations from a large external corpus. Several neural relevance ranking models are proposed to preform ad-hoc retrieval based on word embeddings. Both K-NRM (Xiong et al., 2017) and DRMM (Guo et al., 2016a) estimate the relevance based on the macro-statistics of the hard/soft-match signals in terms of cosine similarity between two word embeddings. DeepRank (Pang et al., 2017) first measures the relevance signals from the query-centric context of each query keyword matching point through convolutional operations. Then, RNN based networks are adopted to aggregate these relevance signals. These works achieve significantly better retrieval performance than the keyword matching based solutions and represent the new state-of-the-art. The relevance between a query and a document can also be considered as a matching task between two pieces of text. There are many deep matching models, *e.g.,* DSSM (Huang et al., 2013), ARC-II (Hu et al., 2014), MatchPyramid (Pang et al., 2016), Match-SRNN (Wan et al., 2016). These models are mainly developed for some specific *semantic matching* tasks, *e.g.,* paraphrase identification. Therefore, information like grammatical structure or sequence of words are often taken into consideration, which is not applicable to seed word based zero-shot document filtering.

5 Conclusion

In this paper, we propose a novel **d**eep relev**a**nce model for **z**ero-shot document filt**er**ing, named DAZER. To enable DAZER to capture conceptual relevance and generalize well to unseen categories, two kinds of feature interactions, a gated convolutional network and an category-independent adversarial learning are devised. The experimental results over two different tasks validate the superiority of the proposed model. In the future, we plan to enrich the architecture of DAZER to allow few-shot document filtering by incorporating several labeled examples.

6 Acknowledgement

This research was supported by National Natural Science Foundation of China (No.61502344), Natural Science Foundation of Hubei Province (No.2017CFB502), Natural Scientific Research Program of Wuhan University (No.2042017kf0225). Chenliang Li is the corresponding author.

References

Krisztian Balog and Heri Ramampiaro. 2013. Cumulative citation recommendation: classification vs. ranking. In *SIGIR*. pages 941–944.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*. pages 830–835.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John A. Carroll. 2015. Dataless text classification with descriptive LDA. In *AAAI*. pages 2224–2231.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *ICML*. pages 933–941.

Gregory Druck, Gideon S. Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR*. pages 595–602.

Carsten Eickhoff, Sebastian Dungs, and Vu Tran. 2015. An eye-tracking study of query reformulation. In *SIGIR*. pages 13–22.

Hui Fang and ChengXiang Zhai. 2006. Semantic term matching in axiomatic approaches to information retrieval. In *SIGIR*. pages 115–122.

John R. Frank, Max Kleiman-Weiner, Daniel A. Roberts, Feng Niu, Ce Zhang, Christopher Ré, and Ian Soboroff. 2012. Building an entity-centric stream filtering test collection for TREC 2012. In *TREC*.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *ACL*. pages 1199–1209.

Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*. pages 1180–1189.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17:59:1–59:35.

Yang Gao, Yue Xu, and Yuefeng Li. 2013. Pattern-based topic models for information filtering. In *ICDM Workshops*. pages 921–928.

Yang Gao, Yue Xu, and Yuefeng Li. 2015. Pattern-based topics for document modelling in information filtering. *IEEE Trans. Knowl. Data Eng.* 27(6):1629–1642.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016a. A deep relevance matching model for ad-hoc retrieval. In *CIKM*. pages 55–64.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016b. Semantic matching by non-linear word transportation for information retrieval. In *CIKM*. pages 701–710.

Donna Harman. 1994. Overview of the third text retrieval conference (TREC-3). In *TREC*. pages 1–20.

Swapnil Hingmire and Sutanu Chakraborti. 2014. Topic labeled text classification: A weakly supervised approach. In *SIGIR*. pages 385–394.

Swapnil Hingmire, Sandeep Chougule, Girish K. Palshikar, and Sutanu Chakraborti. 2013. Document classification by topic labeling. In *SIGIR*. pages 877–880.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*. pages 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. pages 2333–2338.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Chenliang Li, Jian Xing, Aixin Sun, and Zongyang Ma. 2016. Effective document labeling with very few seed words: A topic model approach. In *CIKM*.

Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2004. Text classification by labeling words. In *AAAI*. pages 425–430.

Tomas Mikolov, Kai Chen, Greg Corrada, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Javed Mostafa, Snehasis Mukhopadhyay, Wai Lam, and Mathew J. Palakal. 1997. A multilevel approach to intelligent information filtering: Model, system, and evaluation. *ACM Trans. Inf. Syst.* 15(4):368–399.

Nikolaos Nanas, Manolis Vavalis, and Anne N. De Roeck. 2010. A network-based model for high-dimensional information filtering. In *SIGIR*. pages 202–209.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*. pages 115–124.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *AAAI*. pages 2793–2799.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *CIKM*. pages 257–266.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. pages 1532–1543.

Julia Proskurnia, Ruslan Mavlyutov, Carlos Castillo, Karl Aberer, and Philippe Cudré-Mauroux. 2017. Efficient document filtering using vector space topic expansion and pattern-mining: The case of event detection in microposts. In *CIKM*. pages 457–466.

Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2016. Document filtering for long-tail entities. In *CIKM*. pages 771–780.

Stephen E. Robertson and Ian Soboroff. 2002. The TREC 2002 filtering track report. In *TREC*.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*. pages 232–241.

Ian Soboroff and Stephen E. Robertson. 2003. Building a filtering test collection for TREC 2002. In *SIGIR*. pages 243–250.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*. pages 1579–1585.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial RNN. In *IJCAI*. pages 2922–2928.

Ho Chung Wu, Robert W. P. Luk, Kam-Fai Wong, and K. L. Kwok. 2007. A retrospective study of a hybrid document-context based retrieval model. *Inf. Process. Manage.* 43(5):1308–1331.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR*. pages 55–64.

Mianwei Zhou and Kevin Chen-Chuan Chang. 2013. Entity-centric document filtering: boosting feature mapping through meta-features. In *CIKM*. pages 119–128.

Disconnected Recurrent Neural Networks for Text Categorization

Baoxin Wang
Joint Laboratory of HIT and iFLYTEK, iFLYTEK Research, Beijing, China
bxwang2@iflytek.com

Abstract

Recurrent neural network (RNN) has achieved remarkable performance in text categorization. RNN can model the entire sequence and capture long-term dependencies, but it does not do well in extracting key patterns. In contrast, convolutional neural network (CNN) is good at extracting local and position-invariant features. In this paper, we present a novel model named disconnected recurrent neural network (DRNN), which incorporates position-invariance into RNN. By limiting the distance of information flow in RNN, the hidden state at each time step is restricted to represent words near the current position. The proposed model makes great improvements over RNN and CNN models and achieves the best performance on several benchmark datasets for text categorization.

1 Introduction

Text categorization is a fundamental and traditional task in natural language processing (NLP), which can be applied in various applications such as sentiment analysis (Tang et al., 2015), question classification (Zhang and Lee, 2003) and topic classification (Tong and Koller, 2001). Nowadays, one of the most commonly used methods to handle the task is to represent a text with a low dimensional vector, then feed the vector into a softmax function to calculate the probability of each category. Recurrent neural network (RNN) and convolutional neural network (CNN) are two kinds of neural networks usually used to represent the text.

RNN can model the whole sequence and capture long-term dependencies (Chung et al., 2014). However, modeling the entire sequence sometimes can be a burden, and it may neglect key parts for text categorization (Yin et al., 2017). In contrast, CNN is able to extract local and position-invariant features well (Scherer et al., 2010; Collobert et al., 2011). Table 1 is an example of topic classification, where both sentences should be classified as *Science and Technology*. The key phrase that determines the category is *unsolved mysteries of mathematics*, which can be well extracted by CNN due to position-invariance. RNN, however, doesn't address such issues well because the representation of the key phrase relies on all the previous terms and the representation changes as the key phrase moves.

case1: *One of the seven great* ***unsolved mysteries of mathematics*** *may have been cracked by a reclusive Russian.*
case2: *A reclusive Russian may have cracked one of the seven great* ***unsolved mysteries of mathematics.***

Table 1: Examples of topic classification

In this paper, we incorporate position-invariance into RNN and propose a novel model named Disconnected Recurrent Neural Network (DRNN). Concretely, we disconnect the information transmission of RNN and limit the maximal transmission step length as a fixed value k, so that the representation at each step only depends on the previous $k-1$ words and the current word. In this way, DRNN can also alleviate the burden of modeling the entire document. To maintain the position-invariance, we utilize max pooling to extract the important information, which has been suggested by Scherer et al. (2010).

Our proposed model can also be regarded as a special 1D CNN where convolution kernels are replaced with recurrent units. Therefore, the maximal transmission step length can also be considered as the window size in CNN. Another difference to CNN is that DRNN can increase the window size k arbitrarily without increasing the number of parameters.

We also find that there is a trade-off between position-invariance and long-term dependencies in the DRNN. When the window size is too large, the position-invariance will disappear like RNN. By contrast, when the window size is too small, we will lose the ability to model long-term dependencies just like CNN. We find that the optimal window size is related to the type of task, but affected little by training dataset sizes. Thus, we can search the optimal window size by training on a small dataset.

We conduct experiments on seven large-scale text classification datasets introduced by Zhang et al. (2015). The experimental results show that our proposed model outperforms the other models on all of these datasets.

Our contributions can be concluded as follows:

1. We propose a novel model to incorporate position-variance into RNN. Our proposed model can both capture long-term dependencies and local information well.

2. We study the effect of different recurrent units, pooling operations and window sizes on model performance. Based on this, we propose an empirical method to find the optimal window size.

3. Our proposed model outperforms the other models and achieves the best performance on seven text classification datasets.

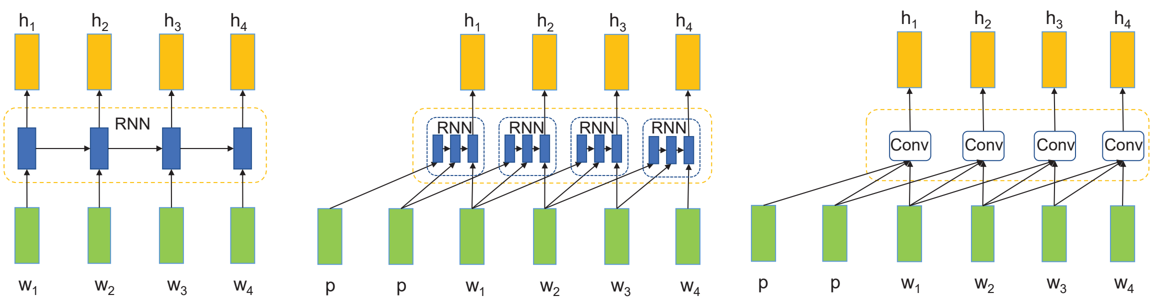
2 Related Work

Deep neural networks have shown great success in many NLP tasks such as machine translation (Bahdanau et al., 2015; Tu et al., 2016), reading comprehension (Hermann et al., 2015), sentiment classification (Tang et al., 2015), etc. Nowadays, nearly most of deep neural networks models are based on CNN or RNN. Below, we will introduce some important works about text classification based on them.

Convolutional Neural Networks CNN has been used in natural language processing because of the local correlation and position-invariance. Collobert et al. (2011) first utilize 1D CNN in part of speech (POS), named entity recognition (NER) and semantic role labeling (SRL). Kim (2014) proposes to classify sentence by encoding a sentence with multiple kinds of convolutional filters. To capture the relation between words, Kalchbrenner et al. (2014) propose a novel CNN model with a dynamic k-max pooling. Zhang et al. (2015) introduce an empirical exploration on the use of character-level CNN for text classification. Shallow CNN cannot encode long-term information well. Therefore, Conneau et al. (2017) propose to use very deep CNN in text classification and achieve good performance. Similarly, Johnson and Zhang (2017) propose a deep pyramid CNN which both achieves good performance and reduces training time.

Recurrent Neural Networks RNN is suitable for handling sequence input like natural language. Thus, many RNN variants are used in text classification. Tang et al. (2015) utilize LSTM to model the relation of sentences. Similarly, Yang et al. (2016) propose hierarchical attention model which incorporates attention mechanism into hierarchical GRU model so that the model can better capture the important information of a document. Wang and Tian (2016) incorporate the residual networks (He et al., 2016) into RNN, which makes the model handle longer sequence. Xu et al. (2016) propose a novel LSTM with a cache mechanism to capture long-range sentiment information.

Hybrid model Some researchers attempt to combine the advantages of CNN and RNN. (Xiao and Cho, 2016) extract local and global features by CNN and RNN separately. (Lai et al., 2015) firstly model sentences by RNN, and then use CNN to get the final representation. Shi et al. (2016) replace convolution filters with deep LSTM, which is similar to what is proposed in this paper. The main differences are as follows. Firstly, they regard their models as CNN and set a small window size of 3, while we propose to use a large window size. We argue that small window size makes the model lose the ability to capture long-term dependencies. Secondly, we utilize max pooling but not mean pooling, because max pooling can maintain position-invariance better (Scherer et al., 2010). Finally, our DRNN model is more general and can make use of different kinds of recurrent units. We find that using GRU as recurrent units outperforms LSTM which is utilized by Shi et al. (2016).



(a) RNN (b) DRNN (c) CNN

Figure 1: Three model architectures. In order to ensure the consistency of the hidden output, we pad $k-1$ zero vectors on the left of the input sequence for DRNN and CNN. Here window size k is 3.

3 Method

3.1 Recurrent Neural Network (RNN)

RNN is a class of neural network which models a sequence by incorporating the notion of time step (Lipton et al., 2015). Figure 1(a) shows the structure of RNN. Hidden states at each step depend on all the previous inputs, which sometimes can be a burden and neglect the key information (Yin et al., 2017).

A variant of RNN has been introduced by Cho et al. (2014) with the name of gated recurrent unit (GRU). GRU is a special type of RNN, capable of learning potential long-term dependencies by using gates. The gating units can control the flow of information and mitigate the vanishing gradients problem. GRU has two types of gates: reset gate $\mathbf{r}_t$ and update gate $\mathbf{z}_t$. The hidden state $\mathbf{h}_t$ of GRU is computed as

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \qquad (1)$$

where $\mathbf{h}_{t-1}$ is the previous state, $\tilde{\mathbf{h}}_t$ is the candidate state computed with new input information and $\odot$ is the element-wise multiplication. The update gate $\mathbf{z}_t$ decides how much new information is updated. $\mathbf{z}_t$ is computed as follows:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \qquad (2)$$

here $\mathbf{x}_t$ is the input vector at step t. The candidate state $\tilde{\mathbf{h}}_t$ is computed by

$$\tilde{\mathbf{h}}_t = tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \qquad (3)$$

where $\mathbf{r}_t$ is the reset gate which controls the flow of previous information. Similarly to the update gate, the reset gate $\mathbf{r}_t$ is computed as:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \qquad (4)$$

We can see that the representation of step t depends upon all the previous input vectors. Thus, we can also express the tth step state shown in Equation (5).

$$\mathbf{h}_t = GRU(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_1) \qquad (5)$$

3.2 Disconntected Recurrent Neural Networks (DRNN)

To reduce the burden of modeling the entire sentence, we limit the distance of information flow in RNN. Like other RNN variants, we feed the input sequence into an RNN model and generate an output vector at each step. One important difference from RNN is that the state of our model at each step is only related to the previous $k-1$ words but not all the previous words. Here k is a hyperparameter called window size that we need to set.

Our proposed model DRNN is illustrated in Figure 1(b). Since the output at each step only depends on the previous $k-1$ words and current word, the output can also be regarded as a representation of a phrase with k words. Phrases with the same k words will always have the same representation no matter where they are. That is, we incorporate the position-invariance into RNN by disconnecting the information flow of RNN.

Similarly, we can get the state $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t = RNN(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-k+1}) \qquad (6)$$

Here k is the window size, and RNN can be naive RNN, LSTM (Hochreiter and Schmidhuber, 1997), GRU or any other kinds of recurrent units.

3.3 Comparison with Convolutional Neural Network (CNN)

DRNN can be considered as a special 1D CNN which replace the convolution filters with recur-

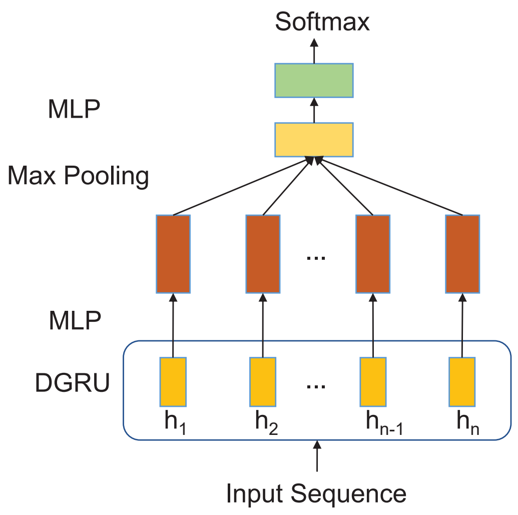


Figure 2: Model architecture

rent units. Let $\mathbf{x}_t$ denote the tth input word vector. Then for each position t we can get a window vector $\mathbf{c}_t$.

$$\mathbf{c}_t = [\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-k+1}] \tag{7}$$

here, we concatenate k word vectors and generate vector $\mathbf{c}_t$. Then we can get the output of convolution as follows:

$$\mathbf{h}_t = \mathbf{W}\mathbf{c}_t + \mathbf{b} \tag{8}$$

where $\mathbf{W}$ is a set of convolution filters and $\mathbf{b}$ is a bias vector. Then a pooling operation can be applied after the convolutional layer and generate a fixed size vector (Kim, 2014). Similarly to RNN and DRNN, we can also represent the context vector of CNN as followings:

$$\mathbf{h}_t = Conv(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_{t-k+1}) \tag{9}$$

Obviously, the parameters of convolution filters $\mathbf{W}$ increase as the window size k increases. By contrast, for DRNN the parameters do not increase with the increase of window size. Hence, DRNN can mitigate overfitting problem caused by the increase of parameters.

3.4 DRNN for Text Classification

DRNN is a general model framework, which can be used for a variety of tasks. In this paper, we only discuss how to apply DRNN in text categorization.

We utilize GRU as recurrent units of DRNN and get the context representation of each step. Every

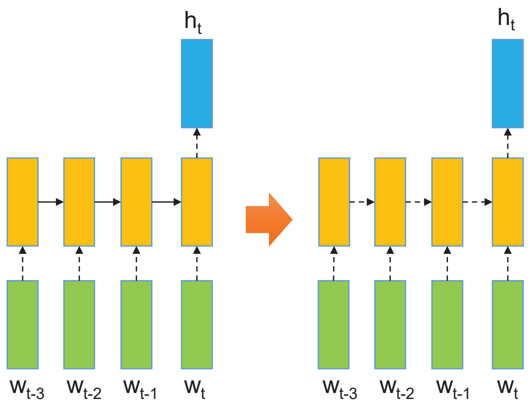


Figure 3: Dropout in DRNN. The dashed arrows indicate connections where dropout is applied. The left model only applies dropout in input and output layers, but the right model applies dropout in hidden states.

context vector can be considered as a representation of a text fragment. Then we feed the context vectors into a multi-layer perceptron (MLP) to extract high-level features as illustrated in Figure 2. Before feeding the vectors into MLP, we utilize Batch Normalization (Ioffe and Szegedy, 2015) after DRNN, so that the model can alleviate the internal covariate shift problem. To get the text representation vector, we apply max pooing after MLP layer to extract the most important information and position-invariant features (Scherer et al., 2010).

Finally, We feed the text representation vector into an MLP with rectified linear unit (ReLU) activation and send the output of MLP to a softmax function to predict the probability of each category. We use cross entropy loss function as follows:

$$H(y, \hat{y}) = \sum_i y_i \log \hat{y}_i \tag{10}$$

where $\hat{y}_i$ is the predicted probability and y_i is the true probability of class i.

To alleviate the overfitting problem, we apply dropout regularization (Srivastava et al., 2014) in DRNN model. Dropout is usually applied in the input and output layers but not the hidden states of RNN, because the number of previous states is variable (Zaremba et al., 2014). In contrast, our DRNN model has a fixed window size for output at each step, so we also apply dropout in the hidden states. In this paper, we apply dropout in the input layer, output layer, and hidden states. The Figure 3 shows the difference to apply dropout between

	AG	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Tasks	News	Ontology	SA	SA	QA	SA	SA
Train dataset	120k	560k	560k	650k	1.4M	3.6M	3M
Test dataset	7.6k	70k	38k	50k	60k	400k	650k
Average Lengths	45	55	153	155	112	93	91
Classes Number	4	14	2	5	10	5	2

Table 2: Dataset information. Here SA refers to sentiment analysis, and QA refers to question answering.

RNN and DRNN.

4 Experiments

4.1 Experimental Settings

Datasets Introduction We use 7 large-scale text classification datasets which are proposed by Zhang et al. (2015). We summarize the datasets in Table 2. AG corpus is news and DBPedia is an ontology which comes from the Wikipedia. Yelp and Amazon corpus are reviews for which we should predict the sentiment. Here *P.* means that we only need to predict the polarities of the dataset, while *F.* indicates that we need predict the star number of the review. *Yahoo! Answers* (Yah. A.) is a question answering dataset. We can see that these datasets contain various domains and sizes, which would be credible to validate our models.

Implementation Details We tokenize all the corpus with NLTK's tokenizer (Bird and Loper, 2004). We limit the vocabulary size of each dataset as shown in Table 3. The words not in vocabulary are replaced with a special token *UNK*. Table 3 also shows the window sizes that we set for these datasets.

We utilize the 300D GloVe 840B vectors (Pennington et al., 2014) as our pre-trained word embeddings. For words that do not appear in GloVe, we average the vector representations of 8 words around the word in training dataset as its word vector, which has been applied by Wang and Jiang (2016). When training our model, word embeddings are updated along with other parameters.

We use Adadelta (Zeiler, 2012) to optimize all the trainable parameters. The hyperparameter of Adadelta is set as Zeiler (2012) suggest that ϵ is $1e-6$ and ρ is 0.95. To avoid the gradient explosion problem, we apply gradient norm clipping (Pascanu et al., 2013). The batch size is set to 128 and all the dimensions of input vectors and hidden states are set to 300.

Corpus	Window size	Vocabulary size
AG	15	100k
DBP.	15	500k
Yelp P.	20	200k
Yelp F.	20	200k
Yah. A.	20	500k
Amz. F.	15	500k
Amz. P.	15	500k

Table 3: Experimental settings

4.2 Experimental Results

Table 4 shows that our proposed model significantly outperforms all the other models in 7 datasets. DRNN does not have too many hyperparameters. The main hyperparameter is the window size which can be determined by an empirical method.

The top block shows the traditional methods and some other neural networks which are not based on RNN or CNN. The linear model (Zhang et al., 2015) achieves a strong baseline in small datasets, but performs not well in large data. FastText (Joulin et al., 2017) and region embedding methods (Qiao et al., 2018) achieve comparable performance with other CNN and RNN based models.

The RNN based models are listed in the second block and CNN based models are in the third block. The D-LSTM (Yogatama et al., 2017) is a discriminative LSTM model. Hierarchical attention network (HAN) (Yang et al., 2016) is a hierarchical GRU model with attentive pooling. We can see that very deep CNN (VDCNN) (Conneau et al., 2017) performs well in large datasets. However, VDCNN is a CNN model with 29 convolutional layers, which needs to be tuned more carefully. By contrast, our proposed model can achieve

Models	AG	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Linear model (Zhang et al., 2015)	7.64	1.31	4.36	40.14	28.96	44.74	7.98
FastText (Joulin et al., 2017)	7.5	1.4	4.3	36.1	27.7	39.8	5.4
Region.emb (Qiao et al., 2018)	7.2	1.1	4.7	35.1	26.3	39.1	4.7
D-LSTM (Yogatama et al., 2017)	7.9	1.3	7.4	40.4	26.3	-	-
HAN (Yang et al., 2016)	-	-	-	-	24.2	36.4	-
char-CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95	28.80	40.43	4.93
word-CNN (Zhang et al., 2015)	8.55	1.37	4.60	39.58	28.84	42.39	5.51
VDCNN (Conneau et al., 2017)	8.67	1.29	4.28	35.28	26.57	37.00	4.28
char-CRNN (Xiao and Cho, 2016)	8.64	1.43	5.51	38.18	28.26	40.77	5.87
DRNN	**5.53**	**0.81**	**2.73**	**30.85**	**23.74**	**35.57**	**3.51**

Table 4: Error rates (%) on seven datasets

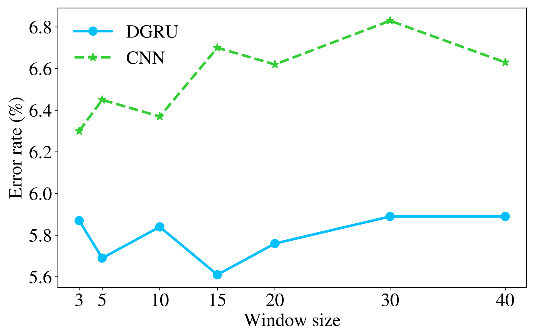


Figure 4: DGRU compared with CNN

Models	AG	DBP.	Yelp P.
CNN	6.30	1.13	4.08
GRU	6.25	0.96	3.41
LSTM	6.20	0.90	3.20
DRNN	**5.53**	**0.81**	**2.73**

Table 5: Comparison with RNN and CNN. Table shows the error rate (%) on three datasets.

better performance in these datasets by simply setting a large window size.

Char-CRNN (Xiao and Cho, 2016) in the fourth block is a model which combines position-invariance of CNN and long-term dependencies of RNN. Nevertheless, they do not achieve great improvements over other models. They first utilize convolution operation to extract position-invariant features, and then use RNN to capture long-term dependencies. Here, modeling the whole sequence with RNN leads to a loss of position-invariance. Compared with their model, our model can better maintain the position-invariance by max pooling (Scherer et al., 2010). Table 4 shows that our model achieves 10-50% relative error reduction compared with char-CRNN in these datasets.

4.3 Comparison with RNN and CNN

In this section, we compare DRNN with CNN, GRU and LSTM (Hochreiter and Schmidhuber, 1997). To make these models comparable, we implement these models with the same architecture shown in Figure 2. We just replace the DRNN with CNN or RNN.

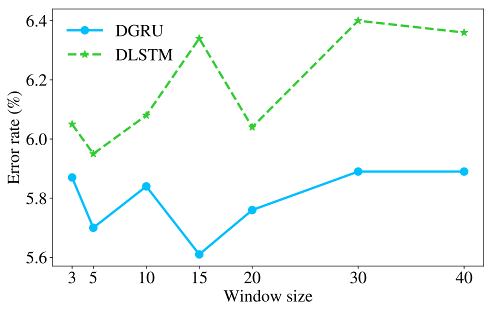
we firstly compare DRNN with CNN on AG dataset. Figure 4 shows that DRNN performs far better than CNN. In addition, the optimal window size of CNN is 3, while for DRNN the optimal window size is 15. It indicates that DRNN can model longer sequence as window size increases. By contrast, simply increasing the window size of CNN only results in overfitting. That is also why Conneau et al. (2017) design complex CNN models to learn long-term dependencies other than simply increase the window size of convolution filters.

In addition, we also compare our model with GRU and LSTM. The experimental results are shown in Table 5. Our model DRNN achieves much better performance than GRU and LSTM.

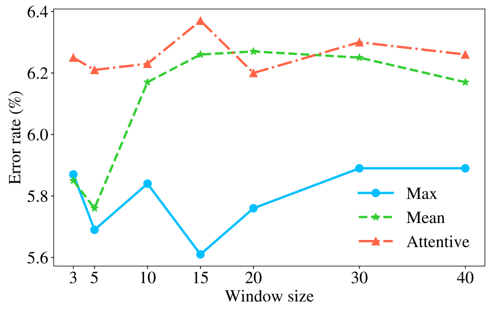
Qualitative Analysis To investigate why DGRU performs better than CNN and GRU, we do some error analysis on Yelp P. dataset. Table 6 shows two examples which have been both

case1: ***I love*** *Hampton Inn* ***but*** *this location is in serious need of remodeling and some deep cleaning. Musty smell everywhere.*
case2: ***Pretty good service***, ***but*** *really busy and noisy!! It gets a little overwhelming because the sales people are very knowledgeable and bombard you with useless techy information to I guess impress you?? Anyways I bought the Ipad 3 and it is* ***freaking awesome*** *and makes up for the store. I would give the Ipad 3 a gazillion stars if I could. I left it at home today and got really sad when I was driving away. Boo Hoo!!*

Table 6: Examples of error analysis. The case 1 is a negative review and case 2 is a positive review. The first example is misclassified by CNN and classified correctly by GRU. The second one is just the contrary. DGRU classify both examples correctly.



(a) Comparison of recurrent units



(b) Comparison of pooling methods

Figure 5: Component comparison

classified correctly by DRNN. The first example is misclassified by CNN and classified correctly by GRU. It is just contrary to the second example. Considering the first example, CNN may extract some key phrases such as *I love* and misclassifies the example as $Positive$, while GRU can model long sequence and capture the information after *but*. For the second example, however, GRU still captures the information after *but* and neglects the key phrases such as *pretty good service* and *freaking awesome*, which leads to the wrong classification.

DGRU can both extract the local key features such as *pretty good service* and capture long-term information such as the sentence after *but*, which makes it perform better than GRU and CNN.

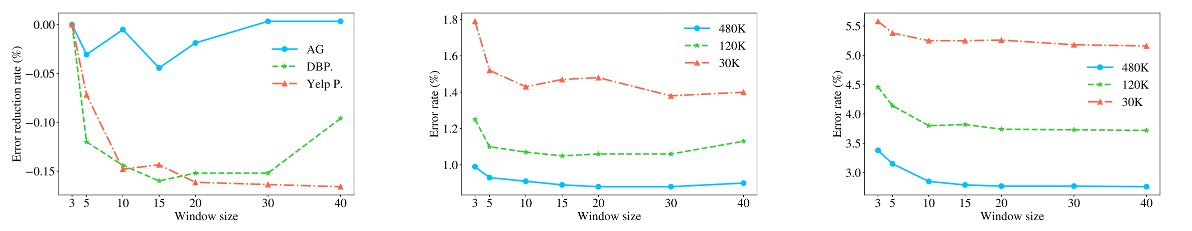
4.4 Component Analysis

Recurrent Unit In this part, we study the impact of different recurrent units on the effectiveness of DRNN. We choose three types of recurrent units: naive RNN, LSTM and GRU which have been compared by Chung et al. (2014). We carry out the experiments with different window sizes to eliminate the impact of window sizes. All the experiments in this part are conducted on the AG dataset.

We find that the disconnected naive RNN performs just a little worse than disconnected LSTM (DLSTM) and disconnected GRU (DGRU) when the window size is lower than 5. However, when the window size is more than 10, its performance decreases rapidly and the error rate becomes even more than 20%. We believe that it is due to vanishing gradient problem of naive RNN.

From Figure 5(a), we can see that window sizes affect the performance of DGRU and DLSTM. DGRU achieves the best performance when the window size is 15, while the best window size for DLSTM is 5. The performance of DGRU is always better than DLSTM no matter what the window size is. We also find that the DGRU model converges faster than DLSTM in the process of training. Therefore, we apply GRU as recurrent units of DRNN in this paper for all the other experiments.

Pooling Method Pooling is a kind of method to subsample the values to capture more important information. In NLP, pooling can also convert a variable-length tensor or vector into a fixed-length



(a) Different tasks (b) Different training sets of DBP. (c) Different training sets of Yelp P.

Figure 6: Window size analysis. For better comparing the trends of different tasks, (a) shows the error reduction rates with different window sizes. (b) and (c) show the error rates of DBP. and Yelp P. with different training set numbers.

one, so that it can be dealt with more easily. There're several kinds of pooling methods such as max pooling, mean pooling and attentive pooling (dos Santos et al., 2016).

We still conduct the experiments on AG dataset. Figure 5(b) shows the experimental results of three pooling methods along with different window sizes. From Figure 5(b), we can see that the DRNN model with max pooling performs better than the others. This may be because that max pooling can capture position-invariant features better (Scherer et al., 2010). We find attentive pooling is not significantly affected by window sizes. However, the performance of mean pooling becomes worse as the window becomes larger.

4.5 Window size analysis

In this section, we mainly study what factors affect the optimal window size. In addition to the recurrent units and pooling methods discussed above, we believe the optimal window size may be also related to the amount of training data and the type of task.

In order to study the factors that affect the optimal window size, we conduct experiments on three datasets: AG, DBP and Yelp Polarity. To eliminate the influence of differrnt training data sizes, we conduct experiments with the same training data size. From Figure 6(a) we can see that the type of task has a great impact on the optimal window size. For AG and DBPedia, the optimal window size is 15. However, for Yelp P. the optimal window size is 40 or even larger. The result is intuitive, because sentiment analysis such as Yelp often involves long-term dependencies (Tang et al., 2015), while topic classification such as AG and DBPedia relys more on the key phrases.

From Figure 6(b) and Figure 6(c) we can see the effect of different training data sizes on the optimal window size. Surprisingly, the effect of different training data sizes on the optimal window size seems little. We can see that for both DBPedia and Yelp corpus, the trend of error rate with the window size is similar. This shows that the number of training data has little effect on the choice of the optimal window size. It also provides a good empirical way for us to choose the optimal window size. That is, conducting experiments on a small dataset first to select the optimal window size.

5 Conclusion

In this paper, we incorporate position-invariance into RNN, so that our proposed model DRNN can both capture key phrases and long-term dependencies. We conduct experiments to compare the effects of different recurrent units and pooling operations. In addition, We also analyze what factors affect the optimal window size of DRNN and present an empirical method to search it. The experimental results show that our proposed model outperforms CNN and RNN models, and achieve the best performance in seven large-scale text classification datasets.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2016YFC0800806). I would like to thank Jianfeng Li, Shijin Wang, Ting liu, Guoping Hu, Shangmin Guo, Ziyue Wang, Xiaoxue Wang and the anonymous reviewers for their insightful comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, page 31.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

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

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. volume 1, pages 1107–1116.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609* 2(3):4.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pages 448–456.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. volume 2, pages 427–431.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1746–1751.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*. volume 333, pages 2267–2273.

Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. A new method of region embedding for text classification. In *International Conference on Learning Representations*.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*. Springer, pages 92–101.

Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1501–1511.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. pages 1422–1432.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research* 2(Nov):45–66.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* .

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of NAACL-HLT*. pages 1442–1451.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 938–943.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367* .

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1660–1669.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1480–1489.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923* .

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* .

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, pages 26–32.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

Joint Embedding of Words and Labels for Text Classification

Guoyin Wang, Chunyuan Li,[*] Wenlin Wang, Yizhe Zhang
Dinghan Shen, Xinyuan Zhang, Ricardo Henao, Lawrence Carin
Duke University
{gw60,cl319,ww107,yz196,ds337,xz139,r.henao,lcarin}@duke.edu

Abstract

Word embeddings are effective intermediate representations for capturing semantic regularities between words, when learning the representations of text sequences. We propose to view text classification as a label-word joint embedding problem: each label is embedded in the same space with the word vectors. We introduce an attention framework that measures the compatibility of embeddings between text sequences and labels. The attention is learned on a training set of labeled samples to ensure that, given a text sequence, the relevant words are weighted higher than the irrelevant ones. Our method maintains the interpretability of word embeddings, and enjoys a built-in ability to leverage alternative sources of information, in addition to input text sequences. Extensive results on the several large text datasets show that the proposed framework outperforms the state-of-the-art methods by a large margin, in terms of both accuracy and speed.

1 Introduction

Text classification is a fundamental problem in natural language processing (NLP). The task is to annotate a given text sequence with one (or multiple) class label(s) describing its textual content. A key intermediate step is the text representation. Traditional methods represent text with hand-crafted features, such as sparse lexical features (*e.g.,* n-grams) (Wang and Manning, 2012). Recently, neural models have been employed to learn text representations, including convolutional neural networks (CNNs) (Kalchbrenner et al., 2014; Zhang et al., 2017b; Shen et al., 2017) and recurrent neural networks (RNNs) based on long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Wang et al., 2018).

To further increase the representation flexibility of such models, attention mechanisms (Bahdanau et al., 2015) have been introduced as an integral part of models employed for text classification (Yang et al., 2016). The attention module is trained to capture the dependencies that make significant contributions to the task, regardless of the distance between the elements in the sequence. It can thus provide complementary information to the distance-aware dependencies modeled by RNN/CNN. The increasing representation power of the attention mechanism comes with increased model complexity.

Alternatively, several recent studies show that the success of deep learning on text classification largely depends on the effectiveness of the word embeddings (Joulin et al., 2016; Wieting et al., 2016; Arora et al., 2017; Shen et al., 2018a). Particularly, Shen et al. (2018a) quantitatively show that the word-embeddings-based text classification tasks can have the similar level of difficulty regardless of the employed models, using the concept of intrinsic dimension (Li et al., 2018). Thus, simple models are preferred. As the basic building blocks in neural-based NLP, word embeddings capture the similarities/regularities between words (Mikolov et al., 2013; Pennington et al., 2014). This idea has been extended to compute embeddings that capture the semantics of word sequences (*e.g.,* phrases, sentences, paragraphs and documents) (Le and Mikolov, 2014; Kiros et al., 2015). These representations are built upon various types of compositions of word vectors, ranging from simple averaging to sophisticated architectures. Further, they suggest that simple models are efficient and interpretable, and have the poten-

[*]Corresponding author

tial to outperform sophisticated deep neural models.

It is therefore desirable to leverage the best of both lines of works: learning text representations to capture the dependencies that make significant contributions to the task, while maintaining low computational cost. For the task of text classification, labels play a central role of the final performance. A natural question to ask is how we can directly use label information in constructing the text-sequence representations.

1.1 Our Contribution

Our primary contribution is therefore to propose such a solution by making use of the label embedding framework, and propose the *Label-Embedding Attentive Model* (LEAM) to improve text classification. While there is an abundant literature in the NLP community on word embeddings (how to describe a word) for text representations, much less work has been devoted in comparison to label embeddings (how to describe a class). The proposed LEAM is implemented by jointly embedding the word and label in the same latent space, and the text representations are constructed directly using the text-label compatibility.

Our label embedding framework has the following salutary properties: (*i*) Label-attentive text representation is informative for the downstream classification task, as it directly learns from a shared joint space, whereas traditional methods proceed in multiple steps by solving intermediate problems. (*ii*) The LEAM learning procedure only involves a series of basic algebraic operations, and hence it retains the interpretability of simple models, especially when the label description is available. (*iii*) Our attention mechanism (derived from the text-label compatibility) has fewer parameters and less computation than related methods, and thus is much cheaper in both training and testing, compared with sophisticated deep attention models. (*iv*) We perform extensive experiments on several text-classification tasks, demonstrating the effectiveness of our label-embedding attentive model, providing state-of-the-art results on benchmark datasets. (*v*) We further apply LEAM to predict the medical codes from clinical text. As an interesting by-product, our attentive model can highlight the informative key words for prediction, which in practice can reduce a doctor's burden on reading clinical notes.

2 Related Work

Label embedding has been shown to be effective in various domains and tasks. In computer vision, there has been a vast amount of research on leveraging label embeddings for image classification (Akata et al., 2016), multimodal learning between images and text (Frome et al., 2013; Kiros et al., 2014), and text recognition in images (Rodriguez-Serrano et al., 2013). It is particularly successful on the task of zero-shot learning (Palatucci et al., 2009; Yogatama et al., 2015; Ma et al., 2016), where the label correlation captured in the embedding space can improve the prediction when some classes are unseen. In NLP, labels embedding for text classification has been studied in the context of heterogeneous networks in (Tang et al., 2015) and multitask learning in (Zhang et al., 2017a), respectively. To the authors' knowledge, there is little research on investigating the effectiveness of label embeddings to design efficient attention models, and how to joint embedding of words and labels to make full use of label information for text classification has not been studied previously, representing a contribution of this paper.

For text representation, the currently best-performing models usually consist of an encoder and a decoder connected through an attention mechanism (Vaswani et al., 2017; Bahdanau et al., 2015), with successful applications to sentiment classification (Zhou et al., 2016), sentence pair modeling (Yin et al., 2016) and sentence summarization (Rush et al., 2015). Based on this success, more advanced attention models have been developed, including hierarchical attention networks (Yang et al., 2016), attention over attention (Cui et al., 2016), and multi-step attention (Gehring et al., 2017). The idea of attention is motivated by the observation that different words in the same context are differentially informative, and the same word may be differentially important in a different context. The realization of "context" varies in different applications and model architectures. Typically, the context is chosen as the target task, and the attention is computed over the hidden layers of a CNN/RNN. Our attention model is directly built in the joint embedding space of words and labels, and the context is specified by the label embedding.

Several recent works (Vaswani et al., 2017; Shen et al., 2018b,c) have demonstrated that simple attention architectures can alone achieve state-of-the-art performance with less computational time, dispensing with recurrence and convolutions entirely. Our work is in the same direction, sharing the similar spirit of retaining model simplicity and interpretability. The major difference is that the aforementioned work focused on self attention, which applies attention to each pair of word tokens from the text sequences. In this paper, we investigate the attention between words and labels, which is more directly related to the target task. Furthermore, the proposed LEAM has much less model parameters.

3 Preliminaries

Throughout this paper, we denote vectors as bold, lower-case letters, and matrices as bold, upper-case letters. We use $\oslash$ for element-wise division when applied to vectors or matrices. We use $\circ$ for function composition, and Δ^p for the set of one hot vectors in dimension p.

Given a training set $\mathcal{S} = \{(\mathbf{X}_n, \boldsymbol{y}_n)\}_{n=1}^N$ of pair-wise data, where $\mathbf{X} \in \mathcal{X}$ is the text sequence, and $\boldsymbol{y} \in \mathcal{Y}$ is its corresponding label. Specifically, $\boldsymbol{y}$ is a one hot vector in single-label problem and a binary vector in multi-label problem, as defined later in Section 4.1. Our goal for text classification is to learn a function $f : \mathcal{X} \mapsto \mathcal{Y}$ by minimizing an empirical risk of the form:

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n=1}^{N} \delta(\boldsymbol{y}_n, f(\mathbf{X}_n)) \tag{1}$$

where $\delta : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ measures the loss incurred from predicting $f(\mathbf{X})$ when the true label is $\boldsymbol{y}$, where f belongs to the functional space $\mathcal{F}$. In the evaluation stage, we shall use the $0/1$ loss as a target loss: $\delta(\boldsymbol{y}, \boldsymbol{z}) = 0$ if $\boldsymbol{y} = \boldsymbol{z}$, and 1 otherwise. In the training stage, we consider surrogate losses commonly used for structured prediction in different problem setups (see Section 4.1 for details on the surrogate losses used in this paper).

More specifically, an input sequence $\mathbf{X}$ of length L is composed of word tokens: $\mathbf{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L\}$. Each token $\boldsymbol{x}_l$ is a one hot vector in the space Δ^D, where D is the dictionary size. Performing learning in Δ^D is computationally expensive and difficult. An elegant framework in NLP, initially proposed in (Mikolov et al., 2013; Le and Mikolov, 2014; Pennington et al., 2014; Kiros et al., 2015), allows to concisely perform learning by mapping the words into an embedding space. The framework relies on so called *word embedding*: $\Delta^D \mapsto \mathbb{R}^P$, where P is the dimensionality of the embedding space. Therefore, the text sequence $\mathbf{X}$ is represented via the respective word embedding for each token: $\mathbf{V} = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_L\}$, where $\boldsymbol{v}_l \in \mathbb{R}^P$. A typical text classification method proceeds in three steps, end-to-end, by considering a function decomposition $f = f_0 \circ f_1 \circ f_2$ as shown in Figure 1(a):

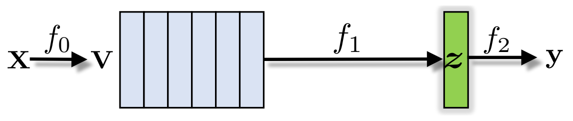
- $f_0 : \mathbf{X} \mapsto \mathbf{V}$, the text sequence is represented as its word-embedding form $\mathbf{V}$, which is a matrix of $P \times L$.
- $f_1 : \mathbf{V} \mapsto \boldsymbol{z}$, a compositional function f_1 aggregates word embeddings into a fixed-length vector representation $\boldsymbol{z}$.
- $f_2 : \boldsymbol{z} \mapsto \boldsymbol{y}$, a classifier f_2 annotates the text representation $\boldsymbol{z}$ with a label.

A vast amount of work has been devoted to devising the proper functions f_0 and f_1, *i.e.,* how to represent a word or a word sequence, respectively. The success of NLP largely depends on the effectiveness of word embeddings in f_0 (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014). They are often pre-trained offline on large corpus, then refined jointly via f_1 and f_2 for task-specific representations. Furthermore, the design of f_1 can be broadly cast into two categories. The popular deep learning models consider the mapping as a "black box," and have employed sophisticated CNN/RNN architectures to achieve state-of-the-art performance (Zhang et al., 2015; Yang et al., 2016). On the contrary, recent studies show that simple manipulation of the word embeddings, *e.g.,* mean or max-pooling, can also provide surprisingly excellent performance (Joulin et al., 2016; Wieting et al., 2016; Arora et al., 2017; Shen et al., 2018a). Nevertheless, these methods only leverage the information from the input text sequence.

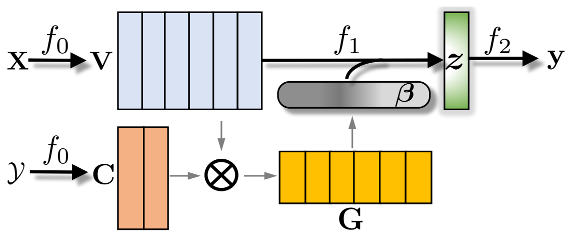
4 Label-Embedding Attentive Model

4.1 Model

By examining the three steps in the traditional pipeline of text classification, we note that the use of label information only occurs in the last step, when learning f_2, and its impact on learning the representations of words in f_0 or word sequences in f_1 is ignored or indirect. Hence, we propose a new pipeline by incorporating label information in every step, as shown in Figure 1(b):



(a) Traditional method



(b) Proposed joint embedding method

Figure 1: Illustration of different schemes for document representations $\boldsymbol{z}$. (a) Much work in NLP has been devoted to directly aggregating word embedding $\mathbf{V}$ for $\boldsymbol{z}$. (b) We focus on learning label embedding $\mathbf{C}$ (how to embed class labels in a Euclidean space), and leveraging the "compatibility" $\mathbf{G}$ between embedded words and labels to derive the attention score $\boldsymbol{\beta}$ for improved $\boldsymbol{z}$. Note that $\otimes$ denotes the cosine similarity between $\mathbf{C}$ and $\mathbf{V}$. In this figure, there are K=2 classes.

- f_0: Besides embedding words, we also embed all the labels in the same space, which act as the "anchor points" of the classes to influence the refinement of word embeddings.
- f_1: The compositional function aggregates word embeddings into $\boldsymbol{z}$, weighted by the compatibility between labels and words.
- f_2: The learning of f_2 remains the same, as it directly interacts with labels.

Under the proposed label embedding framework, we specifically describe a label-embedding attentive model.

Joint Embeddings of Words and Labels We propose to embed both the words and the labels into a joint space *i.e.,* $\Delta^D \mapsto \mathbb{R}^P$ and $\mathcal{Y} \mapsto \mathbb{R}^P$. The label embeddings are $\mathbf{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_K]$, where K is the number of classes.

A simple way to measure the compatibility of label-word pairs is via the cosine similarity

$$\mathbf{G} = (\mathbf{C}^\top \mathbf{V}) \oslash \hat{\mathbf{G}}, \tag{2}$$

where $\hat{\mathbf{G}}$ is the normalization matrix of size $K \times L$, with each element obtained as the multiplication of ℓ_2 norms of the c-th label embedding and l-th word embedding: $\hat{g}_{kl} = \|\boldsymbol{c}_k\|\|\boldsymbol{v}_l\|$.

To further capture the relative spatial information among consecutive words (*i.e.,* phrases[1]) and introduce non-linearity in the compatibility measure, we consider a generalization of (2). Specifically, for a text phase of length $2r+1$ centered at l, the local matrix block $\mathbf{G}_{l-r:l+r}$ in $\mathbf{G}$ measures the label-to-token compatibility for the "label-phrase" pairs. To learn a higher-level compatibility stigmatization $\boldsymbol{u}_l$ between the l-th phrase and all labels, we have:

$$\boldsymbol{u}_l = \text{ReLU}(\mathbf{G}_{l-r:l+r}\mathbf{W}_1 + \boldsymbol{b}_1), \tag{3}$$

where $\mathbf{W}_1 \in \mathbb{R}^{2r+1}$ and $\boldsymbol{b}_1 \in \mathbb{R}^K$ are parameters to be learned, and $\boldsymbol{u}_l \in \mathbb{R}^K$. The largest compatibility value of the l-th phrase *wrt* the labels is collected:

$$m_l = \text{max-pooling}(\boldsymbol{u}_l). \tag{4}$$

Together, $\boldsymbol{m}$ is a vector of length L. The compatibility/attention score for the entire text sequence is:

$$\boldsymbol{\beta} = \text{SoftMax}(\boldsymbol{m}), \tag{5}$$

where the l-th element of SoftMax is $\beta_l = \frac{\exp(m_l)}{\sum_{l'=1}^{L}\exp(m_{l'})}$.

The text sequence representation can be simply obtained via averaging the word embeddings, weighted by label-based attention score:

$$\boldsymbol{z} = \sum_l \beta_l \boldsymbol{v}_l. \tag{6}$$

Relation to Predictive Text Embeddings Predictive Text Embeddings (PTE) (Tang et al., 2015) is the first method to leverage label embeddings to improve the learned word embeddings. We discuss three major differences between PTE and our LEAM: (*i*) The general settings are different. PTE casts the text representation through heterogeneous networks, while we consider text representation through an attention model. (*ii*) In PTE, the text representation $\boldsymbol{z}$ is the averaging of word embeddings. In LEAM, $\boldsymbol{z}$ is weighted averaging of word embeddings through the proposed label-attentive score in (6). (*iii*) PTE only considers the linear interaction between individual words and labels. LEAM greatly improves the performance by considering nonlinear interaction between phrase

[1]We call it "phrase" for convenience; it could be any longer word sequence such as a sentence and paragraph *etc.* when a larger window size r is considered.

and labels. Specifically, we note that the text embedding in PTE is similar with a very special case of LEAM, when our window size $r = 1$ and attention score $\boldsymbol{\beta}$ is uniform. As shown later in Figure 2(c) of the experimental results, LEAM can be significantly better than the PTE variant.

Training Objective The proposed joint embedding framework is applicable to various text classification tasks. We consider two setups in this paper. For a learned text sequence representation $\boldsymbol{z} = f_1 \circ f_0(\mathbf{X})$, we jointly optimize $f = f_0 \circ f_1 \circ f_2$ over $\mathcal{F}$, where f_2 is defined according to the specific tasks:

- *Single-label problem*: categorizes each text instance to precisely one of K classes, $\boldsymbol{y} \in \Delta^K$

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n=1}^{N} \text{CE}(\boldsymbol{y}_n, f_2(\boldsymbol{z}_n)), \quad (7)$$

  where $\text{CE}(\cdot, \cdot)$ is the cross entropy between two probability vectors, and $f_2(\boldsymbol{z}_n) = \text{SoftMax}\,(\boldsymbol{z}'_n)$, with $\boldsymbol{z}'_n = \mathbf{W}_2 \boldsymbol{z}_n + \boldsymbol{b}_2$ and $\mathbf{W}_2 \in \mathbb{R}^{K \times P}, \boldsymbol{b}_2 \in \mathbb{R}^K$ are trainable parameters.

- *Multi-label problem*: categorizes each text instance to a set of K target labels $\{\boldsymbol{y}_k \in \Delta^2 | k = 1, \cdots, K\}$; there is no constraint on how many of the classes the instance can be assigned to, and

$$\min_{f \in \mathcal{F}} \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \text{CE}(\boldsymbol{y}_{nk}, f_2(\boldsymbol{z}_{nk}), \quad (8)$$

  where $f_2(\boldsymbol{z}_{nk}) = \frac{1}{1+\exp(\boldsymbol{z}'_{nk})}$, and $\boldsymbol{z}'_{nk}$ is the kth column of $\boldsymbol{z}'_n$.

To summarize, the model parameters $\boldsymbol{\theta} = \{\mathbf{V}, \mathbf{C}, \mathbf{W}_1, \boldsymbol{b}_1, \mathbf{W}_2, \boldsymbol{b}_2\}$. They are trained end-to-end during learning. $\{\mathbf{W}_1, \boldsymbol{b}_1\}$ and $\{\mathbf{W}_2, \boldsymbol{b}_2\}$ are weights in f_1 and f_2, respectively, which are treated as standard neural networks. For the joint embeddings $\{\mathbf{V}, \mathbf{C}\}$ in f_0, the pre-trained word embeddings are used as initialization if available.

4.2 Learning & Testing with LEAM

Learning and Regularization The quality of the jointly learned embeddings are key to the model performance and interpretability. Ideally, we hope that each label embedding acts as the "anchor" points for each classes: closer to the word/sequence representations that are in the same classes, while farther from those in different classes. To best achieve this property, we consider to regularize the learned label embeddings $\boldsymbol{c}_k$ to be on its corresponding manifold. This is imposed by the fact $\boldsymbol{c}_k$ should be easily classified as the correct label $\boldsymbol{y}_k$:

$$\min_{f \in \mathcal{F}} \frac{1}{K} \sum_{n=1}^{K} \text{CE}(\boldsymbol{y}_k, f_2(\boldsymbol{c}_k)), \quad (9)$$

where f_2 is specficied according to the problem in either (7) or (8). This regularization is used as a penalty in the main training objective in (7) or (8), and the default weighting hyperparameter is set as 1. It will lead to meaningful interpretability of learned label embeddings as shown in the experiments.

Interestingly in text classification, the class itself is often described as a set of E words $\{\boldsymbol{e}_i, i = 1, \cdots, E\}$. These words are considered as the most representative description of each class, and highly distinguishing between different classes. For example, the `Yahoo! Answers Topic` dataset (Zhang et al., 2015) contains ten classes, most of which have two words to precisely describe its class-specific features, such as "Computers & Internet", "Business & Finance" as well as "Politics & Government" *etc.* We consider to use each label's corresponding pre-trained word embeddings as the initialization of the label embeddings. For the datasets without representative class descriptions, one may initialize the label embeddings as random samples drawn from a standard Gaussian distribution.

Testing Both the learned word and label embeddings are available in the testing stage. We clarify that the label embeddings $\mathbf{C}$ of all class candidates $\mathcal{Y}$ are considered as the input in the testing stage; one should distinguish this from the use of groundtruth label $\boldsymbol{y}$ in prediction. For a text sequence $\mathbf{X}$, one may feed it through the proposed pipeline for prediction: (*i*) f_1: harvesting the word embeddings $\mathbf{V}$, (*ii*) f_2: $\mathbf{V}$ interacts with $\mathbf{C}$ to obtain $\mathbf{G}$, pooled as $\boldsymbol{\beta}$, which further attends $\mathbf{V}$ to derive $\boldsymbol{z}$, and (*iii*) f_3: assigning labels based on the tasks. To speed up testing, one may store $\mathbf{G}$ offline, and avoid its online computational cost.

Model	Parameters	Complexity	Seq. Operation
CNN	$m \cdot h \cdot P$	$O(m \cdot h \cdot L \cdot P)$	$O(1)$
LSTM	$4 \cdot h \cdot (h + P)$	$O(L \cdot h^2 + h \cdot L \cdot P)$	$O(L)$
SWEM	0	$O(L \cdot P)$	$O(1)$
Bi-BloSAN	$7 \cdot P^2 + 5 \cdot P$	$O(P^2 \cdot L^2/R + P^2 \cdot L + P^2 \cdot R^2)$	$O(1)$
Our model	$K \cdot P$	$O(K \cdot L \cdot P)$	$O(1)$

Table 1: Comparisons of CNN, LSTM, SWEM and our model architecture. Columns correspond to the number of compositional parameters, computational complexity and sequential operations

4.3 Model Complexity

We compare CNN, LSTM, Simple Word Embeddings-based Models (SWEM) (Shen et al., 2018a) and our LEAM *wrt* the parameters and computational speed. For the CNN, we assume the same size m for all filters. Specifically, h represents the dimension of the hidden units in the LSTM or the number of filters in the CNN; R denotes the number of blocks in the Bi-BloSAN; P denotes the final sequence representation dimension. Similar to (Vaswani et al., 2017; Shen et al., 2018a), we examine the number of compositional parameters, computational complexity and sequential steps of the four methods. As shown in Table 1, both the CNN and LSTM have a large number of compositional parameters. Since $K \ll m, h$, the number of parameters in our models is much smaller than for the CNN and LSTM models. For the computational complexity, our model is almost same order as the most simple SWEM model, and is smaller than the CNN or LSTM by a factor of mh/K or h/K.

5 Experimental Results

Setup We use 300-dimensional GloVe word embeddings Pennington et al. (2014) as initialization for word embeddings and label embeddings in our model. Out-Of-Vocabulary (OOV) words are initialized from a uniform distribution with range $[-0.01, 0.01]$. The final classifier is implemented as an MLP layer followed by a sigmoid or softmax function depending on specific task. We train our model's parameters with the Adam Optimizer (Kingma and Ba, 2014), with an initial learning rate of 0.001, and a minibatch size of 100. Dropout regularization (Srivastava et al., 2014) is employed on the final MLP layer, with dropout rate 0.5. The model is implemented using Tensorflow and is trained on GPU Titan X.

The code to reproduce the experimental results is at `https://github.com/guoyinwang/LEAM`

Dataset	**# Classes**	**# Training**	**# Testing**
AGNews	4	120k	7.6k
Yelp Binary	2	560 k	38k
Yelp Full	5	650k	38k
DBPedia	14	560k	70k
Yahoo	10	1400k	60k

Table 2: Summary statistics of five datasets, including the number of classes, number of training samples and number of testing samples.

5.1 Classification on Benchmark Datasets

We test our model on the same five standard benchmark datasets as in (Zhang et al., 2015). The summary statistics of the data are shown in Table 2, with content specified below:

- `AGNews`: Topic classification over four categories of Internet news articles (Del Corso et al., 2005) composed of titles plus description classified into: World, Entertainment, Sports and Business.

- `Yelp Review Full`: The dataset is obtained from the Yelp Dataset Challenge in 2015, the task is sentiment classification of polarity star labels ranging from 1 to 5.

- `Yelp Review Polarity`: The same set of text reviews from Yelp Dataset Challenge in 2015, except that a coarser sentiment definition is considered: 1 and 2 are negative, and 4 and 5 as positive.

- `DBPedia`: Ontology classification over fourteen non-overlapping classes picked from DBpedia 2014 (Wikipedia).

- `Yahoo! Answers Topic`: Topic classification over ten largest main categories from Yahoo! Answers Comprehensive Questions and Answers version 1.0, including question title, question content and best answer.

We compare with a variety of methods, including (i) the bag-of-words in (Zhang et al., 2015); (ii) sophisticated deep CNN/RNN models: large/small word CNN, LSTM reported in (Zhang et al., 2015; Dai and Le, 2015) and deep CNN (29 layer) (Conneau et al., 2017); (iii) simple compositional methods: fastText (Joulin et al., 2016) and simple word embedding models (SWEM) (Shen et al., 2018a); (iv) deep attention models: hierarchical attention network (HAN) (Yang et al.,

Model	Yahoo	DBPedia	AGNews	Yelp P.	Yelp F.
Bag-of-words (Zhang et al., 2015)	68.90	96.60	88.80	92.20	58.00
Small word CNN (Zhang et al., 2015)	69.98	98.15	89.13	94.46	58.59
Large word CNN (Zhang et al., 2015)	70.94	98.28	91.45	95.11	59.48
LSTM (Zhang et al., 2015)	70.84	98.55	86.06	94.74	58.17
SA-LSTM (word-level) (Dai and Le, 2015)	-	98.60	-	-	-
Deep CNN (29 layer) (Conneau et al., 2017)	73.43	98.71	91.27	**95.72**	**64.26**
SWEM (Shen et al., 2018a)	73.53	98.42	92.24	93.76	61.11
fastText (Joulin et al., 2016)	72.30	98.60	92.50	95.70	63.90
HAN (Yang et al., 2016)	75.80	-	-	-	-
Bi-BloSAN$^\diamond$ (Shen et al., 2018c)	76.28	98.77	**93.32**	94.56	62.13
LEAM	**77.42**	**99.02**	92.45	95.31	64.09
LEAM (linear)	75.22	98.32	91.75	93.43	61.03

Table 3: Test Accuracy on document classification tasks, in percentage. $^\diamond$ We ran Bi-BloSAN using the authors' implementation; all other results are directly cited from the respective papers.

2016); (v) simple attention models: bi-directional block self-attention network (Bi-BloSAN) (Shen et al., 2018c). The results are shown in Table 3.

Testing accuracy Simple compositional methods indeed achieve comparable performance as the sophisticated deep CNN/RNN models. On the other hand, deep hierarchical attention model can improve the pure CNN/RNN models. The recently proposed self-attention network generally yield higher accuracy than previous methods. All approaches are better than traditional bag-of-words method. Our proposed LEAM outperforms the state-of-the-art methods on two largest datasets, *i.e.,* Yahoo and DBPedia. On other datasets, LEAM ranks the 2nd or 3rd best, which are similar to top 1 method in term of the accuracy. This is probably due to two reasons: (i) the number of classes on these datasets is smaller, and (ii) there is no explicit corresponding word embedding available for the label embedding initialization during learning. The potential of label embedding may not be fully exploited. As the ablation study, we replace the nonlinear compatibility (3) to the linear one in (2) . The degraded performance demonstrates the necessity of spatial dependency and nonlinearity in constructing the attentions.

Nevertheless, we argue LEAM is favorable for text classification, by comparing the model size and time cost Table 4, as well as convergence speed in Figure 2(a). The time cost is reported as the wall-clock time for 1000 iterations. LEAM maintains the simplicity and low cost of SWEM, compared with other models. LEAM uses much less model parameters, and converges significantly faster than Bi-BloSAN. We also compare the performance when only a partial dataset is labeled, the results are shown in Figure 2(b). LEAM consistently outperforms other methods with different proportion of labeled data.

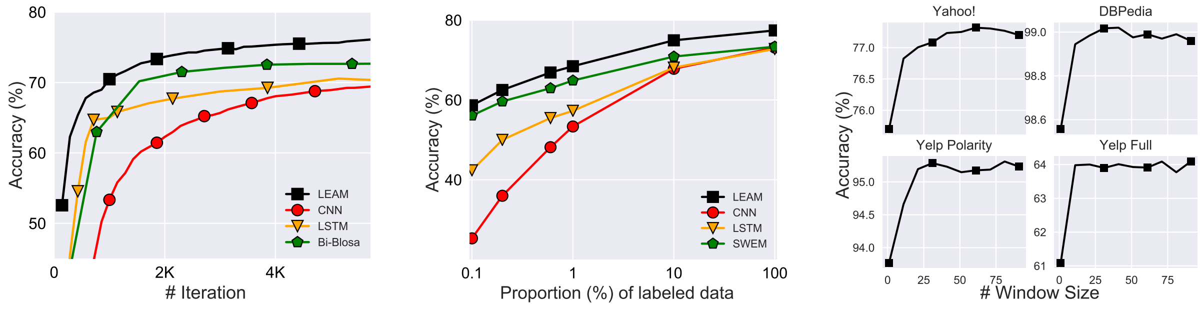
Model	# Parameters	Time cost (s)
CNN	541k	171
LSTM	1.8M	598
SWEM	61K	63
Bi-BloSAN	3.6M	292
LEAM	65K	65

Table 4: Comparison of model size and speed.

Hyper-parameter Our method has an additional hyperparameter, the window size r to define the length of "phase" to construct the attention. Larger r captures long term dependency, while smaller r enforces the local dependency. We study its impact in Figure 2(c). The topic classification tasks generally requires a larger r, while sentiment classification tasks allows relatively smaller r. One may safely choose r around 50 if not finetuning. We report the optimal results in Table 3.

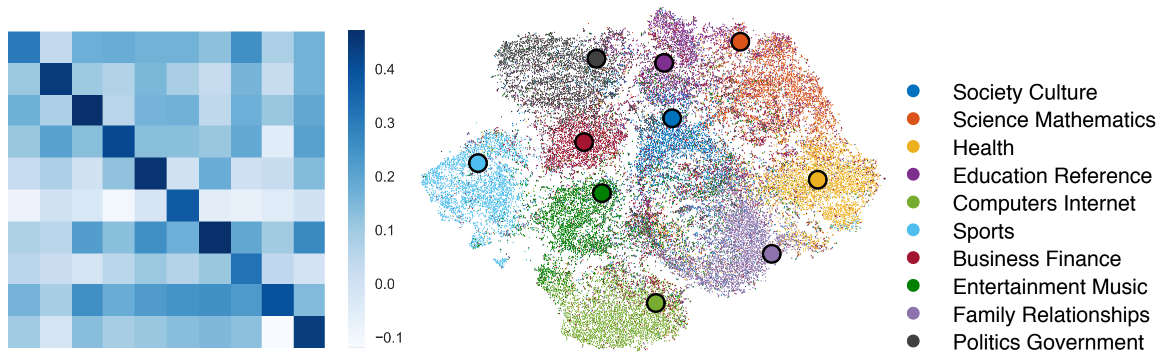
5.2 Representational Ability

Label embeddings are highly meaningful To provide insight into the meaningfulness of the learned representations, in Figure 3 we visualize the correlation between label embeddings and document embeddings based on the Yahoo dateset. First, we compute the averaged document embeddings per class: $\bar{z}_k = \frac{1}{|\mathcal{S}_k|}\sum_{i\in\mathcal{S}_k} z_i$, where $\mathcal{S}_k$ is the set of sample indices belonging to class k. Intuitively, $\bar{z}_k$ represents the center of embedded



(a) Convergence speed (b) Partially labeled data (c) Effects of window size

Figure 2: Comprehensive study of LEAM, including convergence speed, performance vs proportion of labeled data, and impact of hyper-parameter



(a) Cosine similarity matrix (b) t-SNE plot of joint embeddings

Figure 3: Correlation between the learned text sequence representation $\boldsymbol{z}$ and label embedding $\mathbf{V}$. (a) Cosine similarity matrix between averaged $\bar{\boldsymbol{z}}$ per class and label embedding $\mathbf{V}$, and (b) t-SNE plot of joint embedding of text $\boldsymbol{z}$ and labels $\mathbf{V}$.

text manifold for class k. Ideally, the perfect label embedding $\boldsymbol{c}_k$ should be the representative anchor point for class k. We compute the cosine similarity between $\bar{\boldsymbol{z}}_k$ and $\boldsymbol{c}_k$ across all the classes, shown in Figure 3(a). The rows are averaged per-class document embeddings, while columns are label embeddings. Therefore, the on-diagonal elements measure how representative the learned label embeddings are to describe its own classes, while off-diagonal elements reflect how distinctive the label embeddings are to be separated from other classes. The high on-diagonal elements and low off-diagonal elements in Figure 3(a) indicate the superb ability of the label representations learned from LEAM.

Further, since both the document and label embeddings live in the same high-dimensional space, we use t-SNE (Maaten and Hinton, 2008) to visualize them on a 2D map in Figure 3(b). Each color represents a different class, the point clouds are document embeddings, and the label embeddings are the large dots with black circles. As can be seen, each label embedding falls into the internal region of the respective manifold, which again demonstrate the strong representative power of label embeddings.

Interpretability of attention Our attention score $\boldsymbol{\beta}$ can be used to highlight the most informative words *wrt* the downstream prediction task. We visualize two examples in Figure 4(a) for the Yahoo dataset. The darker yellow means more important words. The 1st text sequence is on the topic of "Sports", and the 2nd text sequence is "Entertainment". The attention score can correctly detect the key words with proper scores.

5.3 Applications to Clinical Text

To demonstrate the practical value of label embeddings, we apply LEAM for a real health care scenario: medical code prediction on the Electronic Health Records dataset. A given patient may have multiple diagnoses, and thus multi-label learning is required.

Specifically, we consider an open-access dataset, `MIMIC-III` (Johnson et al., 2016), which

	AUC		F1		
Model	Macro	Micro	Macro	Micro	P@5
Logistic Regression	0.829	0.864	0.477	0.533	0.546
Bi-GRU	0.828	0.868	0.484	0.549	0.591
CNN (Kim, 2014)	0.876	0.907	0.576	0.625	0.620
C-MemNN (Prakash et al., 2017)	0.833	-	-	-	0.42
Attentive LSTM (Shi et al., 2017)	-	0.900	-	0.532	-
CAML (Mullenbach et al., 2018)	0.875	0.909	0.532	0.614	0.609
LEAM	**0.881**	**0.912**	0.540	0.619	0.612

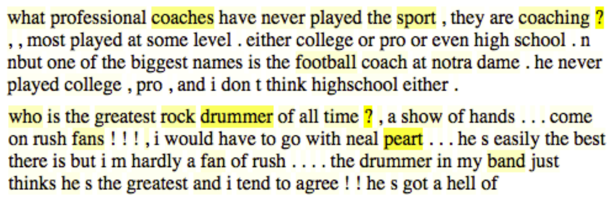
Table 5: Quantitative results for doctor-notes multi-label classification task.

contains text and structured records from a hospital intensive care unit. Each record includes a variety of narrative notes describing a patients stay, including diagnoses and procedures. They are accompanied by a set of metadata codes from the International Classification of Diseases (ICD), which present a standardized way of indicating diagnoses/procedures. To compare with previous work, we follow (Shi et al., 2017; Mullenbach et al., 2018), and preprocess a dataset consisting of the most common 50 labels. It results in 8,067 documents for training, 1,574 for validation, and 1,730 for testing.

Results We compare against the three baselines: a logistic regression model with bag-of-words, a bidirectional gated recurrent unit (Bi-GRU) and a single-layer 1D convolutional network (Kim, 2014). We also compare with three recent methods for multi-label classification of clinical text, including Condensed Memory Networks (C-MemNN) (Prakash et al., 2017), Attentive LSTM (Shi et al., 2017) and Convolutional Attention (CAML) (Mullenbach et al., 2018).

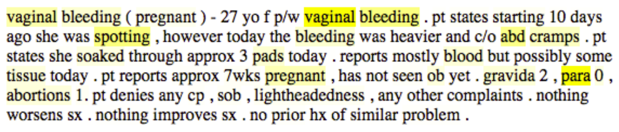
To quantify the prediction performance, we follow (Mullenbach et al., 2018) to consider the micro-averaged and macro-averaged F1 and area under the ROC curve (AUC), as well as the precision at n ($P@n$). Micro-averaged values are calculated by treating each (text, code) pair as a separate prediction. Macro-averaged values are calculated by averaging metrics computed per-label. P@n is the fraction of the n highestscored labels that are present in the ground truth.

The results are shown in Table 5. LEAM provides the best AUC score, and better F1 and P@5 values than all methods except CNN. CNN consistently outperforms the basic Bi-GRU architecture, and the logistic regression baseline performs worse than all deep learning architectures.


what professional coaches have never played the sport , they are coaching ? , , most played at some level . either college or pro or even high school . n nbut one of the biggest names is the football coach at notra dame . he never played college , pro , and i don t think highschool either .

who is the greatest rock drummer of all time ? , a show of hands . . . come on rush fans ! ! ! , i would have to go with neal peart . . . he s easily the best there is but i m hardly a fan of rush . . . . the drummer in my band just thinks he s the greatest and i tend to agree ! ! he s got a hell of

(a) Yahoo dataset


vaginal bleeding (pregnant) - 27 yo f p/w vaginal bleeding . pt states starting 10 days ago she was spotting , however today the bleeding was heavier and c/o abd cramps . pt states she soaked through approx 3 pads today . reports mostly blood but possibly some tissue today . pt reports approx 7wks pregnant , has not seen ob yet . gravida 2 , para 0 , abortions 1. pt denies any cp , sob , lightheadedness , any other complaints . nothing worsens sx . nothing improves sx . no prior hx of similar problem .

(b) Clinical text

Figure 4: Visualization of learned attention $\boldsymbol{\beta}$.

We emphasize that the learned attention can be very useful to reduce a doctor's reading burden. As shown in Figure 4(b), the health related words are highlighted.

6 Conclusions

In this work, we first investigate label embeddings for text representations, and propose the label-embedding attentive models. It embeds the words and labels in the same joint space, and measures the compatibility of word-label pairs to attend the document representations. The learning framework is tested on several large standard datasets and a real clinical text application. Compared with the previous methods, our LEAM algorithm requires much lower computational cost, and achieves better if not comparable performance relative to the state-of-the-art. The learned attention is highly interpretable: highlighting the most informative words in the text sequence for the downstream classification task.

Acknowledgments This research was supported by DARPA, DOE, NIH, ONR and NSF.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2016. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Gianna M Del Corso, Antonio Gulli, and Francesco Romani. 2005. Ranking a stream of news. In *Proceedings of the 14th international conference on World Wide Web*. ACM.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *EACL*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *NIPS 2014 deep learning workshop*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695*.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In *NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Aaditya Prakash, Siyuan Zhao, Sadid A Hasan, Vivek V Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2017. Condensed memory networks for clinical diagnostic inferencing. In *AAAI*.

Jose A Rodriguez-Serrano, Florent Perronnin, and France Meylan. 2013. Label embedding for text recognition. In *Proceedings of the British Machine Vision Conference*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018a. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017. Deconvolutional latent-variable model for text sequence matching. *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018b. Disan: Directional self-attention network for rnn/cnn-free language understanding. *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018c. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *ICLR*.

Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P Xing. 2017. Towards automated icd coding using deep learning. *arXiv preprint arXiv:1711.04075*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2018. Topic compositional neural language model. *AISTATS*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. *ICLR*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *TACL*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *ACL*.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2017a. Multi-task label embedding for text classification. *arXiv preprint arXiv:1710.07210*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017b. Deconvolutional paragraph representation learning. In *NIPS*.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based lstm network for cross-lingual sentiment classification. In *EMNLP*.

Neural Sparse Topical Coding

Min Peng[1], **Qianqian Xie**[1], **Yanchun Zhang**[2], **Hua Wang**[2], **Xiuzheng Zhang**[3], **Jimin Huang**[1] and **Gang Tian**[1]

[1]School of Computer Science, WuHan University, WuHan, China
[2]Centre for Applied Informatics, Victoria University, Melbourne, Australia
[3]School of Science, RMIT University, Melbourne, Australia
{*pengm, xieq, huangjimin, tiang2008*}*@whu.edu.cn*
{*yanchun.zhang, hua.wang*}*@vu.edu.au*, *xiuzhen.zhang@rmit.edu.au*

Abstract

Topic models with sparsity enhancement have been proven to be effective at learning discriminative and coherent latent topics of short texts, which is critical to many scientific and engineering applications. However, the extensions of these models require carefully tailored graphical models and re-deduced inference algorithms, limiting their variations and applications. We propose a novel sparsity-enhanced topic model, Neural Sparse Topical Coding (NSTC) base on a sparsity-enhanced topic model called Sparse Topical Coding (STC). It focuses on replacing the complex inference process with the back propagation, which makes the model easy to explore extensions. Moreover, the external semantic information of words in word embeddings is incorporated to improve the representation of short texts. To illustrate the flexibility offered by the neural network based framework, we present three extensions base on NSTC without re-deduced inference algorithms. Experiments on Web Snippet and 20Newsgroups datasets demonstrate that our models outperform existing methods.

1 Introduction

Topic models with sparsity enhancement have proven to be effective tools for exploratory analysis of the overload of short text content. The latent representations learned by these models are central to many applications. However, these models have trouble to rapidly explore variations for the approximate inference methods of them. Even subtle variations on models can increase the complexity of the inference methods, which is especially apparent for non-conjugate models.

With the development of deep learning, many works combine topic models with neural language model to overcome the computation complexity of topic models (Larochelle and Lauly, 2012a; Cao et al., 2015; Tian et al., 2016). Most of these methods adopt multiple neural network layers to model the generation process of each document. Nevertheless, these methods yield the same poor performance in short texts as traditional topic models. There are also many works introducing new techniques such as word embeddings to traditional topic models. Word embeddings has proven to be effective at capturing syntactic and semantic information of words. Many works (Das et al., 2015; Hu and Tsujii, 2016; Li et al., 2016) have shown that the additional semantics in word embeddings can enhance the performance of traditional topic models. Yet these models have the same trouble in making extensions as traditional topic models.

Base on the above observations, we propose Neural Sparse Topical Coding (NSTC) by jointly utilizing word embeddings and neural network with a sparsity-enhanced topic model, Sparse Topical Coding (STC). We adopt neural network to model the generation process of STC to simplify the complex inference and improve flexibility, and incorporate external semantics provided by word embeddings to improve the overall accuracy. To illustrate the dramatic flexibility offered by the end-to-end neural network, we present three extensions base on NSTC. Our proposed models all benefit from both sides: 1) when compared with the neural based topic models, which stuck in the sparseness of word co-occurrence information, they show how the sparsity mechanism and word embeddings enrich the features and improve the performance; 2) while with topic models with sparsity enhancement, our models present how the black-box inference method of neural network accelerates the training process and increases the flexibility. To evaluate the effectiveness of our models by conducting experiments on 20 Newsgroups and Web Snippet datasets.

2 Related Work

Topic models with sparsity enhancement: The performance of traditional topic models are compromised by the sparse word co-occurrence information when applied in short texts. To overcome the bottleneck, there have been many efforts to address the problem of sparsity in short texts. Based on traditional LDA, (Williamson et al., 2010) introduced a *Spike and Slab* prior to model the sparsity in finite and infinite latent topic structures of text. To consider the dual-sparsity of topics per document and terms per topic, (Lin et al., 2014) proposed a dual-sparse topic model that addresses the sparsity in both the topic mixtures and the word usage. There are also some non-probabilistic sparse topic models, which can directly control the sparsity by imposing regularizers. For example, the non-negative matrix factorization (NMF) (Heiler and Schnörr, 2006) formalized topic modeling as a problem of minimizing loss function regularized by lasso. Similarly, (Zhu and Xing, 2011) presented sparse topical coding (STC) by utilizing the Laplacian prior to directly control the sparsity of inferred representations. Additionally, (Peng et al., 2016) presented sparse topical coding with sparse groups (STCSG) to find sparse word and document representations of texts. However, over complicated inference procedure of these sparse topic models make them difficult to rapidly explore variations.

Topic models with word embeddings: There are many works tried to incorporate word embeddings with topic models to improve the performance. (Das et al., 2015) proposed a new technique for topic modeling by treating the document as a collection of word embeddings and topics itself as multivariate Gaussian distributions in the embedding space. However, the assumption that topics are unimodal in the embedding space is not appropriate, since topically related words can occur distantly from each other in the embedding space. Therefore, (Hu and Tsujii, 2016) proposed latent concept topic model (LCTM), which modeled a topic as a distribution of concepts, where each concept defined another distribution of word vectors. (Nguyen et al., 2015) proposed Latent Feature Topic Modeling (LFTM), which extended LDA to incorporate word embeddings as latent features. (Li et al., 2016) focused on combing the local information of word embeddings and the global information of LDA, thus proposed a model TopicVec yielded by the variational inference method. However, these models also have trouble to rapidly explore variations.

Neural Topic Models: There are also some efforts trying to combine topic models with neural networks to represent words and documents simultaneously. (Larochelle and Lauly, 2012a) proposed a neural network topic model that is similarly inspired by the Replicated Softmax. (Cao et al., 2015) proposed a novel neural topic model (NTM) where the representation of words and documents are efficiently and naturally combined into a uniform framework. (Das et al., 2015) proposed a new technique for topic modeling by treating the document as a collection of word embeddings and topics itself as multivariate Gaussian distributions in the embedding space. To address the limitations of the bag-of-words assumption, (Tian et al., 2016) proposed Sentence Level Recurrent Topic Model (SLRTM) by using a Recurrent Neural Networks (RNN) based framework to model long range dependencies between words. Nevertheless, most of aforementioned works yield poor performance in short texts.

3 Neural Sparse Topical Coding

Firstly, we define that $D = \{1, ..., M\}$ is a document set with size M, $T = \{1, ..., K\}$ is a topic collection with K topics, $V = \{1, .., N\}$ is a vocabulary with N words, and $w_d = \{w_{d,1}, ..., w_{d,|I|}\}$ is a vector of terms representing a document d, where I is the index of words in document d, and $w_{d,n} (n \in I)$ is the frequency of word n in document d. Moreover, we denote $\beta \in \mathbb{R}^{N \times K}$ as a global topic dictionary for the whole document set with K bases, $\theta_d \in \mathbb{R}^K$ is the document code of each document d and $s_{d,n} \in \mathbb{R}^K$ is the word code of each word n in each document d. To yield interpretable patterns, (θ, s, β) are constrained to be non-negative.

3.1 Sparse Topical Coding

STC is a hierarchical non-negative matrix factorization for learning hierarchical latent representations of input samples. In STC, each document and each word is represented as a low-dimensional code in topic space, which can be employed in many tasks. Based on the global topic dictionary β of all documents with K topic bases sampled from a uniform distribution, the generative process of each document d is described as follows:

1. Sample the document code θ_d from a prior $p(\theta_d) \sim Laplace(\lambda^{-1})$.
2. For each observed word n in document d:
   (a) Sample the word code $s_{d,n}$ from a conditional distribution $p(s_{d,n}|\theta_d) \sim supergaussian(\theta_d, \gamma^{-1}, \rho^{-1})$.
   (b) Sample the observed word count $w_{d,n}$ from a distribution $p(w_{d,n}|s_{d,n} * \beta_n) \sim Poisson(s_{d,n} * \beta_n)$

To achieve sparse word codes, STC defines $p(s_{d,n}|\theta_d)$ as a product of two component distributions $p(s_{d,n}|\theta_d) \sim p(s_{d,n}|\theta_d, \gamma)p(s_{d,n}|\rho)$, where $p(s_{d,n}|\theta_d, \gamma)$ is an isotropic Gaussian distribution, and $p(s_{d,n}|\rho)$ is a Laplace distribution. The composite distribution is super-Gaussian: $p(s_{d,n}|\theta_d) \propto exp(\gamma||s_{d,n}\theta_d||_2^2 \rho||s_{d,n}||_1)$. With the Laplace term, the composite distribution tends to yield sparse word codes. For the same purpose, the prior distribution $p(\theta_d)$ of document codes is a Laplace prior. Although STC has closed form coordinate descent equations for parameters (θ, s, β), it is inflexible for its complex inference process.

3.2 Neural Network View of Sparse Topical Coding

We devote to rebuild STC with a neural network to simplify it's inference process via BackPropogation. After generating the topic dictionary from neural network, our model follows the generative story below for each document d:

1. For each word n in document d:
   (a) Sample a latent variable word code $s_{d,n} \sim f_g(d, n)$.
   (b) Sample the observed word count $w_{d,n}$ from $p(w_{d,n}|s_{d,n}, \beta_n) \sim Poisson(s_{d,n} * \beta_n)$

In our model, we have several assumptions:

1) To simplify our model and acclerate the inference process, we collapse the document code from our model. As illuatrated in (Bai et al., 2013) and STC paper (Zhu and Xing, 2011), we can naturally generate each document code via a aggregation of all sampled word codes among all topics, after inferring the global topic dictionary and the word codes of words belong to each document:

$$\theta_d = \sum_{d=1}^{D}\sum_{n=1}^{N_d} s_{d,nk}\,\beta_{kn} / \sum_{d=1}^{D}\sum_{n=1}^{N_d}\sum_{k=1}^{K} s_{d,nk}\,\beta_{kn};$$

2) We replace the composite super-Gaussian prior of the word codes and the uniform distribution of the topic dictionary with the neural network. In the topic dictionary neural network, we introduce the word semantic information via word embeddings to enrich the feature space for short texts;

3) Similar to STC, the observed word count is sampled from Poisson distribution, which is more appropriate for the discrete count data than other exponential family distributions.

3.3 Neural Sparse Topical Coding

In this section, we introduce the detailed layer structures of NSTC in Figure 1. We explicitly explain each layer of NSTC below:

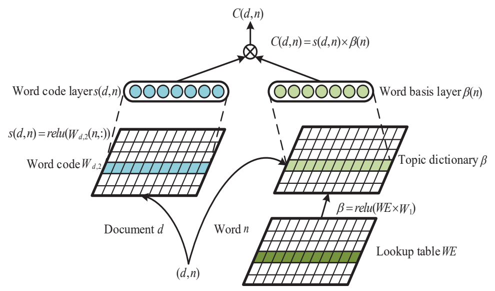


Figure 1: Schematic overview of *NSTC*.

Input layer (n, d): A word n of document $d \in D$, where D is a document set.

Word embedding layer $(WE \in \mathbb{R}^{N\times 300})$: Supposing the word number of the vocabulary is N, this layer devotes to transform each word to a distributed embedding representation. Here, we adopt the pre-trained embeddings by GloVe based on a large Wikipedia dataset[1].

Word code layers $(s_d \in \mathbb{R}^{N\times K})$: These layers generate the K-dimensional word code of input word n in document d.

$$s(d, n) = f_s(d, n) \tag{1}$$

where f_s is a multilayer perceptron. In order to achieve interpretable word codes as in STC,

[1]http://nlp.stanford.edu/projects/glove/

we constrain s to be non-negative, therefore we apply the relu activation function on the output of the neural network. Although imposing non-negativity constraints could potentially result in sparser and more interpretable patterns, we exert l_1 norm regularization on s to further keep the sparse assumption.

Topic dictionary layers ($\beta \in \mathbb{R}^{N \times K}$): These layers aim at converting WE to a topic dictionary similar to the one in STC.

$$\beta(n) = f_\beta(WE) \tag{2}$$

where f_β is a multilayer perceptron. We make a simplex projection among the output of topic dictionary neural network.We normalize each column of the dictionary via the simplex projection as follow:

$$\beta_{.k} = project(\beta_{.k}), \forall k \tag{3}$$

The simplex projection is the same as the sparsemax activation function in (Martins and Astudillo, 2016), providing the theoretical base of its employment in a neural network trained with backpropagation. After the simplex projection, each column of the topic dictionary is promised to be sparse, non-negative and united.

Score layer ($C_{d,n} \in \mathbb{R}^{1 \times 1}$): NSTC outputs the matching score of a word n and a document d with the dot product of $s(d,n)$ and $\beta(n)$ in this layer. The output score is utilized to approximate the observed word count $w_{d,n}$.

$$C(d,n) = s(d,n) * \beta(n) \tag{4}$$

Given the count $w_{d,n}$ of word n in document d, we can directly use it to supervise the training process. According to the architecture of our model, for each word n and each document d, the cost function is:

$$L = l(w_{d,n}, C(d,n)) + \lambda ||s_{d,n}||_1 \tag{5}$$

where l is the log-Poisson loss, λ is the regularization factors.

3.4 Extensions of NSTC

To future illustrate the benefits of a black box inference method, which allows rapidly explore new models, we present three variants of NSTC.

Deep l_1 Approximation. STC is based on the idea of sparse coding, in which the sparse code s of the input w can be obtained by solving the loss function for a given dictionary β. In (Gregor and LeCun, 2010), the parameterized encoder, named learned ISTA (LISTA) was proposed to efficiently approximate the l_1 based sparse code. Based on the consideration, we present a enhanced NSTC via employing the deep l_1 regularized encoder similar to LISTA, named NSTCE. We devise a feed-forward neural network as illustrated in Figure 2, to efficiently approximate the l_1 based sparse code s of the input w.

$$F(w_d; W_d, b_d) = relu(w_d * W_d + b_d) \tag{6}$$

The goal is to make the prediction of neural network predictor F after the fixed depth as close as possible to the optimal set of coefficients s^* in Eq.4. To jointly optimizing all parameters with the dictionary β together, we add another term to the loss function in Eq.4, and enforce the representation s to be as close as possible to the feed forward prediction (Kavukcuoglu et al., 2010):

$$\begin{aligned} L =& l(w_{d,n}, C(d,n)) + \lambda ||s_{d,n}||_1 \\ &+ \alpha \sum_n ||s_d - F(w_d; W_d, b_d)||_2^2 \end{aligned} \tag{7}$$

Minimizing the loss with respect to s produces a sparse representation that simultaneously reconstructs the word count and is not too different from the predicted representation.

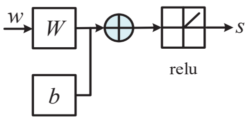


Figure 2: Deep l_1 encoder.

Group Sparse Regularization. Based on STC, (Bai et al., 2013) presented GSTC to discover document-level sparse or admixture proportion for short texts, in which the group sparse is employed to achieve sparse code at document level by taking into account the structure of bag of words. Here, we just need to add the group sparse regularization on the weight matrix, to make a neural network extension of GSTC, called NGSTC. We consider each column of s_d as a group.

$$L = l(w_{d,n}, C(d,n)) + \lambda \sum_{k=1}^{K} ||s_{d,.k}||_2 \tag{8}$$

Sparse Group Lasso. Similar to GSTC, STCSG (Peng et al., 2016) was proposed to learn sparse word and document codes, which relaxes the normalization constraint of the inferred representations with sparse group lasso. Base on STCSG, we propose a novel neural topic model called NSTCSG. We impose the sparse group lasso on the word code, and have the following cost function:

$$L = l(w_{d,n}, C(d,n)) + \lambda_1 ||s_{d,n}||_1 + \lambda_2 \sum_{k=1}^{K} ||s_{d,.k}||_2 \tag{9}$$

These three models have the same neural network structures as NSTC, and can be trained end to end with out re-deduced mathematical inference. Moreover, group and sparse group sparsity can help reduce the intrinsic complexity of the model by eliminating neurons as shown in Figure 3, and thus can help obtain practical speed ups in deep neural networks.

3.5 Optimization

For the first two models with the lasso regularizer, we can directly ulitize the end to end stochastic gradient descent (SGD) to perform optimizing. The last two optimizing objectives of NGSTC and NSTCSG are formed as a combination of both smooth and non-smooth terms, they can be solved via proximal stochastic gradient descent. The proximal gradient algorithm first obtains the intermediate solution via SGD on the loss only, and then optimize for the regularization term via performing Euclidean projection of it to the solution space, as in the following formulation:

$$\min_{s_{d,n}^{t+1}} R(s_{d,n}^{t+1}) + \frac{1}{2}||s_{d,n}^{t+1} - s_{d,n}^{t+\frac{1}{2}}||_2^2 \tag{10}$$

where R is the regularization, $s_{d,n}^{t+\frac{1}{2}}$ the intermediate solution obtained by SGD, $s_{d,n}^{t+1}$ is the variable to obtain after the current iteration. For the group lasso, the above problem has the closed-form solution. The proximal operator for the group lasso regularizer in Eq.8 is given as follow:

$$prox_{GL}(s_{d,nk}) = (1 - \frac{\lambda}{||s_{d,.k}||_2})_+ s_{d,nk} \tag{11}$$

The proximal operator for the sparse group lasso regularizer in Eq.9 is given as follow:

$$\begin{aligned} prox_{SGL}(s_{d,nk}) =& (1 - \frac{\lambda_2}{||sign(s_{d,.k}, \lambda_1)||_2})_+ \\ & sign(s_{d,nk}, \lambda_1) \end{aligned} \tag{12}$$

The detailed algorithm framework of NGSTC and NSTCSG is shown in Algorithm 1.

Algorithm 1 Training Algorithm for our models

```
Require: a document $d \in D$
1: $t = t + 1$
2: repeat
3:     Compute the partial derivatives of weight matrices,$s$, and $\beta$ with a non-regularized objective;
4:     Update weight matrices, $s$, and $\beta$ using SGD.
5:     Update $s$ using proximal operator
6:     Update $\beta$ using simplex projection.
7: until convergence
```

4 Experiments

4.1 Data and Setting

We perform our experiments on two benchmark datasets:

- **20Newsgroups**: is comprised of 18775 newsgroup articles with 20 categories, and contains 60698 unique words[2].
- **Web Snippet**: includes 12340 Web search snippets with 8 categories, we remove the words with fewer than 3 characters and with document frequency less than 3 in the dataset[3].

We compare the performance of the NSTC with the following baselines:

- **LDA** (Blei et al., 2001). A classical probabilistic topic model. We use the LDA package[4] for its implementation. We use the settings with iteration number $n = 2000$, the Dirichlet parameter for distribution over topics $\alpha = 0.1$ and the Dirichlet parameter for distribution over words $\eta = 0.01$.
- **STC** (Zhu and Xing, 2011). It is a sparsity-enhanced non-probabilistic topic model. We use the code released by the authors[5]. We set the regularization constants as $\lambda = 0.3, \rho = 0.0001$ and the maximum number of iterations of hierarchical sparse coding, dictionary learning as 100.

[2]http://www.qwone.com/ jason/20Newsgroups/
[3]http://jwebpro.sourceforge.net/data-web-snippets.tar.gz
[4]https://pypi.python.org/pypi/lda
[5]http://bigml.cs.tsinghua.edu.cn/ jun/stc.shtml/

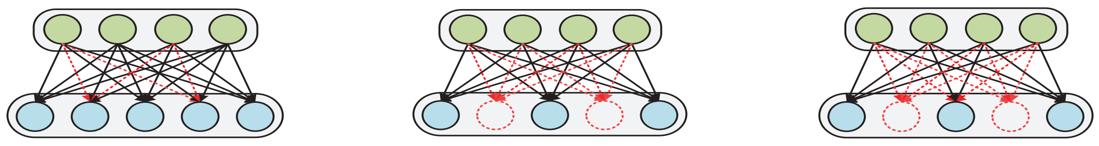

(a) (b) (c)

Figure 3: (a) Lasso: the Lasso penalty removes elements without optimizing neuron-level considerations (highlighted in red). (b) Group lasso: when grouping weights from the the same input neuron into each group, the group sparsity has an effect of completely removing some neurons (highlighted in red). (c) Sparse group lasso: it combines the advantages of the former two formulations, which can remove some neurons and elements (highlighted in red).

- **DocNADE** (Larochelle and Lauly, 2012b). An unsupervised neural network topic model of documents and has shown that it is a competitive model both as a generative model and as a document representation learning algorithm[6]. In DocNADE, the hidden size is 50, the learning rate is 0.0004 , the bath size is 64 and the max training number is 50000.

- **GaussianLDA** (Das et al., 2015). A new technique for topic modeling by treating the document as a collection of word embeddings and topics itself as multivariate Gaussian distributions in the embedding space[7]. We use default values for the parameters.

- **TopicVec** (Li et al., 2016). A model incorporates generative word embedding model with LDA [8]. We also use default values for the parameters.

Our three models are implemented in Python using TensorFlow[9]. For both datasets, we use the pre-trained 300-dimensional word embeddings from Wikipedia by GloVe, and it is fixed during training. For each out-of-vocab word, we sample a random vector from a normal distribution. In practice, we use a regular learning rate 0.00001 for both dataset. We set the regularization factor $\lambda = 1, \alpha = 1, \lambda_1 = 0.6, \lambda_2 = 0.4$. In initialization, all weight matrices are randomly initialized with the uniformed distribution in the interval $[0, 0.001]$ for web snippet, and $[0, 0.0001]$ for 20Newsgroups.

[6]https://github.com/huashiyiqike/TMBP/tree/master/DocNADE

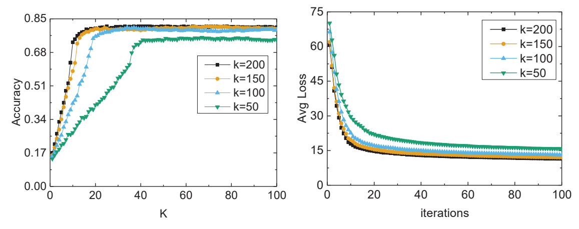
[7]https://github.com/rajarshd/Gaussian_LDA

[8]https://github.com/askerlee/topicvec

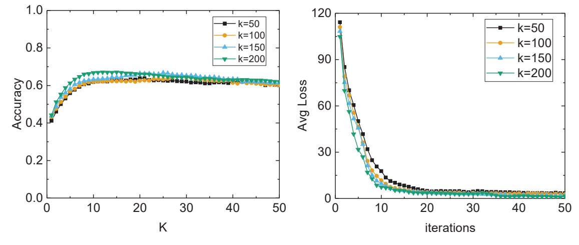
[9]https://www.tensorflow.org/

4.2 Classification Accuracy

We perform text classification tasks on Web Snippet dataset and 20Newsgroups. For the Web Snippet, we follow its original partition that consists of 10060 training documents and 2280 test documents. On 20Newsgroups, we we keep 60% documents for training and 40% for testing as in (Zhu and Xing, 2011). We adopt the SVM as the classifier with the document representations learned by topic models. Figure 4 reports the convergence curves of loss and accuracy over iterations. The results show that the loss and accuracy of our method can achieve convergence after almost 100 epochs on web snippets and 50 epochs on 20Newsgroups with appropriate learning rate. Table 1 reports the classification accuracy on both datasets under different methods with different settings on the number of topics $K = \{50, 100, 150, 200, 250\}$. We can found that 1) The NSTCSG yields the highest accuracy, followed by NGSTC, NSTCE and NSTC which all outperform the DocNADE, GLDA, STC and LDA. 2) The embedding based models (NSTCSG, NGSTC, NSTCE, NSTC, DocNADE and GLDA) generate better document representations than STC and LDA separately, demonstrating the representative power of neural networks based on word embeddings. 3) Sparse models (NSTCSG, NGSTC, NSTCE, NSTC and STC) are superior to non-sparse models NTM and LDA separately. It indicates that sparse topic models are more suitable to short documents. 4) The NSTCSG perform better than NGSTC, followed by NSTC, which illustrates both sparse group lasso and group lasso penalty are contribute to learning the document representations with clear semantic explanations. 5) The accuracies of DocNADE decrease with the increasing of the topic K. This is may because DocNADE may generate the document topic distribution with many indistinct non-zeros due to the data sparsity caused by the increasing number of topics. Notice that LDA has the same performance on the web snippet dataset.

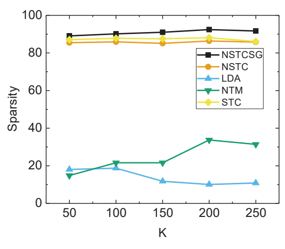


(a) web snippet

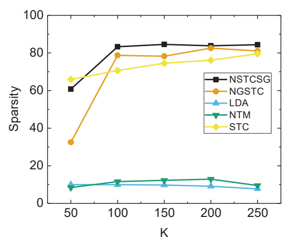


(b) 20Newsgroup

Figure 4: The loss and accuracy curves with the iterations on two datasets,on different number of topic K settings.



(a) word codes



(b) document codes

Figure 5: The average sparsity ratio of word and document codes.

4.3 Sparse Ratio

We further compare the sparsity of the learned latent representations of words and documents from different models on Web Snippet.

Word code: For each word n, we compute the average word code and average sparsity ratio of them as in (Zhu and Xing, 2011). Figure 5a presents the average word sparse ratio of word codes discovered by different models for Web Snippet. Note that the NGSTC can not yield sparse word codes but sparse document codes. We can see that 1) The NSTCSG learns the sparsest word codes, followed by NSTC and STC, which perform much better than NTM and LDA. 2) The word codes discovered by LDA and NTM are very dense for lacking the mechanism to learn the focused topics. The sparsity in both models is mainly caused by the data scarcity. 3)The representations learned by sparse models (NSTCSG, NSTC and STC) are much sparser, which indicates each word concentrates on only a small number of topics in these models, and therefore the word codes are more clear and semantically concentrated. 4) Meanwhile, the sparse ratio of STC and NSTC are lower than NSTCSG. It proves the sparse group lasso penalty can easily allow to provide networks with a high level of sparsity.

Document code: We further quantitatively evaluate the average sparse ratio on latent representations of documents from different models, as shown in Figure 5b. We can see that 1) The NSTCSG yields the highest sparsity ratio, followed by NGSTC and STC, which outperform NTM and LDA by a large margin. 2) The document codes discovered by LDA and NTM are still very dense, while the representations learned by sparse models (NSTC and STC) are much sparser. It indicates the sparse models can discover focused topics and obtain discriminative representations of documents. 3) Similar to the word code, NGSTC outperforms NGSTC and STC. It demonstrates that the sparse group lasso penalty can achieve better sparsity than group lasso and lasso. 4) The sparsity ratios of sparse models increase with the topic numbers. The possible reason is that the sparse models tend to learn the focused topic number which approaches to the real topic number, and an increasing number of redundant topics can be discarded. 5) The NSTCSG inherits the advantages of NSTC and NGSTC, which can achieve the sparse topic representations of words and documents.

4.4 Generative Model Evaluation

To further evaluate our models as a generative model of documents, we show the test document perplexity achieved by each topic model on the 20NewsGroups with 50 topic numbers in table 2. Notice that the topic number in TopicVec can not be specified to a fixed value, thus we follow the set in (Li et al., 2016) with 281 topics. In table 3, we show the top-9 words of learned focused topics in 20 Newsgroups datasets. For each topic, we list top-9 words according to their probabili-

Table 1: Classification accuracy of different models on Web snippet and 20NG, with different number of topic K settings.

Dataset	Snippet					20NG				
k	**50**	**100**	**150**	**200**	**250**	**50**	**100**	**150**	**200**	**250**
LDA	0.682	0.592	0.573	0.615	0.583	0.545	0.615	0.607	0.613	0.623
STC	0.678	0.699	0.724	0.731	0.723	0.602	0.631	0.647	0.652	0.654
DocNADE	0.656	0.656	0.645	0.646	0.647	0.682	0.670	0.646	0.583	0.573
GLDA	0.669	0.689	0.675	0.670	0.623	0.367	0.438	0.465	0.496	0.526
NSTC	0.734	0.756	0.791	0.793	0.789	0.634	0.671	0.682	0.690	0.72
NSTCE	0.739	0.778	0.801	0.803	0.810	0.631	0.681	0.682	0.701	0.721
NGSTC	0.773	0.792	0.813	0.811	0.821	0.670	0.681	0.701	0.712	0.737
NSTCSG	0.788	0.813	0.821	0.823	0.829	0.665	0.687	0.691	0.717	0.735

Table 2: Perplexity on test dataset.

Model	20NG
LDA	1091
STC	611
DocNADE	896
TopicVec	650
NSTC	517

Table 3: Top Words of Learned Topics for 20Newsgroups.

computer	sport	drug	weapon	space-flight
computer	hockey	tobacco	nuclear	nasa
windows	games	drug	guns	flyers
ibm	motorcycl	fallacy	crime	space
drive	team	aids	booming	air
disk	play	hiv	controller	statelite
system	groups	dades	firearms	send
dos	came	illeg	military	launch
key	rom	same	wiring	apartment
hardware	ball	adict	neutral	la

ties under the corresponding topic. It is obvious that the learned topics are clear and meaningful. Such as *economics*, *hockey*, *games*, *play*, *ball* in the topic about sport. In Figure 6, we also use the 2-dimensional t-SNE method to get the visualization of the learned latent representations for Web Snippet and 20Newsgroups Dataset with 200 topics. For Web Snippet, we sample 10% of the whole dataset. For 20newsgroups, we sample 30% of the dataset. It is obvious to see that all documents are clustered into 8 and 20 distinct categories. It proves the semantic effectiveness of the documents codes learned by our model.

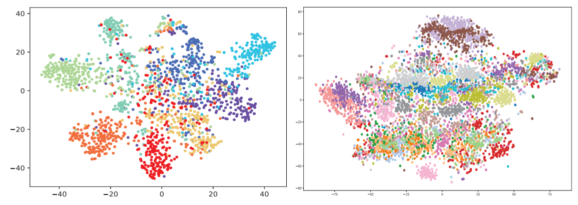


Figure 6: T-SNE embeddings of learned document representations for Web Snippet and 20News-Groups. Different colors mean different categories.

5 Conclusion

In this paper, we propose a novel neural sparsity-enhanced topic model NSTC, which improves STC by incorporating the neural network and word embeddings. Compared with other word embedding based and neural network based topic models, it overcomes the computation complexity of topic models, and improve the generation of representation over short documents. We present three variants of NSTC to illustrate the great flexibility of our framework. Experimental results demonstrate the effectiveness and efficiency of our models. For future work, we are interested in various extensions, including combining STC with autoencoding variational Bayes (AVB).

Acknowledgments

This work is supported by the National Science Foundation of China, under grant No.61472291 and grant No.61272110.

References

Lu Bai, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. Group sparse topical coding: from code to topic. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, pages 315–324.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *AAAI*.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *ACL*.

Karol Gregor and Yann LeCun. 2010. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. pages 399–406.

Matthias Heiler and Christoph Schnörr. 2006. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *Journal of Machine Learning Research* 7(Jul):1385–1407.

Weihua Hu and Junichi Tsujii. 2016. A latent concept topic model for robust topic inference using word embeddings. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 380.

Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2010. Fast inference in sparse coding algorithms with applications to object recognition. *arXiv preprint arXiv:1010.3467* .

Hugo Larochelle and Stanislas Lauly. 2012a. A neural autoregressive topic model. In *NIPS*.

Hugo Larochelle and Stanislas Lauly. 2012b. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*. pages 2708–2716.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative topic embedding: a continuous representation of documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 666–675.

Tianyi Lin, Wentao Tian, Qiaozhu Mei, and Hong Cheng. 2014. The dual-sparse topic model: mining focused topics and focused terms in short text. In *WWW*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*. pages 1614–1623.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics* 3:299–313.

Min Peng, Qianqian Xie, Jiajia Huang, Jiahui Zhu, Shuang Ouyang, Jimin Huang, and Gang Tian. 2016. Sparse topical coding with sparse groups. In *WAIM*.

Fei Tian, Bin Gao, Di He, and Tie-Yan Liu. 2016. Sentence level recurrent topic model: Letting topics speak for themselves. *CoRR* abs/1604.02038.

Sinead Williamson, Chong Wang, Katherine A. Heller, and David M. Blei. 2010. The ibp compound dirichlet process and its application to focused topic modeling. In *ICML*.

Jun Zhu and Eric P. Xing. 2011. Sparse topical coding. *CoRR* abs/1202.3778.

Document Similarity for Texts of Varying Lengths via Hidden Topics

Hongyu Gong[*] **Tarek Sakakini**[*] **Suma Bhat**[*] **Jinjun Xiong**[†]
[*]University of Illinois at Urbana-Champaign, USA
[†]T. J. Watson Research Center, IBM
[*]{hgong6, sakakini, spbhat2}@illinois.edu [†]jinjun@us.ibm.com

Abstract

Measuring similarity between texts is an important task for several applications. Available approaches to measure document similarity are inadequate for document pairs that have non-comparable lengths, such as a long document and its summary. This is because of the lexical, contextual and the abstraction gaps between a long document of rich details and its concise summary of abstract information. In this paper, we present a document matching approach to bridge this gap, by comparing the texts in a common space of hidden topics. We evaluate the matching algorithm on two matching tasks and find that it consistently and widely outperforms strong baselines. We also highlight the benefits of the incorporation of domain knowledge to text matching.

1 Introduction

Measuring the similarity between documents is of key importance in several natural processing applications including information retrieval (Salton and Buckley, 1988), book recommendation (Gopalan et al., 2014), news categorization (Ontrup and Ritter, 2002) and essay scoring (Landauer, 2003). A range of document similarity approaches have been proposed and effectively used in recent applications including (Lai et al., 2015; Bordes et al., 2015). Central to the tasks discussed above is the assumption that the documents being compared are of comparable lengths.

Advances in language processing approaches to transform natural language understanding, such as text summarization and recommendation, have generated new requirements for comparing documents. For instance, summarization techniques (extractive and abstractive) are capable of automatically generating textual summaries by converting a long document of several hundred words into a condensed text of only a few words while preserving the core meaning of the original text (Kedzie and McKeown, 2016). Conceivably, a related aspect of summarization is the task of bidirectional matching of a summary and a document or a set of documents, which is the focus of this study. The document similarity considered in this paper is between texts that have significant differences not only in length, but also in the abstraction level (such as a definition of an abstract concept versus a detailed instance of that abstract concept).

Table 1: A Sample Concept-Project Matching

Concept
Heredity: Inheritance and Variation of Traits
All cells contain genetic information in the form of DNA molecules. Genes are regions in the DNA that contain the instructions that code for the formation of proteins.
Project
Pedigree Analysis: A Family Tree of Traits
Do you have the same hair color or eye color as your mother? When we look at members of a family it is easy to see that some physical characteristics or traits are shared. To start this project, you should draw a pedigree showing the different members of your family. Ideally you should include multiple people from at least three generations.

As an illustration, consider the task of matching a **Concept** with a **Project** as shown in Table 1. Here a **Concept** is a grade-level science curriculum item and represents the summary. A **Project**, listed in a collection of science projects, represents the document. Projects typically are long texts including an introduction, materials and procedures, whereas science concepts are much shorter in comparison having a title and a concise and abstract description. The concepts and projects are described in detail in Section 5.1. The matching task here is to automatically suggest a hands-on project for a given concept in the curriculum, such that the project can help reinforce a learner's basic understanding of the concept. Conversely, given a science project, one may need to identify the concept it covers by matching it to a listed concept in the curriculum. This would be conceivable in the context of an intelligent tutoring system.

Challenges to the matching task mentioned above include: 1) The mismatch in the relative lengths of the documents being compared – a long piece of text (henceforth termed *document*) and a short piece of text (termed *summary*) – gives rise to the vocabulary mismatch problem, where the document and the summary do not share a majority of terms. 2) The context mismatch problem arising because a document provides a reasonable amount of text to infer the contextual meaning of a term, but a summary only provides a limited context, which may or may not involve the same terms considered in the document. These challenges render existing approaches to comparing documents–for instance, those that rely on document representations (e.g., Doc2Vec (Le and Mikolov, 2014))–inadequate, because the predominance of non-topic words in the document introduces noise to its representation while the summary is relatively noise-free, rendering Doc2Vec inadequate for comparing them.

Our approach to the matching problem is to allow a multi-view generalization of the document, where multiple hidden topics are used to establish a common ground to capture as much information of the document and the summary as possible and use this to score the relevance of the pair. We empirically validate our approach on two tasks – that of project-concept matching in grade-level science and that of scientific paper-summary matching – using both custom-made and publicly available datasets. The main contributions of this paper are:

1. We propose an embedding-based hidden topic model to extract topics and measure their importance in long documents.
2. We present a novel geometric approach to compare documents with differing modality (a long document to a short summary) and validate its performance relative to strong baselines.
3. We explore the use of domain-specific word embeddings for the matching task and show the explicit benefit of incorporating domain knowledge in the algorithm.
4. We make available the first dataset[1] on project-concept matching in the science domain to help further research in this area.

2 Related Works

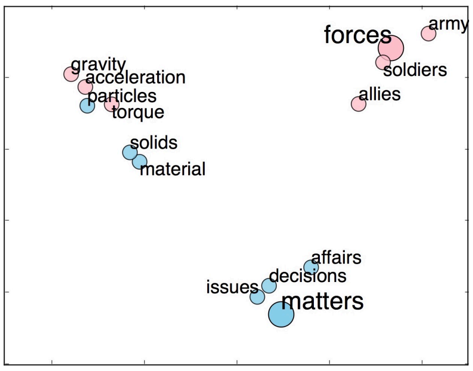
Document similarity approaches quantify the degree of relatedness between two pieces of texts of comparable lengths and thus enable matching between documents. Traditionally, statistical approaches (e.g., (Metzler et al., 2007)) and vector-space-based methods (including the robust Latent Semantic Analysis (LSA) (Dumais, 2004)) have been used for text similarity. More recently, neural network-based methods have been used for document representation and these include average word embeddings (Mikolov et al., 2013), Doc2Vec (Le and Mikolov, 2014), Skip-Thought vectors (Kiros et al., 2015), recursive neural network-based methods (Socher et al., 2014), LSTM architectures (Tai et al., 2015), and convolutional neural networks (Blunsom et al., 2014).

Considering works that avoid using an explicit document representation for comparing documents, the state-of-the-art method is Word Mover's Distance (WMD), which relies on pre-trained word embeddings (Kusner et al., 2015). Given these embeddings, the WMD defines the distance between two documents as the best transport cost of moving all words from one document to another within the space of word embeddings. The advantages of WMD are that it is hyper-parameter free and achieves high retrieval accuracy on document classification tasks with documents of comparable lengths. However, it is computationally expensive for long documents (Kusner et al., 2015).

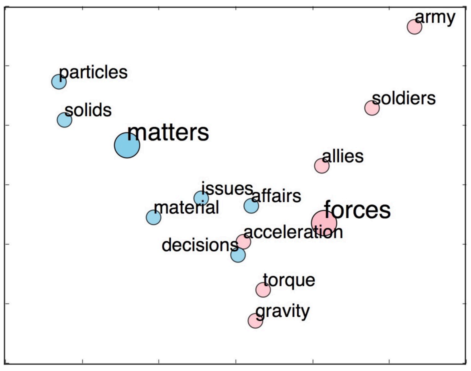
Clearly, what is lacking in prior literature is a study of document similarity approaches that match documents with widely different sizes. It is this gap in literature that we expect to fill by way of this study.

Latent Variable Models. Latent variable models including count-based and probabilistic models have been studied in many previous works. Count-based models such as Latent Semantic Indexing (LSI) compare two documents based on their combined vocabulary (Deerwester et al., 1990). When

[1]Our code and data are available at: `https://github.com/HongyuGong/Document-Similarity-via-Hidden-Topics.git`



(a) word geometry of general embedding



(b) word geometry of science domain embeddings

Figure 1: Two key words "forces" and "matters" are shown in red and blue respectively. Red words represent different senses of "forces", and blue words carry senses of "matters". "forces" mainly refers to "army" and "matters" refers to "issues" in general embedding of (a), whereas "forces" shows its sense of "gravity" and "matters" shows the sense of "solids" in science-domain embedding of (b)

documents have highly mismatched vocabularies such as those that we study, relevant documents might be classified as irrelevant. Our model is built upon word-embeddings which is more robust to such a vocabulary mismatch.

Probabilistic models such as Latent Dirichlet Analysis (LDA) define topics as distributions over words (Blei et al., 2003). In our model, topics are low-dimensional real-valued vectors (more details in Section 4.2).

3 Domain Knowledge

Domain information pertaining to specific areas of knowledge is made available in texts by the use of words with domain-specific meanings or senses. Consequently, domain knowledge has been shown to be critical in many NLP applications such as information extraction and multi-document summarization (Cheung and Penn, 2013a), spoken language understanding (Chen et al., 2015), aspect extraction (Chen et al., 2013) and summarization (Cheung and Penn, 2013b).

As will be described later, our distance metric for comparing a document and a summary relies on word embeddings. We show in this work, that embeddings trained on a science-domain corpus lead to better performance than embeddings on the general corpus (WikiCorpus). Towards this, we extract a science-domain sub-corpus from the WikiCorpus, and the corpus extraction will be detailed in Section 5.

To motivate the domain-specific behavior of polysemous words, we will qualitatively explore how domain-specific embeddings differ from the general embeddings on two polysemous science terms: *forces* and *matters*. Considering the fact that the meaning of a word is dictated by its neighbors, for each set of word embeddings, we plot the neighbors of these two terms in Figure 1 on to 2 dimensions using Locally Linear Embedding (LLE), which preserves word distances (Roweis and Saul, 2000). We then analyze the sense of the focus terms–here, *forces* and *matters*.

From Figure 1(a), we see that for the word *forces*, its general embedding is close to *army*, *soldiers*, *allies* indicating that it is related with violence and power in a general domain. Shifting our attention to Figure 1(b), we see that for the same term, its science embedding is closer to *torque*, *gravity*, *acceleration* implying that its science sense is more about physical interactions. Likewise, for the word *matters*, its general embedding is surrounded by *affairs* and *issues*, whereas, its science embedding is closer to *particles* and *material*, prompting that it represents substances. Thus, we conclude that domain specific embeddings (here, science), is capable of incorporating domain knowledge into word representations. We use this observation in our document-summary matching system to which we turn next.

4 Model

Our model that performs the matching between document and summary is depicted in Figure 2. It is composed of three modules that perform preprocessing, document topic generation, and relevance measurement between a document and a summary. Each of these modules is discussed below.

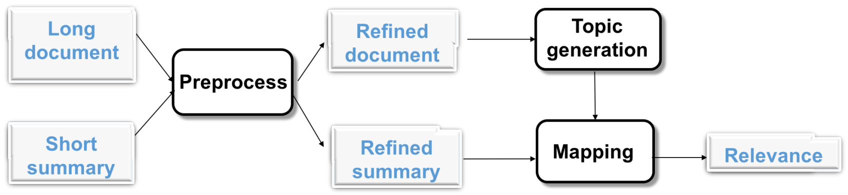


Figure 2: The system for document-summary matching

4.1 Preprocessing

The preprocessing module tokenizes texts and removes stop words and prepositions. This step allows our system to focus on the content words without impacting the meaning of original texts.

4.2 Topic Generation from Documents

We assume that a document (a long text) is a structured collection of words, with the 'structure' brought about by the composition of topics. In some sense, this 'structure' is represented as a set of hidden topics. Thus, we assume that a document is generated from certain hidden "topics", analogous to the modeling assumption in LDA. However, unlike in LDA, the "topics" here are neither specific words nor the distribution over words, but are are essentially a set of *vectors*. In turn, this means that words (represented as vectors) constituting the document structure can be generated from the hidden topic vectors.

Introducing some notation, the word vectors in a document are $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$, and the hidden topic vectors of the document are $\{\mathbf{h}_1, \ldots, \mathbf{h}_K\}$, where $\mathbf{w}_i, \mathbf{h}_k \in \mathbb{R}^d$, $d = 300$ in our experiments.

Linear operations using word embeddings have been empirically shown to approximate their compositional properties (e.g. the embedding of a phrase is nearly the sum of the embeddings of its component words) (Mikolov et al., 2013). This motivates the linear reconstruction of the words from the document's hidden topics while minimizing the reconstruction error. We stack the K topic vectors as a topic matrix $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_K] (K < d)$. We define the reconstructed word vector $\tilde{\mathbf{w}}_i$ for the word $\mathbf{w}_i$ as the optimal linear approximation given by topic vectors: $\tilde{\mathbf{w}}_i = \mathbf{H}\tilde{\boldsymbol{\alpha}}_i$, where

$$\tilde{\boldsymbol{\alpha}}_i = \underset{\boldsymbol{\alpha}_i \in \mathbb{R}^K}{\operatorname{argmin}} \|\mathbf{w}_i - \mathbf{H}\boldsymbol{\alpha}_i\|_2^2. \tag{1}$$

The reconstruction error E for the whole document is the sum of each word's reconstruction error and is given by: $E = \sum_{i=1}^{n} \|\mathbf{w}_i - \tilde{\mathbf{w}}_i\|_2^2$. This being a function of the topic vectors, our goal is to find the optimal $\mathbf{H}^*$ so as to minimize the error E:

$$\begin{aligned} \mathbf{H}^* &= \underset{\mathbf{H} \in \mathbb{R}^{d \times K}}{\operatorname{argmin}} \, E(\mathbf{H}) \\ &= \underset{\mathbf{H} \in \mathbb{R}^{d \times K}}{\operatorname{argmin}} \sum_{i=1}^{n} \min_{\boldsymbol{\alpha}_i} \|\mathbf{w}_i - \mathbf{H}\boldsymbol{\alpha}_i\|_2^2, \end{aligned} \tag{2}$$

where $\|\cdot\|$ is the Frobenius norm of a matrix.

Without loss of generality, we require the topic vectors $\{\mathbf{h}_i\}_{i=1}^K$ to be orthonormal, i.e., $\mathbf{h}_i^T \mathbf{h}_j = 1_{(i=j)}$. As we can see, the optimization problem (2) describes an optimal linear space spanned by the topic vectors, so the norm and the linear dependency of the vectors do not matter. With the orthonormal constraints, we simplify the form of the reconstructed vector $\tilde{\mathbf{w}}_i$ as:

$$\tilde{\mathbf{w}}_i = \mathbf{H}\mathbf{H}^T \mathbf{w}_i. \tag{3}$$

We stack word vectors in the document as a matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_n]$. The equivalent formulation to problem (2) is:

$$\begin{aligned} \min_{\mathbf{H}} \quad & \|\mathbf{W} - \mathbf{H}\mathbf{H}^T\mathbf{W}\|_2^2 \\ \text{s.t.} \quad & \mathbf{H}^T\mathbf{H} = \mathbf{I}, \end{aligned} \tag{4}$$

where $\mathbf{I}$ is an identity matrix.

The problem can be solved by Singular Value Decomposition (SVD), using which, the matrix $\mathbf{W}$ can be decomposed as $\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, where $\mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{V}^T\mathbf{V} = \mathbf{I}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix where the diagonal elements are arranged in a decreasing order of absolute values. We show in the supplementary material that the first K vectors in the matrix $\mathbf{U}$ are exactly the solution to $\mathbf{H}^* = [\mathbf{h}_1^*, \ldots, \mathbf{h}_K^*]$.

We find optimal topic vectors $\mathbf{H}^* = [\mathbf{h}_1^*, \ldots, \mathbf{h}_K^*]$ by solving problem (4). We note that these topic vectors are not equally important, and we say that one topic is more important than another if it can reconstruct words with smaller error. Define E_k as the reconstruction error when we only use topic vector $\mathbf{h}_k^*$ to reconstruct the document:

$$E_k = \|\mathbf{W} - \mathbf{h}_k^* \mathbf{h}_k^{*T} \mathbf{W}\|_2^2. \qquad (5)$$

Now define i_k as the *importance* of topic $\mathbf{h}_k^*$, which measures the topic's ability to reconstruct the words in a document:

$$i_k = \|\mathbf{h}_k^{*T} \mathbf{W}\|_2^2 \qquad (6)$$

We show in the supplementary material that the higher the importance i_k is, the smaller the reconstruction error E_k is. Now we normalize i_k as $\bar{i}_k$ so that the importance does not scale with the norm of the word matrix W, and so that the importances of the K topics sum to 1. Thus,

$$\bar{i}_k = i_k / (\sum_{j=1}^{K} i_j). \qquad (7)$$

The number of topics K is a hyperparameter in our model. A small K may not cover key ideas of the document, whereas a large K may keep trivial and noisy information. Empirically we find that $K = 15$ captures most important information from the document.

4.3 Topic Mapping to Summaries

We have extracted K topic vectors $\{\mathbf{h}_k^*\}_{k=1}^K$ from the document matrix $\mathbf{W}$, whose importance is reflected by $\{\bar{i}_k\}_{k=1}^K$. In this module, we measure the relevance of a document-summary pair. Towards this, a summary that matches the document should also be closely related with the "topics" of that document. Suppose the vectors of the words in a summary are stacked as a $d \times m$ matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_m]$, where $\mathbf{s}_j$ is the vector of the j-th word in a summary. Similar to the reconstruction of the document, the summary can also be reconstructed from the documents' topic vectors as shown in Eq. (3). Let $\tilde{\mathbf{s}}_j^k$ be the reconstruction of the summary word $\mathbf{s}_j$ given by one topic $\mathbf{h}_k^*$: $\tilde{\mathbf{s}}_j^k = \mathbf{h}_k^* \mathbf{h}_k^{*T} \mathbf{s}_j$.

Let $r(\mathbf{h}_k^*, \mathbf{s}_j)$ be the relevance between a topic vector $\mathbf{h}_k^*$ and summary word $\mathbf{s}_j$. It is defined as the cosine similarity between $\tilde{\mathbf{s}}_j^k$ and $\mathbf{s}_j$:

$$r(\mathbf{h}_k^*, \mathbf{s}_j) = \mathbf{s}_j{}^T \tilde{\mathbf{s}}_j^k / (\|\mathbf{s}_j\|_2 \cdot \|\tilde{\mathbf{s}}_j^k\|_2). \qquad (8)$$

Furthermore, let $r(\mathbf{h}_k^*, \mathbf{S})$ be the relevance between a topic vector and the summary, defined to be the average similarity between the topic vector and the summary words:

$$r(\mathbf{h}_k^*, \mathbf{S}) = \frac{1}{m} \sum_{j=1}^{m} r(\mathbf{h}_k^*, \mathbf{s}_j). \qquad (9)$$

The relevance between a topic vector and a summary is a real value between 0 and 1.

As we have shown, the topics extracted from a document are not equally important. Naturally, a summary relevant to more important topics is more likely to better match the document. Therefore, we define $r(\mathbf{W}, \mathbf{S})$ as the relevance between the document $\mathbf{W}$ and the summary $\mathbf{S}$, and $r(\mathbf{W}, \mathbf{S})$ is the sum of topic-summary relevance weighted by the importance of the topic:

$$r(\mathbf{W}, \mathbf{S}) = \sum_{k=1}^{K} \bar{i}_k \cdot r(\mathbf{h}_k^*, \mathbf{S}), \qquad (10)$$

where $\bar{i}_k$ is the importance of topic $\mathbf{h}_k^*$ as defined in (7). The higher $r(\mathbf{W}, \mathbf{S})$ is, the better the summary matches the document.

We provide a visual representation of the documents as shown in Figure 3 to illustrate the notion of hidden topics. The two documents are from science projects: a genetics project, *Pedigree Analysis: A Family Tree of Traits* (ScienceBuddies, 2017a), and a weather project, *How Do the Seasons Change in Each Hemisphere* (ScienceBuddies, 2017b). We project all embeddings to a three-dimensional space for ease of visualization.

As seen in Figure 3, the hidden topics reconstruct the words in their respective documents to the extent possible. This means that the words of a document lie roughly on the plane formed by their corresponding topic vectors. We also notice that the summary words (*heredity* and *weather* respectively for the two projects under consideration) lie very close to the plane formed by the hidden topics of the relevant project while remaining away from the plane of the irrelevant project. This shows that the words in the summary (and hence the summary itself) can also be reconstructed from the hidden topics of documents that match the summary (and are hence 'relevant' to the summary). Figure 3 visually explains the geometric relations between the summaries, the hidden topics and the documents. It also validates the representation power of the extracted hidden topic vectors.

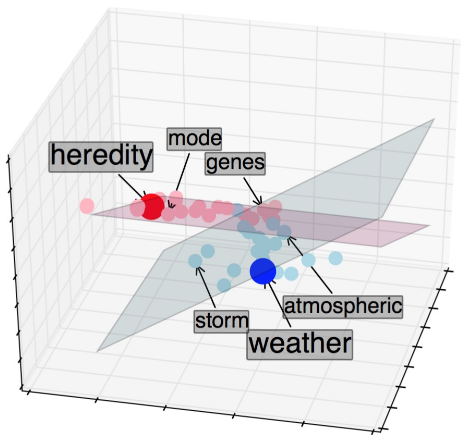


Figure 3: Words *mode* and *genes* from the document on genetics and words *storm* and *atmospheric* from document on weather are represented by pink and blue points respectively. Linear space of hidden topics in genetics form the pink plane, where summary word *heredity* (the red point) roughly lies. Topic vectors of the document on weather form the blue plane, and the summary word *weather* (the darkblue point) lies almost on the same plane.

5 Experiments

In this section, we evaluate our document-summary matching approach on two specific applications where texts of different sizes are compared. One application is that of concept-project matching useful in science education and the other is that of summary-research paper matching.

Word Embeddings. Two sets of 300-dimension word embeddings were used in our experiments. They were trained by the Continuous Bag-of-Words (CBOW) model in word2vec (Mikolov et al., 2013) but on different corpora. One training corpus is the full English WikiCorpus of size 9 GB (Al-Rfou et al., 2013). The second consists of science articles extracted from the WikiCorpus. To extract these science articles, we manually selected the science categories in Wikipedia and considered all subcategories within a depth of 3 from these manually selected root categories. We then extracted all articles in the aforementioned science categories resulting in a science corpus of size 2.4 GB. The word vectors used for documents and summaries are both from the pretrained word2vec embeddings.

Baselines We include two state-of-the-art methods of measuring document similarity for comparison using their implementations available in gensim (Řehůřek and Sojka, 2010).

(1) **Word movers' distance (WMD)** (Kusner et al., 2015). WMD quantifies the distance between a pair of documents based on word embeddings as introduced previously (c.f. Related Work). We take the negative of their distance as a measure of document similarity (here between a document and a summary).

(2) **Doc2Vec** (Le and Mikolov, 2014). Document representations have been trained with neural networks. We used two versions of doc2vec: one trained on the full English Wikicorpus and a second trained on the science corpus, same as the corpora used for word embedding training. We used the cosine similarity between two text vectors to measure their relevance.

For a given document-summary pair, we compare the scores obtained using the above two methods with that obtained using our method.

5.1 Concept-Project matching

Science projects are valuable resources for learners to instigate knowledge creation via experimentation and observation. The need for matching a science concept with a science project arises when learners intending to delve deeper into certain concepts search for projects that match a given concept. Additionally, they may want to identify the concepts with which a set of projects are related.

We note that in this task, science concepts are highly concise summaries of the core ideas in projects, whereas projects are detailed instructions of the experimental procedures, including an introduction, materials and a description of the procedure, as shown in Table 1. Our matching method provides a way to bridge the gap between abstract concepts and detailed projects. The format of the concepts and the projects is discussed below.

Concepts. For the purpose of this study we use the concepts available in the Next Generation Science Standards (NGSS) (NGSS, 2017). Each concept is accompanied by a short description. For example, one concept in life science is *Heredity: Inheritance and Variation of Traits*. Its description is *All cells contain genetic information in the form of DNA molecules. Genes are regions in the DNA that contain the instructions that code for the formation of proteins.* Typical lengths of concepts are around 50 words.

Projects. The website Science Buddies (Science-Buddies, 2017c) provides a list of projects from a variety of science and engineering disciplines such

Table 2: Classification results for the Concept-Project Matching task. All performance differences were statistically significant at $p = 0.01$.

method	topic_science	topic_wiki	wmd_science	wmd_wiki	doc2vec_science	doc2vec_wiki
precision	$\mathbf{0.758 \pm 0.012}$	0.750 ± 0.009	0.643 ± 0.070	0.568 ± 0.055	0.615 ± 0.055	0.661 ± 0.084
recall	$\mathbf{0.885 \pm 0.071}$	0.842 ± 0.010	0.735 ± 0.119	0.661 ± 0.119	0.843 ± 0.066	0.737 ± 0.149
fscore	$\mathbf{0.818 \pm 0.028}$	0.791 ± 0.007	0.679 ± 0.022	0.595 ± 0.020	0.695 ± 0.019	0.681 ± 0.032

as physical sciences, life sciences and social sciences. A typical project consists of an abstract, an introduction, a description of the experiment and the associated procedures. A project typically has more than 1000 words.

Dataset. We prepared a representative dataset 537 pairs of projects and concepts involving 53 unique concepts from NGSS and 230 unique projects from Science Buddies. Engineering undergraduate students annotated each pair with the decision whether it was a good match or not and received research credit. As a result, each concept-project pair received at least three annotations, and upon consolidation, we considered a concept-project pair to be a good match when a majority of the annotators agreed. Otherwise it was not considered a good match. The ratio between good matches and bad matches in the collected data was $44 : 56$.

Classification Evaluation. Annotations from students provided the ground truth labels for the classification task. We randomly split the dataset into tuning and test instances with a ratio of $1 : 9$. A threshold score was tuned on the tuning data, and concept-project pairs with scores higher than this threshold were classified as a good matches during testing. We performed 10-fold cross validation, and report the average precision, recall, F1 score and their standard deviation in Table 2.

Our topic-based metric is denoted as "topic", and the general-domain and science-domain embeddings are denoted as "wiki" and "science" respectively. We show the performance of our method against the two baselines while varying the underlying embeddings, thus resulting in 6 different combinations. For example, "topic_science" refers to our method with science embeddings. From the table (column 1) we notice the following: 1) Our method *significantly outperforms* the two baselines by a wide margin ($\approx$10%) in both the general domain setting as well as the domain-specific setting. 2) Using science domain-specific word embeddings instead of the general word embeddings results in the *best performance across all algorithms*. This performance was observed despite the word embeddings being trained on a *significantly smaller* corpus compared to the general domain corpus.

Besides the classification metrics, we also evaluated the directed matching from concepts to projects with ranking metrics.

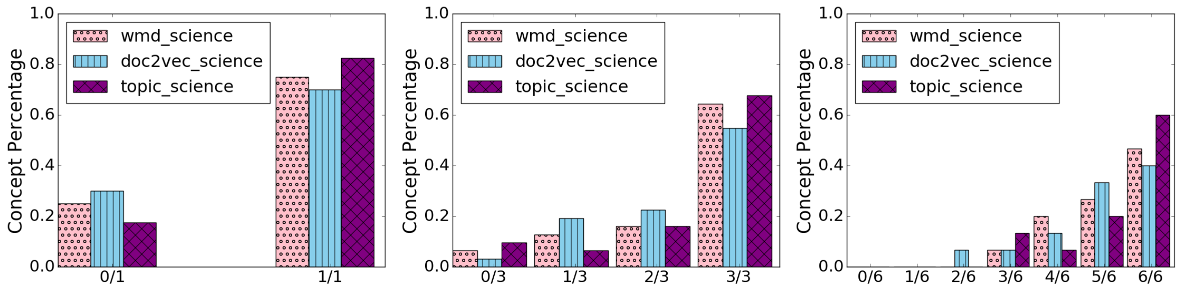
Ranking Evaluation Our collected dataset resulted in having a many-to-many matching between concepts and projects. This is because the same concept was found to be a good match for multiple projects and the same project was found to match many concepts. The previously described classification task evaluated the bidirectional concept-project matching. Next we evaluated the directed matching from concepts to projects, to see how relevant these top ranking projects are to a given input concepts. Here we use precision@k (Radlinski and Craswell, 2010) as the evaluation metric, considering the percentage of relevant ones among top-ranking projects.

For this part, we only considered the methods using science domain embeddings as they have shown superior performance in the classificaiton task. For each concept, we check the precision@k of matched projects and place it in one of k+1 bins accordingly. For example, for k=3, if only two of the three top projects are a correct match, the concept is placed in the bin corresponding to $2/3$. In Figure 4, we show the percentage of concepts that fall into each bin for the three different algorithms for k=1,3,6.

We observe that recommendations using the hidden topic approach fall more in the high value bin compared to others, performing consistently better than two strong baselines. The advantage becomes more obvious at precision@6. It is worth mentioning that wmd_science falls behind doc2vec_science in the classification task while it outperforms in the ranking task.

5.2 Text Summarization

The task of matching summaries and documents is commonly seen in real life. For example, we



(a) Precision@1 (b) Precision@3 (c) Precision@6

Figure 4: Ranking Performance of All Methods

use an event summary "Google's AlphaGo beats Korean player Lee Sedol in Go" to search for relevant news, or use the summary of a scientific paper to look for related research publications. Such matching constitutes an ideal task to evaluate our matching method between texts of different sizes.

Dataset. We use a dataset from the CL-SciSumm Shared Task (Jaidka et al., 2016). The dataset consists of 730 ACL Computational Linguistics research papers covering 50 categories in total. Each category consists of a reference paper (RP) and around 10 citing papers (CP) that contain citations to the RP. A human-generated summary for the RP is provided and we use the 10 CP as being relevant to the summary. The matching task here is between the summary and all CPs in each category.

Evaluation. For each paper, we keep all of its content except the sections of experiments and acknowledgement (these sections were omitted because often their content is often less related to the topic of the summary). The typical summary length is about 100 words, while a paper has more than 2000 words. For each topic, we rank all 730 papers in terms of their relevance generated by our method and baselines using both sets of embeddings. For evaluation, we use the information retrieval measure of precision@k, which considers the number of relevant matches in the top-k matchings (Manning et al., 2008). For each combination of the text similarity approaches and embeddings, we show precision@k for different k's in Figure 5. We observe that our method with science embedding achieves the best performance compared to the baselines, once again showing not only the benefits of our method but also that of incorporating domain knowledge.

6 Discussion

Analysis of Results. From the results of the two tasks we observe that our method outperforms two

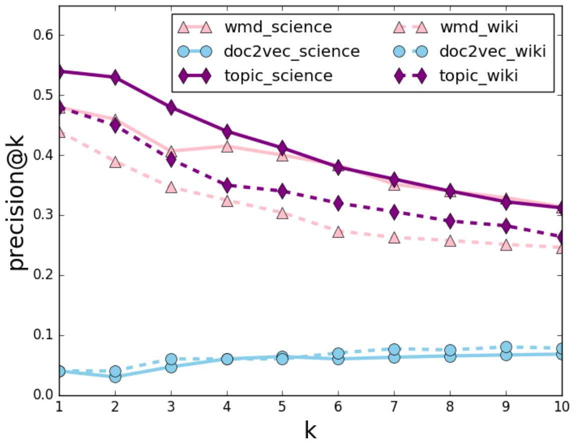


Figure 5: Summary-Article Matching

strong baselines. The reason for WMD's poor performance could be that the many uninformative words (those unrelated to the central topic) make WMD overestimate the distance between the document-summary pair. As for doc2vec, its single vector representation may not be able to capture all the key topics of a document. A project could contain multifaceted information, e.g., a project to study how climate change affects grain production is related to both environmental science and agricultural science.

Effect of Topic Number. The number of hidden topics K is a hyperparameter in our setting. We empirically evaluate the effect of topic number in the task of concept-project mapping. Figure 6 shows the F1 scores and the standard deviations at different K. As we can see, optimal K is 18. When K is too small, hidden topics are too few to capture key information in projects. Thus we can see that the increase of topic number from 3 to 6 brings a big improvement to the performance. Topic numbers larger than the optimal value degrade the performance since more topics incorporate noisy information. We note that the perfor-

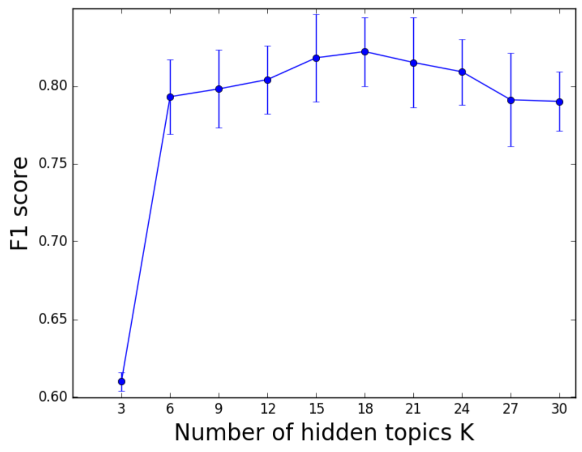


Figure 6: F1 score on concept-project matching with different topic numbers K

mance changes are mild when the number of topics are in the range of [18, 31]. Since topics are weighted by their importance, the effect of noisy information from extra hidden topics is mitigated.

Interpretation of Hidden Topics. We consider the summary-paper matching as an example with around 10 papers per category. We extracted the hidden topics from each paper, reconstructed words with these topics as shown in Eq. (3), and selected the words which had the smallest reconstruction errors. These words are thus closely related to the hidden topics, and we call them *topic words* to serve as an interpretation of the hidden topics. We visualize the cloud of such topic words on the set of papers about word sense disambiguation as shown in Figure 7. We see that the words selected based on the hidden topics cover key ideas such as *disambiguation*, *represent*, *classification* and *sentence*. This qualitatively validates the representation power of hidden topics. More examples are available in the supplementary material.

We interpret this to mean that proposed idea of multiple hidden topics captures the key information of a document. The extracted "hidden topics" represent the essence of documents, suggesting the appropriateness of our relevance metric to measure the similarity between texts of different sizes. Even though our focus in this study was the science domain we point out that the results are more generally valid since we made no domain-specific assumptions.

Varying Sensitivity to Domain. As shown in the results, the science-domain embeddings improved the classification of concept-project matching for

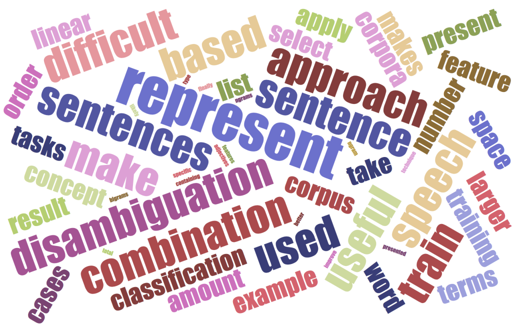


Figure 7: Topic words from papers on word sense disambiguation

the topic-based method by 2% in F1-score, WMD by 8% and doc2vec by 1%, thus underscoring the importance of domain-specific word embeddings.

Doc2vec is less sensitive to the domain, because it provides document-level representation. Even if some words cannot be disambiguated due to the lack of domain knowledge, other words in the same document can provide complementary information so that the document embedding does not deviate too much from its true meaning.

Our method, also a word embedding method, is not as sensitive to domain as WMD. It is robust to the polysemous words with domain-sensitive semantics, since hidden topics are extracted in the document level. Broader contexts beyond just words provide complementary information for word sense disambiguation.

7 Conclusion

We propose a novel approach to matching documents and summaries. The challenge we address is to bridge the gap between detailed long texts and its abstraction with hidden topics. We incorporate domain knowledge into the matching system to gain further performance improvement. Our approach has beaten two strong baselines in two downstream applications, concept-project matching and summary-research paper matching.

Acknowledgments

This work is supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. We thank ACL anonymous reviewers for their constructive suggestions. We thank Mathew Monfort for helping deploy annotation tasks, and thank Jong Yoon Lee for the dataset collection.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I Rudnicky. 2015. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *ACL (1)*, pages 483–494.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *EMNLP*, pages 1655–1667.

Jackie Chi Kit Cheung and Gerald Penn. 2013a. Probabilistic domain modelling with contextualized distributional semantic vectors. In *ACL (1)*, pages 392–401.

Jackie Chi Kit Cheung and Gerald Penn. 2013b. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *ACL (1)*, pages 1233–1242.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Susan T Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

Prem K Gopalan, Laurent Charlin, and David Blei. 2014. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*, pages 3176–3184.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. 2016. Overview of the cl-scisumm 2016 shared task. In *In Proceedings of Joint Workshop on Bibliometric-enhanced Information Retrieval and NLP for Digital Libraries (BIRNDL 2016)*.

Chris Kedzie and Kathleen McKeown. 2016. Extractive and abstractive event summarization over streaming web text. In *IJCAI*, pages 4002–4003.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Thomas K Landauer. 2003. Automatic essay assessment. *Assessment in education: Principles, policy & practice*, 10(3):295–308.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Donald Metzler, Susan Dumais, and Christopher Meek. 2007. Similarity measures for short segments of text. In *European Conference on Information Retrieval*, pages 16–27. Springer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

NGSS. 2017. Available at: `https://www.nextgenscience.org`. Accessed: 2017-06-30.

Jorg Ontrup and Helge Ritter. 2002. Hyperbolic self-organizing maps for semantic navigation. In *Advances in neural information processing systems*, pages 1417–1424.

Filip Radlinski and Nick Craswell. 2010. Comparing the sensitivity of information retrieval metrics. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 667–674. ACM.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New*

Challenges for NLP Frameworks, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

ScienceBuddies. 2017a. Available at: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Genom_p010/genetics-genomics/pedigree-analysis-a-family-tree-of-traits. Accessed: 2017-06-30.

ScienceBuddies. 2017b. Available at: https://www.sciencebuddies.org/science-fair-projects/project-ideas/Weather_p006/weather-atmosphere/how-do-the-seasons-change-in-each-hemisphere. Accessed: 2017-06-30.

ScienceBuddies. 2017c. Available at: http://www.sciencebuddies.org. Accessed: 2017-06-30.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1556–1566.

Eyes are the Windows to the Soul: Predicting the Rating of Text Quality Using Gaze Behaviour

★Sandeep Mathias, ★,♣,♢Diptesh Kanojia, ★Kevin Patel, ★Samarth Agrawal
♠Abhijit Mishra, ★Pushpak Bhattacharyya
★CSE Department, IIT Bombay
♣IITB-Monash Research Academy
♢Monash University, Australia
♠IBM Research, India
★,♣{sam,diptesh,kevin.patel,samartha,pb}@cse.iitb.ac.in
♠abhijimi@in.ibm.com

Abstract

Predicting a reader's rating of text quality is a challenging task that involves estimating different subjective aspects of the text, like structure, clarity, *etc.* Such subjective aspects are better handled using cognitive information. One such source of cognitive information is gaze behaviour. In this paper, we show that gaze behaviour does indeed help in effectively predicting the rating of text quality. To do this, we first model text quality as a function of three properties - organization, coherence and cohesion. Then, we demonstrate how capturing gaze behaviour helps in predicting each of these properties, and hence the overall quality, by reporting improvements obtained by adding gaze features to traditional textual features for score prediction. We also hypothesize that if a reader has fully understood the text, the corresponding gaze behaviour would give a better indication of the assigned rating, as opposed to partial understanding. Our experiments validate this hypothesis by showing greater agreement between the given rating and the predicted rating when the reader has a full understanding of the text.

1 Introduction

Automatically rating the quality of a text is an interesting challenge in NLP. It has been studied since Page's seminal work on automatic essay grading in the mid-1960s (Page, 1966). This is due to the dependence of quality on different aspects such as the overall structure of the text, clarity, *etc.* that are highly qualitative in nature, and whose scoring can vary from person to person (Person, 2013).

Scores for such qualitative aspects cannot be inferred solely from the text and would benefit from psycholinguistic information, such as gaze behaviour. Gaze based features have been used for co-reference resolution (Ross et al., 2016), sentiment analysis (Joshi et al., 2014) and translation annotation complexity estimation (Mishra et al., 2013). They could also be very useful for education applications, like evaluating readability (Mishra et al., 2017) and in automatic essay grading.

In this paper, we consider the following qualitative properties of text: Organization, Coherence and Cohesion. A text is **well-organized** if it begins with an introduction, has a body and ends with a conclusion. One of the other aspects of organization is the fact that it takes into account how the content of the text is split into paragraphs, with each paragraph denoting a single *idea*. If the text is too long, and not split into paragraphs, one could consider the text to be badly organized[1].

A text is **coherent** if it makes sense to a reader. A text is **cohesive** if it is well connected. Coherence and cohesion are two qualities that are closely related. A piece of text that is well-connected usually makes sense. Conversely, a piece of text that makes sense is usually well-connected. However, it is possible for texts to be coherent but lack cohesion. Table 1 provides some examples for texts that are coherent and cohesive, as well as those that lack one of those qualities.

There are different ways to model coherence and cohesion. Since coherence is a measure of how much sense the text makes, it is a semantic property of the text. It requires sentences within the text to be interpreted, by themselves, as well as with other sentences in the text (Van Dijk, 1980).

On the other hand, cohesion makes use of

[1]Refer supplementary material for example. We have placed it there due to space constraints.

Example	Comments
My favourite colour is blue. I like it because it is calming and it relaxes me. I often go outside in the summer and lie on the grass and look into the clear sky when I am stressed. For this reason, I'd have to say my favourite colour is blue.	Coherent and cohesive.
My favourite colour is blue. I'm calm and relaxed. In the summer I lie on the grass and look up.	Coherent but not cohesive. There is no link between the sentences. However, the text makes sense due to a lot of implicit clues (blue, favourite, relaxing, look up (and see the blue sky)).
My favourite colour is *blue*. *Blue* sports cars go **very fast**. Driving in **this way** is dangerous and can cause many *car crashes*. I had a *car accident* once and **broke my leg**. I was very sad because I had to miss a holiday in Europe because of **the injury**.	Cohesive but not coherent. The sentences are linked by words (that are in *italics* or in **bold**) between adjacent sentences. As we can see, every pair of adjacent sentences are connected by words / phrases, but the text does not make sense, since it first starts with blue, and describes missing a holiday due to injury.

Table 1: Examples of coherence and cohesion[2].

linguistic cues, such as references (demonstratives, pronouns, *etc.*), ellipsis (leaving out implicit words - Eg. Sam can type and I can [*type*] too), substitution (use of a word or phrase to replace something mentioned earlier - Eg. How's the croissant? I'd like to have **one** too.), conjunction (and, but, therefore, *etc.*), cohesive nouns (problem, issue, investment, *etc.*) and lexis (linking different pieces of text by synonyms, hyponyms, lexical chains, *etc.*) (Halliday and Hasan, 1976).

Using these properties, we model the overall text quality rating. We make use of a Likert scale (Likert, 1932) with a range of 1 to 4, for measuring each of these properties; the higher the score, the better is the text in terms of that property. We model the text quality rating on a scale of 1 to 10, using the three scores as input. In other words,

$$Quality(T) = Org(T) + Chr(T) + Chs(T) - 2,$$

where $Quality(T)$ is the text quality rating of the text T. $Org(T)$, $Chr(T)$, and $Chs(T)$ correspond to the $\underline{\textbf{Org}}$anization, $\underline{\textbf{C}}$o$\underline{\textbf{h}}$e$\underline{\textbf{r}}$ence, and $\underline{\textbf{C}}$o$\underline{\textbf{h}}$e$\underline{\textbf{s}}$ion scores respectively, for the text T, that are given by a reader. We subtract 2 to scale the scores from a range of 3 - 12, to a range of 1 - 10 for quality.

Texts with poor organization and/or cohesion can force readers to regress *i.e.* go to previous sentences or paragraphs. Texts with poor coherence may lead readers to fixate more on different portions of text to understand them. In other words, such gaze behaviour indirectly captures the effort needed by human readers to comprehend the text (Just and Carpenter, 1980), which, in turn, may influence the ratings given by them. Hence, these properties seem to be a good indicators for overall quality of texts.

In this paper, we address the following question: ***Can information obtained from gaze behaviour help predict reader's rating of quality of text by estimating text's organization, coherence, and cohesion?*** Our work answers that question in the affirmative. We found that using gaze features does contribute in improving the prediction of qualitative ratings of text by users.

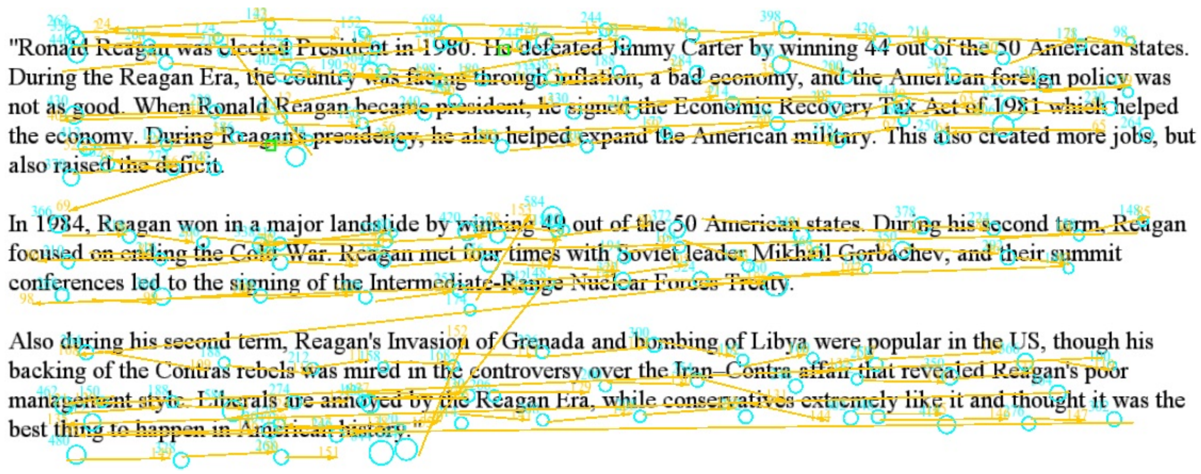
Our work has the following contributions. Firstly, we propose **a novel way to predict readers' rating of text** by recording their eye movements as they read the texts. Secondly, we show that **if a reader has understood the text completely, their gaze behaviour is more reliable**. Thirdly, **we also release our dataset**[3] to help in further research in using gaze features in other tasks involving predicting the quality of texts.

In this paper, we use the following terms related to eye tracking. The **interest area (IA)** is an area of the screen that is under interest. We mainly look at words as interest areas. A **fixation** takes place when the gaze is focused on a point of the screen. A **saccade** is the movement of gaze between two fixations. A **regression** is a special type of saccade in which the reader refers back to something that they had read earlier.

The rest of the paper is organized as follows. Section 2 describes the motivation behind our work. Section 3 describes related work in this field. Section 4 describes the different features that we used. Sections 5 and 6 describes our experiments and results. Section 6 also contains analysis of our experiments. Section 7 concludes our paper and mentions future work.

[2]We took the examples from this site for explaining coherence and cohesion: http://gordonscruton.blogspot.in/2011/08/what-is-cohesion-coherence-cambridge.html

[3]The dataset can be downloaded from http://www.cfilt.iitb.ac.in/cognitive-nlp/


"Ronald Reagan was elected President in 1980. He defeated Jimmy Carter by winning 44 out of the 50 American states. During the Reagan Era, the country was facing through inflation, a bad economy, and the American foreign policy was not as good. When Ronald Reagan became president, he signed the Economic Recovery Tax Act of 1981 which helped the economy. During Reagan's presidency, he also helped expand the American military. This also created more jobs, but also raised the deficit.

In 1984, Reagan won in a major landslide by winning 49 out of the 50 American states. During his second term, Reagan focused on ending the Cold War. Reagan met four times with Soviet leader Mikhail Gorbachev, and their summit conferences led to the signing of the Intermediate-Range Nuclear Forces Treaty.

Also during his second term, Reagan's Invasion of Grenada and bombing of Libya were popular in the US, though his backing of the Contras rebels was mired in the controversy over the Iran–Contra affair that revealed Reagan's poor management style. Liberals are annoyed by the Reagan Era, while conservatives extremely like it and thought it was the best thing to happen in American history."

Figure 1: Sample text showing fixations, saccades and regressions. This text was given scores of 4, 4, and 3 for organization, coherence and cohesion. The circles denote fixations, and the lines are saccades. Radius of the circles denote the duration of the fixation (in milliseconds), which is centred at the centre of the circle. This is the output from SR Research Data Viewer software.

2 Motivation

Reader's perception of text quality is subjective and varies from person to person. Using cognitive information from the reader can help in predicting the score he / she will assign to the text. A well-written text would not have people fixate too long on certain words, or regress a lot to understand, while a badly written text would do so.

Figure 1 shows the gaze behaviour for a sample text. The circles denote fixations, and the arrows denote saccades. If we capture the gaze behaviour, as well as see how well the reader has understood the text, we believe that we can get a clearer picture of the quality rating of the text.

One of the major concerns is *How are we going to get the gaze data?* This is because capability to gather eye-tracking data is not available to the masses. However, top mobile device manufacturers, like Samsung, have started integrating basic eye-tracking software into their smartphones (Samsung Smart Scroll) that are able to detect where the eye is fixated, and can be used in applications like scrolling through a web page. Start-ups, like Cogisen[4], have started using gaze features in their applications, such as using gaze information to improve input to image processing systems. Recently, SR Research has come up with a portable eye-tracking system[5].

[4]www.cogisen.com

[5]https://www.sr-research.com/products/eyelink-portable-duo/

3 Related Work

A number of studies have been done showing how eye tracking can model aspects of text. Word length has been shown to be positively correlated with fixation count (Rayner, 1998) and fixation duration (Henderson and Ferreira, 1993). Word predictability (i.e. how well the reader can predict the next word in a sentence) was also studied by Rayner (1998), where he found that unpredictable words are less likely to be skipped than predictable words.

Shermis and Burstein (2013) gives a brief overview of how text-based features are used in multiple aspects of essay grading, including grammatical error detection, sentiment analysis, short-answer scoring, *etc.* Their work also describes a number of current essay grading systems that are available in the market like *E-rater*® (Attali and Burstein, 2004). In recent years, there has been a lot of work done on evaluating the holistic scores of essays, using deep learning techniques (Alikaniotis et al., 2016; Taghipour and Ng, 2016; Dong and Zhang, 2016).

There has been little work done to model text organization, such as Persing et al. (2010) (using machine learning) and Taghipour (2017) (using neural networks). However, there has been a lot of work done to model coherence and cohesion, using methods like lexical chains (Somasundaran et al., 2014), an entity grid (Barzilay and Lapata, 2005), *etc.* An interesting piece of work to model coherence was done by Soricut and Marcu (2006) where they used a machine translation-based approach to model coherence. Zesch et al. (2015) use topical overlap to model coherence for essay grading. Discourse connectors are used as a heuristic to model cohesion by Zesch et al. (2015) and Persing and Ng (2015). Our work is novel because it makes use of gaze behaviour to model and predict coherence and cohesion in text.

In recent years, there has been some work in using eye-tracking to evaluate certain aspects of the text, like readability (Gonzalez-Garduño and Søgaard, 2017; Mishra et al., 2017), grammaticality (Klerke et al., 2015), *etc.*. Our work uses eye-tracking to predict the score given by a reader to a complete piece of text (rather than just a sentence as done by Klerke et al. (2015)) and show that the scoring is more reliable if the reader has understood the text.

4 Features

In order to predict the scores of the different properties of the text, we use the following text and gaze features.

4.1 Text-based Features

We use a set of text-based features to come up with a baseline system to predict the scores for different properties.

The first set of features that we use are **length and count-based features**, such as word length, word count, sentence length, count of transition phrases[6] *etc.* (Persing and Ng, 2015; Zesch et al., 2015).

The next set of features that we use are **complexity features**, namely the degree of polysemy, coreference distance, and the Flesch Reading Ease Score (FRES) (Flesch, 1948). These features help in normalizing the gaze features for text complexity. These features were extracted using Stanford CoreNLP (Manning et al., 2014), and MorphAdorner (Burns, 2013).

The third set of features that we use are **stylistic features** such as the ratios of the number of adjectives, nouns, prepositions, and verbs to the number of words in the text. These features are used to model the distributions of PoS tags in good and bad texts. These were extracted using NLTK[7] (Loper and Bird, 2002).

The fourth set of features that we use are **word embedding features**. We use the average of word vectors of each word in the essay, using Google News word vectors (Mikolov et al., 2013). The word embeddings are **300 dimensions**. We also calculate the mean and maximum similarities between the word vectors of the content words in adjacent sentences of the text, using GloVe word embeddings[8] (Pennington et al., 2014).

The fifth set of features that we use are **language modeling features**. We use the count of words that are absent in Google News word vectors and misspelled words using the PyEnchant[9] library. In order to check the grammaticality of the text, we construct a 5-gram language model, using the Brown Corpus (Francis and Kucera, 1979).

The sixth set of features are **sequence features**. These features are particularly useful in modeling organization (sentence and paragraph sequence similarity) (Persing et al., 2010), coherence and cohesion (PoS and lemma similarity). Pitler et al. (2010) showed that cosine similarity of adjacent sentences as one of the best predictors of linguistic quality. Hence, we also create vectors for the PoS tags and lemmas for each sentence in the text. The dimension of the vector is the number of distinct PoS tags / lemmas.

The last set of features that we look at are **entity grid features**. We define entities as the nouns in the document, and do coreference resolution to resolve pronouns. We then construct an entity grid (Barzilay and Lapata, 2005) - a 1 or 0 grid that checks whether an entity is present or not in a given sentence. We take into account sequences of entities across sentences that possess ***at least*** one 1, that are either bigrams, trigrams or 4-grams. A sequence with multiple 1s denote entities that are close to each other, while sequences with a solitary 1 denote that an entity is just mentioned once and we do not come across it again for a number of sentences.

4.2 Gaze-based Features

The gaze-based features are dependent on the gaze behaviour of the participant with respect to interest areas.

[6]`https://writing.wisc.edu/Handbook/Transitions.html`

[7]`http://www.nltk.org/`

[8]We found that using GloVe here and Google News for the mean word vectors worked best.

[9]`https://pypi.python.org/pypi/pyenchant/`

Fixation Features

The **First Fixation Duration** (FFD) shows the time the reader fixates on a word when he / she first encounters it. An increased FFD intuitively could mean that the word is more complex and the reader spends more time in understanding the word (Mishra et al., 2016).

The **Second Fixation Duration** (SFD) is the duration in which the reader fixates on a particular interest area the second time. This happens during a regression, when a reader is trying to link the word he / she just read with an earlier word.

The **Last Fixation Duration** (LFD) is the duration in which the reader fixates on a particular interest area the final time. At this point, we believe that the interest area has been processed.

The **Dwell Time** (DT) is the total time the reader fixates on a particular interest area. Like first fixation, this also measures the complexity of the word, not just by itself, but also with regard to the entire text (since it takes into account fixations when the word was regressed, *etc.*)

The **Fixation Count** (FC) is the number of fixations on a particular interest area. A larger fixation count could mean that the reader frequently goes back to read that particular interest area.

Regression Features

IsRegression (IR) is the number of interest areas where a regression happened before reading ahead and **IsRegressionFull** (IRF) is the number of interest areas where a regression happened. The **Regression Count** (RC) is the total number of regressions. The **Regression Time** (RT) is the duration of the regressions from an interest area. These regression features could help in modeling semantic links for coherence and cohesion.

Interest Area Features

The **Skip Count** (SC) is the number of interest areas that have been skipped. The **Run Count** (RC) is the number of interest areas that have at least one fixation. A larger run count means that more interest areas were fixated on. Badly written texts would have higher run counts (and lower skip counts), as well as fixation counts, because the reader will fixate on these texts for a longer time to understand them.

5 Experiment Details

In this section, we describe our experimental setup, creation of the dataset, evaluation metric, classifier details, *etc.*

5.1 Ordinal Classification vs. Regression

For each of the properties - organization, coherence and cohesion, we make use of a Likert scale, with scores of 1 to 4. Details of the scores are given in Table 2. For scoring the quality, we use the formula described in the Introduction. Since we used a Likert scale, we make use of ordinal classification, rather than regression. This is because each of the grades is a discrete value that can be represented as an ordinal class (where $1 < 2 < 3 < 4$), as compared to a continuous real number.

5.2 Evaluation Metric

For the predictions of our experiments, we use Cohen's Kappa with quadratic weights - quadratic weighted Kappa (QWK) (Cohen, 1968) because of the following reasons. Firstly, unlike accuracy and F-Score, Cohen's Kappa takes into account whether or not agreements happen by chance. Secondly, weights (either linear or quadratic) take into account distance between the given score and the expected score, unlike accuracy and F-score where mismatches (either 1 vs. 4, or 1 vs.2) are penalized the same. Quadratic weights reward matches and penalize mismatches more than linear weights.

To measure the Inter-Annotator Agreement of our raters, we make use of Gwet's second-order agreement coefficient (Gwet's AC2) as it can handle ordinal classes, weights, missing values, and multiple raters rating the same document (Gwet, 2014).

5.3 Creation of the Dataset

In this subsection, we describe how we created our dataset. We describe the way we made the texts, the way they were annotated and the inter-annotator agreements for the different properties.

Details of Texts

To the best of our knowledge there isn't a publicly available dataset with gaze features for textual quality. Hence, we decided to create our own. Our dataset consists of a diverse set of **30 texts**, from Simple English Wikipedia (**10 articles**), English Wikipedia (**8 articles**), and online news articles (**12 articles**)[10]. We did not wish to overburden the readers, so we kept the size of texts to

[10]The sources for the articles were `https://simple.wikipedia.org`, `https://en.wikipedia.org`, and `https://newsela.com`

Property	Grade	Guidelines
Organization	1	**Bad**. There is no organization in the text.
	2	**OK**. There is little / no link between the paragraphs, but they each describe an idea.
	3	**Good**. Some paragraphs may be missing, but there is an overall link between them.
	4	**Very Good**. All the paragraphs follow a flow from the Introduction to Conclusion.
Coherence	1	**Bad**. The sentences do not make sense.
	2	**OK**. Groups of sentences may make sense together, but the text still may not make sense.
	3	**Good**. Most of the sentences make sense. The text, overall, makes sense.
	4	**Very Good**. The sentences and overall text make sense.
Cohesion	1	**Bad**. There is little / no link between any 2 adjacent sentences in the same paragraph.
	2	**OK**. There is little / no link between adjacent paragraphs. However, each paragraph is cohesive
	3	**Good**. All the sentences in a paragraph are linked to each other and contribute in understanding the paragraph.
	4	**Very Good**. The text is well connected. All the sentences are linked to each other and help in understanding the text.

Table 2: Annotation guidelines for different properties of text.

approximately **200 words** each. The original articles ranged from a couple hundred words (Simple English Wikipedia) to over a thousand words (English Wikipedia). We first summarized the longer articles manually. Then, for the many articles over 200 words, we removed a few of the paragraphs and sentences. In this way, despite all the texts being published, we were able to introduce some poor quality texts into our dataset. The articles were sampled from a variety of genres, such as History, Science, Law, Entertainment, Education, Sports, *etc.*

Details of Annotators

The dataset was annotated by **20 annotators** in the **age** group of **20-25**. Out of the 20 annotators, the distribution was 9 high school graduates (current college students), 8 college graduates, and 3 annotators with a post-graduate degree.

In order to check the **eyesight** of the annotators, we had each annotator look at different parts of the screen. While they did that, we recorded how their fixations were being detected. Only if their fixations to particular parts of the screen tallied with our requests, would we let them participate in annotation.

All the participants in the experiment were **fluent speakers of English**. A few of them scored over 160 in GRE Verbal test and/or over 110 in TOEFL. Irrespective of their appearance in such exams, each annotator was made to take an English test before doing the experiments. The participants had to read a couple of passages, answer comprehension questions and score them for organization, coherence and cohesion (as either good / medium / bad). In case they either got both comprehension questions wrong, or labeled a good passage bad (or vice versa), they failed the test[11].

[11]25 annotators applied, but we chose only 20. 2 of the rejected annotators failed the test, while the other 3 had bad eyesight.

Property	**Full**	**Overall**
Organization	0.610	0.519
Coherence	0.688	0.633
Cohesion	0.675	0.614

Table 3: Inter-Annotator Agreements (Gwet's AC2) for each of the properties.

In order to help the annotators, they were given 5 **sample texts** to differentiate between good and bad organization, coherence and cohesion. Table 1 has some of those texts[12].

Inter-Annotator Agreement

Each of the properties were scored in the range of 1 to 4. In addition, we also evaluated the participant's understanding of the text by asking them a couple of questions on the text. Table 3 gives the inter-annotator agreement for each of the 3 properties that they rated. The column **Full** shows the agreement only if the participant answered both the questions correct. The **Overall** column shows the agreement irrespective of the participant's comprehension of the text.

5.4 System Details

We conducted the experiment by following standard norms in eye-movement research (Holmqvist et al., 2011). The display screen is kept **about 2 feet** from the reader, and the camera is placed midway between the reader and the screen. The reader is seated and the position of his head is fixed using a chin rest.

Before the text is displayed, we calibrate the camera by having the participant fixate on **13**

[12]The texts for good and bad organization are too long to provide in this paper. They will be uploaded in supplementary material.

points on the screen and validate the calibration so that the camera is able to predict the location of the eye on the screen accurately. After calibration and validation, the text is displayed on the screen in **Times New Roman** typeface with **font size 23**. The reader reads the text and while that happens, we record the reader's eye movements. The readers were allowed to take **as much time as they needed** to finish the text. Once the reader has finished, the reader moves to the next screen.

The next two screens each have a question that is based on the passage. These questions are used to verify that the reader did not just skim through the passage, but understood it as well. The questions were multiple choice, with 4 options[13]. The questions test literal comprehension (where the reader has to recall something they read), and interpretive comprehension (where the reader has to infer the answer from the text they read). After this, the reader scores the texts for organization, coherence and cohesion. The participants then take a short break (about 30 seconds to a couple of minutes) before proceeding with the next text. This is done to prevent reading fatigue over a period of time. After each break, we recalibrate the camera and validate the calibration again.

For obtaining gaze features from a participant, we collect gaze movement patterns using an SR Research Eye Link 1000 eye-tracker (monocular stabilized head mode, sampling rate 500Hz). It is able to collect all the gaze details that we require for our experiments. Reports are generated for keyboard events (message report) and gaze behaviour (interest area report) using SR Research Data Viewer software.

5.5 Classification Details

We also process the articles for obtaining the text features as described in Section 4. Given that we want to show the utility of gaze features, we ran each of the following classifiers with 3 feature sets - only text, only gaze, and all features.

We split the data into a training - test split of sizes **70%** and **30%**. We used a Feed Forward Neural Network with **1 hidden layer** containing **100 neurons** (Bebis and Georgiopoulos, 1994)[14]. The size of the input vector was **361 features**. Out of these, there were **49 text features**, plus **300 dimension word embeddings features**, **11 gaze features**, and **1 class label**. The data was split using stratified sampling, to ensure that there is a similar distribution of classes in each of the training and test splits. The Feed Forward Neural Network was implemented using TensorFlow (Abadi et al., 2015) in Python. We ran the neural network over **10,000 epochs**, with a learning rate of **0.001** in **10 batches**. The loss function that we used was the **mean square error**.

In order to see how much the participant's understanding of the text would reflect on their scoring, we also looked at the data based on how the participant scored in the comprehension questions after they read the article. We split the articles into 2 subsets here - $Full$, denoting that the participant answered both the questions correctly, and $Partial$, denoting that they were able to answer only one of the questions correctly. The readers showed $Full$ understanding in **269 instances** and $Partial$ understanding in **261 instances**. We used the same setup here (same training - test split, stratified sampling, and feed forward neural network). We omit the remaining **70 instances** where the participant got none of the questions correct, as the participant could have scored the texts completely randomly.

6 Results and Analysis

Table 4 shows the results of our experiments using the feed forward neural network classifier. The first column is the property being evaluated. The next 3 columns denote the results for the Text, Gaze and Text+Gaze feature sets.

Property	**Text**	**Gaze**	**Text+Gaze**
Organization	0.237	0.394	**0.563**
Coherence	0.261	0.285	**0.550**
Cohesion	0.120	0.229	**0.451**
Quality	0.230	0.304	**0.552**

Table 4: QWK scores for the three feature sets on different properties.

The QWK scores are the predictions which we obtain with respect to the scores of all the 30 documents, scored by all 20 raters. Textual features when augmented with gaze based features show significant improvement for all the properties.

[13]**Example Passage Text**: The text in Figure 1
Question: "How many states did Ronald Reagan win in both his Presidential campaigns?"
Correct Answer: "93" (44+49)

[14]We also used other classifiers, like Naive Bayes, Logistic Regression and Random Forest. However, the neural network outperformed them.

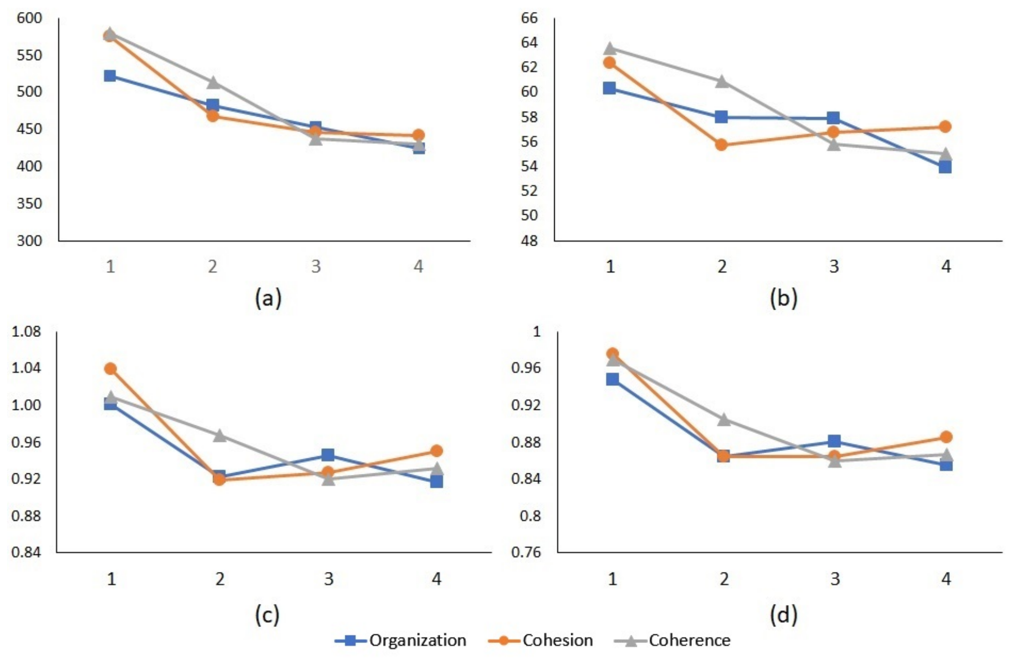


Figure 2: Relation between some of the different gaze features and the score. The gaze features are (a) RD, (b) SFD, (c) FC and (d) RC. For figures (a) and (b), the units on the y-axis are milliseconds. For figures (c) and (d) the numbers are a ratio to the number of interest areas in the text. The x-axis in all 4 graphs is the score given by the annotators.

We check the statistical significance of improvement of adding gaze based features for the results in Table 4. To test our hypothesis - that adding gaze features make a statistically significant improvement - we run the t-test. Our null hypothesis: Gaze based features do not help in prediction, any more than text features themselves, and whatever improvements happen when gaze based features are added to the textual features, are not statistically significant. We choose a significance level of $p < 0.001$. For all the improvements, we found them to be statistically significant above this α level, rejecting our null hypothesis.

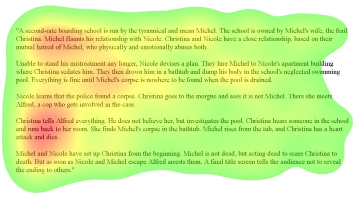
We also evaluate how the participant's understanding of the text affects the way they score the text. Table 5 shows the results of our experiments taking the reader's comprehension into account. The first column is the property being evaluated. The second column is the level of comprehension - $Full$ for the passages where the participant answered both the questions correctly, and $Partial$ for the passages where the participant answered one question correctly. The next 3 columns show the results using the Text feature set, the Gaze feature set, and both (Text+Gaze) feature sets. From this table, we see that wherever the gaze features are used, there is far greater agreement for those with $Full$ understanding as compared to $Partial$ understanding.

Property	**Comp.**	**Text**	**Gaze**	**Text+Gaze**
Organization	Full	**0.319**	**0.319**	**0.563**
	Partial	0.115	0.179	0.283
Coherence	Full	0.255	**0.385**	**0.601**
	Partial	**0.365**	0.343	0.446
Cohesion	Full	**0.313**	**0.519**	**0.638**
	Partial	0.161	0.155	0.230
Quality	Full	**0.216**	**0.624**	**0.645**
	Partial	0.161	0.476	0.581

Table 5: QWK scores for the three feature sets on different properties categorized on the basis of reader comprehension.

Figure 2 shows a clear relationship between some of the gaze features and the scores given by readers for the properties - organization, cohesion and coherence. In all the charts, we see that texts with the lowest scores have the longest durations (regression / fixation) as well as counts (of fixations and interest areas fixated).

Figure 3 shows the fixation heat maps for 3 texts whose quality scores were good (10), medium (6) and bad (3), read by the same participant. From these heat maps, we see that the text rated good has highly dense fixations for only a part of the text,



"A second-rate boarding school is run by the tyrannical and mean Michel. The school is owned by Michel's wife, the frail Christina. Michel flaunts his relationship with Nicole. Christina and Nicole have a close relationship, based on their mutual hatred of Michel, who physically and emotionally abuses both.

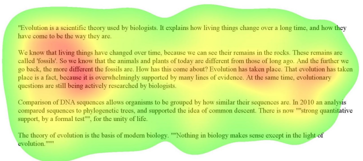
Unable to stand his mistreatment any longer, Nicole devises a plan. They lure Michel to Nicole's apartment building where Christina sedates him. They then drown him in a bathtub and dump his body in the school's neglected swimming pool. Everything is fine until Michel's corpse is nowhere to be found when the pool is drained.

Nicole learns that the police found a corpse. Christina goes to the morgue and sees it is not Michel. There she meets Alfred, a cop who gets involved in the case.

Christina tells Alfred everything. He does not believe her, but investigates the pool. Christina hears someone in the school and runs back to her room. She finds Michel's corpse in the bathtub. Michel rises from the tub, and Christina has a heart attack and dies.

Michel and Nicole have set up Christina from the beginning. Michel is not dead, but acting dead to scare Christina to death. But as soon as Nicole and Michel escape Alfred arrests them. A final title screen tells the audience not to reveal the ending to others."

(a) Good (rated 10)



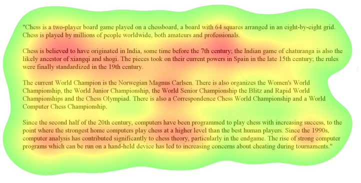
"Evolution is a scientific theory used by biologists. It explains how living things change over a long time, and how they have come to be the way they are.

We know that living things have changed over time, because we can see their remains in the rocks. These remains are called 'fossils'. So we know that the animals and plants of today are different from those of long ago. And the further we go back, the more different the fossils are. How has this come about? Evolution has taken place. That evolution has taken place is a fact, because it is overwhelmingly supported by many lines of evidence. At the same time, evolutionary questions are still being actively researched by biologists.

Comparison of DNA sequences allows organisms to be grouped by how similar their sequences are. In 2010 an analysis compared sequences to phylogenetic trees, and supported the idea of common descent. There is now ""strong quantitative support, by a formal test"", for the unity of life.

The theory of evolution is the basis of modern biology. ""Nothing in biology makes sense except in the light of evolution."""

(b) Medium (rated 6)



"Chess is a two-player board game played on a chessboard, a board with 64 squares arranged in an eight-by-eight grid. Chess is played by millions of people worldwide, both amateurs and professionals.

Chess is believed to have originated in India, some time before the 7th century; the Indian game of chaturanga is also the likely ancestor of xiangqi and shogi. The pieces took on their current powers in Spain in the late 15th century; the rules were finally standardized in the 19th century.

The current World Champion is the Norwegian Magnus Carlsen. There is also organizes the Women's World Championship, the World Junior Championship, the World Senior Championship the Blitz and Rapid World Championships and the Chess Olympiad. There is also a Correspondence Chess World Championship and a World Computer Chess Championship.

Since the second half of the 20th century, computers have been programmed to play chess with increasing success, to the point where the strongest home computers play chess at a higher level than the best human players. Since the 1990s, computer analysis has contributed significantly to chess theory, particularly in the endgame. The rise of strong computer programs which can be run on a hand-held device has led to increasing concerns about cheating during tournaments."

(c) Bad (rated 3)

Figure 3: Fixation heatmap examples for one of the participants from SR Research Data Viewer software.

as compared to the medium and bad texts. This shows that badly written texts force the readers to fixate a lot more than well-written texts.

6.1 Ablation Tests

In order to see which of the gaze feature sets is important, we run a set of ablation tests. We ablate the fixations, regressions and interest area feature sets one at a time. We also ablated each of the **individual gaze features**.

Property	Fixation	Regression	Interest Areas
Organization	-0.102	-0.017	**-0.103**
Coherence	-0.049	-0.077	**-0.088**
Cohesion	-0.015	**-0.040**	0.037
Quality	0.002	0.016	**-0.056**

Table 6: **Difference** in QWK scores when ablating three gaze behaviour feature sets for different properties.

Table 6 gives the result of our ablation tests on the three feature sets - fixation, regression and interest area feature sets. The first column is the property that we are measuring. The next 3 columns denote the **difference** between the predicted QWK that we got from ablating the fixation, regression and interest area feature sets. We found that the Interest Area feature set was the most important, followed by fixation and regression.

Among the individual features, **Run Count** (RC) was found to be the most important for organization and quality. **First Fixation Duration** (FFD) was the most important feature for coherence, and **IsRegressionFull** (IRF) was the most important feature for cohesion. We believe that this is because the number of interest areas that are fixated on at least once and the number of interest areas that are skipped play an important role in determining how much of the text was read and how much was skipped. However, for cohesion, regression features are the most important, because they show a link between the cohesive clues (like lexis, references, *etc.*) in adjacent sentences.

7 Conclusion and Future Work

We presented a novel approach to predict reader's rating of texts. The approach estimates the overall quality on the basis of three properties - organization, coherence and cohesion. Although well defined, predicting the score of these properties for a text is quite challenging. It has been established that cognitive information such as gaze behaviour can help in such subjective tasks (Mishra et al., 2013, 2016). We hypothesized that gaze behavior will assist in predicting the scores of text quality. To evaluate this hypothesis, we collected gaze behaviour data and evaluated the predictions using only the text-based features. When we took gaze behaviour into account, we were able to significantly improve our predictions of organization, coherence, cohesion and quality. We found out that, in all cases, there was an improvement in the agreement scores when the participant who rated the text showed full understanding, as compared to partial understanding, using only the Gaze features and the Text+Gaze features. This indicated that gaze behaviour is more reliable when the reader has understood the text.

To the best of our knowledge, our work is pioneering in using gaze information for predicting text quality rating. In future, we plan to use use approaches, like multi-task learning (Mishra et al., 2018), in estimating gaze features and using those estimated features for text quality prediction.

Acknowledgements

We'd like to thank all the anonymous reviewers for their constructive feedback in helping us improve our paper. We'd also like to thank Anoop Kunchukuttan, a research scholar from the Centre for Indian Language Technology, IIT Bombay for his valuable input.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 715–725.

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series* 2004(2).

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 141–148. https://doi.org/10.3115/1219840.1219858.

George Bebis and Michael Georgiopoulos. 1994. Feed-forward neural networks. *IEEE Potentials* 13(4):27–31.

Philip R Burns. 2013. Morphadorner v2: A java library for the morphological adornment of english language texts. *Northwestern University, Evanston, IL* .

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70(4):213.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring – an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1072–1077.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology* 32(3):221.

W Nelson Francis and Henry Kucera. 1979. The brown corpus: A standard corpus of present-day edited american english. *Providence, RI: Department of Linguistics, Brown University [producer and distributor]* .

Ana Valeria Gonzalez-Garduño and Anders Søgaard. 2017. Using gaze to predict text readability. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Copenhagen, Denmark, pages 438–443. http://www.aclweb.org/anthology/W17-5050.

Kilem L Gwet. 2014. *Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters*. Advanced Analytics, LLC.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. 1976. *Cohesion in english*. Longman Group Ltd.

John M Henderson and Fernanda Ferreira. 1993. Eye movement control during reading: Fixation measures reflect foveal but not parafoveal processing difficulty. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale* 47(2):201.

Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford.

Aditya Joshi, Abhijit Mishra, Nivvedan Senthamilselvan, and Pushpak Bhattacharyya. 2014. Measuring sentiment annotation complexity of text. In *ACL (2)*. pages 36–41.

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

Sigrid Klerke, Héctor Martínez Alonso, and Anders Søgaard. 2015. Looking hard: Eye tracking for detecting grammaticality of automatically compressed sentences. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*. Linköping University Electronic Press, Sweden, Vilnius, Lithuania, pages 97–105. http://www.aclweb.org/anthology/W15-1814.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* .

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*. Association for Computational Linguistics, pages 63–70.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Abhijit Mishra, Pushpak Bhattacharyya, and Michael Carl. 2013. Automatically predicting sentence translation difficulty. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 346–351.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. Predicting readers' sarcasm understandability by modeling gaze behavior. In *AAAI*. pages 3747–3753.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Scanpath complexity: Modeling reading effort using gaze information. In *AAAI*. pages 4429–4436.

Abhijit Mishra, Srikanth Tamilselvam, Riddhiman Dasgupta, Seema Nagar, and Kuntal Dey. 2018. Cognition-cognizant sentiment analysis with multitask subjectivity summarization based on annotators gaze behavior .

Ellis B Page. 1966. The imminence of... grading essays by computer. *The Phi Delta Kappan* 47(5):238–243.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 229–239.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 543–552. http://www.aclweb.org/anthology/P15-1053.

Robert Person. 2013. Blind truth: An examination of grading bias.

Emily Pitler, Annie Louis, and Ani Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 544–554. http://www.aclweb.org/anthology/P10-1056.

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

Joe Cheri Ross, Abhijit Mishra, and Pushpak Bhattacharyya. 2016. Leveraging annotators gaze behaviour for coreference resolution. In *Proceedings of the 7th Workshop on Cognitive Aspects of Computational Language Learning*. pages 22–26.

Mark D Shermis and Jill Burstein. 2013. *Handbook of automated essay evaluation: Current applications and new directions*. Routledge.

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 950–961.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Association for Computational Linguistics, Sydney, Australia, pages 803–810. http://www.aclweb.org/anthology/P/P06/P06-2103.

Kaveh Taghipour. 2017. *Robust Trait-Specific Essay Scoring Using Neural Networks and Density Estimators*. Ph.D. thesis.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1882–1891.

Teun Adrianus Van Dijk. 1980. Text and context explorations in the semantics and pragmatics of discourse .

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Denver, Colorado, pages 224–232. http://www.aclweb.org/anthology/W15-0626.

Multi-Input Attention for Unsupervised OCR Correction

Rui Dong **David A. Smith**
College of Computer Information and Science
Northeastern University
{dongrui, dasmith}@ccs.neu.edu

Abstract

We propose a novel approach to OCR post-correction that exploits repeated texts in large corpora both as a source of noisy target outputs for unsupervised training and as a source of evidence when decoding. A sequence-to-sequence model with attention is applied for single-input correction, and a new decoder with multi-input attention averaging is developed to search for consensus among multiple sequences. We design two ways of training the correction model without human annotation, either training to match noisily observed textual variants or bootstrapping from a uniform error model. On two corpora of historical newspapers and books, we show that these unsupervised techniques cut the character and word error rates nearly in half on single inputs and, with the addition of multi-input decoding, can rival supervised methods.

1 Introduction

Optical character recognition (OCR) software has made vast quantities of printed material available for retrieval and analysis, but severe recognition errors in corpora with low quality of printing and scanning or physical deterioration often hamper accessibility (Chiron et al., 2017). Many digitization projects have employed manual proofreading to further correct OCR output (Holley, 2009), but this is time consuming and depends on fostering a community of volunteer workers. These problems with OCR are exacerbated in library-scale digitization by commercial (e.g., Google Books, Newspapers.com), government (e.g., Library of Congress, Bibliothèque nationale de France), and nonprofit (e.g., Internet Archive) organizations. The scale of these projects not only makes it difficult to adapt OCR models to their diverse layouts and typefaces but also makes it impractical to present any OCR output other than a single-best transcript.

Existing methods for automatic OCR post-correction are mostly supervised methods that correct recognition errors in a single OCR output (Kolak and Resnik, 2002; Kolak et al., 2003; Yamazoe et al., 2011). Those systems are not scalable since human annotations are expensive to acquire, and they are not capable of utilizing complementary sources of information. Another line of work is ensemble methods (Lund et al., 2013, 2014) combining OCR results from multiple scans of the same document. Most of these ensemble methods, however, require aligning multiple OCR outputs (Lund and Ringger, 2009; Lund et al., 2011), which is intractable in general and might introduce noise into the later correction stage. Furthermore, voting-based ensemble methods (Lund and Ringger, 2009; Wemhoener et al., 2013; Xu and Smith, 2017) only work where the correct output exists in one of the inputs, while classification methods (Boschetti et al., 2009; Lund et al., 2011; Al Azawi et al., 2015) are also trained on human annotations.

To address these challenges, we propose an *unsupervised* OCR post-correction framework both to correct *single* input text sequences and also to exploit *multiple* candidate texts by simultaneously *aligning, correcting, and voting among input sequences*. Our proposed method is based on the observation that significant number of duplicate and near-duplicate documents exist in many corpora (Xu and Smith, 2017), resulting in OCR output containing repeated texts with various quality. As shown by the example in Table 1, different errors (characters in red) are introduced when the OCR system scans the same text in multiple editions, each with its own layout, fonts, etc. For example, `in` is recognized as `m` in the first output and `a` is recognized as `u` in the third output, while the second output is correctly recognized. Therefore, duplicated texts with diverse errors could serve as complementary information sources for each other.

OCR Output	eor**y that I have been slam in battle, for 1 sorry that I have been slain in battle, for I sorry tha' I have been s_uin in battle, f_r I
Original Text	sorry that I have been slain in battle, for I

Table 1: Example duplicate texts in OCR'd digital corpora

In this paper, we aim to train an unsupervised correction model via utilizing the duplication in OCR output. We propose to map each erroneous OCR'd text unit to either its high-quality duplication or a consensus correction among its duplications via bootstrapping from an uniform error model. The baseline correction system is a sequence-to-sequence model with attention (Bahdanau et al., 2015), which has been shown to be effective in text correction tasks (Chollampatt et al., 2016; Xie et al., 2016).

We also seek to improve the correction performance for duplicated texts by integrating multiple inputs. Previous work on combining multiple inputs in neural translation deal with data from different domains, e.g., multilingual (Zoph and Knight, 2016) or multimodal (Libovický and Helcl, 2017) data. Therefore, their models need to be trained on multiple inputs to learn parameters to combine inputs from each domain. Given that the inputs of our task are all from the same domain, our model is trained on a single input and introduces multi-input attention to generate a consensus result merely for decoding. It does not require learning extra parameters for attention combination and thus is more efficient to train. Furthermore, average attention combination, a simple multi-input attention mechanism, is proposed to improve both the effectiveness and efficiency of multi-input combination on the OCR post-correction task.

We experiment with both **supervised and unsupervised training** and with **single- and multi-input decoding** on data from two manually transcribed collections in English with diverse typefaces, genres, and time periods: newspaper articles from the Richmond (Virginia) Daily Dispatch (RDD) from 1860–1865 and books from 1500–1800 from the Text Creation Partnership (TCP). For both collections, which were manually transcribed by other researchers and are in the public domain, we aligned the one-best output of an OCR system to the manual transcripts. We also aligned the OCR in the training and evaluation sets to other public-domain newspaper issues (from the Library of Congress) and books (from the Internet Archive) to find multiple duplicates as "witnesses", where available, for each line. Experimental results on both datasets show that our proposed averarge attention combination mechanism is more effective than existing methods in integrating multiple inputs. Moreover, our noisy error correction model achieves comparable performance with the supervised model via multiple-input decoding on duplicated texts.

In summary, our contributions are: (1) a scalable framework needing no supervision from human annotations to train the correction model; (2) a multi-input attention mechanism incorporating aligning, correcting, and voting on multiple sequences simultaneously for consensus decoding, which is more efficient and effective than existing ensemble methods; and (3) a method that corrects text either with or without duplicated versions, while most existing methods can only deal with one of these cases.

2 Data Collection

We perform experiments on one-best OCR output from two sources: two million issues from the Chronicling America collection of historic U.S. newspapers, which is the largest public-domain full-text collection in the Library of Congress;[1] and three million public-domain books in the Internet Archive.[2]

For supervised training and for evaluation, we aligned manually transcribed texts to these one-best OCR transcripts: 1384 issues of the Richmond (Virginia) Daily Dispatch from 1860–1865 (RDD)[3] and 934 books from 1500–1800 from the

[1] `chroniclingamerica.loc.gov`: Historical newspapers also constitute the largest digitized text collections in the Australian National Library (Trove) and the Europeana consortium.

[2] `https://archive.org/details/texts`. Google Books and the Hathi Trust consortium also hold many in-copyright books and require licensing agreements to access public-domain materials.

[3] `dlxs.richmond.edu/d/ddr/`: the transcription from the University of Richmond includes all articles but only some advertisements.

Text Creation Partnership (TCP).[4] Both of these manually transcribed collections, which were produced independently from the current authors, are in the public domain and in English, although both Chronicling America and the Internet Archive also contain much non-English text.

To get more evidence for the correct reading of an OCR'd line, we aligned each OCR'd RDD issue to *other* issues of the RDD and other newspapers from Chronicling America and aligned each OCR'd TCP page to other pre-1800 books in the Internet Archive. To perform these alignments between noisy OCR transcripts efficiently, we used methods from our earlier work on text-reuse analysis (Smith et al., 2014; Wilkerson et al., 2015). An inverted index of hashes of word 5-grams was produced, and then all pairs from different pages in the same posting list were extracted. Pairs of pages with more than five shared hashed 5-grams were aligned with the Smith-Waterman algorithm with equal costs for insertion, deletion, and substitution, which returns a maximally aligned subsequence in each pair of pages (Smith and Waterman, 1981). Aligned passages that were at least five lines long in the target RDD or TCP text were output. For each target OCR line—i.e., each line in the training or test set—there are thus, in addition to the ground-truth manual transcript, zero or more **witnesses** from similar texts, to use the term from textual criticism.

In our experiments on OCR correction, each training and test example is a line of text following the layout of the scanned image documents[5]. The average number of characters per line is 42.4 for the RDD newspapers and 53.2 for the TCP books. Table 2 lists statistics for the number of OCR'd text lines with manual transcriptions and additional witnesses. 43% of the manually transcribed lines have witnesses in the RDD newspapers, and 64% of them have witnesses in the TCP books. In the full Chronicling America data, 44% of lines align to at least one other witness. Although not all OCR collections will have this level of repetition, it is notable that these collections, which are some of the largest public-domain digital libraries, do exhibit this kind of reprinting. Similarly, at least 25% of the pages in Google's web crawls are duplicates (Henzinger, 2006). Although we exploit text reuse, where available, to improve decoding and unsupervised training, we also show (Table 5) significant improvements to OCR accuracy with only a single transcript.

Dataset	# Lines w/manual	# Lines w/manual & witnesses
RDD	2.2M	0.95M (43%)
TCP	8.6M	5.5M (64%)

Table 2: Statistics for the number of OCR'd lines in million (M) from the Richmond Dispatch and TCP Books with manual transcriptions (Column 1) or with both transcriptions and multiple witnesses (Column 2).

3 Methods

In this section, we first define our problem in §3.1, followed by model description. In general, we train an OCR error correction model via an attention-based RNN encoder-decoder, which takes a single erroneous OCR'd line as input and outputs the corrected text (§3.2). At decoding time, multi-input attention combination strategies are introduced to allow the decoder to integrate information from multiple inputs (§3.3). Finally, we discuss several unsupervised settings for training the correction model in §3.4.

3.1 Problem Definition

Given a line of OCR'd text $\mathbf{x}$, comprising the sequence of characters $[x_1, \cdots, x_{T_S}]$, our goal is to map it to an error-free text $\mathbf{y} = [y_1, \cdots, y_{T_T}]$ via modeling $p(\mathbf{y}|\mathbf{x})$. Given $p(\mathbf{y}|\mathbf{x})$, we also seek to model $p(\mathbf{y}|\mathbf{X})$ to search for consensus among duplicated texts $\mathbf{X}$, where $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ are duplicated lines of OCR'd text.

3.2 Attention-based Seq2Seq Model

Similar to previous work (Bahdanau et al., 2015), the encoder is a bidrectional RNN (Schuster and Paliwal, 1997) that converts source sequence $\mathbf{x} = [x_1, \cdots, x_{T_S}]$ into a sequence of RNN states $\mathbf{h} = [h_1, \cdots, h_{T_S}]$, where $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$ is a concatenation of both forward and backward hidden states at time step $i (1 \leq i \leq T_S)$. We have

$$\overrightarrow{h}_i = f(x_i, \overrightarrow{h}_{i-1}); \quad \overleftarrow{h}_i = f(x_i, \overleftarrow{h}_{i+1}), \quad (1)$$

here f is the dynamic function, e.g., LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014).

The decoder RNN predicts the output sequence $\mathbf{y} = [y_1, \cdots, y_{T_T}]$, through the following dynamics and prediction model:

$$s_t = f(y_{t-1}, s_{t-1}, c_t); \quad (2)$$

[4] www.textcreationpartnership.org

[5] The datasets can be downloaded from http://www.ccs.neu.edu/home/dongrui/ocr.html

$$p(y_t|y_{<t}, \mathbf{x}) = g(y_{t-1}, s_t, c_t), \tag{3}$$

where s_t is the RNN state and c_t is the context vector at time t. y_t is the predicted symbol from the target vocabulary at time t via prediction function $g(\cdot)$. The context vector is given as a linear combination of the encoder hidden states:

$$c_t = \sum_{i=1}^{T_S} \alpha_{t,i} h_i; \quad \alpha_{t,i} = \frac{e^{\eta(s_{t-1}, h_i)}}{\sum_\tau e^{\eta(s_{t-1}, h_\tau)}} \tag{4}$$

where $\alpha_{t,i}$ is the weight for each hidden state h_i and η is the function that computes the strength of each encoder hidden state according to current decoder hidden state. The loss function is the cross-entropy loss per time step summed over the output sequence y:

$$L(\mathbf{x}, \mathbf{y}) = -\sum_{t=1}^{T_S} \log p(y_t|\mathbf{x}, y_{<t}) \tag{5}$$

3.3 Multi-input Attention

Given a trained Seq2Seq model $p(\mathbf{y}|\mathbf{x})$, our goal is to combine multiple input sequences $\mathbf{X}$ to generate the target sequence $\mathbf{y}$, i.e., to utilize information from multiple sources at decoding time. Assume that N relevant source sequences $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ are observed, where each sequence $\mathbf{x}_l = [x_{l,1}, \cdots, x_{l,T_l}]$ $(1 \le l \le N)$ and T_l is the length of the l^{th} sequence. Then, a sequence of hidden states $\mathbf{h}_l = [h_{l,1}, \cdots, h_{l,T_l}]$ is generated by the encoder network for each input sequence $\mathbf{x}_l$. At each decoding time step t, the decoder searches through encoder hidden states $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_N]$ to compute a global context vector c_t. Different strategies to combine attention from multiple encoders are described as follows.

Flat Attention Combination. Flat attention combination assigns a weight $\alpha_{t,l,i}$ to each encoder hidden state $h_{l,i}$ for each input sequence $\mathbf{x}_l$ as:

$$\alpha_{t,l,i} = \frac{e^{\eta(s_{t-1}, h_{l,i})}}{\sum_{l'=1}^{N} \sum_{\tau=1}^{T_{l'}} e^{\eta(s_{t-1}, h_{l',\tau})}}. \tag{6}$$

Therefore, the global context vector is given by

$$c_t = \sum_{l=1}^{N} \sum_{i=1}^{T_l} \alpha_{t,l,i} h_{l,i}. \tag{7}$$

Flat attention combination is similar to single-input decoding in that it concatenates all inputs into a long sequence, except that the encoder hidden states are computed independently for each input.

Hierarchical Attention Combination. The structure of hierarchical attention combination is presented in Figure 1. We first compute a context vector for each input as:

$$\mathbf{c}_{t,l} = \sum_{i=1}^{T_l} \alpha_{t,l,i} h_{l,i}; \quad \alpha_{t,l,i} = \frac{e^{\eta(s_{t-1}, h_{l,i})}}{\sum_{\tau=1}^{T_l} e^{\eta(s_{t-1}, h_{l,\tau})}}. \tag{8}$$

Then a global context vector c_t is computed as a weighted sum of all the context vectors:

$$c_t = \sum_{i=1}^{N} \beta_{t,l} c_{t,l}, \tag{9}$$

where $\beta_{t,l}$ is the weight assigned to each context vector $c_{t,l}$ and computed in different ways as follows:

(a) *Weighted Attention Combination.* In weighted attention combination, the weight for each context vector is given by its dot product with the decoder state in the transformed common space:

$$\beta_{t,l} = \frac{e^{\eta(s_{t-1}, c_{t,l})}}{\sum_{l'=1}^{N} e^{\eta(s_{t-1}, \mathbf{c}_{t,l'})}}. \tag{10}$$

(b) *Average Attention Combination.* In average attention combination, each input sequence is treated as equally weighted. Thus $\beta_{t,l} = \frac{1}{N}$ for each input sequence $\mathbf{x}_l$. It is more efficient than the weighted attention combination in that it does not need to compute a weight for each input.

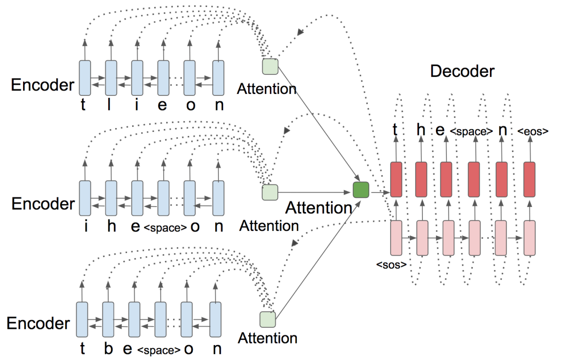


Figure 1: Hierarchical attention combination.

These attention-combination methods do not have parameters trained on multiple inputs and are only introduced at decoding time. In contrast, Libovický and Helcl (2017) and Zoph and Knight (2016) introduce parameters for each type of input and require training and decoding with the same number of inputs.

3.4 Training Settings

In this section, we introduce different settings for training our correction model, a single-input attention-based Seq2Seq model (§3.2), which transforms each OCR'd text line into a corrected version generated via different mechanisms.

Supervised Training. In this setting, the correction model is trained to map each OCR'd line into the corresponding manual transcription, i.e., the human annotation. We call the correction model trained in this setting Seq2Seq-Super.

Unsupervised Training. In the absence of ground truth transcriptions, we can use different methods to generate a noisy corrected version for each OCR'd line.

(a) *Noisy Training.* In this setting, the correction model is trained to transform each OCR'd text line to a selected high-quality witness. The quality of the witnesses is measured by a 5-gram character language model built on the New York Time Corpus (Sandhaus, 2008) with KenLM toolkit (Heafield, 2011). For each OCR'd line with multiple witnesses, a score is assigned to each witness by the language model, divided by the number of characters in it to reduce the effect of the length of a witness. Then a witness with the highest score is chosen as the noisy ground truth for each line. Those lines with low score for all witnesses are removed. We call the correction model trained in this setting Seq2Seq-Noisy.

(b) *Synthetic Training.* In this setting, the error correction model is trained to recover a manually corrupted out-of-domain corpus. We construct the synthetic dataset by injecting uniformly distributed insertion, deletion and substitution errors into the New York Times corpus. Firstly, the news articles are split into lines with random length between $[1, 70]$ following a Gaussian distribution $N(45, 5)$, which is similar to that of the real world dataset. Then, a certain number of lines are randomly selected and injected with equal number of insertion, deletion and substitution errors. The correction model is then trained to recover the original line from each corrupted line. We call this model Seq2Seq-Syn.

(c) *Synthetic Training with Bootstrapping.* In this setting, we propose to further improve the performance of synthetic training via bootstrapping. The correction model trained on synthetic dataset does not perform well when correcting a given input from real world dataset, due to their difference in error distributions. But it achieves comparable performance with the supervised model when decoding lines with multiple witnesses, since the model could further benefit from jointly aligning and voting among multiple inputs. Thus, with the multi-input attention mechanism introduced in §3.3, we first generate a high-quality consensus correction for each OCR'd line with witnesses via the correction model trained on synthetic data. Then, the a bootstrapped model is trained to transform those lines into their consensus correction results. We call the correction model trained in this setting Seq2Seq-Bootstrap.

4 Experiments

In this section, we first introduce the details of our experimental setup (§4.1). Then, the results of preliminary experiments comparing the performance of different options for the single-input Seq2Seq model and the multi-input attention combination strategies are presented in §4.2. The main experimental results for evaluating the correction model trained in different training settings and decoded with/without multi-input attention are reported and explained in §4.3. Further discussions of our model are described in §4.4.

4.1 Experimental Setup

We begin by describing the data split, training details, baseline systems, and evaluation metrics.

4.1.1 Training Details

For both RDD newspapers and TCP books, we randomly split the OCR'd lines into 80% training and 20% test either by the date of the newspaper or by the name of the books. For the RDD newspapers, we have 1.7M training lines and 0.44M test lines. For the TCP books, 2.8M lines are randomly sampled from the whole training set for different training settings to conduct a fair comparison with noisy training, and about 1.6M lines are used for testing.

Both the encoder and decoder of our model has 3 layers with 400 hidden units for each layer, where GRU is applied as the dynamic function. Adam optimizer with a learning rate of 0.0003 and default decay rates is used to train the correction model . We train up to 40 epochs with a minibatch size of 128 and select the model with the lowest perplexity on the development set. The decoder implements beam search with a beam width of 100.

4.1.2 Baselines and Comparisons

In preliminary experiments, we first compare the neural translation model (§3.2) with a commonly used Seq2Seq model, pruned conditional random fields (PCRF) (Schnober et al., 2016) on the single-input correction task. CRF models have been shown to be very competitve on tasks such as OCR post-correction, spelling correction, and lemmatization. After that, we compare the different multi-input attention strategies introduced in §3.3 on multi-input correction task to choose the best strategy for the main experiments.

In the main experiment, we compare the performance of correction models trained in different training settings and decode with and without multiple witnesses. Two ensembles methods, language model ranking (LMR) and majority vote (Xu and Smith, 2017), are also considered as unsupervised baseline methods. LMR chooses a single high-quality witness for each OCR'd line by a language model as the correction for that line. Majority vote first aligns multiple input sequences using a greedy pairwise algorithm (since multiple sequence alignment is intractable) and then votes on each position in the alignment, with a slight advantage given to the original OCR output in case of ties. We also tried to use an exact unsupervised method for consensus decoding based on dual decomposition (Paul and Eisner, 2012). Their implementation, unfortunately, turned out not to return a certificate of completion on most lines in our data even after thousands of iterations.

4.1.3 Evaluation Metrics

Word error rate (WER) and character error rate (CER) are used to compare the performance of each method. Case is ignored. Lattice word error rate (LWER) and lattice character error rate (LCER) are also computed as the oracle performance for each method, which could reveal the capability of each model to be applied to downstream tasks taking lattices as input, e.g., reranking or retrieval of the correction hypotheses (Taghva et al., 1996; Lam-Adesina and Jones, 2006). We compute the macro average for each type of error rate, which allows us to use a paired permutation significance test.

4.2 Preliminary Experiments

In this section, we conduct two preliminary experiments to study different options for both the single-input correction models and the multi-input attention combination strategies.

4.2.1 Single Input Correction Model

Model	CER	WER
None	0.18133	0.41780
PCRF$_{(order=5,w=4)}$	0.11403	0.25116
PCRF$_{(order=5,w=6)}$	0.11535	0.25617
Attn	**0.11028***	**0.23405***

Table 3: CER and WER on single-input correction for PCRF and Attn-Seq2Seq on RDD newspapers. Results from Attn-Seq2Seq that are significantly better than the PCRF are highlighted with *($p < 0.05$, paired permutation test). The best result for each column is in **bold**.

We first compare the attention-based Seq2Seq (Attn-Seq2Seq) model, with a traditional Seq2Seq model, PCRF, on single input correction task. As the PCRF implementation of Schnober et al. (2016) is highly memory and time consuming for training on long sequences, we compare it with Attn-Seq2Seq model on a smaller dataset with 100K lines randomly sampled from RDD newspapers training set. The trained correction model is then applied to correct the full test set. CER and WER of the correction results from both models are listed in Table 3. We can find that the Attn-Seq2Seq neural translation model works significantly better than the PCRF when trained on a dataset of the same size. The performance of the Attn-Seq2seq model could be further improved by including more training data or by multi-input decoding for duplicated texts, while the PCRF could only be trained on limited data and is not able to work on multiple inputs. Thus, we choose Attn-Seq2Seq as our error correction model.

4.2.2 Multi-input Attention Combination

We also compare different attention combination strategies on a multi-input decoding task. The results from Table 4 reveal that average attention combination performs best among all the decoding strategies on RDD newspapers and TCP books datasets. It reduces the CER of single input decoding by 41.5% for OCR'd lines in RDD newspapers and 9.76% for TCP books. The comparison between two hierarchical attention combination strategies shows that averaging evidence from each input works better than a weighted summation mechanism. Flat attention combination, which merges all the inputs into a long sequence when computing the strength of each encoder hidden state, obtains the worst performance in terms

Decode	RDD Newspapers				TCP Books			
	CER	LCER	WER	LWER	CER	LCER	WER	LWER
None	0.15149	0.04717	0.37111	0.13799	0.10590	0.07666	0.30549	0.23495
Single	0.07199	0.03300	0.14906	0.06948	0.04508	0.01407	0.11283	0.03392
Flat	0.07238	0.02904*	0.15818	0.06241*	0.05554	0.01727	0.13487	0.04079
Weighted	0.06882*	0.02145*	0.15221	0.05375	0.05516	0.01392*	0.1330	0.03669
Average	**0.04210***	**0.01399 ***	**0.09397**	**0.02863***	**0.04072***	**0.01021***	**0.09786***	**0.02092***

Table 4: Results of correcting lines in the RDD newspapers and TCP books with multiple witnesses when decoding with different strategies using the same supervised model. Attention combination strategies that statistically significantly outperform single-input decoding are highlighted with * ($p < 0.05$, paired-permutation test). Best result for each column is in **bold**.

of both CER and WER.

4.3 Main Results

We now present results on the full training and test sets for the Richmond Daily Dispatch newspapers and Text Creation Partnership books. All results are on the same test set. The multi-input decoding experiments have access to additional witnesses for each line, where available, but fall back to single-input decoding when no additional witnesses are present for a given line.

Table 5 presents the results for our model trained in different training settings as well as the baseline language model reranking (LMR) and majority vote methods. Multiple input decoding performs better than single input decoding for every training setting, and the model trained in supervised mode with multi-input decoding achieves the best performance. The majority vote baseline, which works only on more than two inputs, performs worst on both the TCP books and RDD newspapers. Our proposed unsupervised framework Seq2Seq-Noisy and Seq2Seq-Boots achieves performance comparable with the supervised model via multi-input decoding on the RDD newspaper dataset. The performance of Seq2Seq-Noisy is worse on the TCP Books than the RDD newspapers, since those old books contain the character long **s** [6], which is formerly used where **s** occurred in the middle or at the beginning of a word. These characters are recognized as **f** in all the witnesses because of similar shape. Thus, the model trained on noisy data are unable to correct them into **s**. Nonetheless, by removing the factor of long **s**, i.e., replacing the long **s** in the ground truth with **f**, Seq2Seq-Noisy could achieve a CER of 0.062 for single-input decoding and 0.058 for multi-input decoding on the TCP books. Both Seq2Seq-Syn and Seq2Seq-Boots work better on the RDD newspapers than the TCP books dataset. We conjecture that it is because the synthetic dataset is trained on (modern) newspapers, which are more similar to the nineteenth-century RDD newspapers. The long **s** problem also makes it more difficult for the model trained on synthetic data to work on the TCP books.

[6] https://en.wikipedia.org/wiki/Long_s

4.4 Discussion

In this section, we provide further analysis on different aspects of our method.

Does Corruption Rate Affect Synthetic Training? We first examine how the corruption rate of the synthetic dataset would affect the performance of the correction model. Figure 2 presents the results of single-input correction and multi-input correction tasks on the RDD newspapers and TCP books when trained on synthetic data corrupted with different error rate: 0.9, 0.12, 0.15. For both tasks, the character error rate increases a little bit when the correction model is trained to recover the synthetic date with higher corruption rate. However, the performance is more stable on the RDD newspapers than the TCP books when more errors are introduced.

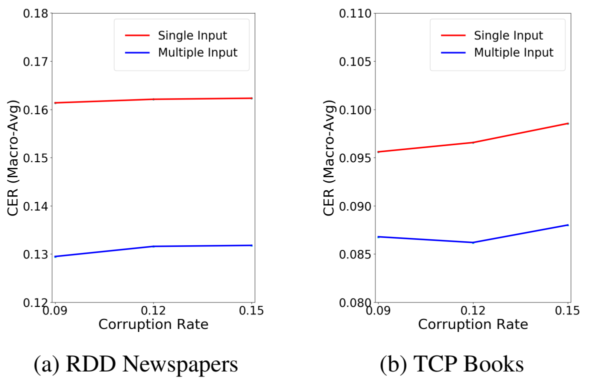
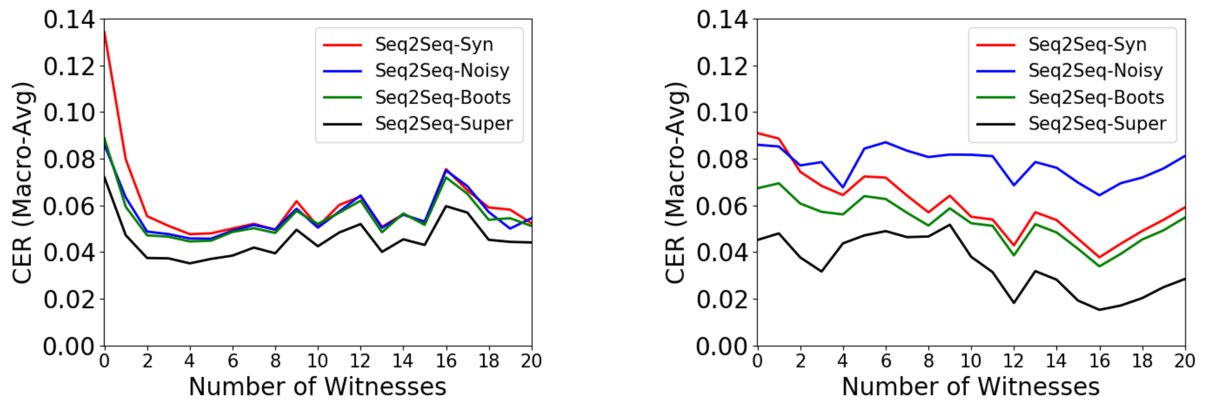


Figure 2: Performance of Seq2Seq-Syn trained on synthetic data with different corruption rates.

Does Number of Witnesses Matter for Multiple-Input Decoding? Here we want to study the impact of the number of witnesses on

Decode	Model	RDD Newspapers				TCP Books			
		CER	LCER	WER	LWER	CER	LCER	WER	LWER
	None	0.18133	0.13552	0.41780	0.31544	0.10670	0.08800	0.31734	0.27227
Single	*Seq2Seq-Super*	*0.09044*	*0.04469*	*0.17812*	*0.09063*	*0.04944*	*0.01498*	*0.12186*	*0.03500*
	Seq2Seq-Noisy	0.10524	0.05565	0.20600	0.11416	0.08704	0.05889	0.25994	0.15725
	Seq2Seq-Syn	0.16136	0.11986	0.35802	0.26547	0.09551	0.06160	0.27845	0.18221
	Seq2Seq-Boots	0.11037	0.06149	0.22750	0.13123	0.07196	0.03684	0.21711	0.11233
Multi	LMR	0.15507	0.13552	0.34653	0.31544	0.10862	0.08800	0.33983	0.27227
	Majority Vote	0.16285	0.13552	0.40063	0.31544	0.11096	0.08800	0.34151	0.27227
	Seq2Seq-Super	*0.07731*	*0.03634*	*0.15393*	*0.07269*	*0.04668*	*0.01252*	*0.11236*	*0.02667*
	Seq2Seq-Noisy	**0.09203***	**0.04554***	**0.17940**	**0.09269**	0.08317	0.05588	0.24824	0.14885
	Seq2Seq-Syn	0.12948	0.09112	0.28901	0.19977	0.08506	0.05002	0.24942	0.15169
	Seq2Seq-Boots	0.09435	0.04976	0.19681	0.10604	**0.06824***	**0.03343***	**0.20325***	**0.09995***

Table 5: Results from model trained under different settings on single-input decoding and multiple-input decoding for both the RDD newspapers and TCP books. All training is unsupervised except for supervised results in *italics*. Unsupervised training settings with multi-input decoding that are significantly better than other unsupervised counterparts are highlighted with * ($p < 0.05$, paired-permutation test). Best result among **unsupervised** training in each column is in **bold**.

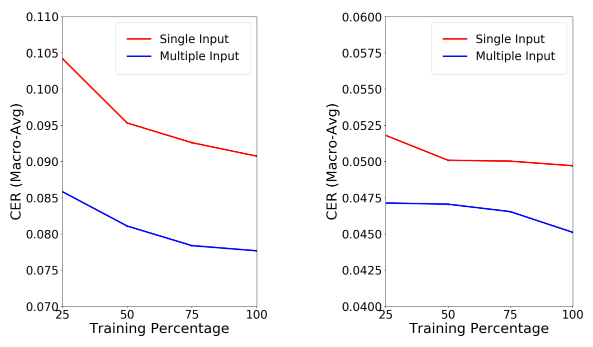


(a) RDD Newspapers (b) TCP Books

Figure 3: Performance of different models on multiple decoding of lines with different number of witnesses.

the performance of multiple-input decoding. The test set is divided into subgroups with varying size according to their number of witnesses. Figure 3 presents the performance of multi-input correction on subgroups with different number of witnesses. We can see that supervised training achieves the best performance on each subgroup for both datasets. On the RDD newspapers, the performance of each training setting is significantly improved when the number of witnesses increases from 0 to 2, then the error rate tends to be flat when more witnesses are observed. For the TCP books, the character error rate for both Seq2Seq-Syn and Seq2Seq-Boots decreases with small fluctuation when the number of witnesses increases. Seq2Seq-Noisy performs the worst almost on all subgroups on the TCP books since all the witnesses suffers from the long **s** problem.

Can More Training Data Benefit Learning? Figure 4 shows the test results for our correction model trained on datasets of different size. As the size of the training set increases, the CER of our model decreases consistently for both single and multiple input correction on the RDD newspapers. However, the performance curve of correction model on TCP books dataset is flatter since it is larger overall than RDD newspapers.



(a) RDD Newspapers (b) TCP Books

Figure 4: Performance of the supervised correction model trained on different proportions of the RDD newspapers and TCP books dataset.

5 Related Work

Multi-Input OCR Correction. Ensemble methods have been shown to be effective in OCR post-correction by combining OCR output from multiple scans of the same document (Lopresti and Zhou, 1997; Klein and Kopel, 2002; Cecotti and Belaïd, 2005; Lund et al., 2013). Existing methods aim at generating consensus results by aligning multiple inputs, followed by supervised methods such as classification (Boschetti et al., 2009; Lund et al., 2011; Al Azawi et al., 2015), or unsupervised methods such as dictionary-based selection (Lund and Ringger, 2009) and voting (Wemhoener et al., 2013; Xu and Smith, 2017). While supervised ensemble methods require human annotation for training, unsupervised selection methods work only when the correct word or character exists in one of the inputs. Furthermore, those methods could not correct single inputs.

Multi-Input Attention. Multi-input attention has already been explored in tasks such as machine translation (Zoph and Knight, 2016; Libovický and Helcl, 2017) and summarization (Wang and Ling, 2016). Wang and Ling (2016) propose to concatenate multiple inputs to generate a summary; this flat attention combination model might be affected by the order of input sequences. Zoph and Knight (2016) aims at developing a multi-source translation model on a trilingual corpus where the encoder for each language is combined to pass to the decoder; however, it requires the same number of inputs at training and decoding time since the parameters depend on the number of inputs. Libovický and Helcl (2017) explore different attention combination strategies for multiple information sources such as image and text. In contrast, our method does not require multiple inputs for training, and the attention combination strategies are used to integrate multiple inputs when decoding.

6 Conclusions

We have proposed an unsupervised framework for OCR error correction, which can handle both single-input and multi-input correction tasks. An attention-based sequence-to-sequence model is applied for single-input correction, based on which a strategy of multi-input attention combination is designed to correct multiple input sequences simultaneously. The proposed strategy naturally incorporates aligning, correcting, and voting among multiple sequences, and is thus effective in improving the correction performance for corpora containing duplicated text. We propose two ways of training the correction model without human annotation by exploiting the duplication in the corpus. Experimental results on historical books and newspapers show that these unsupervised approaches significantly improve OCR accuracy and, when multiple inputs are available, achieve performance comparable to supervised methods.

Acknowledgements

This work was supported by NIH grant 2R01DC009834-06A1, the Andrew W. Mellon Foundation's Scholarly Communications and Information Technology program, and a Google Faculty Research Award. Any views, findings, conclusions, or recommendations expressed do not necessarily reflect those of the NIH, Mellon, or Google. We would like to thank the anonymous reviewers for their valuable comments.

References

Mayce Al Azawi, Marcus Liwicki, and Thomas M. Breuel. 2015. Combination of multiple aligned recognition outputs using WFST and LSTM. In *ICDAR*, pages 31–35.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Federico Boschetti, Matteo Romanello, Alison Babeu, David Bamman, and Gregory Crane. 2009. Improving OCR accuracy for classical critical editions. In *JCDL*, pages 156–167.

Hubert Cecotti and Abdel Belaïd. 2005. Hybrid OCR combination approach complemented by a specialized ICR applied on ancient documents. In *ICDAR*, pages 1045–1049.

Guillaume Chiron, Antoine Doucet, Mickael Coustaty, Muriel Visani, and Jean-Philippe Moreux. 2017. Impact of OCR errors on the use of digital libraries: Towards a better access to information. In *JCDL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *IJCAI*.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proc. Workshop on Statistical Machine Translation*, pages 187–197.

Monika Henzinger. 2006. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *SIGIR*, pages 284–291.

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

Rose Holley. 2009. Many hands make light work: Public collaborative OCR text correction in Australian historic newspapers. Technical report, National Library of Australia.

Shmuel T. Klein and Miri Kopel. 2002. A voting system for automatic OCR correction. In *Proc. SIGIR Workshop on Information Retrieval and OCR*.

Okan Kolak, William Byrne, and Philip Resnik. 2003. A generative probabilistic OCR model for NLP applications. In *HLT-NAACL*, pages 55–62.

Okan Kolak and Philip Resnik. 2002. OCR error correction using a noisy channel model. In *HLT*, pages 257–262.

Adenike M. Lam-Adesina and Gareth J. F. Jones. 2006. Examining and improving the effectiveness of relevance feedback for retrieval of scanned text documents. *Information Processing & Management*, 42(3):633–649.

Jindřich Libovický and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *ACL*.

Daniel Lopresti and Jiangying Zhou. 1997. Using consensus sequence voting to correct OCR errors. *Computer Vision and Image Understanding*, 67(1):39–47.

William B. Lund, Douglas J. Kennard, and Eric K. Ringger. 2013. Combining multiple thresholding binarization values to improve OCR output. In *Proc. Document Recognition and Retrieval (DRR)*.

William B. Lund and Eric K. Ringger. 2009. Improving optical character recognition through efficient multiple system alignment. In *JCDL*, pages 231–240.

William B Lund, Eric K Ringger, and Daniel D Walker. 2014. How well does multiple OCR error correction generalize? In *Proc. Document Recognition and Retrieval (DRR)*.

William B. Lund, Daniel D. Walker, and Eric K. Ringger. 2011. Progressive alignment and discriminative error correction for multiple OCR engines. In *ICDAR*, pages 764–768.

Michael J. Paul and Jason Eisner. 2012. Implicitly intersecting weighted automata using dual decomposition. In *NAACL*, pages 232–242.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium*, 6(12):e26752.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *COLING*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

David A. Smith, Ryan Cordell, Elizabeth Maddock Dillon, Nick Stramp, and John Wilkerson. 2014. Detecting and modeling local text reuse. In *JCDL*.

T. F. Smith and M. S. Waterman. 1981. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Kazem Taghva, Julie Borsack, and Allen Condit. 1996. Effects of ocr errors on ranking and feedback using the vector space model. *Information Processing & Management*, 32(3):317–327.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *NAACL*, pages 47–57.

David Wemhoener, Ismet Zeki Yalniz, and R. Manmatha. 2013. Creating an improved version using noisy OCR from multiple editions. In *ICDAR*, pages 160–164.

John Wilkerson, David A. Smith, and Nick Stramp. 2015. Tracing the flow of policy ideas on legislatures: A text reuse approach. *American Journal of Political Science*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Shaobin Xu and David A. Smith. 2017. Retrieving and combining repeated passages to improve OCR. In *JCDL*.

Takafumi Yamazoe, Minoru Etoh, Takeshi Yoshimura, and Kousuke Tsujino. 2011. Hypothesis preservation approach to scene text recognition with weighted finite-state transducer. In *ICDAR*, pages 359–363.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *NAACL*.

Building Language Models for Text with Named Entities

Md Rizwan Parvez
University of California Los Angeles
rizwan@cs.ucla.edu

Saikat Chakraborty
University of Virginia
saikatc@virginia.edu

Baishakhi Ray
Columbia University
rayb@cs.columbia.edu

Kai-Wei Chang
University of California Los Angeles
kwchang@cs.ucla.edu

Abstract

Text in many domains involves a significant amount of named entities. Predicting the entity names is often challenging for a language model as they appear less frequent on the training corpus. In this paper, we propose a novel and effective approach to building a discriminative language model which can learn the entity names by leveraging their entity type information. We also introduce two benchmark datasets based on recipes and Java programming codes, on which we evaluate the proposed model. Experimental results show that our model achieves 52.2% better perplexity in recipe generation and 22.06% on code generation than the state-of-the-art language models.

1 Introduction

Language model is a fundamental component in Natural Language Processing (NLP) and it supports various applications, including document generation (Wiseman et al., 2017), text auto-completion (Arnold et al., 2017), spelling correction (Brill and Moore, 2000), and many others. Recently, language models are also successfully used to generate software source code written in programming languages like Java, C, etc. (Hindle et al., 2016; Yin and Neubig, 2017; Hellendoorn and Devanbu, 2017; Rabinovich et al., 2017). These models have improved the language generation tasks to a great extent, e.g., (Mikolov et al., 2010; Galley et al., 2015). However, while generating text or code with a large number of named entities (e.g., different variable names in source code), these models often fail to predict the entity names properly due to their wide variations. For instance, consider building a language model for generating recipes. There are numerous similar, yet slightly different cooking ingredients (e.g., *olive oil, canola oil, grape oil, etc.*—all are different varieties of oil). Such diverse vocabularies of the ingredient names hinder the language model from predicting them properly.

To address this problem, we propose a novel language model for texts with many entity names. Our model learns the probability distribution over all the candidate words by leveraging the entity type information. For example, oil is the *type* for named entities like olive oil, canola oil, grape oil, etc.[1] Such type information is even more prevalent for source code corpus written in statically typed programming languages (Bruce, 1993), since all the variables are by construct associated with types like integer, float, string, etc.

Our model exploits such deterministic type information of the named entities and learns the probability distribution over the candidate words by decomposing it into two sub-components: (i) *Type Model.* Instead of distinguishing the individual names of the same type of entities, we first consider all of them equal and represent them by their type information. This reduces the vocab size to a great extent and enables to predict the type of each entity more accurately. (ii) *Entity Composite Model.* Using the entity type as a prior, we learn the conditional probability distribution of the actual entity names at inference time. We depict our model in Fig. 1.

To evaluate our model, we create two benchmark datasets that involve many named entities. One is a cooking recipe corpus[2] where each recipe contains a number of ingredients which are cate-

[1] Entity type information is often referred as category information or group information. In many applications, such information can be easily obtained by an ontology or by a pre-constructed entity table.

[2] Data is crawled from http://www.ffts.com/recipes.htm.

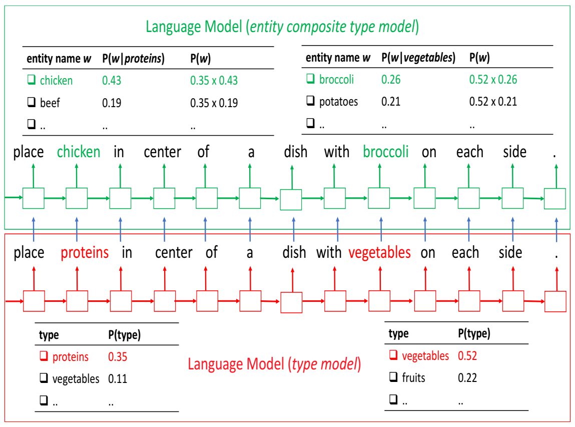

Language Model (entity composite type model)
entity name w | P(w|proteins) | P(w)
chicken 0.43 0.35 x 0.43
beef 0.19 0.35 x 0.19
entity name w | P(w|vegetables) | P(w)
broccoli 0.26 0.52 x 0.26
potatoes 0.21 0.52 x 0.21
place chicken in center of a dish with broccoli on each side .
place proteins in center of a dish with vegetables on each side .
type | P(type)
proteins 0.35
vegetables 0.11
Language Model (type model)
type | P(type)
vegetables 0.52
fruits 0.22

Figure 1: **An example illustrates the proposed model. For a given context (i.e., types of context words as input), the *type model* (in bottom red block) generates the type of the next word (i.e., the probability of the type of the next word as output). Further, for a given context *and* type of each candidate (i.e., context words, corresponding types of the context words, and type of the next word generated by the *type model* as input), the *entity composite model* (in upper green block) predicts the next word (actual entity name) by estimating the conditional probability of the next word as output. The proposed approach conducts joint inference over both models to leverage type information for generating text.**

gorized into 8 super-ingredients (i.e., type); e.g., “proteins”, “vegetables”, “fruits”, “seasonings”, “grains”, etc. Our second dataset comprises a source code corpus of 500 open-source Android projects collected from GitHub. We use an Abstract Syntax Tree (AST) (Parsons, 1992) based approach to collect the type information of the code identifiers.

Our experiments show that although state-of-the-art language models are, in general, good to learn the frequent words with enough training instances, they perform poorly on the entity names. A simple addition of type information as an extra feature to a neural network does not guarantee to improve the performance because more features may overfit or need more model parameters on the same data. In contrast, our proposed method significantly outperforms state-of-the-art neural network based language models and also the models with type information added as an extra feature.

Overall, followings are our contributions:

- We analyze two benchmark language corpora where each consists of a reasonable number of entity names. While we leverage an existing corpus for recipe, we curated the code corpus. For both datasets, we created auxiliary corpora with entity type information. All the code and datasets are released.[3]
- We design a language model for text consisting of many entity names. The model learns to mention entities names by leveraging the entity type information.
- We evaluate our model on our benchmark datasets and establish a new baseline performance which significantly outperforms state-of-the-art language models.

2 Related Work and Background

Class Based Language Models. Building language models by leveraging the deterministic or probabilistic class properties of the words (a.k.a, class-based language models) is an old idea (Brown et al., 1992; Goodman, 2001). However, the objective of our model is different from the existing class-based language models. The key differences are two-folds: 1) Most existing class-based language models (Brown et al., 1992; Pereira et al., 1993; Niesler et al., 1998; Baker and McCallum, 1998; Goodman, 2001; Maltese et al., 2001) are generative n-gram models whereas ours is a discriminative language model based on neural networks. The modeling principle and assumptions are very different. For example, we cannot calculate the conditional probability by statistical occurrence counting as these papers did. 2) Our approaches consider building two models and perform joint inference which makes our framework general and easy to extend. In Section 4, we demonstrate that our model can be easily incorporated with the state-of-art language model. The closest work in this line is hierarchical neural language models (Morin and Bengio, 2005), which model language with word clusters. However, their approaches do not focus on dealing with named entities as our model does. A recent work (Ji et al., 2017) studied the problem of building up a dynamic representation of named entity by updating the representation for every contextualized mention of that entity. Nonetheless, their approach does not deal with the sparsity issue and their goal is different from ours.

Language Models for Named Entities. In some generation tasks, recently developed language models address the problem of predict-

[3]https://github.com/uclanlp/NamedEntityLanguageModel

ing entity names by copying/matching the entity names from the reference corpus. For example, Vinyals et al. (2015) calculates the conditional probability of discrete output token sequence corresponding to positions in an input sequence. Gu et al. (2016) develops a seq2seq alignment mechanism which directly copies entity names or long phrases from the input sequence. Wiseman et al. (2017) generates document from structured table like basketball statistics using copy and reconstruction method as well. Another related code generation model (Yin and Neubig, 2017) parses natural language descriptions into source code considering the grammar and syntax in the target programming language (e.g., Python). Kiddon et al. (2016) generates recipe for a given goal, and agenda by making use of items on the agenda. While generating the recipe it continuously monitors the agenda coverage and focus on increasing it. All of them are sequence-to-sequence learning or end-to-end systems which differ from our general purpose (free form) language generation task (e.g., text auto-completion, spelling correction).

Code Generation. The way developers write codes is not only just writing a bunch of instructions to run a machine, but also a form of communication to convey their thought. As observed by Donald E. Knuth (Knuth, 1992), *"The practitioner of literate programming can be regarded as an essayist, whose main concern is exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what such variable means."* Such comprehensible software corpora show surprising regularity (Ray et al., 2015; Gabel and Su, 2010) that is quite similar to the statistical properties of natural language corpora and thus, amenable to large-scale statistical analysis (Hindle et al., 2012). (Allamanis et al., 2017) presented a detailed survey.

Although similar, source code has some unique properties that differentiate it from natural language. For example, source code often shows more regularities in local context due to common development practices like copy-pasting (Gharehyazie et al., 2017; Kim et al., 2005). This property is successfully captured by cache based language models (Hellendoorn and Devanbu, 2017; Tu et al., 2014). Code is also less ambiguous than natural language so that it can be interpreted by a compiler. The constraints for generating correct code is implemented by combining language model and program analysis technique (Raychev et al., 2014). Moreover, code contains open vocabulary—developers can coin new variable names without changing the semantics of the programs. Our model aims to addresses this property by leveraging variable types and scope.

LSTM Language Model. In this paper, we use LSTM language model as a running example to describe our approach. Our language model uses the LSTM cells to generate latent states for a given context which captures the necessary features from the text. At the output layer of our model, we use Softmax probability distribution to predict the next word based on the latent state. Merity et al. (2017) is a LSTM-based language model which achieves the state-of-the-art performance on Penn Treebank (PTB) and WikiText-2 (WT2) datasets. To build our recipe language model we use this as a blackbox and for our code generation task we use the simple LSTM model both in forward and backward direction. A forward directional LSTM starts from the beginning of a sentence and goes from left to right sequentially until the sentence ends, and vice versa. However, our approach is general and can be applied with other types of language models.

3 A Probabilistic Model for Text with Named Entities

In this section, we present our approach to build a language model for text with name entities. Given previous context $\bar{w} = \{w_1, w_2, .., w_{t-1}\}$, the goal of a language model is to predict the probability of next word $P(w_t|\bar{w})$ at time step t, where $w_t \in V^{text}$ and V^{text} is a fixed vocabulary set. Because the size of vocabulary for named entities is large and named entities often occur less frequently in the training corpus, the language model cannot generate these named entities accurately. For example, in our recipe test corpus the word "apple" occurs only 720 times whereas any kind of "fruits" occur 27,726 times. Existing approaches often either only generate common named entities or omit entities when generating text (Jozefowicz et al., 2016).

To overcome this challenge, we propose to leverage the entity type information when modeling text with many entities. We assume each entity is associated with an entity type in a finite set of categories $S = \{s_1, s_2, .., s_i, .., s_k\}$. Given a word w, $s(w)$ reflects its entity type. If the word is a named entity, then we denote $s(w) \in S$; otherwise the type function returns the words itself (i.e, $s(w) = w$). To simplify the notations, we use $s(w) \notin S$ to represent the case where the word is not an entity. The entity type information given by $s(w)$ is an auxiliary information that we can use to improve the language model. We use $s(\bar{w})$ to represent the entity type information of all the words in context $\bar{w}$ and use w to represent the current word w_t. Below, we show that a language model for text with typed information can be decomposed into the following two models: 1) a *type model* θ_t that predicts the entity type of the next word and 2) an *entity composite model* θ_v that predicts the next word based on a given entity type.

Our goal is to model the probability of next word w given previous context $\bar{w}$:

$$P\left(w|\bar{w};\theta_t,\theta_v\right), \tag{1}$$

where θ_t and θ_v are the parameters of the two aforementioned models. As we assume the typed information is given on the data, Eq. (1) is equivalent to

$$P\left(w, s(w)|\bar{w}, s(\bar{w});\theta_t,\theta_v\right). \tag{2}$$

A word can be either a named entity or not; therefore, we consider the following two cases.

Case 1: next word is a named entity. In this case, Eq. (2) can be rewritten as

$$\begin{aligned} &P\left(s(w) = s|\bar{w}, s(\bar{w});\theta_t,\theta_v\right) \times \\ &P\left(w|\bar{w}, s(\bar{w}), s(w) = s;\theta_v,\theta_t\right) \end{aligned} \tag{3}$$

based on the rules of conditional probability.

We assume the type of the next token $s(w)$ can be predicted by a model θ_t using information of $s(\bar{w})$, and we can approximate the first term in Eq. (3)

$$P(s(w)|\bar{w}, s(\bar{w});\theta_t,\theta_v) \approx P(s(w)|s(\bar{w}),\theta_t) \tag{4}$$

Similarly, we can make a modeling assumption to simplify the second term as

$$\begin{aligned} &P(w|\bar{w}, s(\bar{w}), s(w),\theta_v,\theta_t) \\ &\approx P(w|\bar{w}, s(\bar{w}), s(w),\theta_v). \end{aligned} \tag{5}$$

Case 2: next word is not a named entity. In this case, we can rewrite Eq. (2) to be

$$\begin{aligned} &P\left(s(w) \notin S|\bar{w}, s(\bar{w}),\theta_t\right) \times \\ &P\left(w|\bar{w}, s(\bar{w}), s(w) \notin S,\theta_v\right). \end{aligned} \tag{6}$$

The first term in Eq. (6) can be modeled by

$$1 - \sum_{s\in S} P(s(w) = s|s(\bar{w}),\theta_t),$$

which can be computed by the *type* model[4]. The second term can be again approximated by (5) and further estimated by an *entity composition model*.

Typed Language Model. Combine the aforementioned equations, the proposed language model estimates $P(w|\bar{w};\theta_t,\theta_v)$ by

$$P(w|\bar{w}, s(\bar{w}), s(w),\theta_v)\times \\ \begin{cases} P(s(w)|s(\bar{w}),\theta_t) & \text{if } s(w) \in S \\ (1-\sum_{s\in S} P(s(w){=}s|s(\bar{w}),\theta_t)) & \text{if } s(w) \notin S \end{cases} \tag{7}$$

The first term can be estimated by an *entity composite model* and the second term can be estimated by a *type model* as discussed below.

3.1 Type model

The *type model* θ_t estimates the probability of $P(s(w)|s(\bar{w}),\theta_t)$. It can be viewed as a language model builds on a corpus with all entities replaced by their type. That is, assume the training corpus consists of $x = \{w_1, w_2, .., w_n\}$. Using the type information provided in the auxiliary source, we can replace each word w with their corresponding type $s(w)$ and generate a corpus of $\mathcal{T} = \{s(w_i), s(w_2), .., s(w_n)\}$. Note that if w_i is not an named entity (i.e., $s(w) \notin S$), $s(w) = w$ and the vocabulary on $\mathcal{T}$ is $V^{text} \cup S$.[5] Any language modeling technique can be used in modeling the *type model* on the modified corpus $\mathcal{T}$. In this paper, we use the state-of-the-art model for each individual task. The details will be discussed in the experiment section.

3.2 Entity Composite Model

The *entity composite model* predicts the next word based on modeling the conditional probability $P(w|\bar{w}, s(\bar{w}), s(w),\theta_v)$, which can be derived by

$$\frac{P(w|\bar{w}, s(\bar{w});\theta_v)}{\sum_{w_s\in\Omega(s(w))} P(w_s|\bar{w}, s(\bar{w});\theta_v)}, \tag{8}$$

[4]Empirically for the non-entity words, $\sum_{s\in S} P(s(w) = s|s(\bar{w}) \approx 0$

[5]In a preliminary experiment, we consider putting all words with $s(w) \notin S$ in a category "N/A". However, because most words on the training corpus are not named entities, the type "N/A" dominates others and hinder the *type model* to make accurate predictions.

where $\Omega(s(w))$ is the set of words of the same type with w.

To model the types of context word $s(\bar{w})$ in $P(w|\bar{w}, s(\bar{w}); \theta_v)$, we consider learning a type embedding along with the word embedding by augmenting each word vector with a type vector when learning the underlying word representation. Specifically, we represent each word w as a vector of $[v_w(w)^T; v_t(s(w))^T]^T$, where $v_w(\cdot)$ and $v_t(\cdot)$ are the word vectors and type vectors learned by the model from the training corpus, respectively. Finally, to estimate Eq. (8) using θ_v, when computing the Softmax layer, we normalize over only words in $\Omega(s(w))$. In this way, the conditional probability $P(w|\bar{w}, s(\bar{w}), s(w), \theta_v)$ can be derived.

3.3 Training and Inference Strategies

We learn model parameters θ_t and θ_v independently by training two language models *type model* and *entity composite model* respectively. Given the context of type, *type model* predicts the type of the next word. Given the context and the type information of the all candidate words, *entity composite model* predicts the conditional actual word (e.g., entity name) as depicted in Fig 1. At inference time the generated probabilities from these two models are combined according to conditional probability (i.e., Eq. (7)) which gives the final probability distribution over all candidate words[6].

Our proposed model is flexible to any language model, training strategy, and optimization. As per our experiments, we use ADAM stochastic mini-batch optimization (Kingma and Ba, 2014). In Algorithm 1, we summarize the language generation procedure.

Algorithm 1: **Language Generation**

```
Input: Language corpus
       $\mathcal{X} = \{w_1, w_2, .., w_n\}$, type $s(w)$ of
       the words, integer number $m$.
Output: $\theta_t$, $\theta_v$, $\{W_1, W_2, .., W_m\}$
1  Training Phase:
2  Generate $\mathcal{T}$ = { $s(w_1), s(w_2), .., s(w_n)\}$
3  Train type model $\theta_t$ on $\mathcal{T}$
4  Train entity composite model $\theta_v$ on $\mathcal{X}$ using
     $[w_i; s(w_i)]$ as input
5  Test Phase (Generation Phase):
6  for $i = 1$ to $m$ do
7      for $w \in V^{text}$ do
8          Compute $P(s(w)|s(\bar{w}), \theta_t)$
9          Compute $P(w|\bar{w}, s(\bar{w}), s(w), \theta_v)$
10         Compute $P(w|\bar{w}; \theta_t, \theta_v)$ using Eq.(7)
11     end
12     $W_i \leftarrow \text{argmax}_w P(w|\bar{w}; \theta_t, \theta_v)$
13 end
```

4 Experiments

We evaluate our proposed model on two different language generation tasks where there exist a lot of entity names in the text. In this paper, we release all the codes and datasets. The first task is recipe generation. For this task, we analyze a cooking recipe corpus. Each instance in this corpus is an individual recipe and consists of many ingredients'. Our second task is code generation. We construct a Java code corpus where each instance is a Java method (i.e., function). These tasks are challenging because they have the abundance of entity names and state-of-the-art language models fail to predict them properly as a result of insufficient training observations. Although in this paper, we manually annotate the types of the recipe ingredients, in other applications it can be acquired automatically. For example: in our second task of code generation, the types are found using Eclipse JDT framework. In general, using DBpedia ontology (e.g., "Berlin" has an ontology "Location"), Wordnet hierarchy (e.g., "Dog" is an "Animal"), role in sports (e.g., "Messi" plays in "Forward"; also available in DBpedia[7]), Thesaurus (e.g., "renal cortex", "renal pelvis", "renal vein", all are related to "kidney"), Medscape (e.g., "Advil" and "Motrin" are actually "Ibuprofen"), we can get the necessary type information. As for the applications where the entity types cannot be extracted automatically by these frameworks (e.g., recipe ingredients), although there is no exact strategy, any reasonable design can work. Heuristically, while annotating manually in our first task, we choose the total number of types in such a way that each type has somewhat balanced (similar) size.

We use the same dimensional word embedding

[6]While calculating the final probability distribution over all candidate words, with our joint inference schema, a strong state-of-art language model, without the type information, itself can work sufficiently well and replace the *entity composite model*. Our experiments using (Merity et al., 2017) in Section 4.1 validate this claim.

[7] http://dbpedia.org/page/Lionel_Messi

(400 for recipe corpus, 300 for code corpus) to represent both of the entity name (e.g., "apple") and their entity type (e.g., "fruits") in all the models. Note that in our approach, the *type model* only replaces named entities with entity type when it generates next word. If next word is not a named entity, it will behave like a regular language model. Therefore, we set both models with the same dimensionality. Accordingly, for the *entity composite model* which takes the concatenation of the entity name and the entity type, the concatenated input dimension is 800 and 600 respectively for recipe and code corpora.

4.1 Recipe Generation

Recipe Corpus Pre-processing: Our recipe corpus collection is inspired by (Kiddon et al., 2016). We crawl the recipes from "Now Youre Cooking! Recipe Software" [8]. Among more than 150,000 recipes in this dataset, we select similarly structured/formatted (e.g, title, blank line then ingredient lists followed by a recipe) 95,786 recipes. We remove all the irrelevant information (e.g., author's name, data source) and keep only two information: ingredients and recipes. We set aside the randomly selected 20% of the recipes for testing and from the rest, we keep randomly selected 80% for the training and 20% for the development. Similar to (Kiddon et al., 2016), we preprocess the dataset and filter out the numerical values, special tokens, punctuation, and symbols.[9] Quantitatively, the data we filter out is negligible; in terms of words, we keep 9,994,365 words out of 10,231,106 and the number of filter out words is around $\sim$2%. We release both of the raw and cleaned data for future challenges. As the ingredients are the entity names in our dataset, we process it separately to get the type information.

Retrieving Ingredient Type: As per our type model, for each word w, we require its type $s(w)$. We only consider ingredient type for our experiment. First, we tokenize the ingredients and consider each word as an ingredient. We manually classify the ingredients into 8 super-ingredients: "fruits", "proteins", "sides", "seasonings", "vegetables", "dairy", "drinks", and "grains". Sometimes, ingredients are expressed using multiple words; for such ingredient phrase, we classify each word in the same group (e.g., for "boneless beef" both "boneless" and "beef" are classified as "proteins"). We classify the most frequent 1,224 unique ingredients, [10] which cover 944,753 out of 1,241,195 mentions (top 76%) in terms of frequency of the ingredients. In our experiments, we omit the remainder 14,881 unique ingredients which are less frequent and include some misspelled words. The number of unique ingredients in the 8 super ingredients is 110, 316, 140, 180, 156, 80, 84, and 158 respectively. We prepare the modified *type corpus* by replacing each actual ingredient's name w in the original recipe corpus by the type (i.e., super ingredients $s(w)$) to train the *type model*.

Recipe Statistics: In our corpus, the total number of distinct words in vocabulary is 52,468; number of unique ingredients (considering splitting phrasal ingredients also) is 16,105; number of tokens is 8,716,664. In number of instances train/dev/test splits are 61,302/15,326/19,158. The average instance size of a meaningful recipe is 91 on the corpus.

Configuration: We consider the state-of-the art LSTM-based language model proposed in (Merity et al., 2017) as the basic component for building the *type model*, and *entity composite model*. We use 400 dimensional word embedding as described in Section 4. We train the embedding for our dataset. We use a minibatch of 20 instances while training and back-propagation through time value is set to 70. Inside of this (Merity et al., 2017) language model, it uses 3 layered LSTM architecture where the hidden layers are 1150 dimensional and has its own optimization and regularization mechanism. All the experiments are done using PyTorch and Python 3.5.

Baselines: Our first baseline is ASGD Weight-Dropped LSTM (AWD_LSTM) (Merity et al., 2017), which we also use to train our models (see 'Configuration' in 4.1). This model achieves the state-of-the-art performance on benchmark Penn Treebank (PTB), and WikiText-2 (WT2) language corpus. Our second baseline is the same language model (AWD_LSTM) with the type information added as an additional feature (i.e., same as *entity composite model*).

[8]http://www.ffts.com/recipes.htm

[9]For example, in our crawled raw dataset, we find that some recipes have lines like "===MMMMM===" which are totally irrelevant to our task. For the words with numerical values like "100 ml", we only remove the "100" and keep the "ml" since our focus is not to predict the exact number.

[10]We consider both singular and plural forms. The number of singular formed annotated ingredients are 797.

Model	Dataset (Recipe Corpus)	Vocabulary Size	Perplexity
AWD_LSTM	original	52,472	20.23
AWD_LSTM *type model*	modified type	51,675	17.62
AWD_LSTM with type feature	original	52,472	18.23
our model	original	52,472	**9.67**

Table 1: **Comparing the performance of recipe generation task. All the results are on the test set of the corresponding corpus. AWD_LSTM (*type model*) is our *type model* implemented with the baseline language model AWD_LSTM (Merity et al., 2017). Our second baseline is the same language model (AWD_LSTM) with the type information added as an additional feature for each word.**

Results of Recipe Generation. We compare our model with the baselines using *perplexity* metric—lower perplexity means the better prediction. Table 1 summarizes the result. The 3^{rd} row shows that adding type as a simple feature does not guarantee a significant performance improvement while our proposed method significantly outperforms both baselines and achieves 52.2% improvement with respect to baseline in terms of perplexity. To illustrate more, we provide an example snippet of our test corpus: "place onion and ginger inside chicken . allow chicken to marinate for hour .". Here, for the last mention of the word "chicken", the standard language model assigns probability 0.23 to this word, while ours assigns probability 0.81.

4.2 Code Generation

Code Corpus Pre-processing. We crawl 500 Android open source projects from GitHub[11]. GitHub is the largest open source software forge where anyone can contribute (Ray et al., 2014). Thus, GitHub also contains trivial projects like student projects, etc. In our case, we want to study the coding practices of practitioners so that our model can learn to generate quality code. To ensure this, we choose only those Android projects from GitHub that are also present in Google Play Store[12]. We download the source code of these projects from GitHub using an off the shelf tool GitcProc (Casalnuovo et al., 2017).

Since real software continuously evolves to cater new requirements or bug fixes, to make our modeling task more realistic, we further study different project versions. We partition the codebase of a project into multiple versions based on the code commit history retrieved from GitHub; each version is taken at an interval of 6 months. For example, anything committed within the first six months of a project will be in the first version, and so on. We then build our code suggestion task mimicking how a developer develops code in an evolving software—based on the past project history, developers add new code. To implement that we train our language model on past project versions and test it on the most recent version, at method granularity. However, it is quite difficult for any language model to generate a method from the scratch if the method is so new that even the method signature (i.e., method declaration statement consisting of method name and parameters) is not known. Thus, during testing, we only focus on the methods that the model has seen before but some new tokens are added to it. This is similar to the task when a developer edits a method to implement a new feature or bug-fix.

Since we focus on generating the code for every method, we train/test the code prediction task at method level—each method is similar to a sentence and each token in the method is equivalent to a word. Thus, we ignore the code outside the method scope like global variables, class declarations, etc. We further clean our dataset by removing user-defined "String" tokens as they increase the diversity of the vocabularies significantly, although having the same type. For example, the word sequences "Hello World!" and "Good wishes for ACL2018!!" have the same type `java.lang.String.VAR`.

Retrieving Token Type: For every token w in a method, we extract its type information $s(w)$. A token type can be Java built-in data types (e.g., *int, double, float, boolean* etc.,) or user or framework defined classes (e.g., *java.lang.String, io.segment.android.flush.FlushThread* etc.). We extract such type information for each token by parsing the Abstract Syntax Tree (AST) of the source code[13]. We extract the AST type information of each token using Eclipse JDT framework[14]. Note that, language keywords like `for`, `if`, etc. are not associated with any type. Next, we prepare the type corpus by replacing the

[11]https://github.com

[12]https://play.google.com/store?hl=en

[13]AST represents source code as a tree by capturing its abstract syntactic structure, where each node represents a construct in the source code.

[14]https://www.eclipse.org/jdt/

variable names with corresponding type information. For instance, if variable `var` is of type *java.lang.Integer*, in the type corpus we replace *var* by *java.lang.Integer*. Since multiple packages might contain classes of the same name, we retain the fully qualified name for each type[15].

Code Corpus Statistics: In our corpus, the total number of distinct words in vocabulary is 38,297; the number of unique AST type (including all user-defined classes) is 14,177; the number of tokens is 1,440,993. The number of instances used for train and testing is 26,600 and 3,546. Among these 38,297 vocabulary words, 37,411 are seen at training time while the rests are new.

Configuration: To train both *type model* and *entity composite model*, we use forward and backward LSTM (See Section 2) and combine them at the inference/generation time. We train 300-dimensional word embedding for each token as described in Section 4 initialized by GLOVE (Pennington et al., 2014). Our LSTM is single layered and the hidden size is 300. We implement our model on using PyTorch and Python 3.5. Our training corpus size 26,600 and we do not split it further into smaller train and development set; rather we use them all to train for one single epoch and record the result on the test set.

Baselines: Our first baseline is standard LSTM language model which we also use to train our modules (see ‘Configuration’ in 4.2). Similar to our second baseline for recipe generation we also consider LSTM with the type information added as more features[16] as our another baseline. We further compare our model with state-of-the-art token-based language model for source code SLP-Core (Hellendoorn and Devanbu, 2017).

Results of Code Generation: Table 2 shows that adding type as simple features does not guarantee a significant performance improvement while our proposed method significantly outperforms both forward and backward LSTM baselines. Our approach with backward LSTM has 40.3% better perplexity than original backward LSTM and forward has 63.14% lower (i.e., better) perplexity than original forward LSTM. With respect to SLP-Core performance, our model is 22.06% better in perplexity. We compare our model with SLP-Core details in case study-2.

[15]Also the AST type of a very same variable may differ in two different methods. Hence, the context is limited to each method.

[16]LSTM with type is same as *entity composite model*.

Model	Dataset (Code Corpus)	Vocabulary Size	Perplexity
SLP-Core	original	38,297	3.40
fLSTM	original	38,297	21.97
fLSTM [*type model*]	modified type	14,177	7.94
fLSTM with type feature	original	38,297	20.05
our model (fLSTM)	original	38,297	**12.52**
bLSTM	original	38,297	7.19
bLSTM [*type model*]	modified type	14,177	2.58
bLSTM with type feature	original	38,297	6.11
our model (bLSTM)	original	38,297	**2.65**

Table 2: **Comparing the performance of code generation task. All the results are on the test set of the corresponding corpus. fLSTM, bLSTM denotes forward and backward LSTM respectively. SLP-Core refers to (Hellendoorn and Devanbu, 2017).**

5 Quantitative Error Analysis

To understand the generation performance of our model and interpret the meaning of the numbers in Table 1 and 2, we further perform the following case studies.

5.1 Case Study-1: Recipe Generation

As the reduction of the perplexity does not necessarily mean the improvement of the accuracy, we design a “fill in the blank task” task to evaluate our model. A blank place in this task will contain an ingredient and we check whether our model can predict it correctly. In particular, we choose six ingredients from different frequency range (low, mid, high) based on how many times they have appeared in the training corpus. Following Table shows two examples with four blanks (underlined with the true answer).

Example fill in the blank task
1. Sprinkle <u>chicken</u> pieces lightly with <u>salt</u>.
2. Mix egg and <u>milk</u> and pour over <u>bread</u>.

We further evaluate our model on a multiple choice questioning (MCQ) strategy where the fill in the blank problem remains same but the options for the correct answers are restricted to the six ingredients. Our intuition behind this case-study is to check when there is an ingredient whether our model can learn it. If yes, we then quantify the learning using standard *accuracy* metric and compare with the state-of-the-art model to evaluate how much it improves the performance. We also measure how much the accuracy improvement depends on the training frequency.

Table 3 shows the result. Our model outperforms the fill in the blank task for both cases,

Ingredient	Train Freq.	#Blanks	Accuracy Free-Form AWD_LSTM	Accuracy Free-Form Our	Accuracy MCQ AWD_LSTM	Accuracy MCQ Our
Milk	14, 136	4,001	26.94	**59.34**	80.83	**94.90**
Salt	33,906	9,888	37.12	**62.47**	89.29	**95.75**
Apple	7,205	720	1.94	**30.28**	37.65	**89.86**
Bread	11,673	3,074	32.43	**52.64**	78.85	**94.53**
Tomato	12,866	1,815	2.20	**35.76**	43.53	**88.76**
Chicken	19,875	6,072	22.50	**45.24**	77.70	**94.63**

Table 3: **Performance of fill in the blank task.**

i.e., without any options (free-form) and MCQ. Note that, the percentage of improvement is inversely proportional to the training frequencies of the ingredients—less-frequent ingredients achieve a higher accuracy improvement (e.g., "Apple" and "Tomato"). This validates our intuition of learning to predict the type first more accurately with lower vocabulary set and then use conditional probability to predict the actual entity considering the type as a prior.

5.2 Case Study-2: Code Generation

Programming language source code shows regularities both in local and global context (e.g., variables or methods used in one source file can also be created or referenced from another library file). SLP-Core (Hellendoorn and Devanbu, 2017) is a state-of-the-art code generation model that captures this global and local information using a nested cache based n-gram language model. They further show that considering such code structure into account, a simple n-gram based SLP-Core outperforms vanilla deep learning based models like RNN, LSTM, etc.

In our case, as our example instance is a Java method, we only have the local context. Therefore, to evaluate the efficiency of our proposed model, we further analyze that exploiting only the type information are we even learning any global code pattern? If yes, then how much in comparison to the baseline (SLP-Core)? To investigate these questions, we provide all the full project information to SLP-Core (Hellendoorn and Devanbu, 2017) corresponding to our train set. However, at test-time, to establish a fair comparison, we consider the perplexity metric for the same methods. SLP-Core achieves a perplexity 3.40 where our backward LSTM achieves 2.65. This result shows that appropriate type information can actually capture many inherent attributes which can be exploited to build a good language model for programming language.

6 Conclusion

Language model often lacks in performance to predict entity names correctly. Applications with lots of named entities, thus, obviously suffer. In this work, we propose to leverage the type information of such named entities to build an effective language model. Since similar entities have the same type, the vocabulary size of a type based language model reduces significantly. The prediction accuracy of the type model increases significantly with such reduced vocabulary size. Then, using the entity type information as prior we build another language model which predicts the true entity name according to the conditional probability distribution. Our evaluation and case studies confirm that the type information of the named entities captures inherent text features too which leads to learn intrinsic text pattern and improve the performance of overall language model.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We also thank Wasi Uddin Ahmad, Peter Kim, Shou-De Lin, and Paul Mineiro for helping us implement, annotate, and design the experiments. This work was supported in part by National Science Foundation Grants IIS-1760523, CCF-16-19123, CNS-16-18771 and an NVIDIA hardware grant.

References

Miltiadis Allamanis, Earl T Barr, Premkumar Devanbu, and Charles Sutton. 2017. A survey of machine learning for big code and naturalness. *arXiv preprint arXiv:1709.06182* .

Kenneth C. Arnold, Kai-Wei Chang, and Adam Kalai. 2017. Counterfactual language model adaptation for suggesting phrases. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017*. pages 49–54.

L Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 96–103.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 286–293.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Kim B Bruce. 1993. Safe type checking in a statically-typed object-oriented programming language. In *Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, pages 285–298.

Casey Casalnuovo, Yagnik Suchak, Baishakhi Ray, and Cindy Rubio-González. 2017. Gitcproc: a tool for processing and classifying github commits. ACM, pages 396–399.

Mark Gabel and Zhendong Su. 2010. A study of the uniqueness of source code. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, pages 147–156.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863* .

Mohammad Gharehyazie, Baishakhi Ray, and Vladimir Filkov. 2017. Some from here, some from there: cross-project code reuse in github. In *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, pages 291–301.

Joshua Goodman. 2001. Classes for fast maximum entropy training. *CoRR* cs.CL/0108006. http://arxiv.org/abs/cs.CL/0108006.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR* abs/1603.06393. http://arxiv.org/abs/1603.06393.

Vincent J. Hellendoorn and Premkumar Devanbu. 2017. Are deep neural networks the best choice for modeling source code? In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, New York, NY, USA, ESEC/FSE 2017, pages 763–773. https://doi.org/10.1145/3106237.3106290.

Abram Hindle, Earl T. Barr, Mark Gabel, Zhendong Su, and Premkumar Devanbu. 2016. On the naturalness of software. *Commun. ACM* 59(5):122–131. https://doi.org/10.1145/2902362.

Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. 2012. On the naturalness of software. In *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, pages 837–847.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. *arXiv preprint arXiv:1708.00781* .

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* .

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 329–339.

Miryung Kim, Vibha Sazawal, David Notkin, and Gail Murphy. 2005. An empirical study of code clone genealogies. In *ACM SIGSOFT Software Engineering Notes*. ACM, volume 30, pages 187–196.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Donald E Knuth. 1992. Literate programming. *CSLI Lecture Notes, Stanford, CA: Center for the Study of Language and Information (CSLI), 1992* .

Giulio Maltese, P Bravetti, Hubert Crépy, BJ Grainger, M Herzog, and Francisco Palou. 2001. Combining word-and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques. In *Seventh European Conference on Speech Communication and Technology*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* .

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*. Citeseer, volume 5, pages 246–252.

Thomas R Niesler, Edward WD Whittaker, and Philip C Woodland. 1998. Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, volume 1, pages 177–180.

Thomas W Parsons. 1992. *Introduction to compiler construction*. Computer Science Press New York.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing. pages 1532–1543.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 183–190.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. *CoRR* abs/1704.07535. http://arxiv.org/abs/1704.07535.

Baishakhi Ray, Meiyappan Nagappan, Christian Bird, Nachiappan Nagappan, and Thomas Zimmermann. 2015. The uniqueness of changes: Characteristics and applications. ACM, MSR '15.

Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. 2014. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, pages 155–165.

Veselin Raychev, Martin Vechev, and Eran Yahav. 2014. Code completion with statistical language models. In *Acm Sigplan Notices*. ACM, volume 49, pages 419–428.

Zhaopeng Tu, Zhendong Su, and Premkumar Devanbu. 2014. On the localness of software. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, pages 269–280.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2692–2700. http://papers.nips.cc/paper/5866-pointer-networks.pdf.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. *CoRR* abs/1707.08052. http://arxiv.org/abs/1707.08052.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *CoRR* abs/1704.01696. http://arxiv.org/abs/1704.01696.

`hyperdoc2vec`: Distributed Representations of Hypertext Documents

Jialong Han♠, Yan Song♠, Wayne Xin Zhao♦, Shuming Shi♠, Haisong Zhang♠
♠Tencent AI Lab
♦School of Information, Renmin University of China
{jialonghan,batmanfly}@gmail.com,{clksong,shumingshi,hansonzhang}@tencent.com

Abstract

Hypertext documents, such as web pages and academic papers, are of great importance in delivering information in our daily life. Although being effective on plain documents, conventional text embedding methods suffer from information loss if directly adapted to hyper-documents. In this paper, we propose a general embedding approach for hyper-documents, namely, `hyperdoc2vec`, along with four criteria characterizing necessary information that hyper-document embedding models should preserve. Systematic comparisons are conducted between `hyperdoc2vec` and several competitors on two tasks, *i.e.,* paper classification and citation recommendation, in the academic paper domain. Analyses and experiments both validate the superiority of `hyperdoc2vec` to other models w.r.t. the four criteria.

1 Introduction

The ubiquitous World Wide Web has boosted research interests on hypertext documents, *e.g.,* personal webpages (Lu and Getoor, 2003), Wikipedia pages (Gabrilovich and Markovitch, 2007), as well as academic papers (Sugiyama and Kan, 2010). Unlike independent plain documents, a hypertext document (hyper-doc for short) links to another hyper-doc by a hyperlink or citation mark in its textual content. Given this essential distinction, hyperlinks or citations are worth specific modeling in many tasks such as link-based classification (Lu and Getoor, 2003), web retrieval (Page et al., 1999), entity linking (Cucerzan, 2007), and citation recommendation (He et al., 2010).

To model hypertext documents, various efforts (Cohn and Hofmann, 2000; Kataria et al., 2010; Perozzi et al., 2014; Zwicklbauer et al., 2016; Wang et al., 2016) have been made to depict networks of hyper-docs as well as their content. Among potential techniques, distributed representation (Mikolov et al., 2013; Le and Mikolov, 2014) tends to be promising since its validity and effectiveness are proven for plain documents on many natural language processing (NLP) tasks.

Conventional attempts on utilizing embedding techniques in hyper-doc-related tasks generally fall into two types. The first type (Berger et al., 2017; Zwicklbauer et al., 2016) simply downcasts hyper-docs to plain documents and feeds them into `word2vec` (Mikolov et al., 2013) (`w2v` for short) or `doc2vec` (Le and Mikolov, 2014) (`d2v` for short). These approaches involve downgrading hyperlinks and inevitably omit certain information in hyper-docs. However, no previous work investigates the information loss, and how it affects the performance of such downcasting-based adaptations. The second type designs sophisticated embedding models to fulfill certain tasks, *e.g.,* citation recommendation (Huang et al., 2015b), paper classification (Wang et al., 2016), and entity linking (Yamada et al., 2016), *etc.* These models are limited to specific tasks, and it is yet unknown whether embeddings learned for those particular tasks can generalize to others. Based on the above facts, we are interested in two questions:

- What information should hyper-doc embedding models preserve, and what nice property should they possess?
- Is there a general approach to learning task-independent embeddings of hyper-docs?

To answer the two questions, we formalize the hyper-doc embedding task, and propose four criteria, *i.e., content awareness*, *context awareness*, *newcomer friendliness*, and *context intent aware-*

ness, to assess different models. Then we discuss simple downcasting-based adaptations of existing approaches w.r.t. the above criteria, and demonstrate that none of them satisfy all four. To this end, we propose hyperdoc2vec (h-d2v for short), a general embedding approach for hyper-docs. Different from most existing approaches, h-d2v learns two vectors for each hyper-doc to characterize its roles of citing others and being cited. Owning to this, h-d2v is able to directly model hyperlinks or citations without downgrading them. To evaluate the learned embeddings, we employ two tasks in the academic paper domain[1], *i.e.,* paper classification and citation recommendation. Experimental results demonstrate the superiority of h-d2v. Comparative studies and controlled experiments also confirm that h-d2v benefits from satisfying the above four criteria.

We summarize our contributions as follows:

- We propose four criteria to assess different hyper-document embedding models.
- We propose hyperdoc2vec, a general embedding approach for hyper-documents.
- We systematically conduct comparisons with competing approaches, validating the superiority of h-d2v in terms of the four criteria.

2 Related Work

Network representation learning is a related topic to ours since a collection of hyper-docs resemble a network. To embed nodes in a network, Perozzi et al. (2014) propose DeepWalk, where nodes and random walks are treated as pseudo words and texts, and fed to w2v for node vectors. Tang et al. (2015b) explicitly embed second-order proximity via the number of common neighbors of nodes. Grover and Leskovec (2016) extend DeepWalk with second-order Markovian walks. To improve classification tasks, Tu et al. (2016) explore a semi-supervised setting that accesses partial labels. Compared with these models, h-d2v learns from both documents' connections and contents while they mainly focus on network structures.

Document embedding for classification is another focused area to apply document embeddings. Le and Mikolov (2014) employ learned d2v vectors to build different text classifiers. Tang et al. (2015a) apply the method in (Tang et al., 2015b) on word co-occurrence graphs for word embeddings, and average them for document vectors. For hyper-docs, Ganguly and Pudi (2017) and Wang et al. (2016) target paper classification in unsupervised and semi-supervised settings, respectively. However, unlike h-d2v, they do not explicitly model citation contexts. Yang et al. (2015)'s approach also addresses embedding hyper-docs, but involves matrix factorization and does not scale.

Citation recommendation is a direct downstream task to evaluate embeddings learned for a certain kind of hyper-docs, *i.e.,* academic papers. In this paper we concentrate on context-aware citation recommendation (He et al., 2010). Some previous studies adopt neural models for this task. Huang et al. (2015b) propose Neural Probabilistic Model (NPM) to tackle this problem with embeddings. Their model outperforms non-embedding ones (Kataria et al., 2010; Tang and Zhang, 2009; Huang et al., 2012). Ebesu and Fang (2017) also exploit neural networks for citation recommendation, but require author information as additional input. Compared with h-d2v, these models are limited in a task-specific setting.

Embedding-based entity linking is another topic that exploits embeddings to model certain hyper-docs, *i.e.,* Wikipedia (Huang et al., 2015a; Yamada et al., 2016; Sun et al., 2015; Fang et al., 2016; He et al., 2013; Zwicklbauer et al., 2016), for entity linking (Shen et al., 2015). It resembles citation recommendation in the sense that linked entities highly depend on the contexts. Meanwhile, it requires extra steps like candidate generation, and can benefit from sophisticated techniques such as collective linking (Cucerzan, 2007).

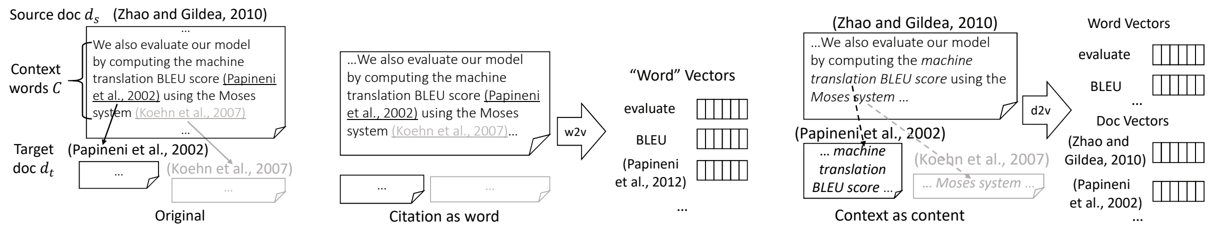
3 Preliminaries

We introduce notations and definitions, then formally define the embedding problem. We also propose four criteria for hyper-doc embedding models w.r.t their appropriateness and informativeness.

3.1 Notations and Definitions

Let $w \in W$ be a word from a vocabulary W, and $d \in D$ be a *document id* (*e.g.,* web page URLs and paper DOIs) from an id collection D. After filtering out non-textual content, a *hyper-document* H is reorganized as a sequence of words and doc ids,

[1]Although limited in tasks and domains, we expect that our embedding approach can be potentially generalized to, or serve as basis to more sophisticated methods for, similar tasks in the entity domain, *e.g.,* Wikipedia page classification and entity linking. We leave them for future work.



(a) Hyper-documents. (b) Citation as word. (c) Context as content.

Figure 1: An example of Zhao and Gildea (2010) citing Papineni et al. (2002) and existing approaches.

i.e., $W \cup D$. For example, web pages could be simplified as streams of words and URLs, and papers are actually sequences of words and cited DOIs.

If a document id d_t with some surrounding words C appear in the hyper-doc of d_s, *i.e.,* H_{d_s}, we stipulate that a *hyper-link* $\langle d_s, C, d_t \rangle$ is formed. Herein $d_s, d_t \in D$ are ids of the *source* and *target* documents, respectively; $C \subseteq W$ are *context words*. Figure 1(a) exemplifies a hyperlink.

3.2 Problem Statement

Given a corpus of hyper-docs $\{H_d\}_{d \in D}$ with D and W, we want to learn document and word embedding matrices $\mathbf{D} \in \mathbb{R}^{k \times |D|}$ and $\mathbf{W} \in \mathbb{R}^{k \times |W|}$ simultaneously. The i-th column $\mathbf{d}_i$ of $\mathbf{D}$ is a k-dimensional embedding vector for the i-th hyper-doc with id d_i. Similarly, $\mathbf{w}_j$, the j-th column of $\mathbf{W}$, is the vector for word w_j. Once embeddings for hyper-docs and words are learned, they can facilitate applications like hyper-doc classification and citation recommendation.

3.3 Criteria for Embedding Models

A reasonable model should learn how contents and hyperlinks in hyper-docs impact both $\mathbf{D}$ and $\mathbf{W}$. We propose the following criteria for models:

- **Content aware.** Content words of a hyper-doc play the main role in describing it, so the document representation should depend on its own content. For example, the words in Zhao and Gildea (2010) should affect and contribute to its embedding.

- **Context aware.** Hyperlink contexts usually provide a summary for the target document. Therefore, the target document's vector should be impacted by words that others use to summarize it, *e.g.,* paper Papineni et al. (2002) and the word "*BLEU*" in Figure 1(a).

- **Newcomer friendly.** In a hyper-document network, it is inevitable that some documents are not referred to by any hyperlink in other hyper-docs. If such "newcomers" do not get embedded properly, downstream tasks involving them are infeasible or deteriorated.

- **Context intent aware.** Words around a hyperlink, *e.g.,* "evaluate ... by" in Figure 1(a), normally indicate why the source hyper-doc makes the reference, *e.g.,* for general reference or to follow/oppose the target hyper-doc's opinion or practice. Vectors of those context words should be influenced by both documents to characterize such semantics or intents between the two documents.

We note that the first three criteria are for hyper-docs, while the last one is desired for word vectors.

4 Representing Hypertext Documents

In this section, we first give the background of two prevailing techniques, `word2vec` and `doc2vec`. Then we present two conversion approaches for hyper-documents so that `w2v` and `d2v` can be applied. Finally, we address their weaknesses w.r.t. the aforementioned four criteria, and propose our `hyperdoc2vec` model. In the remainder of this paper, when the context is clear, we mix the use of terms hyper-doc/hyperlink with paper/citation.

4.1 `word2vec` and `doc2vec`

`w2v` (Mikolov et al., 2013) has proven effective for many NLP tasks. It integrates two models, *i.e.,* `cbow` and `skip-gram`, both of which learn two types of word vectors, *i.e.,* IN and OUT vectors. `cbow` sums up IN vectors of context words and make it predictive of the current word's OUT vector. `skip-gram` uses the IN vector of the current word to predict its context words' OUT vectors.

As a straightforward extension to `w2v`, `d2v` also has two variants: `pv-dm` and `pv-dbow`. `pv-dm` works in a similar manner as `cbow`, except that the IN vector of the current document

Desired Property	Impacts Task?		Addressed by Approach?			
	Classification	Citation Recommendation	w2v	d2v-nc	d2v-cac	h-d2v
Context aware	✓	✓	✓	×	✓	✓
Content aware	✓	✓	×	✓	✓	✓
Newcomer friendly	✓	✓	×	✓	✓	✓
Context intent aware	×	✓	×	×	×	✓

Table 1: Analysis of tasks and approaches w.r.t. desired properties.

Model	Output			
	$\mathbf{D}^I$	$\mathbf{D}^O$	$\mathbf{W}^I$	$\mathbf{W}^O$
w2v	✓	✓	✓	✓
d2v (pv-dm)	✓	×	✓	✓
d2v (pv-dbow)	✓	×	×	✓
h-d2v	✓	✓	✓	✓

Table 2: Output of models.

is regarded as a special context vector to average. Analogously, `pv-dbow` uses IN document vector to predict its words' OUT vectors, following the same structure of `skip-gram`. Therefore in `pv-dbow`, words' IN vectors are omitted.

4.2 Adaptation of Existing Approaches

To represent hyper-docs, a straightforward strategy is to convert them into plain documents in a certain way and apply `w2v` and `d2v`. Two conversions following this strategy are illustrated below.

Citation as word. This approach is adopted by Berger et al. (2017).[2] As Figure 1(b) shows, document ids D are treated as a collection of special words. Each citation is regarded as an occurrence of the target document's special word. After applying standard word embedding methods, *e.g.,* `w2v`, we obtain embeddings for both ordinary words and special "words", *i.e.,* documents. In doing so, this approach allows target documents interacting with context words, thus produces context-aware embeddings for them.

Context as content. It is often observed in academic papers when citing others' work, an author briefly summarizes the cited paper in its citation context. Inspired by this, we propose a context-as-content approach as in Figure 1(c). To start, we remove all citations. Then all citation contexts of a target document d_t are copied into d_t as additional contents to make up for the lost information. Finally, `d2v` is applied to the augmented documents to generate document embeddings. With this approach, the generated document embeddings are both context- and content-aware.

4.3 `hyperdoc2vec`

Besides citation-as-word with `w2v` and context-as-content with `d2v` (denoted by `d2v-cac` for short), there is also an alternative using `d2v` on documents with citations removed (`d2v-nc` for short). We made a comparison of these approaches in Table 1 in terms of the four criteria stated in Section 3.3. It is observed that none of them satisfy all criteria, where the reasons are as follows.

First, `w2v` is not content aware. Following our examples in the academic paper domain, consider the paper (hyper-doc) Zhao and Gildea (2010) in Figure 1(a), from `w2v`'s perspective in Figure 1(b), "…computing the machine translation BLEU …" and other text no longer have association with Zhao and Gildea (2010), thus not contributing to its embedding. In addition, for papers being just published and having not obtained citations yet, they will not appear as special "words" in any text. This makes `w2v` newcomer-unfriendly, *i.e.,* unable to produce embeddings for them. Second, being trained on a corpus without citations, `d2v-nc` is obviously not context aware. Finally, in both `w2v` and `d2v-cac`, context words interact with the target documents without treating the source documents as backgrounds, which forces IN vectors of words with context intents, *e.g.,* "*evaluate*" and "*by*" in Figure 1(a), to simply remember the target documents, rather than capture the semantics of the citations.

The above limitations are caused by the conversions of hyper-docs where certain information in citations is lost. For a citation $\langle d_s, C, d_t\rangle$, citation-as-word only keeps the co-occurrence information between C and d_t. Context-as-content, on the other hand, mixes C with the original content of d_t. Both approaches implicitly downgrade citations $\langle d_s, C, d_t\rangle$ to $\langle C, d_t\rangle$ for adaptation purposes.

To learn hyper-doc embeddings without such limitations, we propose `hyperdoc2vec`. In this model, two vectors of a hyper-doc d, *i.e.,* IN and OUT vectors, are adopted to represent the document of its two roles. The IN vector $\mathbf{d}^I$ characterizes d being a source document. The OUT vector $\mathbf{d}^O$ encodes its role as a target document. We note that learning those two types of vectors is advantageous. It enables us to model citations and con-

[2]It is designed for document visualization purposes.

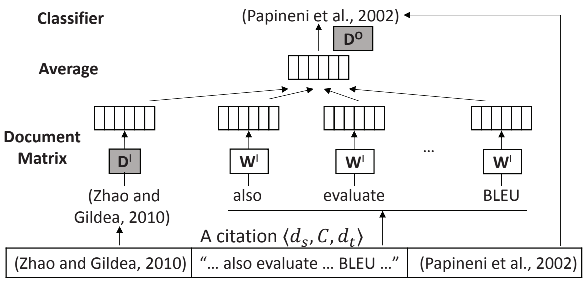


Figure 2: The `hyperdoc2vec` model.

tents simultaneously without sacrificing information on either side. Next, we describe the details of `h-d2v` in modeling citations and contents.

To model citations, we adopt the architecture in Figure 2. It is similar to `pv-dm`, except that documents rather than words are predicted at the output layer. For a citation $\langle d_s, C, d_t \rangle$, to allow context words C interacting with both vectors, we average $\mathbf{d}_s^I$ of d_s with word vectors of C, and make the resulted vector predictive of $\mathbf{d}_t^O$ of d_t. Formally, for all citations $\mathcal{C} = \{\langle d_s, C, d_t \rangle\}$, we aim to optimize the following average log probability objective:

$$\max_{\mathbf{D}^I,\mathbf{D}^O,\mathbf{W}^I} \frac{1}{|\mathcal{C}|} \sum_{\langle d_s,C,d_t\rangle \in \mathcal{C}} \log P(d_t|d_s, C) \quad (1)$$

To model the probability $P(d_t|d_s, C)$ where d_t is cited in d_s with C, we average their IN vectors

$$\mathbf{x} = \frac{1}{1+|C|}\left(\mathbf{d}_s^I + \sum_{w\in C} \mathbf{w}^I\right) \quad (2)$$

and use $\mathbf{x}$ to compose a multi-class softmax classifier on all OUT document vectors

$$P(d_t|d_s, C) = \frac{\exp(\mathbf{x}^\top \mathbf{d}_t^O)}{\sum_{d\in D}\exp(\mathbf{x}^\top \mathbf{d}^O)} \quad (3)$$

To model contents' impact on document vectors, we simply consider an additional objective function that is identical to `pv-dm`, *i.e.,* enumerate words and contexts, and use the same input architecture as Figure 2 to predict the OUT vector of the current word. Such convenience owes to the fact that using two vectors makes the model parameters compatible with those of `pv-dm`. Note that combining the citation and content objectives leads to a joint learning framework. To facilitate easier and faster training, we adopt an alternative pre-training/fine-tuning or *retrofitting* framework (Faruqui et al., 2015). We initialize with a predefined number of `pv-dm` iterations, and then optimize Eq. 1 based on the initialization.

Dataset		Docs	Citations	Years
NIPS	Train	1,590	512	Up to 1998
	Test	150	89	1999
	Total	1,740	601	Up to 1999
ACL	Train	18,845	91,792	Up to 2012
	Test	1,563	16,937	2013
	Total	20,408	108,729	Up to 2013
DBLP	Train	593,378	2,565,625	Up to 2009
	Test	55,736	308,678	From 2010
	Total	649,114	2,874,303	All years

Table 3: The statistics of three datasets.

Finally, similar to `w2v` (Mikolov et al., 2013) and `d2v` (Le and Mikolov, 2014), to make training efficient, we adopt negative sampling:

$$\log\sigma(\mathbf{x}^\top \mathbf{d}_t^O) + \sum_{i=1}^{n} \mathbb{E}_{d_i\sim P_N(d)} \log\sigma(-\mathbf{x}^\top \mathbf{d}_i^O) \quad (4)$$

and use it to replace every $\log P(d_t|d_s, C)$. Following Huang et al. (2015b), we adopt a uniform distribution on D as the distribution $P_N(d)$.

Unlike the other models in Table 1, `h-d2v` satisfies all four criteria. We refer to the example in Figure 2 to make the points clear. First, when optimizing Eq. 1 with the instance in Figure 2, the update to $\mathbf{d}^O$ of Papineni et al. (2002) depends on $\mathbf{w}^I$ of context words such as "*BLEU*". Second, we pre-train $\mathbf{d}^I$ with contents, which makes the document embeddings content aware. Third, newcomers can depend on their contents for $\mathbf{d}^I$, and update their OUT vectors when they are sampled[3] in Eq. 4. Finally, the optimization of Eq. 1 enables mutual enhancement between vectors of hyper-docs and context intent words, *e.g.,* "*evaluate by*". Under the background of a machine translation paper Zhao and Gildea (2010), the above two words help point the citation to the BLEU paper (Papineni et al., 2002), thus updating its OUT vector. The intent "*adopting tools/algorithms*" of "evaluate by" is also better captured by iterating over many document pairs with them in between.

5 Experiments

In this section, we first introduce datasets and basic settings used to learn embeddings. We then discuss additional settings and present experimental results of the two tasks, *i.e.,* document classification and citation recommendation, respectively.

[3]Given a relatively large n.

Model	Original		w/ DeepWalk	
	Macro	Micro	Macro	Micro
DeepWalk	61.67	69.89	61.67	69.89
w2v (I)	10.83	41.84	31.06	50.93
w2v (I+O)	9.36	41.26	25.92	49.56
d2v-nc	70.62	77.86	70.64	78.06
d2v-cac	71.83	78.09	71.57	78.59
h-d2v (I)	68.81	76.33	**73.96**	**79.93**
h-d2v (I+O)	**72.89**	**78.99**	73.24	79.55

Table 4: F_1 scores on DBLP.

Model	Content Aware/ Newcomer Friendly	Original		w/ DeepWalk	
		Macro	Micro	Macro	Micro
DeepWalk	-	66.57	**76.56**	66.57	76.56
w2v (I)	× / ×	19.77	47.32	59.80	72.90
w2v (I+O)	× / ×	15.97	45.66	50.77	70.08
d2v-nc	✓/ ✓	61.54	73.73	69.37	78.22
d2v-cac	✓/ ✓	65.23	75.93	**70.43**	**78.75**
h-d2v (I)	✓/ ✓	58.59	69.79	66.99	75.63
h-d2v (I+O)	✓/ ✓	**66.64**	75.19	68.96	76.61

Table 5: F_1 on DBLP when newcomers are discarded.

5.1 Datasets and Experimental Settings

We use three datasets from the academic paper domain, *i.e.,* NIPS[4], ACL anthology[5] and DBLP[6], as shown in Table 3. They all contain full text of papers, and are of small, medium, and large size, respectively. We apply ParsCit[7] (Councill et al., 2008) to parse the citations and bibliography sections. Each identified citation string referring to a paper in the same dataset, *e.g.,* [1] or (Author et al., 2018), is replaced by a global paper id. Consecutive citations like [1, 2] are regarded as multiple ground truths occupying one position. Following He et al. (2010), we take 50 words before and after a citation as the citation context.

Gensim (Řehůřek and Sojka, 2010) is used to implement all `w2v` and `d2v` baselines as well as `h-d2v`. We use `cbow` for `w2v` and `pv-dbow` for `d2v`, unless otherwise noted. For all three baselines, we set the (half) context window length to 50. For `w2v`, `d2v`, and the `pv-dm`-based initialization of `h-d2v`, we run 5 epochs following Gensim's default setting. For `h-d2v`, its iteration is set to 100 epochs with 1000 negative samples. The dimension size k of all approaches is 100. All other parameters in Gensim are kept as default.

5.2 Document Classification

In this task, we classify the research fields of papers given their vectors learned on DBLP. To obtain labels, we use Cora[8], a small dataset of Computer Science papers and their field categories. We keep the first levels of the original categories, *e.g.,* "Artificial Intelligence" of "Artificial Intelligence - Natural Language Processing", leading to 10 unique classes. We then intersect the dataset with DBLP, and obtain 5,975 labeled papers.

For `w2v` and `h-d2v` outputing both IN and OUT document vectors, we use IN vectors or concatenations of both vectors as features. For newcomer papers without `w2v` vectors, we use zero vectors instead. To enrich the features with network structure information, we also try concatenating them with the output of DeepWalk (Perozzi et al., 2014), a representative network embedding model. The model is trained on the citation network of DBLP with an existing implementation[9] and default parameters. An SVM classifier with RBF kernel is used. We perform 5-fold cross validation, and report Macro- and Micro-F_1 scores.

5.2.1 Classification Performance

In Table 4, we demonstrate the classification results. We have the following observations.

First, adding DeepWalk information almost always leads to better classification performance, except for Macro-F_1 of the `d2v-cac` approach.

Second, owning to different context awareness, `d2v-cac` consistently outperforms `d2v-nc` in terms of all metrics and settings.

Third, `w2v` has the worst performance. The reason may be that `w2v` is neither content aware nor newcomer friendly. We will elaborate more on the impacts of the two properties in Section 5.2.2.

Finally, no matter whether DeepWalk vectors are used, `h-d2v` achieves the best F_1 scores. However, when OUT vectors are involved, `h-d2v` with DeepWalk has slightly worse performance. A possible explanation is that, when `h-d2v` IN and DeepWalk vectors have enough information to train the SVM classifiers, adding another 100 features (OUT vectors) only increase the parameter

[4]https://cs.nyu.edu/ roweis/data.html
[5]http://clair.eecs.umich.edu/aan/index.php (2013 release)
[6]http://zhou142.myweb.cs.uwindsor.ca/academicpaper.html This page has been unavailable recently. They provide a larger CiteSeer dataset and a collection of DBLP paper ids. To better interpret results from the Computer Science perspective, we intersect them and obtain the DBLP dataset.
[7]https://github.com/knmnyn/ParsCit
[8]http://people.cs.umass.edu/~mccallum/data.html
[9]https://github.com/phanein/deepwalk

Model	NIPS				ACL Anthology				DBLP			
	Rec	MAP	MRR	nDCG	Rec	MAP	MRR	nDCG	Rec	MAP	MRR	nDCG
w2v (cbow, I4I)	5.06	1.29	1.29	2.07	12.28	5.35	5.35	6.96	3.01	1.00	1.00	1.44
w2v (cbow, I4O)	12.92	**6.97**	**6.97**	8.34	15.68	8.54	8.55	10.23	13.26	7.29	7.33	8.58
d2v-nc (pv-dbow, cosine)	14.04	3.39	3.39	5.82	21.09	9.65	9.67	12.29	7.66	3.25	3.25	4.23
d2v-cac (same as d2v-nc)	14.61	4.94	4.94	7.14	28.01	11.82	11.84	15.59	15.67	7.34	7.36	9.16
NPM (Huang et al., 2015b)	7.87	2.73	3.13	4.03	12.86	5.98	5.98	7.59	6.87	3.28	3.28	4.07
h-d2v (random init, I4O)	3.93	0.78	0.78	1.49	30.98	16.76	16.77	20.12	17.22	8.82	8.87	10.65
h-d2v (pv-dm retrofitting, I4O)	**15.73**	6.68	6.68	**8.80**	**31.93**	**17.33**	**17.34**	**20.76**	**21.32**	**10.83**	**10.88**	**13.14**

Table 6: Top-10 citation recommendation results (dimension size $k = 100$).

space of the classifiers and the training variance. For `w2v` with or without DeepWalk, it is also the case. This may be because information in `w2v`'s IN and OUT vectors is fairly redundant.

5.2.2 Impacts of Content Awareness and Newcomer Friendliness

Because content awareness and newcomer friendliness are highly correlated in Table 1, to isolate and study their impacts, we decouple them as follows. In the 5,975 labeled papers, we keep 2,052 with at least one citation, and redo experiments in Table 4. By carrying out such controlled experiments, we expect to remove the impact of newcomers, and compare all approaches only with respect to different content awareness. In Table 5, we provide the new scores obtained.

By comparing Tables 4 and 5, we observe that `w2v` benefits from removing newcomers with zero vectors, while all newcomer friendly approaches get lower scores because of fewer training examples. Even though the change, `w2v` still cannot outperform the other approaches, which reflects the positive impact of content awareness on the classification task. It is also interesting that DeepWalk becomes very competitive. This implies that structure-based methods favor networks with better connectivity. Finally, we note that Table 5 is based on controlled experiments with intentionally skewed data. The results are not intended for comparison among approaches in practical scenarios.

5.3 Citation Recommendation

When writing papers, it is desirable to recommend proper citations for a given context. This could be achieved by comparing the vectors of the context and previous papers. We use all three datasets for this task. Embeddings are trained on papers before 1998, 2012, and 2009, respectively. The remaining papers in each dataset are used for testing.

We compare `h-d2v` with all approaches in Section 4.2, as well as NPM[10] (Huang et al., 2015b) mentioned in Section 2, the first embedding-based approach for the citation recommendation task. Note that the inference stage involves interactions between word and document vectors and is nontrivial. We describe our choices as below.

First, for `w2v` vectors, Nalisnick et al. (2016) suggest that the IN-IN similarity favors word pairs with similar functions (*e.g.,* “red” and “blue”), while the IN-OUT similarity characterizes word co-occurrence or compatibility (*e.g.,* “red” and “bull”). For citation recommendation that relies on the compatibility between context words and cited papers, we hypothesize that the IN-for-OUT (or I4O for short) approach will achieve better results. Therefore, for `w2v`-based approaches, we average IN vectors of context words, then score and and rank OUT document vectors by dot product.

Second, for `d2v`-based approaches, we use the learned model to infer a document vector $\mathbf{d}$ for the context words, and use $\mathbf{d}$ to rank IN document vectors by cosine similarity. Among multiple attempts, we find this choice to be optimal.

Third, for `h-d2v`, we adopt the same scoring and ranking configurations as for `w2v`.

Finally, for NPM, we adopt the same ranking strategy as in Huang et al. (2015b). Following them, we focus on top-10 results and report the Recall, MAP, MRR, and nDCG scores.

5.3.1 Recommendation Performance

In Table 6, we report the citation recommendation results. Our observations are as follows.

First, among all datasets, all methods perform relatively well on the medium-sized ACL dataset. This is because the smallest NIPS dataset provides

[10]Note that the authors used $n = 1000$ for negative sampling, and did not report the number of training epoches. After many trials, we find that setting the number of both the negative samples and epoches at 100 to be relatively effective and affordable w.r.t. training time.

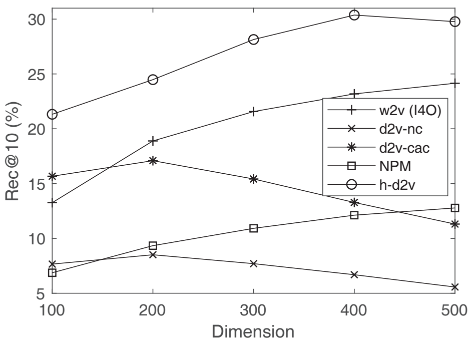


Figure 3: Varying k on DBLP. The scores of w2v keeps increasing to 26.63 at $k = 1000$, and then begins to drop. Although at the cost of a larger model and longer training/inference time, it still cannot outperform h-d2v of 30.37 at $k = 400$.

too few citation contexts to train a good model. Moreover, DBLP requires a larger dimension size k to store more information in the embedding vectors. We increase k and report the Rec@10 scores in Figure 3. We see that all approaches have better performance when k increases to 200, though d2v-based ones start to drop beyond this point.

Second, the I4I variant of w2v has the worst performance among all approaches. This observation validates our hypothesis in Section 5.3.

Third, the d2v-cac approach outperforms its variant d2v-nc in terms of all datasets and metrics. This indicates that context awareness matters in the citation recommendation task.

Fourth, the performance of NPM is sandwiched between those of w2v's two variants. We have tried our best to reproduce it. Our explanation is that NPM is citation-as-word-based, and only depends on citation contexts for training. Therefore, it is only context aware but neither content aware nor newcomer friendly, and behaves like w2v.

Finally, when retrofitting pv-dm, h-d2v generally has the best performance. When we substitute pv-dm with random initialization, the performance is deteriorated by varying degrees on different datasets. This implies that content awareness is also important, if not so important than context awareness, on the citation recommendation task.

5.3.2 Impact of Newcomer Friendliness

Table 7 analyzes the impact of newcomer friendliness. Opposite from what is done in Section 5.2.2, we only evaluate on testing examples where at least a ground-truth paper is a newcomer. Please note that newcomer unfriendly approaches do not necessarily get zero scores. The table shows that newcomer friendly approaches are superior to unfriendly ones. Note that, like Table 5, this table is also based on controlled experiments and not intended for comparing approaches.

Model	Newcomer Friendly	Rec	MAP	MRR	nDCG
w2v (I4O)	×	3.64	3.23	3.41	2.73
NPM	×	1.37	1.13	1.15	0.92
d2v-nc	✓	6.48	3.52	3.54	3.96
d2v-cac	✓	**8.16**	**5.13**	**5.24**	**5.21**
h-d2v	✓	6.41	4.95	5.21	4.49

Table 7: DBLP results evaluated on 63,342 citation contexts with newcomer ground-truth.

Category	Description
Weak	Weakness of cited approach
CoCoGM	Contrast/Comparison in Goals/Methods (neutral)
CoCo-	Work stated to be superior to cited work
CoCoR0	Contrast/Comparison in Results (neutral)
CoCoXY	Contrast between 2 cited methods
PBas	Author uses cited work as basis or starting point
PUse	Author uses tools/algorithms/data/definitions
PModi	Author adapts or modifies tools/algorithms/data
PMot	This citation is positive about approach used or problem addressed (used to motivate work in current paper)
PSim	Author's work and cited work are similar
PSup	Author's work and cited work are compatible/provide support for each other
Neut	Neutral description of cited work, or not enough textual evidence for above categories, or unlisted citation function

Table 8: Annotation scheme of citation functions in Teufel et al. (2006).

5.3.3 Impact of Context Intent Awareness

In this section, we analyze the impact of context intent awareness. We use Teufel et al. (2006)'s 2,824 citation contexts[11] with annotated citation functions, *e.g.,* emphasizing weakness (Weak) or using tools/algorithms (PBas) of the cited papers. Table 8 from Teufel et al. (2006) describes the full annotating scheme. Teufel et al. (2006) also use manual features to evaluate citation function classification. To test all models on capturing context intents, we average all context words' IN vectors (trained on DBLP) as features. Noticing that pv-dbow does not output IN word vectors, and OUT vectors do not provide reasonable results, we use pv-dm here instead. We use SVM with RBF

[11]The number is 2,829 in the original paper. The inconsistency may be due to different regular expressions we used.

Query and Ground Truth	**Result Ranking of** `w2v`	**Result Ranking of** `d2v-cac`	**Result Ranking of** `h-d2v`
…We also evaluate our model by computing the machine translation BLEU score (Papineni et al., 2002) using the Moses system (Koehn et al., 2007)… (Papineni et al., 2002) **BLEU: a Method for Automatic Evaluation of Machine Translation** (Koehn et al., 2007) **Moses: Open Source Toolkit for Statistical Machine Translation**	1. HMM-Based Word Alignment in Statistical Translation 2. Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems 3. The Alignment Template Approach to Statistical Machine Translation … 9. **Moses: Open Source Toolkit for Statistical Machine Translation** 57. **BLEU: a Method for Automatic Evaluation of Machine Translation**	1. Discriminative Reranking for Machine Translation 2. Learning Phrase-Based Head Transduction Models for Translation of Spoken Utterances 3. Cognates Can Improve Statistical Translation Models … 6. **BLEU: a Method for Automatic Evaluation of Machine Translation** 29. **Moses: Open Source Toolkit for Statistical Machine Translation**	1. **BLEU: a Method for Automatic Evaluation of Machine Translation** 2. Statistical Phrase-Based Translation 3. Improved Statistical Alignment Models 4. HMM-Based Word Alignment in Statistical Translation 5. **Moses: Open Source Toolkit for Statistical Machine Translation**

Table 9: Papers recommended by different approaches for a citation context in Zhao and Gildea (2010).

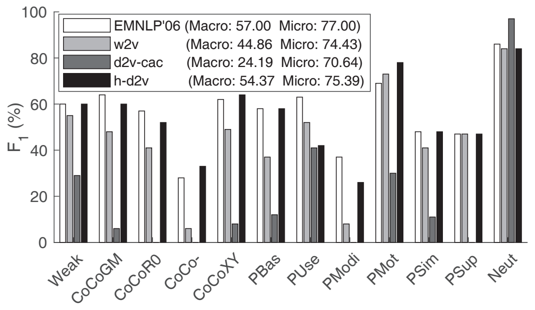


Figure 4: F_1 of citation function classification.

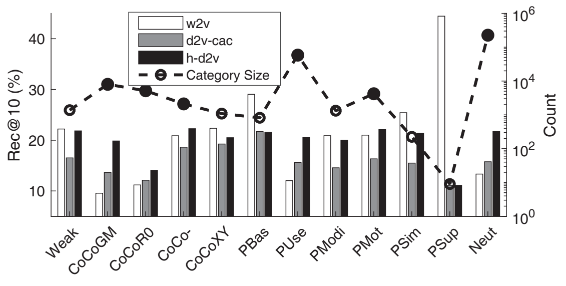


Figure 5: Rec@10 w.r.t. citation functions.

kernels and default parameters. Following Teufel et al. (2006), we use 10-fold cross validation.

Figure 4 depicts the F_1 scores. Scores of Teufel et al. (2006)'s approach are from the original paper. We omit `d2v-nc` because it is very inferior to `d2v-cac`. We have the following observations.

First, Teufel et al. (2006)'s feature-engineering-based approach has the best performance. Note that we cannot obtain their original cross validation split, so the comparison may not be fair and is only for consideration in terms of numbers.

Second, among all embedding-based methods, `h-d2v` has the best citation function classification results, which is close to Teufel et al. (2006)'s.

Finally, the `d2v-cac` vectors are only good at Neutral, the largest class. On the other classes and global F_1, they are outperformed by `w2v` vectors.

To study how citation function affects citation recommendation, we combine the 2,824 labeled citation contexts and another 1,075 labeled contexts the authors published later to train an SVM, and apply it to the DBLP testing set to get citation functions. We evaluate citation recommendation performance of `w2v` (I4O), `d2v-cac`, and `h-d2v` on a per-citation-function basis. In Figure 5, we break down Rec@10 scores on citation functions. On the six largest classes (marked by solid dots), `h-d2v` outperforms all competitors.

To better investigate the impact of context intent awareness, Table 9 shows recommended papers of the running example of this paper. Here, Zhao and Gildea (2010) cited the BLEU metric (Papineni et al., 2002) and Moses tools (Koehn et al., 2007) of machine translation. However, the additional words "machine translation" lead both `w2v` and `d2v-cac` to recommend many machine translation papers. Only our `h-d2v` manages to recognize the citation function "using tools/algorithms (PBas)", and concentrates on the citation intent to return the right papers in top-5 results.

6 Conclusion

We focus on the hyper-doc embedding problem. We propose that hyper-doc embedding algorithms should be content aware, context aware, newcomer friendly, and context intent aware. To meet all four criteria, we propose a general approach, `hyperdoc2vec`, which assigns two vectors to each hyper-doc and models citations in a straightforward manner. In doing so, the learned embeddings satisfy all criteria, which no existing model is able to. For evaluation, paper classification and citation recommendation are conducted on three academic paper datasets. Results confirm the effectiveness of our approach. Further analyses also demonstrate that possessing the four properties helps `h-d2v` outperform other models.

References

Matthew Berger, Katherine McDonough, and Lee M. Seversky. 2017. cite2vec: Citation-driven document exploration via word embeddings. *IEEE Trans. Vis. Comput. Graph.* 23(1):691–700.

David A. Cohn and Thomas Hofmann. 2000. The missing link - A probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000*. pages 430–436.

Isaac G. Councill, C. Lee Giles, and Min-Yen Kan. 2008. Parscit: an open-source CRF reference string parsing package. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 708–716.

Travis Ebesu and Yi Fang. 2017. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 1093–1096.

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*. pages 260–269.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1606–1615.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*. pages 1606–1611.

Soumyajit Ganguly and Vikram Pudi. 2017. Paper2vec: Combining graph and text information for scientific paper representation. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017*. pages 383–395.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 855–864.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and C. Lee Giles. 2010. Context-aware citation recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010*. pages 421–430.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Volume 2: Short Papers*. pages 30–34.

Hongzhao Huang, Larry P. Heck, and Heng Ji. 2015a. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *CoRR* abs/1504.07678.

Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles, and Lior Rokach. 2012. Recommending citations: translating papers into references. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12*. pages 1910–1914.

Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C. Lee Giles. 2015b. A neural probabilistic model for context based citation recommendation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. pages 2404–2410.

Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. 2010. Utilizing context in generative bayesian models for linked corpus. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*. pages 1188–1196.

Qing Lu and Lise Getoor. 2003. Link-based classification. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003)*. pages 496–503.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*. pages 3111–3119.

Eric T. Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Companion Volume*. pages 83–84.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pages 311–318.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*. pages 701–710.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. http://is.muni.cz/publication/884893/en.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.* 27(2):443–460.

Kazunari Sugiyama and Min-Yen Kan. 2010. Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 2010 Joint International Conference on Digital Libraries, JCDL 2010*. pages 29–38.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*. pages 1333–1339.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. PTE: predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 1165–1174.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015b. LINE: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015*. pages 1067–1077.

Jie Tang and Jing Zhang. 2009. A discriminative approach to topic-based citation recommendation. In *Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009*. pages 572–579.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. pages 103–110.

Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-margin deepwalk: Discriminative learning of network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*. pages 3889–3895.

Suhang Wang, Jiliang Tang, Charu C. Aggarwal, and Huan Liu. 2016. Linked document embedding for classification. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016*. pages 115–124.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*. pages 250–259.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network representation learning with rich text information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*. pages 2111–2117.

Shaojun Zhao and Daniel Gildea. 2010. A fast fertility hidden markov model for word alignment using MCMC. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010*. pages 596–605.

Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016*. pages 425–434.

Entity-Duet Neural Ranking: Understanding the Role of Knowledge Graph Semantics in Neural Information Retrieval

Zhenghao Liu[1] **Chenyan Xiong**[2] **Maosong Sun**[1] [*] **Zhiyuan Liu**[1]
[1]State Key Laboratory of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Language Technologies Institute, Carnegie Mellon University

Abstract

This paper presents the Entity-Duet Neural Ranking Model (`EDRM`), which introduces knowledge graphs to neural search systems. `EDRM` represents queries and documents by their words and entity annotations. The semantics from knowledge graphs are integrated in the distributed representations of their entities, while the ranking is conducted by interaction-based neural ranking networks. The two components are learned end-to-end, making `EDRM` a natural combination of entity-oriented search and neural information retrieval. Our experiments on a commercial search log demonstrate the effectiveness of `EDRM`. Our analyses reveal that knowledge graph semantics significantly improve the generalization ability of neural ranking models.

1 Introduction

The emergence of large scale knowledge graphs has motivated the development of *entity-oriented search*, which utilizes knowledge graphs to improve search engines. The recent progresses in entity-oriented search include better text representations with entity annotations (Xiong et al., 2016; Raviv et al., 2016), richer ranking features (Dalton et al., 2014), entity-based connections between query and documents (Liu and Fang, 2015; Xiong and Callan, 2015), and soft-match query and documents through knowledge graph relations or embeddings (Xiong et al., 2017c; Ensan and Bagheri, 2017). These approaches bring in entities and semantics from knowledge graphs and have greatly improved the effectiveness of feature-based search systems.

[*]Corresponding author: M. Sun (sms@tsinghua.edu.cn)

Another frontier of information retrieval is the development of neural ranking models (*neural-IR*). Deep learning techniques have been used to learn distributed representations of queries and documents that capture their relevance relations (*representation-based*) (Shen et al., 2014), or to model the query-document relevancy directly from their word-level interactions (*interaction-based*) (Guo et al., 2016a; Xiong et al., 2017b; Dai et al., 2018). Neural-IR approaches, especially the *interaction-based* ones, have greatly improved the ranking accuracy when large scale training data are available (Dai et al., 2018).

Entity-oriented search and neural-IR push the boundary of search engines from two different aspects. Entity-oriented search incorporates human knowledge from entities and knowledge graph semantics. It has shown promising results on feature-based ranking systems. On the other hand, neural-IR leverages distributed representations and neural networks to learn more sophisticated ranking models form large-scale training data. However, it remains unclear how these two approaches interact with each other and whether the entity-oriented search has the same advantage in neural-IR methods as in feature-based systems.

This paper explores the role of entities and semantics in neural-IR. We present an Entity-Duet Neural Ranking Model (`EDRM`) that incorporates entities in interaction-based neural ranking models. `EDRM` first learns the distributed representations of entities using their semantics from knowledge graphs: descriptions and types. Then it follows a recent state-of-the-art entity-oriented search framework, the word-entity duet (Xiong et al., 2017a), and matches documents to queries with both bag-of-words and bag-of-entities. Instead of manual features, `EDRM` uses interaction-based neural models (Dai et al., 2018) to match query and documents with word-entity duet representations. As a result, EDRM combines entity-oriented search and the interaction based neural-IR; it brings the knowledge graph semantics to neural-IR and enhances entity-oriented search with neural networks.

One advantage of being neural is that EDRM can be learned end-to-end. Given a large amount of user feedback from a commercial search log, the integration of knowledge graph semantics to neural ranker, is learned jointly with the modeling of query-document relevance in EDRM. It provides a convenient data-driven way to leverage external semantics in neural-IR.

Our experiments on a Sogou query log and CN-DBpedia demonstrate the effectiveness of entities and semantics in neural models. EDRM significantly outperforms the word-interaction-based neural ranking model, K-NRM (Xiong et al., 2017a), confirming the advantage of entities in enriching word-based ranking. The comparison with Conv-KNRM (Dai et al., 2018), the recent state-of-the-art neural ranker that models phrase level interactions, provides a more interesting observation: Conv-KNRM predicts user clicks reasonably well, but integrating knowledge graphs using EDRM significantly improves the neural model's generalization ability on more difficult scenarios.

Our analyses further revealed the source of EDRM's generalization ability: the knowledge graph semantics. If only treating entities as ids and ignoring their semantics from the knowledge graph, the entity annotations are only a cleaner version of phrases. In neural-IR systems, the embeddings and convolutional neural networks have already done a decent job in modeling phrase-level matches. However, the knowledge graph semantics brought by EDRM can not yet be captured solely by neural networks; incorporating those human knowledge greatly improves the generalization ability of neural ranking systems.

2 Related Work

Current neural ranking models can be categorized into two groups: representation based and interaction based (Guo et al., 2016b). The earlier works mainly focus on representation based models. They learn good representations and match them in the learned representation space of query and documents. DSSM (Huang et al., 2013) and its convolutional version CDSSM (Shen et al., 2014) get representations by hashing letter-tri-grams to a low dimension vector. A more recent work uses pseudo-labeling as a weak supervised signal to train the representation based ranking model (Dehghani et al., 2017).

The interaction based models learn word-level interaction patterns from query-document pairs. ARC-II (Hu et al., 2014) and MatchPyramind (Pang et al., 2016) utilize Convolutional Neural Network (CNN) to capture complicated patterns from word-level interactions. The Deep Relevance Matching Model (DRMM) (Guo et al., 2016b) uses pyramid pooling (histogram) to summarize the word-level similarities into ranking models. K-NRM and Conv-KNRM use kernels to summarize word-level interactions with word embeddings and provide soft match signals for learning to rank. There are also some works establishing position-dependent interactions for ranking models (Pang et al., 2017; Hui et al., 2017). Interaction based models and representation based models can also be combined for further improvements (Mitra et al., 2017).

Recently, large scale knowledge graphs such as DBpedia (Auer et al., 2007), Yago (Suchanek et al., 2007) and Freebase (Bollacker et al., 2008) have emerged. Knowledge graphs contain human knowledge about real-word entities and become an opportunity for search system to better understand queries and documents. There are many works focusing on exploring their potential for ad-hoc retrieval. They utilize knowledge as a kind of pseudo relevance feedback corpus (Cao et al., 2008) or weight words to better represent query according to well-formed entity descriptions. Entity query feature expansion (Dietz and Verga, 2014) uses related entity attributes as ranking features.

Another way to utilize knowledge graphs in information retrieval is to build the additional connections from query to documents through related entities. Latent Entity Space (LES) builds an unsupervised model using latent entities' descriptions (Liu and Fang, 2015). EsdRank uses related entities as a latent space, and performs learning to rank with various information retrieval features (Xiong and Callan, 2015). AttR-Duet develops a four-way interaction to involve cross matches between entity and word representations to catch more semantic relevance patterns (Xiong et al., 2017a).

There are many other attempts to integrate knowledge graphs in neural models in related tasks (Miller et al., 2016; Gupta et al., 2017; Ghazvininejad et al., 2018). Our work shares a similar spirit and focuses on exploring the effectiveness of knowledge graph semantics in neural-IR.

3 Entity-Duet Neural Ranking Model

This section first describes the standard architecture in current interaction based neural ranking models. Then it presents our Entity-Duet Neural Ranking Model, including the semantic entity representation which integrates the knowledge graph semantics, and then the entity-duet ranking framework. The overall architecture of EDRM is shown in Figure 1.

3.1 Interaction based Ranking Models

Given a query q and a document d, interaction based models first build the word-level translation matrix between q and d (Berger and Lafferty, 1999). The translation matrix describes word pairs similarities using word correlations, which are captured by word embedding similarities in interaction based models.

Typically, interaction based ranking models first map each word t in q and d to an L-dimensional embedding $\vec{v}_t$ with an embedding layer Emb_w:

$$\vec{v}_t = \text{Emb}_w(t). \tag{1}$$

It then constructs the interaction matrix M based on query and document embeddings. Each element M^{ij} in the matrix, compares the ith word in q and the jth word in d, e.g. using the cosine similarity of word embeddings:

$$M^{ij} = \cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d}). \tag{2}$$

With the translation matrix describing the term level matches between query and documents, the next step is to calculate the final ranking score from the matrix. Many approaches have been developed in interaction base neural ranking models, but in general, that would include a feature extractor $\phi()$ on M and then one or several ranking layers to combine the features to the ranking score.

3.2 Semantic Entity Representation

EDRM incorporates the semantic information about an entity from the knowledge graphs into its representation. The representation includes three embeddings: entity embedding, description embedding, and type embedding, all in L dimension and are combined to generate the semantic representation of the entity.

Entity Embedding uses an L-dimensional embedding layer Emb_e to get an entity embedding $\vec{v}_e^{\text{emb}}$ for e:

$$\vec{v}_e^{\text{emb}} = \text{Emb}_e(e). \tag{3}$$

Description Embedding encodes an entity description which contains m words and explains the entity. EDRM first employs the word embedding layer Emb_w to embed the description word w to $\vec{v}_w$. Then it combines all embeddings in text to an embedding matrix $\vec{V}_w$. Next, it leverages convolutional filters to slide over the text and compose the h length n-gram as $\vec{g}_e^j$:

$$\vec{g}_e^j = \text{ReLu}(W_{\text{CNN}} \cdot \vec{V}_w^{j:j+h} + \vec{b}_{\text{CNN}}), \tag{4}$$

where W_{CNN} and $\vec{b}_{\text{CNN}}$ are two parameters of the covolutional filter.

Then we use max pooling after the convolution layer to generate the description embedding $\vec{v}_e^{\text{des}}$:

$$\vec{v}_e^{\text{des}} = \max(\vec{g}_e^1, ..., \vec{g}_e^j, ..., \vec{g}_e^m). \tag{5}$$

Type Embedding encodes the categories of entities. Each entity e has n kinds of types $F_e = \{f_1, ..., f_j, ..., f_n\}$. EDRM first gets the f_j embedding $\vec{v}_{f_j}$ through the type embedding layer Emb_{tp}:

$$\vec{v}_{f_j}^{\text{emb}} = \text{Emb}_{\text{tp}}(e). \tag{6}$$

Then EDRM utilizes an attention mechanism to combine entity types to the type embedding $\vec{v}_e^{\text{type}}$:

$$\vec{v}_e^{\text{type}} = \sum_j^n a_j \vec{v}_{f_j}, \tag{7}$$

where a_j is the attention score, calculated as:

$$a_j = \frac{\exp(P_j)}{\sum_l^n \exp(P_l)}, \tag{8}$$

$$P_j = (\sum_i W_{\text{bow}} \vec{v}_{t_i}) \cdot \vec{v}_{f_j}. \tag{9}$$

P_j is the dot product of the query or document representation and type embedding f_j. We leverage bag-of-words for query or document encoding. W_{bow} is a parameter matrix.

Combination. The three embeddings are combined by an linear layer to generate the semantic representation of the entity:

$$\vec{v}_e^{\text{sem}} = \vec{v}_e^{\text{emb}} + W_e(\vec{v}_e^{\text{des}} \oplus \vec{v}_e^{\text{type}})^T + \vec{b}_e. \tag{10}$$

W_e is an $L \times 2L$ matrix and $\vec{b}_e$ is an L-dimensional vector.

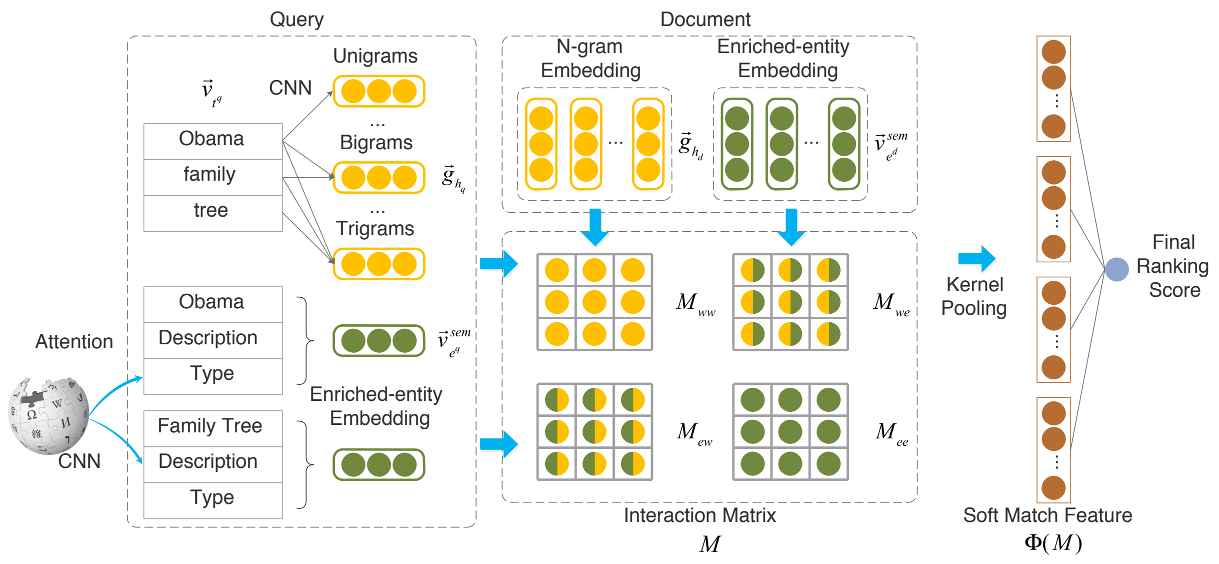


Figure 1: The architecture of EDRM.

3.3 Neural Entity-Duet Framework

Word-entity duet (Xiong et al., 2017a) is a recently developed framework in entity-oriented search. It utilizes the duet representation of bag-of-words and bag-of-entities to match q-d with hand crafted features. This work introduces it to neural-IR.

We first construct bag-of-entities q^e and d^e with entity annotation as well as bag-of-words q^w and d^w for q and d. The duet utilizes a four-way interaction: query words to document words (q^w-d^w), query words to documents entities (q^w-d^e), query entities to document words (q^e-d^w) and query entities to document entities (q^e-d^e).

Instead of features, EDRM uses a translation layer that calculates similarity between a pair of query-document terms: ($\vec{v}^i_{w^q}$ or $\vec{v}^i_{e^q}$) and ($\vec{v}^j_{w^d}$ or $\vec{v}^j_{e^d}$). It constructs the interaction matrix $M = \{M_{ww}, M_{we}, M_{ew}, M_{ee}\}$. And $M_{ww}, M_{we}, M_{ew}, M_{ee}$ denote interactions of q^w-d^w, q^w-d^e, q^e-d^w, q^e-d^e respectively. And elements in them are the cosine similarities of corresponding terms:

$$M^{ij}_{ww} = \cos(\vec{v}^i_{w^q}, \vec{v}^j_{w^d}); M^{ij}_{ee} = \cos(\vec{v}^i_{e^q}, \vec{v}^j_{e^d})$$
$$M^{ij}_{ew} = \cos(\vec{v}^i_{e^q}, \vec{v}^j_{w^d}); M^{ij}_{we} = \cos(\vec{v}^i_{w^q}, \vec{v}^j_{e^d}). \quad (11)$$

The final ranking feature $\Phi(\mathcal{M})$ is a concatenation ($\oplus$) of four cross matches ($\phi(M)$):

$$\Phi(\mathcal{M}) = \phi(M_{ww}) \oplus \phi(M_{we}) \oplus \phi(M_{ew}) \oplus \phi(M_{ee}), \quad (12)$$

where the ϕ can be any function used in interaction based neural ranking models.

The entity-duet presents an effective way to cross match query and document in entity and word spaces. In EDRM, it introduces the knowledge graph semantics representations into neural-IR models.

4 Integration with Kernel based Neural Ranking Models

The duet translation matrices provided by EDRM can be plugged into any standard interaction based neural ranking models. This section expounds special cases where it is integrated with K-NRM (Xiong et al., 2017b) and Conv-KNRM (Dai et al., 2018), two recent state-of-the-arts.

K-NRM uses K Gaussian kernels to extract the matching feature $\phi(M)$ from the translation matrix M. Each kernel K_k summarizes the translation scores as soft-TF counts, generating a K-dimensional feature vector $\phi(M) = \{K_1(M), ..., K_K(M)\}$:

$$K_k(M) = \sum_j \exp(-\frac{M^{ij} - \mu_k}{2\delta_k^2}). \quad (13)$$

μ_k and δ_k are the mean and width for the kth kernel. Conv-KNRM extend K-NRM incorporating h-gram compositions $\vec{g}^i_h$ from text embedding $\vec{V}_T$ using CNN:

$$\vec{g}^i_h = \text{relu}(W_h \cdot \vec{V}^{i:i+h}_T + \vec{v}_h). \quad (14)$$

Then a translation matrix M_{h_q,h_d} is constructed. Its elements are the similarity scores of h-gram pairs between query and document:

$$M_{h_q,h_d} = \cos(\vec{g}^{i}_{h_q}, \vec{g}^{j}_{h_d}). \quad (15)$$

We also extend word n-gram cross matches to word entity duet matches:

$$\Phi(\mathcal{M}) = \phi(M_{1,1}) \oplus ... \oplus \phi(M_{h_q,h_d}) \oplus ... \oplus \phi(M_{ee}). \quad (16)$$

Each ranking feature $\phi(M_{h_q,h_d})$ contains three parts: query h_q-gram and document h_d-gram match feature ($\phi(M_{ww^{h_q,h_d}})$), query entity and document h_d-gram match feature ($\phi(M_{ew^{1,h_d}})$), and query h_q-gram and document entity match feature ($\phi(M_{ww^{h_q,1}})$):

$$\phi(M_{h_q,h_d}) = \phi(M_{ww^{h_q,h_d}}) \oplus \phi(M_{ew^{1,h_d}}) \oplus \phi(M_{we^{h_q,1}}). \quad (17)$$

We then use learning to rank to combine ranking feature $\Phi(\mathcal{M})$ to produce the final ranking score:

$$f(q,d) = \tanh(\omega_r^T \Phi(\mathcal{M}) + b_r). \quad (18)$$

ω_r and b_r are the ranking parameters. tanh is the activation function.

We use standard pairwise loss to train the model:

$$l = \sum_q \sum_{d^+,d^- \in D_q^{+,-}} \max(0, 1 - f(q,d^+) + f(q,d^-)), \quad (19)$$

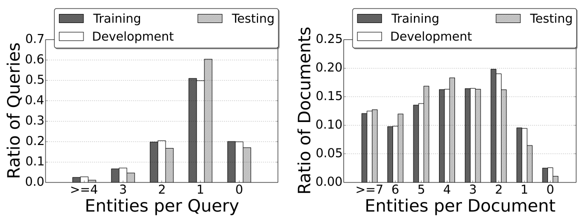
where the d^+ is a document ranks higher than d^-.

With sufficient training data, the whole model is optimized end-to-end with back-propagation. During the process, the integration of the knowledge graph semantics, entity embedding, description embeddings, type embeddings, and matching with entities-are learned jointly with the ranking neural network.

5 Experimental Methodology

This section describes the dataset, evaluation metrics, knowledge graph, baselines, and implementation details of our experiments.

Dataset. Our experiments use a query log from Sogou.com, a major Chinese searching engine (Luo et al., 2017). The exact same dataset and training-testing splits in the previous research (Xiong et al., 2017b; Dai et al., 2018) are used. They defined the ad-hoc ranking task in this dataset as re-ranking the candidate documents provided by the search engine. All Chinese texts are segmented by ICTCLAS (Zhang et al., 2003), after that they are treated the same as English.



(a) Statistic of queries (b) Statistic of documents

Figure 2: Query and document distributions. Queries and documents are grouped by the number of entities.

Prior research leverages clicks to model user behaviors and infer reliable relevance signals using click models (Chuklin et al., 2015). DCTR and TACM are two click models: DCTR calculates the relevance scores of a query-document pair based on their click through rates (CTR); TACM (Wang et al., 2013) is a more sophisticated model that uses both clicks and dwell times. Following previous research (Xiong et al., 2017b), both DCTR and TACM are used to infer labels. DCTR inferred relevance labels are used in training. Three testing scenarios are used: Testing-SAME, Testing-DIFF and Testing-RAW.

Testing-SAME uses DCTR labels, the same as in training. Testing-DIFF evaluates models performance based on TACM inferred relevance labels. Testing-RAW evaluates ranking models through user clicks, which tests ranking performance for the most satisfying document. Testing-DIFF and Testing-RAW are harder scenarios that challenge the generalization ability of all models, because their training labels and testing labels are generated differently (Xiong et al., 2017b).

Evaluation Metrics. NDCG@1 and NDCG@10 are used in Testing-SAME and Testing-DIFF. MRR is used for Testing-Raw. Statistic significances are tested by permutation test with P$<$ 0.05. All are the same as in previous research (Xiong et al., 2017b).

Knowledge Graph. We use CN-DBpedia (Xu et al., 2017), a large scale Chinese knowledge graph based on Baidu Baike, Hudong Baike, and Chinese Wikipedia. CN-DBpedia contains 10,341,196 entities and 88,454,264 relations. The query and document entities are annotated by CMNS, the commonness (popularity) based entity linker (Hasibi et al., 2017). CN-DBpedia and CMNS provide good coverage on our queries and

Table 1: Ranking accuracy of EDRM-KNRM, EDRM-CKNRM and baseline methods. Relative performances compared with K-NRM are in percentages. †, ‡, §, ¶, $*$ indicate statistically significant improvements over DRMM†, CDSSM‡, MP§, K-NRM¶ and Conv-KNRM* respectively.

	Testing-SAME				Testing-DIFF				Testing-RAW	
Method	**NDCG@1**		**NDCG@10**		**NDCG@1**		**NDCG@10**		**MRR**	
BM25	0.1422	−46.24%	0.2868	−31.67%	0.1631	−45.63%	0.3254	−23.04%	0.2280	−33.86%
RankSVM	0.1457	−44.91%	0.3087	−26.45%	0.1700	−43.33%	0.3519	−16.77%	0.2241	−34.99%
Coor-Ascent	0.1594	−39.74%	0.3547	−15.49%	0.2089	−30.37%	0.3775	−10.71%	0.2415	−29.94%
DRMM	0.1367	−48.34%	0.3134	−25.34%	0.2126‡	−29.14%	0.3592§	−15.05%	0.2335	−32.26%
CDSSM	0.1441	−45.53%	0.3329	−20.69%	0.1834	−38.86%	0.3534	−16.41%	0.2310	−33.00%
MP	0.2184†‡	−17.44%	0.3792†‡	−9.67%	0.1969	−34.37%	0.3450	−18.40%	0.2404	−30.27%
K-NRM	0.2645	–	0.4197	–	0.3000	–	0.4228	–	0.3447	–
Conv-KNRM	0.3357†‡§¶	+26.90%	0.4810†‡§¶	+14.59%	0.3384†‡§¶	+12.81%	0.4318†‡§	+2.14%	0.3582†‡§	+3.91%
EDRM-KNRM	0.3096†‡§¶	+17.04%	0.4547†‡§¶	+8.32%	0.3327†‡§¶	+10.92%	0.4341†‡§¶	+2.68%	0.3616†‡§¶	+4.90%
EDRM-CKNRM	**0.3397**†‡§¶	+28.42%	**0.4821**†‡§¶	+14.86%	**0.3708**†‡§¶*	+23.60%	**0.4513**†‡§¶*	+6.74%	**0.3892**†‡§¶*	+12.90%

documents. As shown in Figure 2, the majority of queries have at least one entity annotation; the average number of entity annotated per document title is about four.

Baselines. The baselines include feature-based ranking models and neural ranking models. Most of the baselines are borrowed from previous research (Xiong et al., 2017b; Dai et al., 2018).

Feature-based baselines include two learning to rank systems, RankSVM (Joachims, 2002) and coordinate ascent (Coor-Accent) (Metzler and Croft, 2006). The standard word-based unsupervised retrieval model, BM25, is also compared.

Neural baselines include CDSSM (Shen et al., 2014), MatchPyramid (MP) (Pang et al., 2016), DRMM (Grauman and Darrell, 2005), K-NRM (Xiong et al., 2017b) and Conv-KNRM (Dai et al., 2018). CDSSM is representation based. It uses CNN to build query and document representations on word letter-tri-grams (or Chinese characters). MP and DRMM are both interaction based models. They use CNNs or histogram pooling to extract features from embedding based translation matrix.

Our main baselines are K-NRM and Conv-KNRM, the recent state-of-the-art neural models on the Sogou-Log dataset. The goal of our experiments is to explore the effectiveness of knowledge graphs in these state-of-the-art interaction based neural models.

Implementation Details. The dimension of word embedding, entity embedding and type embedding are 300. Vocabulary size of entities and words are 44,930 and 165,877. Conv-KNRM uses one layer CNN with 128 filter size for the n-gram composition. Entity description encoder is a one layer CNN with 128 and 300 filter size for Conv-KNRM and K-NRM respectively.

All models are implemented with PyTorch. Adam is utilized to optimize all parameters with learning rate $= 0.001$, $\epsilon = 1e - 5$ and early stopping with the practice of 5 epochs.

There are two versions of EDRM: EDRM-KNRM and EDRM-CKNRM, integrating with K-NRM and Conv-KNRM respectively. The first one (K-NRM) enriches the word based neural ranking model with entities and knowledge graph semantics; the second one (Conv-KNRM) enriches the n-gram based neural ranking model.

6 Evaluation Results

Four experiments are conducted to study the effectiveness of EDRM: the overall performance, the contributions of matching kernels, the ablation study, and the influence of entities in different scenarios. We also do case studies to show effect of EDRM on document ranking.

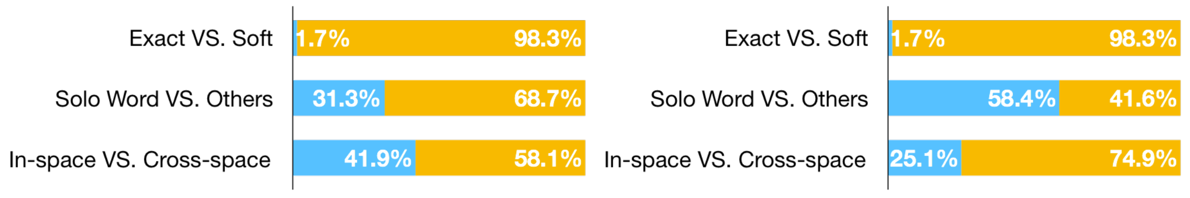
6.1 Ranking Accuracy

The ranking accuracies of the ranking methods are shown in Table 1. K-NRM and Conv-KNRM outperform other baselines in all testing scenarios by large margins as shown in previous research.

EDRM-KNRM out performs K-NRM by over 10% improvement in Testing-SAME and Testing-DIFF. EDRM-CKNRM has almost same performance on Testing-SAME with Conv-KNRM. A possible reason is that, entity annotations provide effective phrase matches, but Conv-KNRM is also able to learn phrases matches automatically from data. However, EDRM-CKNRM has significant improvement on Testing-DIFF and Testing-RAW. Those demonstrate that EDRM has strong ability to overcome domain differences from different labels.

Table 2: Ranking accuracy of adding diverse semantics based on K-NRM and Conv-KNRM. Relative performances compared are in percentages. †, ‡, §, ¶, $*$, $**$ indicate statistically significant improvements over K-NRM† (or Conv-KNRM†), +Embed‡, +Type§, +Description¶, +Embed+Type* and +Embed+Description** respectively.

	Testing-SAME				Testing-DIFF				Testing-RAW	
Method	**NDCG@1**		**NDCG@10**		**NDCG@1**		**NDCG@10**		**MRR**	
`K-NRM`	0.2645	–	0.4197	–	0.3000	–	0.4228	–	0.3447	–
`+Embed`	0.2743	+3.68%	0.4296	+2.35%	0.3134	+4.48%	0.4306	+1.86%	0.3641†	+5.62%
`+Type`	0.2709	+2.41%	0.4395†	+4.71%	0.3126	+4.20%	0.4373†	+3.43%	0.3531	+2.43%
`+Description`	0.2827	+6.86%	0.4364†	+3.97%	0.3181	+6.04%	0.4306	+1.86%	**0.3691**†§*	+7.06%
`+Embed+Type`	0.2924†	+10.52%	0.4533†‡§¶	+8.00%	0.3034	+1.13%	0.4297	+1.65%	0.3544	+2.79%
`+Embed+Description`	0.2891	+9.29%	0.4443†‡	+5.85%	0.3197	+6.57%	0.4304	+1.80%	0.3564	+3.38%
`Full Model`	**0.3096**†‡§	+17.04%	**0.4547**†‡§¶	+8.32%	**0.3327**†*	+10.92%	**0.4341**†	+2.68%	0.3616†	+4.90%
`Conv-KNRM`	0.3357	–	0.4810	–	0.3384	–	0.4318	–	0.3582	–
`+Embed`	0.3382	+0.74%	**0.4831**	+0.44%	0.3450	+1.94%	0.4413	+2.20%	0.3758†	+4.91%
`+Type`	0.3370	+0.38%	0.4762	−0.99%	0.3422	+1.12%	0.4423†	+2.42%	0.3798†	+6.02%
`+Description`	0.3396	+1.15%	0.4807	−0.05%	0.3533	+4.41%	0.4468†	+3.47%	0.3819†	+6.61%
`+Embed+Type`	**0.3420**	+1.88%	0.4828	+0.39%	0.3546	+4.79%	0.4491†	+4.00%	0.3805†	+6.22%
`+Embed+Description`	0.3382	+0.73%	0.4805	−0.09%	0.3608	+6.60%	0.4494†	+4.08%	0.3868†	+7.99%
`Full Model`	0.3397	+1.19%	0.4821	+0.24%	**0.3708**†‡§	+9.57%	**0.4513**†‡	+4.51%	**0.3892**†‡	+8.65%



(a) Kernel weight distribution for EDRM-KNRM. (b) Kernel weight distribution for EDRM-CKNRM.

Figure 3: Ranking contribution for EDRM. Three scenarios are presented: Exact VS. Soft compares the weights of exact match kernel and others; Solo Word VS. Others shows the proportion of only text based matches; In-space VS. Cross-space compares in-space matches and cross-space matches.

These results show the effectiveness and the generalization ability of `EDRM`. In the following experiments, we study the source of this generalization ability.

6.2 Contributions of Matching Kernels

This experiment studies the contribution of knowledge graph semantics by investigating the weights learned on the different types of matching kernels.

As shown in Figure 3(a), most of the weight in `EDRM-KNRM` goes to soft match (Exact VS. Soft); entity related matches play an as important role as word based matches (Solo Word VS. Others); cross-space matches are more important than in-space matches (In-space VS. Cross-space). As shown in Figure 3(b), the percentages of word based matches and cross-space matches are more important in `EDRM-CKNRM` compared to in `EDRM-KNRM`.

The contribution of each individual match type in `EDRM-CKNRM` is shown in Figure 4. The weight of unigram, bigram, trigram, and entity is almost uniformly distributed, indicating the effectiveness of entities and all components are important in `EDRM-CKNRM`.

6.3 Ablation Study

This experiment studies which part of the knowledge graph semantics leads to the effectiveness and generalization ability of `EDRM`.

There are three types of embeddings incorporating different aspects of knowledge graph information: entity embedding (Embed), description embedding (Description) and type embedding (Type). This experiment starts with the word-only `K-NRM` and `Conv-KNRM`, and adds these three types of embedding individually or two-by-two (Embed+Type and Embed+Description).

The performances of `EDRM` with different groups of embeddings are shown in Table 2. The description embeddings show the greatest improvement among the three embeddings. Entity

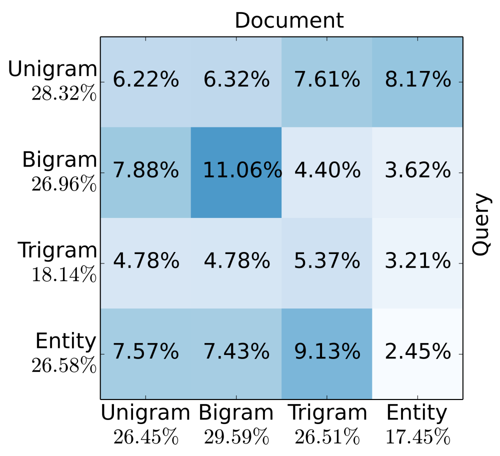


Figure 4: Individual kernel weight for EDRM-CKNRM. X-axis and y-axis denote document and query respectively.

type plays an important role only combined with other embeddings. Entity embedding improves `K-NRM` while has little effect on `Conv-KNRM`. This result further confirms that the signal from entity names are captured by the n-gram CNNs in `Conv-KNRM`. Incorporating all of three embeddings usually gets the best ranking performance.

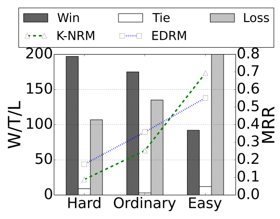
This experiments shows that knowledge graph semantics are crucial to `EDRM`'s effectiveness. `Conv-KNRM` learns good phrase matches that overlap with the entity embedding signals. However, the knowledge graph semantics (descriptions and types) is hard to be learned just from user clicks.

6.4 Performance on Different Scenarios

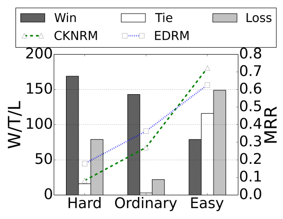
This experiment analyzes the influence of knowledge graphs in two different scenarios: multiple difficulty degrees and multiple length degrees.

Query Difficulty Experiment studies `EDRM`'s performance on testing queries at different difficulty, partitioned by `Conv-KNRM`'s MRR value: Hard (MRR $<$ 0.167), Ordinary (MRR $\in [0.167, 0.382]$, and Easy (MRR $>$ 0.382). As shown in Figure 5, `EDRM` performs the best on hard queries.

Query Length Experiment evaluates `EDRM`'s effectiveness on Short (1 words), Medium (2-3 words) and Long (4 or more words) queries. As shown in Figure 6, `EDRM` has more win cases and achieves the greatest improvement on short queries. Knowledge embeddings are more crucial when limited information is available from the original query text.

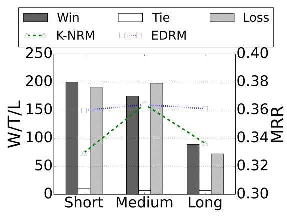


(a) K-NRM VS. EDRM

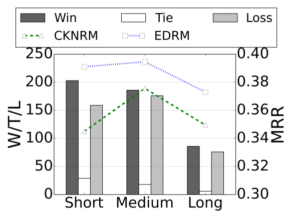


(b) Conv-KNRM VS. EDRM

Figure 5: Performance VS. Query Difficulty. The x-axises mark three query difficulty levels. The y-axises are the Win/Tie/Loss (left) and MRR (right) in the corresponding group.



(a) K-NRM VS. EDRM



(b) Conv-KNRM VS. EDRM

Figure 6: Performance VS. Query Length. The x-axises mark three query length levels, and y-axises are the Win/Tie/Loss (left) and MRR (right) in the corresponding group.

These two experiments reveal that the effectiveness of `EDRM` is more observed on harder or shorter queries, whereas the word-based neural models either find it difficult or do not have sufficient information to leverage.

6.5 Case Study

Table 3 provide examples reflecting two possible ways, in which the knowledge graph semantics could help the document ranking.

First, the entity descriptions explain the meaning of entities and connect them through the word space. *Meituxiuxiu web version* and *Meilishuo* are two websites providing image processing and shopping services respectively. Their descriptions provide extra ranking signals to promote the related documents.

Second, entity types establish underlying relevance patterns between query and documents. The underlying patterns can be captured by cross-space matches. For example, the types of the query entity *Crayon Shin-chan* and *GINTAMA* overlaps with the bag-of-words in the relevant documents. They can also be captured by the entity-based matches through their type overlaps,

Table 3: Examples of entity semantics connecting query and title. All the examples are correctly ranked by EDRM-CKNRM. Table 3a shows query-document pairs. Table 3b lists the related entity semantics that include useful information to match the query-document pair. The examples and related semantics are picked by manually examining the ranking changes between different variances of `EDRM-CKNRM`.

(a) Query and document examples. *Entities* are emphasized.

Query	Document
Meituxiuxiu web version	*Meituxiuxiu web version*: An online picture processing tools
Home page of *Meilishuo*	Home page of *Meilishuo* - Only the correct popular fashion
Master Lu	Master Lu official website: *System optimization*, hardware test, phone evaluation
Crayon Shin-chan: The movie	*Crayon Shin-chan*: The movie online - Anime
GINTAMA	*GINTAMA*: The movie online - Anime - Full HD online watch

(b) Semantics of related entities. The first two rows and last two rows show entity descriptions and entity types respectively.

Entity	Content
Meituxiuxiu web version	Description: Meituxiuxiu is the most popular Chinese image processing software, launched by the Meitu company
Meilishuo	Description: Meilishuo, the largest women's fashion e-commerce platform, dedicates to provide the most popular fashion shopping experience
Crayon Shin-chan, *GINTAMA*	Type: Anime; Cartoon characters; Comic
Master Lu, *System Optimization*	Type: Hardware test; Software; System tool

for example, between the query entity *Master Lu* and the document entity *System Optimization*.

7 Conclusions

This paper presents `EDRM`, the Entity-Duet Neural Ranking Model that incorporating knowledge graph semantics into neural ranking systems. `EDRM` inherits entity-oriented search to match query and documents with bag-of-words and bag-of-entities in neural ranking models. The knowledge graph semantics are integrated as distributed representations of entities. The neural model leverages these semantics to help document ranking. Using user clicks from search logs, the whole model—the integration of knowledge graph semantics and the neural ranking networks–is trained end-to-end. It leads to a data-driven combination of entity-oriented search and neural information retrieval.

Our experiments on the Sogou search log and CN-DBpedia demonstrate `EDRM`'s effectiveness and generalization ability over two state-of-the-art neural ranking models. Our further analyses reveal that the generalization ability comes from the integration of knowledge graph semantics. The neural ranking models can effectively model n-gram matches between query and document, which overlaps with part of the ranking signals from entity-based matches: Solely adding the entity names may not improve the ranking accuracy much. However, the knowledge graph semantics, introduced by the description and type embeddings, provide novel ranking signals that greatly improve the generalization ability of neural rankers in difficult scenarios.

This paper preliminarily explores the role of structured semantics in deep learning models. Though mainly fouced on search, we hope our findings shed some lights on a potential path towards more intelligent neural systems and will motivate more explorations in this direction.

Acknowledgments

This work[1] is supported by the Major Project of the National Social Science Foundation of China (No.13&ZD190) as well as the China-Singapore Joint Research Project of the National Natural Science Foundation of China (No. 61661146007) under the umbrella of the NexT Joint Research Center of Tsinghua University and National University of Singapore. Chenyan Xiong is supported by National Science Foundation (NSF) grant IIS-1422676. We thank Sogou for providing access to the search log.

[1]Source codes of this work are available at `http://github.com/thunlp/EntityDuetNeuralRanking`

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *DBpedia: A nucleus for a web of open data*. Springer.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*. ACM, pages 222–229.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008)*. ACM, pages 1247–1250.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*. ACM, pages 243–250.

Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7(3):1–115.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018)*. ACM, pages 126–134.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014)*. ACM, pages 365–374.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM, pages 65–74.

Laura Dietz and Patrick Verga. 2014. Umass at TREC 2014: Entity query feature expansion using knowledge base links. In *Proceedings of The 23st Text Retrieval Conference (TREC 2014)*. NIST.

Faezeh Ensan and Ebrahim Bagheri. 2017. Document retrieval model through semantic linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*. ACM, pages 181–190.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Scott Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Kristen Grauman and Trevor Darrell. 2005. The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. IEEE, volume 2, pages 1458–1465.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016a. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM 2016)*. ACM, pages 701–710.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W.Bruce Croft. 2016b. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM 2016)*. ACM, pages 55–64.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*. pages 2681–2690.

Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Entity linking in queries: Efficiency vs. effectiveness. In *European Conference on Information Retrieval*. Springer, pages 40–53.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS 2014)*. MIT Press, pages 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management (CIKM 2013)*. ACM, pages 2333–2338.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Pacrr: A position-aware neural ir model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*. pages 1060–1069.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*. ACM, pages 133–142.

Xitong Liu and Hui Fang. 2015. Latent entity space: A novel retrieval approach for entity-bearing queries. *Information Retrieval Journal* 18(6):473–503.

Cheng Luo, Yukun Zheng, Yiqun Liu, Xiaochuan Wang, Jingfang Xu, Min Zhang, and Shaoping Ma. 2017. Sogout-16: A new web corpus to embrace ir research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM, pages 1233–1236.

Donald Metzler and W. Bruce Croft. 2006. Linear feature-based models for information retrieval. *Information Retrieval* 10(3):257–274.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. pages 1400–1409.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*. ACM, pages 1291–1299.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*. pages 2793–2799.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM 2017)*. ACM, pages 257–266.

Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document retrieval using entity-based language models. In *Proceedings of the 39th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*. ACM, pages 65–74.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014)*. ACM, pages 101–110.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web (WWW 2007)*. ACM, pages 697–706.

Hongning Wang, ChengXiang Zhai, Anlei Dong, and Yi Chang. 2013. Content-aware click modeling. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW 2013)*. ACM, pages 1365–1376.

Chenyan Xiong and Jamie Callan. 2015. EsdRank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015)*. ACM, pages 951–960.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2016. Bag-of-entities representation for ranking. In *Proceedings of the sixth ACM International Conference on the Theory of Information Retrieval (ICTIR 2016)*. ACM, pages 181–184.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017a. Word-entity duet representations for document ranking. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM, pages 763–772.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017b. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM, pages 55–64.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017c. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*. ACM, pages 1271–1279.

Bo Xu, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui, and Yanghua Xiao. 2017. Cn-dbpedia: A never-ending chinese knowledge extraction system. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, pages 428–438.

Hua Ping Zhang, Hong Kui Yu, De Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *Sighan Workshop on Chinese Language Processing*. pages 758–759.

Neural Natural Language Inference Models Enhanced with External Knowledge

Qian Chen
University of Science and Technology of China
cq1231@mail.ustc.edu.cn

Xiaodan Zhu
ECE, Queen's University
xiaodan.zhu@queensu.ca

Zhen-Hua Ling
University of Science and Technology of China
zhling@ustc.edu.cn

Diana Inkpen
University of Ottawa
diana@site.uottawa.ca

Si Wei
iFLYTEK Research
siwei@iflytek.com

Abstract

Modeling natural language inference is a very challenging task. With the availability of large annotated data, it has recently become feasible to train complex models such as neural-network-based inference models, which have shown to achieve the state-of-the-art performance. Although there exist relatively large annotated data, can machines learn all knowledge needed to perform natural language inference (NLI) from these data? If not, how can neural-network-based NLI models benefit from external knowledge and how to build NLI models to leverage it? In this paper, we enrich the state-of-the-art neural natural language inference models with external knowledge. We demonstrate that the proposed models improve neural NLI models to achieve the state-of-the-art performance on the SNLI and MultiNLI datasets.

1 Introduction

Reasoning and inference are central to both human and artificial intelligence. Natural language inference (NLI), also known as recognizing textual entailment (RTE), is an important NLP problem concerned with determining inferential relationship (e.g., entailment, contradiction, or neutral) between a premise p and a hypothesis h. In general, modeling informal inference in language is a very challenging and basic problem towards achieving true natural language understanding.

In the last several years, larger annotated datasets were made available, e.g., the SNLI (Bowman et al., 2015) and MultiNLI datasets (Williams et al., 2017), which made it feasible to train rather complicated neural-network-based models that fit a large set of parameters to better model NLI. Such models have shown to achieve the state-of-the-art performance (Bowman et al., 2015, 2016; Yu and Munkhdalai, 2017b; Parikh et al., 2016; Sha et al., 2016; Chen et al., 2017a,b; Tay et al., 2018).

While neural networks have been shown to be very effective in modeling NLI with large training data, they have often focused on end-to-end training by assuming that all inference knowledge is learnable from the provided training data. In this paper, we relax this assumption and explore whether external knowledge can further help NLI. Consider an example:

- p: A lady standing in a *wheat* field.
- h: A person standing in a *corn* field.

In this simplified example, when computers are asked to predict the relation between these two sentences and if training data do not provide the knowledge of relationship between "wheat" and "corn" (e.g., if one of the two words does not appear in the training data or they are not paired in any premise-hypothesis pairs), it will be hard for computers to correctly recognize that the premise contradicts the hypothesis.

In general, although in many tasks learning *tabula rasa* achieved state-of-the-art performance, we believe complicated NLP problems such as NLI could benefit from leveraging knowledge accumulated by humans, particularly in a foreseeable future when machines are unable to learn it by themselves.

In this paper we enrich neural-network-based NLI models with external knowledge in co-attention, local inference collection, and inference composition components. We show the proposed model improves the state-of-the-art NLI models to achieve better performances on the SNLI and MultiNLI datasets. The advantage of using external knowledge is more significant when the size of training data is restricted, suggesting that if more knowledge can be obtained, it may bring more benefit. In addition to attaining the state-of-the-art performance, we are also interested in understanding how external knowledge contributes to the major components of typical neural-network-based NLI models.

2 Related Work

Early research on natural language inference and recognizing textual entailment has been performed on relatively small datasets (refer to MacCartney (2009) for a good literature survey), which includes a large bulk of contributions made under the name of RTE, such as (Dagan et al., 2005; Iftene and Balahur-Dobrescu, 2007), among many others.

More recently the availability of much larger annotated data, e.g., SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017), has made it possible to train more complex models. These models mainly fall into two types of approaches: sentence-encoding-based models and models using also inter-sentence attention. Sentence-encoding-based models use Siamese architecture (Bromley et al., 1993). The parameter-tied neural networks are applied to encode both the premise and the hypothesis. Then a neural network classifier is applied to decide relationship between the two sentences. Different neural networks have been utilized for sentence encoding, such as LSTM (Bowman et al., 2015), GRU (Vendrov et al., 2015), CNN (Mou et al., 2016), BiLSTM and its variants (Liu et al., 2016c; Lin et al., 2017; Chen et al., 2017b; Nie and Bansal, 2017), self-attention network (Shen et al., 2017, 2018), and more complicated neural networks (Bowman et al., 2016; Yu and Munkhdalai, 2017a,b; Choi et al., 2017). Sentence-encoding-based models transform sentences into fixed-length vector representations, which may help a wide range of tasks (Conneau et al., 2017).

The second set of models use inter-sentence attention (Rocktäschel et al., 2015; Wang and Jiang, 2016; Cheng et al., 2016; Parikh et al., 2016; Chen et al., 2017a). Among them, Rocktäschel et al. (2015) were among the first to propose neural attention-based models for NLI. Chen et al. (2017a) proposed an enhanced sequential inference model (ESIM), which is one of the best models so far and is used as one of our baselines in this paper.

In this paper we enrich neural-network-based NLI models with external knowledge. Unlike early work on NLI (Jijkoun and de Rijke, 2005; MacCartney et al., 2008; MacCartney, 2009) that explores external knowledge in conventional NLI models on relatively small NLI datasets, we aim to merge the advantage of powerful modeling ability of neural networks with extra external inference knowledge. We show that the proposed model improves the state-of-the-art neural NLI models to achieve better performances on the SNLI and MultiNLI datasets. The advantage of using external knowledge is more significant when the size of training data is restricted, suggesting that if more knowledge can be obtained, it may have more benefit. In addition to attaining the state-of-the-art performance, we are also interested in understanding how external knowledge affect major components of neural-network-based NLI models.

In general, external knowledge has shown to be effective in neural networks for other NLP tasks, including word embedding (Chen et al., 2015; Faruqui et al., 2015; Liu et al., 2015; Wieting et al., 2015; Mrksic et al., 2017), machine translation (Shi et al., 2016; Zhang et al., 2017b), language modeling (Ahn et al., 2016), and dialogue systems (Chen et al., 2016b).

3 Neural-Network-Based NLI Models with External Knowledge

In this section we propose neural-network-based NLI models to incorporate external inference knowledge, which, as we will show later in Section 5, achieve the state-of-the-art performance. In addition to attaining the leading performance we are also interested in investigating the effects of external knowledge on major components of neural-network-based NLI modeling.

Figure 1 shows a high-level general view of the proposed framework. While specific NLI systems vary in their implementation, typical state-of-the-art NLI models contain the main components (or equivalents) of representing premise and hypothesis sentences, collecting local (e.g., lexical) inference information, and aggregating and composing local information to make the global decision at the sentence level. We incorporate and investigate external knowledge accordingly in these major NLI components: computing co-attention, collecting local inference information, and composing inference to make final decision.

3.1 External Knowledge

As discussed above, although there exist relatively large annotated data for NLI, can machines learn all inference knowledge needed to perform NLI from the data? If not, how can neural network-based NLI models benefit from external knowledge and how to build NLI models to leverage it?

We study the incorporation of external, inference-related knowledge in major components of neural networks for natural language inference. For example, intuitively knowledge about *synonymy*, *antonymy*, *hypernymy* and *hyponymy* between given words may help model soft-alignment between premises and hypotheses; knowledge about *hypernymy* and *hyponymy* may help capture entailment; knowledge about *antonymy* and *co-hyponyms* (words sharing the same hypernym) may benefit the modeling of contradiction.

In this section, we discuss the incorporation of basic, lexical-level semantic knowledge into neural NLI components. Specifically, we consider external lexical-level inference knowledge between word w_i and w_j, which is represented as a vector $\boldsymbol{r}_{ij}$ and is incorporated into three specific components shown in Figure 1. We will discuss the details of how $\boldsymbol{r}_{ij}$ is constructed later in the experiment setup section (Section 4) but instead focus on the proposed model in this section. Note that while we study lexical-level inference knowledge in the paper, if inference knowledge about larger pieces of text pairs (e.g., inference relations between phrases) are available, the proposed model can be easily extended to handle that. In this paper, we instead let the NLI models to compose lexical-level knowledge to obtain inference relations between larger pieces of texts.

3.2 Encoding Premise and Hypothesis

Same as much previous work (Chen et al., 2017a,b), we encode the premise and the hypothesis with bidirectional LSTMs (BiLSTMs). The premise is represented as $\boldsymbol{a} = (a_1, \ldots, a_m)$ and the hypothesis is $\boldsymbol{b} = (b_1, \ldots, b_n)$, where m and n are the lengths of the sentences. Then $\boldsymbol{a}$ and $\boldsymbol{b}$ are embedded into d_e-dimensional vectors $[\mathbf{E}(a_1), \ldots, \mathbf{E}(a_m)]$ and $[\mathbf{E}(b_1), \ldots, \mathbf{E}(b_n)]$ using the embedding matrix $\mathbf{E} \in \mathbb{R}^{d_e \times |V|}$, where $|V|$ is the vocabulary size and $\mathbf{E}$ can be initialized with the pre-trained word embedding. To represent words in its context, the premise and the hypothesis are fed into BiLSTM encoders (Hochreiter and Schmidhuber, 1997) to obtain context-dependent hidden states $\boldsymbol{a}^s$ and $\boldsymbol{b}^s$:

$$\boldsymbol{a}_i^s = \text{Encoder}(\mathbf{E}(\boldsymbol{a}), i)\,, \tag{1}$$

$$\boldsymbol{b}_j^s = \text{Encoder}(\mathbf{E}(\boldsymbol{b}), j)\,. \tag{2}$$

where i and j indicate the i-th word in the premise and the j-th word in the hypothesis, respectively.

3.3 Knowledge-Enriched Co-Attention

As discussed above, soft-alignment of word pairs between the premise and the hypothesis may benefit from knowledge-enriched co-attention mechanism. Given the relation features $\boldsymbol{r}_{ij} \in \mathbb{R}^{d_r}$ between the premise's i-th word and the hypothesis's j-th word derived from the external knowledge, the co-attention is calculated as:

$$e_{ij} = (\boldsymbol{a}_i^s)^{\mathrm{T}} \boldsymbol{b}_j^s + F(\boldsymbol{r}_{ij})\,. \tag{3}$$

The function F can be any non-linear or linear functions. In this paper, we use $F(\boldsymbol{r}_{ij}) = \lambda \mathbb{1}(\boldsymbol{r}_{ij})$, where λ is a hyper-parameter tuned on the development set and $\mathbb{1}$ is the indication function as follows:

$$\mathbb{1}(\boldsymbol{r}_{ij}) = \begin{cases} 1 & \text{if } \boldsymbol{r}_{ij} \text{ is not a zero vector}\,; \\ 0 & \text{if } \boldsymbol{r}_{ij} \text{ is a zero vector}\,. \end{cases} \tag{4}$$

Intuitively, word pairs with semantic relationship, e.g., synonymy, antonymy, hypernymy, hyponymy and co-hyponyms, are probably aligned together. We will discuss how we construct external knowledge later in Section 4. We have also tried a two-layer MLP as a universal function approximator in function F to learn the underlying combination function but did not observe further improvement over the best performance we obtained on the development datasets.

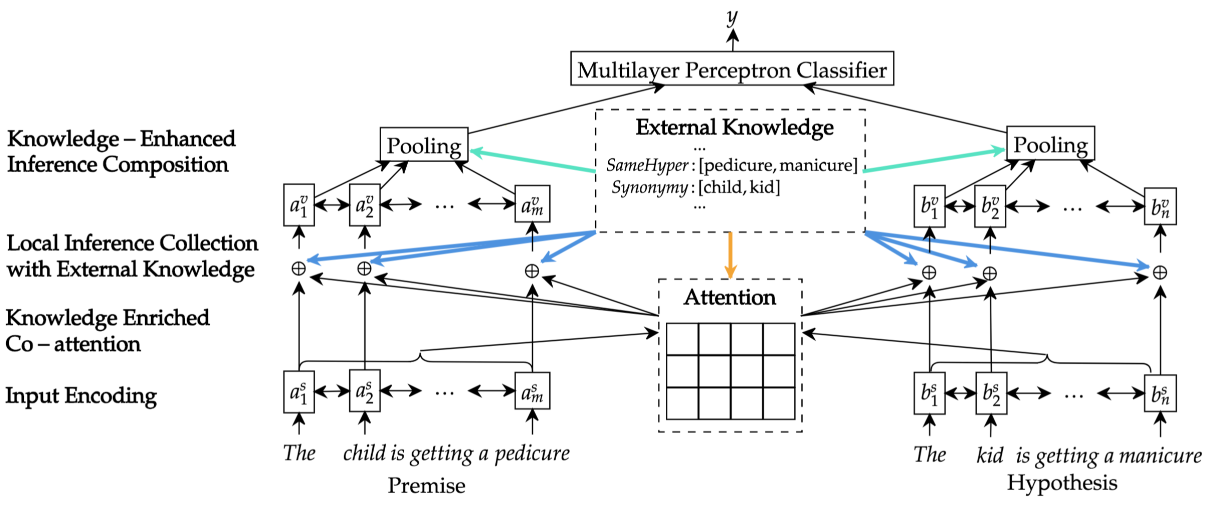

y
Multilayer Perceptron Classifier
Knowledge – Enhanced Inference Composition
Pooling
External Knowledge
SameHyper: [pedicure, manicure]
Synonymy: [child, kid]
Pooling
Local Inference Collection with External Knowledge
Knowledge Enriched Co – attention
Attention
Input Encoding
The child is getting a pedicure
Premise
The kid is getting a manicure
Hypothesis

Figure 1: A high-level view of neural-network-based NLI models enriched with external knowledge in co-attention, local inference collection, and inference composition.

Soft-alignment is determined by the co-attention matrix $\mathbf{e} \in \mathbb{R}^{m\times n}$ computed in Equation (3), which is used to obtain the local relevance between the premise and the hypothesis. For the hidden state of the i-th word in the premise, i.e., $\boldsymbol{a}_i^s$ (already encoding the word itself and its context), the relevant semantics in the hypothesis is identified into a context vector $\boldsymbol{a}_i^c$ using e_{ij}, more specifically with Equation (5).

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n}\exp(e_{ik})}\,,\ \boldsymbol{a}_i^c = \sum_{j=1}^{n}\alpha_{ij}\boldsymbol{b}_j^s\,, \quad (5)$$

$$\beta_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{m}\exp(e_{kj})}\,,\ \boldsymbol{b}_j^c = \sum_{i=1}^{m}\beta_{ij}\boldsymbol{a}_i^s\,, \quad (6)$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{\beta} \in \mathbb{R}^{m\times n}$ are the normalized attention weight matrices with respect to the 2-axis and 1-axis. The same calculation is performed for each word in the hypothesis, i.e., $\boldsymbol{b}_j^s$, with Equation (6) to obtain the context vector $\boldsymbol{b}_j^c$.

3.4 Local Inference Collection with External Knowledge

By way of comparing the inference-related semantic relation between $\boldsymbol{a}_i^s$ (individual word representation in premise) and $\boldsymbol{a}_i^c$ (context representation from hypothesis which is align to word $\boldsymbol{a}_i^s$), we can model local inference (i.e., word-level inference) between aligned word pairs. Intuitively, for example, knowledge about hypernymy or hyponymy may help model entailment and knowledge about antonymy and co-hyponyms may help model contradiction. Through comparing $\boldsymbol{a}_i^s$ and $\boldsymbol{a}_i^c$, in addition to their relation from external knowledge, we can obtain word-level inference information for each word. The same calculation is performed for $\boldsymbol{b}_j^s$ and $\boldsymbol{b}_j^c$. Thus, we collect knowledge-enriched local inference information:

$$\boldsymbol{a}_i^m = G([\boldsymbol{a}_i^s; \boldsymbol{a}_i^c; \boldsymbol{a}_i^s - \boldsymbol{a}_i^c; \boldsymbol{a}_i^s \circ \boldsymbol{a}_i^c; \sum_{j=1}^{n}\alpha_{ij}\boldsymbol{r}_{ij}])\,, \quad (7)$$

$$\boldsymbol{b}_j^m = G([\boldsymbol{b}_j^s, \boldsymbol{b}_j^c; \boldsymbol{b}_j^s - \boldsymbol{b}_j^c; \boldsymbol{b}_j^s \circ \boldsymbol{b}_j^c; \sum_{i=1}^{m}\beta_{ij}\boldsymbol{r}_{ji}])\,, \quad (8)$$

where a heuristic matching trick with difference and element-wise product is used (Mou et al., 2016; Chen et al., 2017a). The last terms in Equation (7)(8) are used to obtain word-level inference information from external knowledge. Take Equation (7) as example, $\boldsymbol{r}_{ij}$ is the relation feature between the i-th word in the premise and the j-th word in the hypothesis, but we care more about semantic relation between aligned word pairs between the premise and the hypothesis. Thus, we use a soft-aligned version through the soft-alignment weight α_{ij}. For the i-th word in the premise, the last term in Equation (7) is a word-level inference information based on external knowledge between the i-th word and the aligned word. The same calculation for hypothesis is performed in Equation (8). G is a non-linear mapping function to reduce dimensionality. Specifically, we use a 1-layer feed-forward neural network with the ReLU activation function with a shortcut connection, i.e., concatenate the hidden states after ReLU with the input $\sum_{j=1}^{n}\alpha_{ij}\boldsymbol{r}_{ij}$ (or $\sum_{i=1}^{m}\beta_{ij}\boldsymbol{r}_{ji}$) as the output $\boldsymbol{a}_i^m$ (or $\boldsymbol{b}_j^m$).

3.5 Knowledge-Enhanced Inference Composition

In this component, we introduce knowledge-enriched inference composition. To determine the overall inference relationship between the premise and the hypothesis, we need to explore a composition layer to compose the local inference vectors ($\boldsymbol{a}^m$ and $\boldsymbol{b}^m$) collected above:

$$\boldsymbol{a}_i^v = \text{Composition}(\boldsymbol{a}^m, i)\,, \tag{9}$$

$$\boldsymbol{b}_j^v = \text{Composition}(\boldsymbol{b}^m, j)\,. \tag{10}$$

Here, we also use BiLSTMs as building blocks for the composition layer, but the responsibility of BiLSTMs in the inference composition layer is completely different from that in the input encoding layer. The BiLSTMs here read local inference vectors ($\boldsymbol{a}^m$ and $\boldsymbol{b}^m$) and learn to judge the types of local inference relationship and distinguish crucial local inference vectors for overall sentence-level inference relationship. Intuitively, the final prediction is likely to depend on word pairs appearing in external knowledge that have some semantic relation. Our inference model converts the output hidden vectors of BiLSTMs to the fixed-length vector with pooling operations and puts it into the final classifier to determine the overall inference class. Particularly, in addition to using mean pooling and max pooling similarly to ESIM (Chen et al., 2017a), we propose to use weighted pooling based on external knowledge to obtain a fixed-length vector as in Equation (11)(12).

$$\boldsymbol{a}^{\mathrm{w}} = \sum_{i=1}^{m} \frac{\exp(H(\sum_{j=1}^{n} \alpha_{ij}\boldsymbol{r}_{ij}))}{\sum_{i=1}^{m}\exp(H(\sum_{j=1}^{n} \alpha_{ij}\boldsymbol{r}_{ij}))}\boldsymbol{a}_i^v\,, \tag{11}$$

$$\boldsymbol{b}^{\mathrm{w}} = \sum_{j=1}^{n} \frac{\exp(H(\sum_{i=1}^{m} \beta_{ij}\boldsymbol{r}_{ji}))}{\sum_{j=1}^{n}\exp(H(\sum_{i=1}^{m} \beta_{ij}\boldsymbol{r}_{ji}))}\boldsymbol{b}_j^v\,. \tag{12}$$

In our experiments, we regard the function H as a 1-layer feed-forward neural network with ReLU activation function. We concatenate all pooling vectors, i.e., mean, max, and weighted pooling, into the fixed-length vector and then put the vector into the final multilayer perceptron (MLP) classifier. The MLP has one hidden layer with *tanh* activation and *softmax* output layer in our experiments. The entire model is trained end-to-end, through minimizing the cross-entropy loss.

4 Experiment Set-Up

4.1 Representation of External Knowledge

Lexical Semantic Relations As described in Section 3.1, to incorporate external knowledge (as a knowledge vector $\boldsymbol{r}_{ij}$) to the state-of-the-art neural network-based NLI models, we first explore semantic relations in WordNet (Miller, 1995), motivated by MacCartney (2009). Specifically, the relations of lexical pairs are derived as described in (1)-(4) below. Instead of using Jiang-Conrath WordNet distance metric (Jiang and Conrath, 1997), which does not improve the performance of our models on the development sets, we add a new feature, i.e., *co-hyponyms*, which consistently benefit our models.

(1) *Synonymy*: It takes the value 1 if the words in the pair are synonyms in WordNet (i.e., belong to the same synset), and 0 otherwise. For example, [felicitous, good] = 1, [dog, wolf] = 0.

(2) *Antonymy*: It takes the value 1 if the words in the pair are antonyms in WordNet, and 0 otherwise. For example, [wet, dry] = 1.

(3) *Hypernymy*: It takes the value $1 - n/8$ if one word is a (direct or indirect) hypernym of the other word in WordNet, where n is the number of edges between the two words in hierarchies, and 0 otherwise. Note that we ignore pairs in the hierarchy which have more than 8 edges in between. For example, [dog, canid] = 0.875, [wolf, canid] = 0.875, [dog, carnivore] = 0.75, [canid, dog] = 0

(4) *Hyponymy*: It is simply the inverse of the hypernymy feature. For example, [canid, dog] = 0.875, [dog, canid] = 0.

(5) *Co-hyponyms*: It takes the value 1 if the two words have the same hypernym but they do not belong to the same synset, and 0 otherwise. For example, [dog, wolf] = 1.

As discussed above, we expect features like *synonymy*, *antonymy*, *hypernymy*, *hyponymy* and *co-hyponyms* would help model co-attention alignment between the premise and the hypothesis. Knowledge of *hypernymy* and *hyponymy* may help capture entailment; knowledge of *antonymy* and *co-hyponyms* may help model contradiction. Their final contributions will be learned in end-to-end model training. We regard the vector $\boldsymbol{r} \in \mathbb{R}^{d_r}$ as the relation feature derived from external knowledge, where d_r is 5 here. In addition, Table 1 reports some key statistics of these features.

Feature	#Words	#Pairs
Synonymy	84,487	237,937
Antonymy	6,161	6,617
Hypernymy	57,475	753,086
Hyponymy	57,475	753,086
Co-hyponyms	53,281	3,674,700

Table 1: Statistics of lexical relation features.

In addition to the above relations, we also use more relation features in WordNet, including *instance*, *instance of*, *same instance*, *entailment*, *member meronym*, *member holonym*, *substance meronym*, *substance holonym*, *part meronym*, *part holonym*, summing up to 15 features, but these additional features do not bring further improvement on the development dataset, as also discussed in Section 5.

Relation Embeddings In the most recent years graph embedding has been widely employed to learn representation for vertexes and their relations in a graph. In our work here, we also capture the relation between any two words in WordNet through relation embedding. Specifically, we employed TransE (Bordes et al., 2013), a widely used graph embedding methods, to capture relation embedding between any two words. We used two typical approaches to obtaining the relation embedding. The first directly uses 18 relation embeddings pretrained on the WN18 dataset (Bordes et al., 2013). Specifically, if a word pair has a certain type relation, we take the corresponding relation embedding. Sometimes, if a word pair has multiple relations among the 18 types; we take an average of the relation embedding. The second approach uses TransE's word embedding (trained on WordNet) to obtain relation embedding, through the objective function used in TransE, i.e., $\boldsymbol{l} \approx \boldsymbol{t} - \boldsymbol{h}$, where $\boldsymbol{l}$ indicates relation embedding, $\boldsymbol{t}$ indicates tail entity embedding, and $\boldsymbol{h}$ indicates head entity embedding.

Note that in addition to relation embedding trained on WordNet, other relational embedding resources exist; e.g., that trained on Freebase (WikiData) (Bollacker et al., 2007), but such knowledge resources are mainly about facts (e.g., relationship between Bill Gates and Microsoft) and are less for commonsense knowledge used in general natural language inference (e.g., the color yellow potentially contradicts red).

4.2 NLI Datasets

In our experiments, we use Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) and Multi-Genre Natural Language Inference (MultiNLI) (Williams et al., 2017) dataset, which focus on three basic relations between a premise and a potential hypothesis: the premise entails the hypothesis (*entailment*), they contradict each other (*contradiction*), or they are not related (*neutral*). We use the same data split as in previous work (Bowman et al., 2015; Williams et al., 2017) and classification accuracy as the evaluation metric. In addition, we test our models (trained on the SNLI training set) on a new test set (Glockner et al., 2018), which assesses the lexical inference abilities of NLI systems and consists of 8,193 samples. WordNet 3.0 (Miller, 1995) is used to extract semantic relation features between words. The words are lemmatized using Stanford CoreNLP 3.7.0 (Manning et al., 2014). The premise and the hypothesis sentences fed into the input encoding layer are tokenized.

4.3 Training Details

For duplicability, we release our code[1]. All our models were strictly selected on the development set of the SNLI data and the in-domain development set of MultiNLI and were then tested on the corresponding test set. The main training details are as follows: the dimension of the hidden states of LSTMs and word embeddings are 300. The word embeddings are initialized by *300D GloVe 840B* (Pennington et al., 2014), and out-of-vocabulary words among them are initialized randomly. All word embeddings are updated during training. Adam (Kingma and Ba, 2014) is used for optimization with an initial learning rate of 0.0004. The mini-batch size is set to 32. Note that the above hyperparameter settings are same as those used in the baseline ESIM (Chen et al., 2017a) model. ESIM is a strong NLI baseline framework with the source code made available at *https://github.com/lukecq1231/nli* (the ESIM core code has also been adapted to summarization (Chen et al., 2016a) and question-answering tasks (Zhang et al., 2017a)).

The trade-off λ for calculating co-

[1]https://github.com/lukecq1231/kim

attention in Equation (3) is selected in $[0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50]$ based on the development set. When training TransE for WordNet, relations are represented with vectors of 20 dimension.

5 Experimental Results

5.1 Overall Performance

Table 2 shows the results of state-of-the-art models on the SNLI dataset. Among them, ESIM (Chen et al., 2017a) is one of the previous state-of-the-art systems with an 88.0% test-set accuracy. The proposed model, namely Knowledge-based Inference Model (**KIM**), which enriches ESIM with external knowledge, obtains an accuracy of 88.6%, the best single-model performance reported on the SNLI dataset. The difference between ESIM and KIM is statistically significant under the one-tailed paired t-test at the 99% significance level. Note that the KIM model reported here uses five semantic relations described in Section 4. In addition to that, we also use 15 semantic relation features, which does not bring additional gains in performance. These results highlight the effectiveness of the five semantic relations described in Section 4. To further investigate external knowledge, we add TransE relation embedding, and again no further improvement is observed on both the development and test sets when TransE relation embedding is used (concatenated) with the semantic relation vectors. We consider this is due to the fact that TransE embedding is not specifically sensitive to inference information; e.g., it does not model co-hyponyms features, and its potential benefit has already been covered by the semantic relation features used.

Table 3 shows the performance of models on the MultiNLI dataset. The baseline ESIM achieves 76.8% and 75.8% on in-domain and cross-domain test set, respectively. If we extend the ESIM with external knowledge, we achieve significant gains to 77.2% and 76.4% respectively. Again, the gains are consistent on SNLI and MultiNLI, and we expect they would be orthogonal to other factors when external knowledge is added into other state-of-the-art models.

5.2 Ablation Results

Figure 2 displays the ablation analysis of different components when using the external knowledge. To compare the effects of external knowledge under different training data scales, we randomly sample different ratios of the entire training set, i.e., 0.8%, 4%, 20% and 100%. "A" indicates adding external knowledge in calculating the co-attention matrix as in Equation (3), "I" indicates adding external knowledge in collecting local inference information as in Equation (7)(8), and "C" indicates adding external knowledge in composing inference as in Equation (11)(12). When we only have restricted training data, i.e., 0.8% training set (about 4,000 samples), the baseline ESIM has a poor accuracy of 62.4%. When we only add external knowledge in calculating co-attention ("A"), the accuracy increases to 66.6% (+ absolute 4.2%). When we only utilize external knowledge in collecting local inference information ("I"), the accuracy has a significant gain, to 70.3% (+ absolute 7.9%). When we only add external knowledge in inference composition ("C"), the accuracy gets a smaller gain to 63.4% (+ absolute 1.0%). The comparison indicates that "I" plays the most important role among the three components in using external knowledge. Moreover, when we com-

Model	**Test**
LSTM Att. (Rocktäschel et al., 2015)	83.5
DF-LSTMs (Liu et al., 2016a)	84.6
TC-LSTMs (Liu et al., 2016b)	85.1
Match-LSTM (Wang and Jiang, 2016)	86.1
LSTMN (Cheng et al., 2016)	86.3
Decomposable Att. (Parikh et al., 2016)	86.8
NTI (Yu and Munkhdalai, 2017b)	87.3
Re-read LSTM (Sha et al., 2016)	87.5
BiMPM (Wang et al., 2017)	87.5
DIIN (Gong et al., 2017)	88.0
BCN + CoVe (McCann et al., 2017)	88.1
CAFE (Tay et al., 2018)	88.5
ESIM (Chen et al., 2017a)	88.0
KIM (This paper)	**88.6**

Table 2: Accuracies of models on SNLI.

Model	**In**	**Cross**
CBOW (Williams et al., 2017)	64.8	64.5
BiLSTM (Williams et al., 2017)	66.9	66.9
DiSAN (Shen et al., 2017)	71.0	71.4
Gated BiLSTM (Chen et al., 2017b)	73.5	73.6
SS BiLSTM (Nie and Bansal, 2017)	74.6	73.6
DIIN * (Gong et al., 2017)	77.8	**78.8**
CAFE (Tay et al., 2018)	**78.7**	77.9
ESIM (Chen et al., 2017a)	76.8	75.8
KIM (This paper)	**77.2**	**76.4**

Table 3: Accuracies of models on MultiNLI. * indicates models using extra SNLI training set.

pose the three components ("A,I,C"), we obtain the best result of 72.6% (+ absolute 10.2%). When we use more training data, i.e., 4%, 20%, 100% of the training set, only "I" achieves a significant gain, but "A" or "C" does not bring any significant improvement. The results indicate that external semantic knowledge only helps co-attention and composition when limited training data is limited, but always helps in collecting local inference information. Meanwhile, for less training data, λ is usually set to a larger value. For example, the optimal λ on the development set is 20 for 0.8% training set, 2 for the 4% training set, 1 for the 20% training set and 0.2 for the 100% training set.

Figure 3 displays the results of using different ratios of external knowledge (randomly keep different percentages of whole lexical semantic relations) under different sizes of training data. Note that here we only use external knowledge in collecting local inference information as it always works well for different scale of the training set. Better accuracies are achieved when using more external knowledge. Especially under the condition of restricted training data (0.8%), the model obtains a large gain when using more than half of external knowledge.

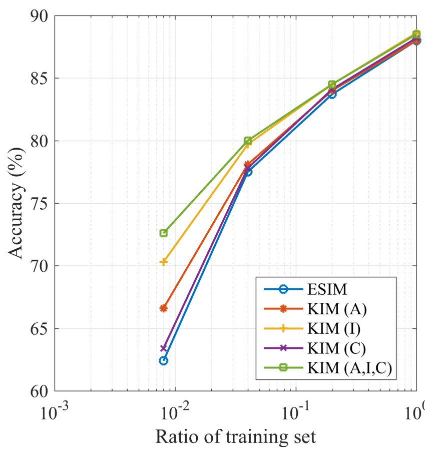


Figure 2: Accuracies of models of incorporating external knowledge into different NLI components, under different sizes of training data (0.8%, 4%, 20%, and the entire training data).

5.3 Analysis on the (Glockner et al., 2018) Test Set

In addition, Table 4 shows the results on a newly published test set (Glockner et al., 2018). Compared with the performance on the SNLI test

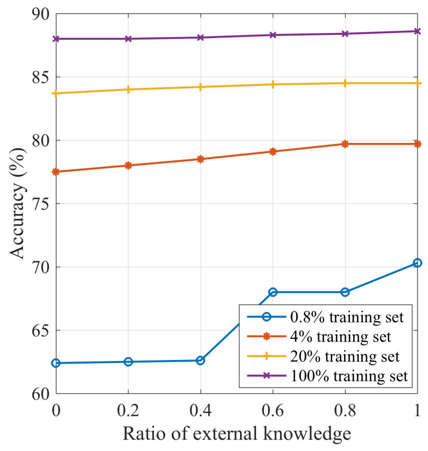


Figure 3: Accuracies of models under different sizes of external knowledge. More external knowledge corresponds to higher accuracies.

Model	SNLI	Glockner's(Δ)
(Parikh et al., 2016)*	84.7	51.9 (-32.8)
(Nie and Bansal, 2017)*	86.0	62.2 (-23.8)
ESIM *	87.9	65.6 (-22.3)
KIM (This paper)	**88.6**	**83.5** (-5.1)

Table 4: Accuracies of models on the SNLI and (Glockner et al., 2018) test set. * indicates the results taken from (Glockner et al., 2018).

set, the performance of the three baseline models dropped substantially on the (Glockner et al., 2018) test set, with the differences ranging from 22.3% to 32.8% in accuracy. Instead, the proposed KIM achieves 83.5% on this test set (with only a 5.1% drop in performance), which demonstrates its better ability of utilizing lexical level inference and hence better generalizability.

Figure 5 displays the accuracy of ESIM and KIM in each replacement-word category of the (Glockner et al., 2018) test set. KIM outperforms ESIM in 13 out of 14 categories, and only performs worse on synonyms.

5.4 Analysis by Inference Categories

We perform more analysis (Table 6) using the supplementary annotations provided by the MultiNLI dataset (Williams et al., 2017), which have 495 samples (about 1/20 of the entire development set) for both in-domain and out-domain set. We compare against the model outputs of the ESIM model across 13 categories of inference. Table 6 reports the results. We can see that KIM outperforms ESIM on overall accuracies on both in-domain and

Category	Instance	ESIM	KIM
Antonyms	1,147	70.4	**86.5**
Cardinals	759	75.5	**93.4**
Nationalities	755	35.9	**73.5**
Drinks	731	63.7	**96.6**
Antonyms WordNet	706	74.6	**78.8**
Colors	699	96.1	**98.3**
Ordinals	663	21.0	**56.6**
Countries	613	25.4	**70.8**
Rooms	595	69.4	**77.6**
Materials	397	89.7	**98.7**
Vegetables	109	31.2	**79.8**
Instruments	65	90.8	**96.9**
Planets	60	3.3	**5.0**
Synonyms	894	**99.7**	92.1
Overall	8,193	65.6	**83.5**

Table 5: The number of instances and accuracy per category achieved by ESIM and KIM on the (Glockner et al., 2018) test set.

Category	In-domain		Cross-domain	
	ESIM	KIM	ESIM	KIM
Active/Passive	**93.3**	**93.3**	**100.0**	**100.0**
Antonym	**76.5**	**76.5**	70.0	**75.0**
Belief	72.7	**75.8**	75.9	**79.3**
Conditional	**65.2**	**65.2**	61.5	**69.2**
Coreference	**80.0**	76.7	**75.9**	**75.9**
Long sentence	**82.8**	78.8	69.7	**73.4**
Modal	**80.6**	79.9	77.0	**80.2**
Negation	76.7	**79.8**	**73.1**	71.2
Paraphrase	**84.0**	72.0	86.5	**89.2**
Quantity/Time	**66.7**	**66.7**	56.4	**59.0**
Quantifier	**79.2**	78.4	73.6	**77.1**
Tense	74.5	**78.4**	**72.2**	66.7
Word overlap	**89.3**	85.7	**83.8**	81.1
Overall	77.1	**77.9**	76.7	**77.4**

Table 6: Detailed Analysis on MultiNLI.

cross-domain subset of development set. KIM outperforms or equals ESIM in 10 out of 13 categories on the cross-domain setting, while only 7 out of 13 categories on in-domain setting. It indicates that external knowledge helps more in cross-domain setting. Especially, for antonym category in cross-domain set, KIM outperform ESIM significantly (+ absolute 5.0%) as expected, because antonym feature captured by external knowledge would help unseen cross-domain samples.

5.5 Case Study

Table 7 includes some examples from the SNLI test set, where KIM successfully predicts the inference relation and ESIM fails. In the first example, the premise is "An African person standing in a **wheat** field" and the hypothesis "A person standing in a **corn** field". As the KIM model knows that "wheat" and "corn" are both a kind of cereal, i.e, the *co-hyponyms* relationship in our relation features, KIM therefore predicts the premise contradicts the hypothesis. However, the baseline ESIM cannot learn the relationship between "wheat" and "corn" effectively due to lack of enough samples in the training sets. With the help of external knowledge, i.e., "wheat" and "corn" having the same hypernym "cereal", KIM predicts contradiction correctly.

P/G	Sentences
e/c	*p*: An African person standing in a **wheat** field. *h*: A person standing in a **corn** field.
e/c	*p*: Little girl is flipping an **omelet** in the kitchen. *h*: A young girl cooks **pancakes**.
c/e	*p*: A middle eastern **marketplace**. *h*: A middle easten **store**.
c/e	*p*: Two boys are swimming with boogie **boards**. *h*: Two boys are swimming with their **floats**.

Table 7: Examples. Word in bold are key words in making final prediction. **P** indicates a predicted label and **G** indicates gold-standard label. *e* and *c* denote *entailment* and *contradiction*, respectively.

6 Conclusions

Our neural-network-based model for natural language inference with external knowledge, namely KIM, achieves the state-of-the-art accuracies. The model is equipped with external knowledge in its main components, specifically, in calculating co-attention, collecting local inference, and composing inference. We provide detailed analyses on our model and results. The proposed model of infusing neural networks with external knowledge may also help shed some light on tasks other than NLI.

Acknowledgments

We thank Yibo Sun and Bing Qin for early helpful discussion.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR*, abs/1608.00318.

Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. 2007. Freebase: A shared database of structured general human knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1962–1963.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2787–2795.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 737–744.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016a. Distraction-based neural networks for modeling document. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2754–2760.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017a. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 36–40.

Yun-Nung Chen, Dilek Z. Hakkani-Tür, Gökhan Tür, Asli Çelikyilmaz, Jianfeng Gao, and Li Deng. 2016b. Knowledge as a teacher: Knowledge-guided structural attention networks. *CoRR*, abs/1609.03286.

Zhigang Chen, Wei Lin, Qian Chen, Xiaoping Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. 2015. Revisiting word embedding for contrasting meaning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 106–115.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *CoRR*, abs/1707.02786.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pages 177–190.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1606–1615.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *CoRR*, abs/1709.04348.

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

Adrian Iftene and Alexandra Balahur-Dobrescu. 2007. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, chapter Hypothesis Transformation and Semantic Variability Rules Used in Recognizing Textual Entailment. Association for Computational Linguistics.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th Research on Computational Linguistics International Conference, ROCLING 1997, Taipei, Taiwan, August 1997*, pages 19–33.

Valentin Jijkoun and Maarten de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion lstms for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016b. Modelling interaction of sentence pair with coupled-lstms. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1703–1712.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1501–1511.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016c. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 802–811.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6297–6308.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Nikola Mrksic, Ivan Vulic, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gasic, Anna Korhonen, and Steve J. Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *CoRR*, abs/1706.00374.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 41–45.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29,* *2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2870–2879.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *CoRR*, abs/1801.10296.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. A compare-propagate architecture with alignment factorization for natural language inference. *CoRR*, abs/1801.00102.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR*, abs/1511.06361.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1442–1451.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4144–4150.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*, 3:345–358.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426.

Hong Yu and Tsendsuren Munkhdalai. 2017a. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 397–407.

Hong Yu and Tsendsuren Munkhdalai. 2017b. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 11–21.

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, Si Wei, and Hui Jiang. 2017a. Exploring question understanding and adaptation in neural-network-based question answering. *CoRR*, abs/arXiv:1703.04617v2.

Shiyue Zhang, Gulnigar Mahmut, Dong Wang, and Askar Hamdulla. 2017b. Memory-augmented chinese-uyghur neural machine translation. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2017, Kuala Lumpur, Malaysia, December 12-15, 2017*, pages 1092–1096.

AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples

Dongyeop Kang[1] **Tushar Khot**[2] **Ashish Sabharwal**[2] **Eduard Hovy**[1]
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Allen Institute for Artificial Intelligence, Seattle, WA, USA
{dongyeok,hovy}@cs.cmu.edu {tushark,ashishs}@allenai.org

Abstract

We consider the problem of learning textual entailment models with limited supervision (5K-10K training examples), and present two complementary approaches for it. First, we propose knowledge-guided adversarial example generators for incorporating large lexical resources in entailment models via only a handful of rule templates. Second, to make the entailment model—a discriminator—more robust, we propose the first GAN-style approach for training it using a natural language example generator that iteratively adjusts based on the discriminator's performance. We demonstrate effectiveness using two entailment datasets, where the proposed methods increase accuracy by 4.7% on SciTail and by 2.8% on a 1% training sub-sample of SNLI. Notably, even a single hand-written rule, NEGATE, improves the accuracy on the negation examples in SNLI by 6.1%.

1 Introduction

The impressive success of machine learning models on large natural language datasets often does not carry over to moderate training data regimes, where models often struggle with infrequently observed patterns and simple adversarial variations. A prominent example of this phenomenon is *textual entailment*, the fundamental task of deciding whether a *premise* text entails ($\models$) a *hypothesis* text. On certain datasets, recent deep learning entailment systems (Parikh et al., 2016; Wang et al., 2017; Gong et al., 2018) have achieved close to human level performance. Nevertheless, the problem is far from solved, as evidenced by how easy it is to generate minor adversarial examples that break even the best systems. As Table 1 illustrates, a state-of-the-art neural system for this task, namely the Decomposable Attention Model (Parikh et al., 2016), fails when faced with simple linguistic phenomena such as negation, or a re-ordering of words. This is not unique to a particular model or task. Minor adversarial examples have also been found to easily break neural systems on other linguistic tasks such as reading comprehension (Jia and Liang, 2017).

Table 1: Failure examples from the SNLI dataset: negation (Top) and re-ordering (Bottom). **P** is premise, **H** is hypothesis, and **S** is prediction made by an entailment system (Parikh et al., 2016).

P: The dog did not eat all of the chickens.
H: The dog ate all of the chickens.
S: entails (score 56.5%)
P: The red box is in the blue box.
H: The blue box is in the red box.
S: entails (score 92.1%)

A key contributor to this brittleness is the use of specific datasets such as SNLI (Bowman et al., 2015) and SQuAD (Rajpurkar et al., 2016) to drive model development. While large and challenging, *these datasets also tend to be homogeneous*. E.g., SNLI was created by asking crowd-source workers to generate entailing sentences, which then tend to have limited linguistic variations and annotation artifacts (Gururangan et al., 2018). Consequently, models overfit to sufficiently repetitive patterns—and sometimes idiosyncrasies—in the datasets they are trained on. They fail to cover long-tail and rare patterns in the training distribution, or linguistic phenomena such as negation that would be obvious to a layperson.

To address this challenge, we propose to *train textual entailment models more robustly using ad-*

versarial examples generated in two ways: (a) by incorporating knowledge from large linguistic resources, and (b) using a sequence-to-sequence neural model in a GAN-style framework.

The motivation stems from the following observation. While deep-learning based textual entailment models lead the pack, they generally do not incorporate intuitive rules such as negation, and ignore large-scale linguistic resources such as PPDB (Ganitkevitch et al., 2013) and WordNet (Miller, 1995). These resources could help them generalize beyond specific words observed during training. For instance, while the SNLI dataset contains the pattern *two men* $\vDash$ *people*, it does not contain the analogous pattern *two dogs* $\vDash$ *animals* found easily in WordNet.

Effectively integrating simple rules or linguistic resources in a deep learning model, however, is challenging. Doing so directly by substantially adapting the model architecture (Sha et al., 2016; Chen et al., 2018) can be cumbersome and limiting. Incorporating such knowledge indirectly via modified word embeddings (Faruqui et al., 2015; Mrkšić et al., 2016), as we show, can have little positive impact and can even be detrimental.

Our proposed method, which is task-specific but model-independent, is inspired by data-augmentation techniques. We generate new training examples by applying knowledge-guided rules, via only a handful of rule templates, to the original training examples. Simultaneously, we also use a sequence-to-sequence or seq2seq model for each entailment class to generate new hypotheses from a given premise, adaptively creating new adversarial examples. These can be used with any entailment model without constraining model architecture.

We also introduce the first approach to train a robust entailment model using a Generative Adversarial Network or GAN (Goodfellow et al., 2014) style framework. We iteratively improve both the entailment system (the *discriminator*) and the differentiable part of the data-augmenter (specifically the neural *generator*), by training the generator based on the discriminator's performance on the generated examples. Importantly, unlike the typical use of GANs to create a strong generator, we use it as a mechanism to create a strong and robust discriminator.

Our new entailment system, called AdvEntuRe, demonstrates that in the moderate data regime, adversarial iterative data-augmentation via only a handful of linguistic rule templates can be surprisingly powerful. Specifically, we observe 4.7% accuracy improvement on the challenging SciTail dataset (Khot et al., 2018) and a 2.8% improvement on 10K-50K training subsets of SNLI. An evaluation of our algorithm on the negation examples in the test set of SNLI reveals a 6.1% improvement from just a single rule.

2 Related Work

Adversarial example generation has recently received much attention in NLP. For example, Jia and Liang (2017) generate adversarial examples using manually defined templates for the SQuAD reading comprehension task. Glockner et al. (2018) create an adversarial dataset from SNLI by using WordNet knowledge. Automatic methods (Iyyer et al., 2018) have also been proposed to generate adversarial examples through paraphrasing. These works reveal how neural network systems trained on a large corpus can easily break when faced with carefully designed unseen adversarial patterns at test time. Our motivation is different. We use adversarial examples at training time, in a data augmentation setting, to train a more robust entailment discriminator. The generator uses explicit knowledge or hand written rules, and is trained in a end-to-end fashion along with the discriminator.

Incorporating external rules or linguistic resources in a deep learning model generally requires substantially adapting the model architecture (Sha et al., 2016; Liang et al., 2017; Kang et al., 2017). This is a model-dependent approach, which can be cumbersome and constraining. Similarly non-neural textual entailment models have been developed that incorporate knowledge bases. However, these also require model-specific engineering (Raina et al., 2005; Haghighi et al., 2005; Silva et al., 2018).

An alternative is the model- and task-independent route of incorporating linguistic resources via word embeddings that are *retro-fitted* (Faruqui et al., 2015) or *counter-fitted* (Mrkšić et al., 2016) to such resources. We demonstrate, however, that this has little positive impact in our setting and can even be detrimental. Further, it is unclear how to incorporate knowledge sources into advanced representations such as contextual embeddings (McCann et al., 2017; Peters et al., 2018). We thus focus on a task-specific but model-independent approach.

Logical rules have also been defined to label existing examples based on external resources (Hu et al., 2016). Our focus here is on generating *new* training examples.

Our use of the GAN framework to create a better discriminator is related to CatGANs (Wang and Zhang, 2017) and TripleGANs (Chongxuan et al., 2017) where the discriminator is trained to classify the original training image classes as well as a new 'fake' image class. We, on the other hand, generate examples belonging to the same classes as the training examples. Further, unlike the earlier focus on the vision domain, this is the first approach to train a discriminator using GANs for a natural language task with discrete outputs.

3 Adversarial Example Generation

We present three different techniques to create adversarial examples for textual entailment. Specifically, we show how external knowledge resources, hand-authored rules, and neural language generation models can be used to generate such examples. Before describing these generators in detail, we introduce the notation used henceforth.

We use lower-case letters for single instances (e.g., x, p, h), upper-case letters for sets of instances (e.g., X, P, H), blackboard bold for models (e.g., $\mathbb{D}$), and calligraphic symbols for discrete spaces of possible values (e.g., class labels $\mathcal{C}$). For the textual entailment task, we assume each example is represented as a triple (p, h, c), where p is a premise (a natural language sentence), h is a hypothesis, and c is an entailment label: (a) *entails* ($\sqsubseteq$) if h is true whenever p is true; (b) *contradicts* ($\curlywedge$) if h is false whenever p is true; or (c) *neutral* (#) if the truth value of h cannot be concluded from p being true.[1]

We will introduce various example generators in the rest of this section. Each such generator, $\mathbb{G}_\rho$, is defined by a partial function f_ρ and a label g_ρ. If a sentence s has a certain property required by f_ρ (e.g., contains a particular string), f_ρ transforms it into another sentence s' and g_ρ provides an entailment label from s to s'. Applied to a sentence s, $\mathbb{G}_\rho$ thus either "fails" (if the pre-requisite isn't met) or generates a new entailment example triple, $\big(s, f_\rho(s), g_\rho\big)$. For instance, consider the generator for ρ:=hypernym(car, vehicle) with the (partial) transformation function f_ρ:="Replace *car* with *vehicle*" and the label g_ρ:=*entails*. f_ρ would fail when applied to a sentence not containing the word "car". Applying f_ρ to the sentence s="*A man is driving the car*" would generate s'="*A man is driving the vehicle*", creating the example $(s, s', \textit{entails})$.

[1]The symbols are based on Natural Logic (Lakoff, 1970) and use the notation of MacCartney and Manning (2012).

Source	ρ	$f_\rho(\mathrm{s})$	g_ρ
Knowledge Base, $\mathbb{G}^{\mathrm{KB}}$			
WordNet	hyper(x, y)	Replace x with y in s	$\sqsubseteq$
	anto(x, y)		$\curlywedge$
	syno(x, y)		$\sqsubseteq$
PPDB	$x \equiv y$		$\sqsubseteq$
SICK	$c(x, y)$		c
Hand-authored, $\mathbb{G}^{\mathrm{H}}$			
Domain knowledge	NEG	NEGATE(s)	$\curlywedge$
Neural Model, $\mathbb{G}^{\mathrm{s2s}}$			
Training data	(s2s, c)	$\mathbb{G}^{\mathrm{s2s}}_c(s)$	c

Table 2: Various generators $\mathbb{G}_\rho$ characterized by their source, (partial) transformation function f_ρ as applied to a sentence s, and entailment label g_ρ

The seven generators we use for experimentation are summarized in Table 2 and discussed in more detail subsequently. While these particular generators are simplistic and one can easily imagine more advanced ones, we show that training using adversarial examples created using even these simple generators leads to substantial accuracy improvement on two datasets.

3.1 Knowledge-Guided Generators

Large knowledge-bases such as WordNet and PPDB contain lexical equivalences and other relationships highly relevant for entailment models. However, even large datasets such as SNLI generally do not contain most of these relationships in the training data. E.g., that *two dogs* entails *animals* isn't captured in the SNLI data. We define simple generators based on lexical resources to create adversarial examples that capture the underlying knowledge. This allows models trained on these examples to learn these relationships.

As discussed earlier, there are different ways of incorporating such symbolic knowledge into neural models. Unlike task-agnostic ways of approaching this goal from a word embedding perspective (Faruqui et al., 2015; Mrkšić et al., 2016) or the model-specific approach (Sha et al., 2016; Chen et al., 2018), we use this knowledge to generate task-specific examples. This allows any entailment model to learn how to use these relationships *in the context of the entailment task*, helping them outperform the above task-agnostic alternative.

Our knowledge-guided example generators, $\mathbb{G}_{\rho}^{\text{KB}}$, use lexical relations available in a knowledge-base: $\rho := r(x, y)$ where the relation r (such as synonym, hypernym, etc.) may differ across knowledge bases. We use a simple (partial) transformation function, $f_\rho(s)$:="Replace x in s with y", as described in an earlier example. In some cases, when part-of-speech (POS) tags are available, the partial function requires the tags for x in s and in $r(x, y)$ to match. The entailment label g_ρ for the resulting examples is also defined based on the relation r, as summarized in Table 2.

This idea is similar to Natural Logic Inference or NLI (Lakoff, 1970; Sommers, 1982; Angeli and Manning, 2014) where words in a sentence can be replaced by their hypernym/hyponym to produce entailing/neutral sentences, depending on their context. We propose a context-agnostic use of lexical resources that, despite its simplicity, already results in significant gains. We use three sources for generators:

WordNet (Miller, 1995) is a large, hand-curated, semantic lexicon with synonymous words grouped into *synsets*. Synsets are connected by many semantic relations, from which we use *hyponym* and *synonym* relations to generate entailing sentences, and *antonym* relations to generate contradicting sentences[2]. Given a relation $r(x, y)$, the (partial) transformation function f_ρ is the POS-tag matched replacement of x in s with y, and requires the POS tag to be noun or verb. NLI provides a more robust way of using these relations based on context, which we leave for future work.

PPDB (Ganitkevitch et al., 2013) is a large resource of lexical, phrasal, and syntactic paraphrases. We use 24,273 lexical paraphrases in their smallest set, PPDB-S (Pavlick et al., 2015), as equivalence relations, $x \equiv y$. The (partial) transformation function f_ρ for this generator is POS-tagged matched replacement of x in s with y, and the label g_ρ is *entails*.

[2]A similar approach was used in a parallel work to generate an adversarial dataset from SNLI (Glockner et al., 2018).

SICK (Marelli et al., 2014) is dataset with entailment examples of the form (p, h, c), created to evaluate an entailment model's ability to capture compositional knowledge via hand-authored rules. We use the 12,508 patterns of the form $c(x, y)$ extracted by Beltagy et al. (2016) by comparing sentences in this dataset, with the property that for each SICK example (p, h, c), replacing (when applicable) x with y in p produces h. For simplicity, we ignore positional information in these patterns. The (partial) transformation function f_ρ is replacement of x in s with y, and the label g_ρ is c.

3.2 Hand-Defined Generators

Even very large entailment datasets have no or very few examples of certain otherwise common linguistic constructs such as negation,[3] causing models trained on them to struggle with these constructs. A simple model-agnostic way to alleviate this issue is via a negation example generator whose transformation function $f_\rho(s)$ is NEGATE(s), described below, and the label g_ρ is *contradicts*.

NEGATE(s): If s contains a 'be' verb (e.g., is, was), add a "not" after the verb. If not, also add a "did" or "do" in front based on its tense. E.g., change "A person is crossing" to "A person is not crossing" and "A person crossed" to "A person did not cross." While many other rules could be added, we found that this single rule covered a majority of the cases. Verb tenses are also considered[4] and changed accordingly. Other functions such as dropping adverbial clauses or changing tenses could be defined in a similar manner.

Both the knowledge-guided and hand-defined generators make local changes to the sentences based on simple rules. It should be possible to extend the hand-defined rules to cover the long tail (as long as they are procedurally definable). However, a more scalable approach would be to extend our generators to trainable models that can cover a wider range of phenomena than hand-defined rules. Moreover, the applicability of these rules generally depends on the context which can also be incorporated in such trainable generators.

3.3 Neural Generators

For each entailment class c, we use a trainable sequence-to-sequence neural model (Sutskever

[3]Only 211 examples (2.11%) in the SNLI training set contain negation triggers such as not, 'nt, etc.

[4]https://www.nodebox.net/code/index.php/Linguistics

et al., 2014; Luong et al., 2015) to generate an entailment example (s, s', c) from an input sentence s. The seq2seq model, trained on examples labeled c, itself acts as the transformation function f_ρ of the corresponding generator $\mathbb{G}_c^{\text{s2s}}$. The label g_ρ is set to c. The joint probability of seq2seq model is:

$$\mathbb{G}_c^{\text{s2s}}(X_c; \phi_c) = \mathbb{G}_c^{\text{s2s}}(H_c, P_c; \phi_c) \quad (1)$$
$$= \Pi_i P(h_{i,c}|p_{i,c}; \phi_c)P(h_i) \quad (2)$$

The loss function for training the seq2seq is:

$$\hat{\phi}_c = \underset{\phi_c}{\text{argmin}}\, L(H_c, \mathbb{G}_c^{\text{s2s}}(X_c; \phi_c)) \quad (3)$$

where L is the cross-entropy loss between the original hypothesis H_c and the predicted hypothesis. Cross-entropy is computed for each predicted word w_i against the same in H_c given the sequence of previous words in H_c . $\hat{\phi}_c$ are the optimal parameters in $\mathbb{G}_c^{\text{s2s}}$ that minimize the loss for class c. We use the single most likely output to generate sentences in order to reduce decoding time.

3.4 Example Generation

The generators described above are used to create new entailment examples from the training data. For each example (p, h, c) in the data, we can create two new examples: $\big(p, f_\rho(p), g_\rho\big)$ and $\big(h, f_\rho(h), g_\rho\big)$.

The examples generated this way using $\mathbb{G}^{\text{KB}}$ and $\mathbb{G}^{\text{H}}$ can, however, be relatively easy, as the premise and hypothesis would differ by only a word or so. We therefore compose such simple ("first-order") generated examples with the original input example to create more challenging "second-order" examples. We can create second-order examples by composing the original example (p, h, c) with a generated sentence from hypothesis, $f_\rho(h)$ and premise, $f_\rho(p)$. Figure 1 depicts how these two kinds of examples are generated from an input example (p, h, c).

First, we consider the second-order example between the original premise and the transformed hypothesis: $(p, f_\rho(h), \bigoplus(c, g_\rho))$, where $\bigoplus$, defined in the left half of Table 3, composes the input example label c (connecting p and h) and the generated example label g_ρ to produce a new label. For instance, if p entails h and h entails $f_\rho(h)$, p would entail f_ρ. In other words, $\bigoplus(\sqsubseteq, \sqsubseteq)$ is $\sqsubseteq$. For example, composing ("A man is playing

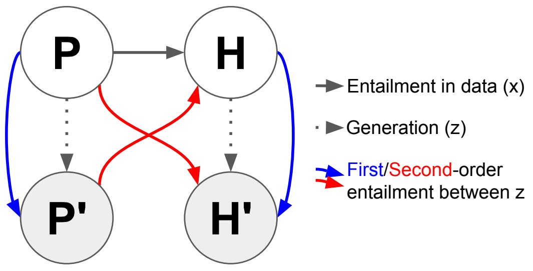


Figure 1: Generating first-order (blue) and second-order (red) examples.

$p \Rightarrow h$	$h \Rightarrow h'$	$p \Rightarrow h'$	$p \Rightarrow h$	$p \Rightarrow p'$	$p' \Rightarrow h$
c	g_ρ	$\bigoplus$	c	g_ρ	$\bigotimes$
⊑	⊑	⊑	⊑	⊑	?
⊑	⋏	⋏	⊑	⋏	?
⊑	#	#	⊑	#	#
⋏	⊑	?	⋏	⊑	?
⋏	⋏	?	⋏	⋏	?
⋏	#	#	⋏	#	#
#	⊑	#	#	⊑	#
#	⋏	#	#	⋏	#
#	#	#	#	#	#

Table 3: Entailment label composition functions $\bigoplus$ (left) and $\bigotimes$ (right) for creating second-order examples. c and g_ρ are the original and generated labels, resp. ⊑: *entails*, ⋏: *contradicts*, #: *neutral*, ?: undefined

soccer", "A man is playing a game", ⊑) with a generated hypothesis $f_\rho(h)$: "A person is playing a game." will give a new second-order entailment example: ("A man is playing soccer", "A person is playing a game", ⊑).

Second, we create an example from the generated premise to the original hypothesis: $(f_\rho(p), h, \bigotimes(g_\rho, c))$. The composition function here, denoted $\bigotimes$ and defined in the right half of Table 3, is often undetermined. For example, if p entails $f_\rho(p)$ and p entails h, the relation between $f_\rho(p)$ and h is undetermined i.e. $\bigotimes(\sqsubseteq, \sqsubseteq) = ?$. While this particular composition $\bigotimes$ often leads to undetermined or neutral relations, we use it here for completeness. For example, composing the previous example with a generated *neutral* premise, $f_\rho(p)$: "A person is wearing a cap" would generate an example ("A person is wearing a cap", "A man is playing a game", #)

The composition function $\bigoplus$ is the same as the "join" operation in natural logic reasoning (Icard III and Moss, 2014), except for two differences: (a) relations that do not belong to our three entailment classes are mapped to '?', and (b) the exclusivity/alternation relation is mapped to *contradicts*. The composition function $\bigotimes$, on the other hand, does not map to the join operation.

3.5 Implementation Details

Given the original training examples X, we generate the examples from each premise and hypothesis in a batch using $\mathbb{G}^{\text{KB}}$ and $\mathbb{G}^{\text{H}}$. We also generate new hypothesis per class for each premise using $\mathbb{G}_c^{\text{s2s}}$. Using all the generated examples to train the model would, however, overwhelm the original training set. For examples, our knowledge-guided generators $\mathbb{G}^{\text{KB}}$ can be applied in 17,258,314 different ways.

To avoid this, we sub-sample our synthetic examples to ensure that they are proportional to the input examples X, specifically they are bounded to $\alpha|X|$ where α is tuned for each dataset. Also, as seen in Table 3, our knowledge-guided generators are more likely to generate *neutral* examples than any other class. To make sure that the labels are not skewed, we also sub-sample the examples to ensure that our generated examples have the same class distribution as the input batch. The SciTail dataset only contains two classes: *entails* mapped to $\sqsubseteq$ and *neutral* mapped to $\curlywedge$. As a result, generated examples that do not belong to these two classes are ignored.

The sub-sampling, however, has a negative side-effect where our generated examples end up using a small number of lexical relations from the large knowledge bases. On moderate datasets, this would cause the entailment model to potentially just memorize these few lexical relations. Hence, we generate new entailment examples for each mini-batch and update the model parameters based on the training+generated examples in this batch.

The overall example generation procedure goes as follows: For each mini-batch X (1) randomly choose 3 applicable rules per source and sentence (e.g., replacing men with people based on PPDB in premise is one rule), (2) produce examples Z_{all} using $\mathbb{G}^{\text{KB}}$, $\mathbb{G}^{\text{H}}$ and $\mathbb{G}^{\text{s2s}}$, (3) randomly sub-select examples Z from Z_{all} to ensure the balance between classes and $|Z|= \alpha|X|$.

4 AdvEntuRe

Figure 2 shows the complete architecture of our model, AdvEntuRe (ADVersarial training for textual ENTailment Using Rule-based Examples.). The entailment model $\mathbb{D}$ is shown with the white box and two proposed generators are shown using black boxes. We combine the two symbolic untrained generators, $\mathbb{G}^{\text{KB}}$ and $\mathbb{G}^{\text{H}}$ into a single $\mathbb{G}^{\text{rule}}$ model. We combine the generated adversarial examples Z with the original training examples X to train the discriminator. Next, we describe how the individual models are trained and finally present our new approach to train the generator based on the discriminator's performance.

4.1 Discriminator Training

We use one of the state-of-the-art entailment models (at the time of its publication) on SNLI, decomposable attention model (Parikh et al., 2016) with intra-sentence attention as our discriminator $\mathbb{D}$. The model attends each word in hypothesis with each word in the premise, compares each pair of the attentions, and then aggregates them as a final representation. This discriminator model can be easily replaced with any other entailment model without any other change to the AdvEntuRe architecture. We pre-train our discriminator $\mathbb{D}$ on the original dataset, X=(P, H, C) using:

$$\mathbb{D}(X;\theta) = \underset{\hat{C}}{\operatorname{argmax}}\, \mathbb{D}(\hat{C}|P,H;\theta) \tag{4}$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, L(C, \mathbb{D}(X;\theta)) \tag{5}$$

where L is cross-entropy loss function between the true labels, Y and the predicted classes, and $\hat{\theta}$ are the learned parameters.

4.2 Generator Training

Our knowledge-guided and hand-defined generators are symbolic parameter-less methods which are not currently trained. For simplicity, we will refer to the set of symbolic rule-based generators as $\mathbb{G}^{\text{rule}} := \mathbb{G}^{\text{KB}} \cup \mathbb{G}^{\text{H}}$. The neural generator $\mathbb{G}^{\text{s2s}}$, on the other hand, can be trained as described earlier. We leave the training of the symbolic models for future work.

4.3 Adversarial Training

We now present our approach to iteratively train the discriminator and generator in a GAN-style framework. Unlike traditional GAN (Goodfellow et al., 2014) on image/text generation that aims to obtain better generators, our goal is to build a robust discriminator regularized by the generators ($\mathbb{G}^{\text{s2s}}$ and $\mathbb{G}^{\text{rule}}$). The discriminator and generator are iteratively trained against each other to

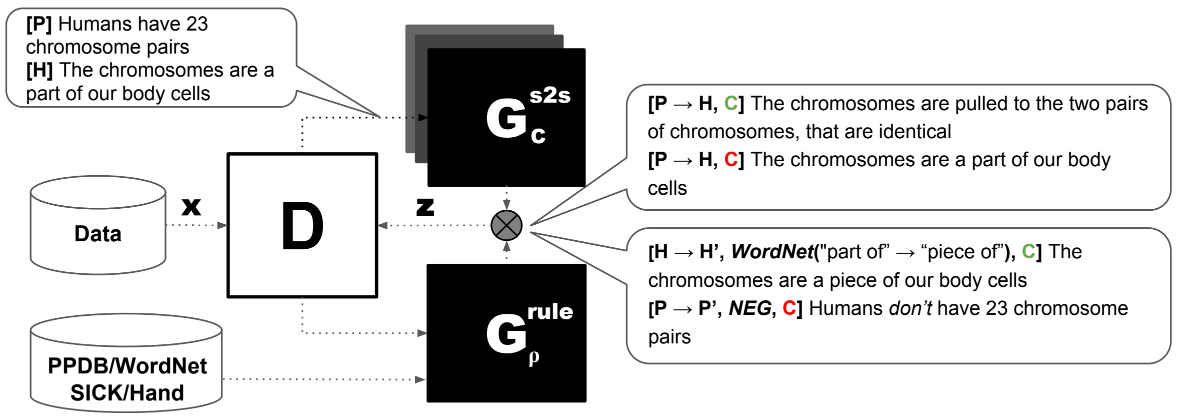

[P] Humans have 23 chromosome pairs
[H] The chromosomes are a part of our body cells
Data
x
D
z
PPDB/WordNet SICK/Hand
G s2s c
G rule ρ
[P → H, C] The chromosomes are pulled to the two pairs of chromosomes, that are identical
[P → H, C] The chromosomes are a part of our body cells
[H → H', WordNet("part of" → "piece of"), C] The chromosomes are a piece of our body cells
[P → P', NEG, C] Humans don't have 23 chromosome pairs

Figure 2: Overview of AdvEntuRe, our model for knowledge-guided textual entailment.

Algorithm 1 Training procedure for AdvEntuRe.

```
1:  pretrain discriminator $\mathbb{D}(\hat{\theta})$ on $\mathbf{X}$;
2:  pretrain generators $\mathbb{G}_c^{s2s}(\hat{\phi})$ on $\mathbf{X}$;
3:  for number of training iterations do
4:    for mini-batch $B \leftarrow X$ do
5:      generate examples from $\mathbb{G}$
6:        $Z_G \Leftarrow \mathbb{G}(B;\phi)$,
7:      balance $X$ and $Z_G$ s.t. $|Z_G| \le \alpha|X|$
8:      optimize discriminator:
9:        $\hat{\theta} = \mathrm{argmin}_\theta\, L_{\mathbb{D}}(X + Z_G;\theta)$
10:     optimize generator:
11:       $\hat{\phi} = \mathrm{argmin}_\phi\, L_{\mathbb{G}^{s2s}}(\mathcal{Z}_G; L_{\mathbb{D}};\phi)$
12:     Update $\theta \leftarrow \hat{\theta}; \phi \leftarrow \hat{\phi}$
```

achieve better discrimination on the augmented data from the generator and better example generation against the learned discriminator. Algorithm 1 shows our training procedure.

First, we pre-train the discriminator $\mathbb{D}$ and the seq2seq generators $\mathbb{G}^{s2s}$ on the original data X. We alternate the training of the discriminator and generators over K iterations (set to 30 in our experiments).

For each iteration, we take a mini-batch B from our original data X. For each mini-batch, we generate new entailment examples, Z_G using our adversarial examples generator. Once we collect all the generated examples, we balance the examples based on their source and label (as described in Section 3.5). In each training iteration, we optimize the discriminator against the augmented training data, $X + Z_G$ and use the discriminator loss to guide the generator to pick challenging examples. For every mini-batch of examples $X + Z_G$, we compute the discriminator loss $L(C; \mathbb{D}(X + Z_G; \theta))$ and apply the negative of this loss to each word of the generated sentence in $\mathbb{G}^{s2s}$. In other words, the discriminator loss value replaces the cross-entropy loss used to train the seq2seq model (similar to a REINFORCE (Williams, 1992) reward). This basic approach uses the loss over the entire batch to update the generator, ignoring whether specific examples were hard or easy for the discriminator. Instead, one could update the generator per example based on the discriminator's loss on that example. We leave this for future work.

5 Experiments

Our empirical assessment focuses on two key questions: (a) Can a handful of rule templates improve a state-of-the-art entailment system, especially with moderate amounts of training data? (b) Can iterative GAN-style training lead to an improved discriminator?

To this end, we assess various models on the two entailment **datasets** mentioned earlier: SNLI (570K examples) and SciTail (27K examples).[5] To test our hypothesis that adversarial example based training prevents overfitting in small to moderate training data regimes, we compare model accuracies on the test sets when using 1%, 10%, 50%, and 100% subsamples of the train and dev sets.

We consider two baseline **models**: $\mathbb{D}$, the Decomposable Attention model (Parikh et al., 2016) with intra-sentence attention using pre-trained word embeddings (Pennington et al., 2014); and $\mathbb{D}_{\text{retro}}$ which extends $\mathbb{D}$ with word embeddings initialized by retrofitted vectors (Faruqui et al., 2015). The vectors are retrofitted on PPDB, Word-

[5]SNLI has a 96.4%/1.7%/1.7% split and SciTail has a 87.3%/4.8%/7.8% split on train, valid, and test sets, resp.

Table 4: Test accuracies with different subsampling ratios on SNLI (top) and SciTail (bottom).

SNLI	1%	10%	50%	100%
$\mathbb{D}$	57.68	75.03	82.77	84.52
$\mathbb{D}_{\text{retro}}$	57.04	73.45	81.18	84.14
AdvEntuRe				
∟ $\mathbb{D}$ + $\mathbb{G}^{\text{s2s}}$	58.35	75.66	82.91	**84.68**
∟ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$	**60.45**	**77.11**	**83.51**	84.40
∟ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ + $\mathbb{G}^{\text{s2s}}$	59.33	76.03	83.02	83.25

SciTail	1%	10%	50%	100%
$\mathbb{D}$	56.60	60.84	73.24	74.29
$\mathbb{D}_{\text{retro}}$	59.75	67.99	69.05	72.63
AdvEntuRe				
∟ $\mathbb{D}$ + $\mathbb{G}^{\text{s2s}}$	**65.78**	**70.77**	74.68	76.92
∟ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$	61.74	66.53	73.99	**79.03**
∟ $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ + $\mathbb{G}^{\text{s2s}}$	63.28	66.78	**74.77**	78.60

Net, FrameNet, and all of these, with the best results for each dataset reported here.

Our proposed model, AdvEntuRe, is evaluated in three flavors: $\mathbb{D}$ augmented with examples generated by $\mathbb{G}^{\text{rule}}$, $\mathbb{G}^{\text{s2s}}$, or both, where $\mathbb{G}^{\text{rule}} = \mathbb{G}^{\text{KB}} \cup \mathbb{G}^{\text{H}}$. In the first two cases, we create new examples for each batch in every epoch using a fixed generator (cf. Section 3.5). In the third case ($\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ + $\mathbb{G}^{\text{s2s}}$), we use the GAN-style training.

We uses grid search to find the best hyper-parameters for $\mathbb{D}$ based on the validation set: hidden size 200 for LSTM layer, embedding size 300, dropout ratio 0.2, and fine-tuned embeddings.

The ratio between the number of generated vs. original examples, α is empirically chosen to be 1.0 for SNLI and 0.5 for SciTail, based on validation set performance. Generally, very few generated examples (small α) has little impact, while too many of them overwhelm the original dataset resulting in worse scores (cf. Appendix for more details).

5.1 Main Results

Table 4 summarizes the test set accuracies of the different models using various subsampling ratios for SNLI and SciTail training data.

We make a few observations. First, $\mathbb{D}_{\text{retro}}$ is ineffective or even detrimental in most cases, except on SciTail when 1% (235 examples) or 10% (2.3K examples) of the training data is used. The gain in these two cases is likely because retrofitted lexical rules are helpful with extremely less data training while not as data size increases.

On the other hand, our method always achieves the best result compared to the baselines ($\mathbb{D}$ and $\mathbb{D}_{\text{retro}}$). Especially, significant improvements are made in less data setting: +2.77% in SNLI (1%) and 9.18% in SciTail (1%). Moreover, $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$'s accuracy on SciTail (100%) also outperforms the previous state-of-the-art model (DGEM (Khot et al., 2018), which achieves 77.3%) for that dataset by 1.7%.

Among the three different generators combined with $\mathbb{D}$, both $\mathbb{G}^{\text{rule}}$ and $\mathbb{G}^{\text{s2s}}$ are useful in SciTail, while $\mathbb{G}^{\text{rule}}$ is much more useful than $\mathbb{G}^{\text{s2s}}$ on SNLI. We hypothesize that seq2seq model trained on large training sets such as SNLI will be able to reproduce the input sentences. Adversarial examples from such a model are not useful since the entailment model uses the same training examples. However, on smaller sets, the seq2seq model would introduce noise that can improve the robustness of the model.

Table 5: Test accuracies across various rules $\mathcal{R}$ and classes $\mathcal{C}$. Since SciTail has two classes, we only report results on two classes of $\mathbb{G}^{\text{s2s}}$

	$\mathcal{R}/\mathcal{C}$	SNLI (5%)	SciTail (10%)
$\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$	$\mathbb{D}$	69.18	60.84
	+ PPDB	**72.81 (+3.6%)**	65.52 (+4.6%)
	+ SICK	71.32 (+2.1%)	67.49 (+6.5%)
	+ WordNet	71.54 (+2.3%)	64.67 (+3.8%)
	+ HAND	71.15 (+1.9%)	**69.05 (+8.2%)**
	+ all	71.31 (+2.1%)	64.16 (+3.3%)
$\mathbb{D}$ + $\mathbb{G}^{\text{s2s}}$	$\mathbb{D}$	69.18	60.84
	+ positive	71.21 (+2.0%)	67.49 (+6.6%)
	+ negative	71.76 (+2.6%)	68.95 (+8.1%)
	+ neutral	71.72 (+2.5%)	-
	+ all	**72.28 (+3.1%)**	**70.77 (+9.9%)**

5.2 Ablation Study

To evaluate the impact of each generator, we perform ablation tests against each symbolic generator in $\mathbb{D}$ + $\mathbb{G}^{\text{rule}}$ and the generator $\mathbb{G}_c^{\text{s2s}}$ for each entailment class c. We use a 5% sample of SNLI and a 10% sample of SciTail. The results are summarized in Table 5.

Interestingly, while PPDB (phrasal paraphrases) helps the most (**+3.6%**) on SNLI, simple negation rules help significantly (**+8.2%**) on SciTail dataset. Since most entailment examples in SNLI are minor rewrites by Turkers, PPDB often contains these simple paraphrases. For SciTail, the sentences are authored independently with limited gains from simple paraphrasing. However, a model trained on only 10% of the dataset (2.3K

Table 6: Given a premise **P** (underlined), examples of hypothesis sentences **H'** generated by seq2seq generators $\mathbb{G}^{\mathrm{s2s}}$, and premise sentences **P'** generated by rule based generators $\mathbb{G}^{\mathrm{rule}}$, on the full SNLI data. Replaced words or phrases are shown in **bold**. This illustrates that even simple, easy-to-define rules can generate useful adversarial examples.

P	a person on a horse jumps over a broken down airplane
H': $\mathbb{G}^{\mathrm{s2s}}_{c=\sqsubseteq}$	a person is on a horse jumps over a rail, a person jumping over a plane
H': $\mathbb{G}^{\mathrm{s2s}}_{c=\curlywedge}$	a person is riding a horse in a field with a dog in a red coat
H': $\mathbb{G}^{\mathrm{s2s}}_{c=\#}$	a person is in a blue dog is in a park
P (or **H**)	a dirt bike rider catches some air going off a large hill
P': $\mathbb{G}^{\mathrm{KB(PPDB)}}_{\rho=\equiv,g_\rho=\sqsubseteq}$	a dirt **motorcycle** rider catches some air going off a large hill
P': $\mathbb{G}^{\mathrm{KB(SICK)}}_{\rho=c,g_\rho=\#}$	a dirt bike **man on yellow bike** catches some air going off a large hill
P': $\mathbb{G}^{\mathrm{KB(WordNet)}}_{\rho=syno,g_\rho=\sqsubseteq}$	a dirt bike rider catches some **atmosphere** going off a large hill
P': $\mathbb{G}^{\mathrm{Hand}}_{\rho=\mathrm{NEG},g_\rho=\curlywedge}$	a dirt bike rider **do not catch** some air going off a large hill

examples) would end up learning a model relying on purely word overlap. We believe that the simple negation examples introduce *neutral* examples with high lexical overlap, forcing the model to find a more informative signal.

On the other hand, using all classes for $\mathbb{G}^{\mathrm{s2s}}$ results in the best performance, supporting the effectiveness of the GAN framework for penalizing or rewarding generated sentences based on $\mathbb{D}$'s loss. Preferential selection of rules within the GAN framework remains a promising direction.

5.3 Qualitative Results

Table 6 shows examples generated by various methods in AdvEntuRe. As shown, both seq2seq and rule based generators produce reasonable sentences according to classes and rules. As expected, seq2seq models trained on very few examples generate noisy sentences. The quality of our knowledge-guided generators, on the other hand, does not depend on the training set size and they still produce reliable sentences.

5.4 Case Study: Negation

For further analysis of the negation-based generator in Table 1, we collect only the negation examples in test set of SNLI, henceforth referred to as nega-SNLI. Specifically, we extract examples where either the premise or the hypothesis contains "not", "no", "never", or a word that ends with "n't'. These do not cover more subtle ways of expressing negation such as "seldom" and the use of antonyms. nega-SNLI contains 201 examples with the following label distribution: 51 (25.4%) neutral, 42 (20.9%) entails, 108 (53.7%) contradicts. Table 7 shows examples in each category.

Table 7: Negation examples in nega-SNLI

$\sqsubseteq$	P: several women are playing volleyball. H: this doesn't look like soccer.
#	P: a man with no shirt on is performing with a baton. H: a man is trying his best at the national championship of baton.
$\curlywedge$	P: island native fishermen reeling in their nets after a long day's work. H: the men did not go to work today but instead played bridge.

While $\mathbb{D}$ achieves an accuracy of only 76.64%[6] on nega-SNLI, $\mathbb{D}$ + $\mathbb{G}^{\mathrm{H}}$ with NEGATE is substantially more successful (+6.1%) at handling negation, achieving an accuracy of 82.74%.

6 Conclusion

We introduced an adversarial training architecture for textual entailment. Our seq2seq and knowledge-guided example generators, trained in an end-to-end fashion, can be used to make any base entailment model more robust. The effectiveness of this approach is demonstrated by the significant improvement it achieves on both SNLI and SciTail, especially in the low to medium data regimes. Our rule-based generators can be expanded to cover more patterns and phenomena, and the seq2seq generator extended to incorporate per-example loss for adversarial training.

[6]This is much less than the full test accuracy of 84.52%.

References

Gabor Angeli and Christopher D Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *EMNLP*, pages 534–545.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42:763–808.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2018. Natural language inference with external knowledge. In *ACL*.

LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. 2017. Triple generative adversarial nets. In *NIPS*, pages 4091–4101.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. *NAACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL-HLT*, pages 758–764.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *ACL*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. *ICLR*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*, pages 2672–2680.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL*.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *EMNLP*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *ACL*.

Thomas Icard III and Lawrence Moss. 2014. Recent progress in monotonicity. *LiLT (Linguistic Issues in Language Technology)*, 9.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke S. Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*.

R. Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and explaining causes from text for a time series event. In *EMNLP*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. *AAAI*.

George Lakoff. 1970. Linguistics and Natural Logic. *Synthese*, 22(1-2):151–271.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Bill MacCartney and Christopher D. Manning. 2012. Natural logic and natural language inference. In *Computing Meaning. Text, Speech and Language Technology*, volume 47.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

George A Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *HLT-NAACL*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher D Manning, and Andrew Y Ng. 2005. Robust textual inference using diverse knowledge sources. In *1st PASCAL Recognition Textual Entailment Challenge Workshop*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Recognizing textual entailment via multi-task knowledge assisted lstm. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 285–298. Springer.

Vivian S Silva, André Freitas, and Siegfried Handschuh. 2018. Recognizing and justifying text entailment through distributional navigation on definition graphs. In *AAAI*.

Fred Sommers. 1982. The logic of natural language.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Shanshan Wang and Lei Zhang. 2017. CatGAN: Coupled adversarial transfer for domain generation. *CoRR*, abs/1711.08904.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Subword-level Word Vector Representations for Korean

Sungjoon Park [1], Jeongmin Byun [1], Sion Baek [2], Yongseok Cho [3], Alice Oh [1]
[1] Department of Computing, KAIST, Republic of Korea
[2] Program in Cognitive Science, Seoul National University, Republic of Korea
[3] Natural Language Processing team, Adecco, Republic of Korea
{sungjoon.park, jmbyun}@kaist.ac.kr, sioning1122@snu.ac.kr
yongseok.cho84@gmail.com, alice.oh@kaist.edu

Abstract

Research on distributed word representations is focused on widely-used languages such as English. Although the same methods can be used for other languages, language-specific knowledge can enhance the accuracy and richness of word vector representations. In this paper, we look at improving distributed word representations for Korean using knowledge about the unique linguistic structure of Korean. Specifically, we decompose Korean words into the *jamo* level, beyond the character-level, allowing a systematic use of subword information. To evaluate the vectors, we develop Korean test sets for word similarity and analogy and make them publicly available. The results show that our simple method outperforms word2vec and character-level Skip-Grams on semantic and syntactic similarity and analogy tasks and contributes positively toward downstream NLP tasks such as sentiment analysis.

1 Introduction

Word vector representations built from a large corpus embed useful semantic and syntactic knowledge. They can be used to measure the similarity between words and can be applied to various downstream tasks such as document classification (Yang et al., 2016), conversation modeling (Serban et al., 2016), and machine translation (Neishi et al., 2017). Most previous research for learning the vectors focuses on English (Collobert and Weston, 2008; Mikolov et al., 2013b,a; Pennington et al., 2014; Liu et al., 2015; Cao and Lu, 2017) and thus leads to difficulties and limitations in directly applying those techniques to a language with a different internal structure from that of English.

The mismatch is especially significant for morphologically rich languages such as Korean where the morphological richness could be captured by subword level embedding such as character embedding. It has been already shown that decomposing a word into subword units and using them as inputs improves performance for downstream NLP such as text classification (Zhang et al., 2015), language modeling (Kim et al., 2016), and machine translation (Ling et al., 2015; Lee et al., 2017). Despite their effectiveness in capturing syntactic features of diverse languages, decomposing a word into a set of n-grams and learning n-gram vectors does not consider the unique linguistic structures of various languages. Thus, researchers have integrated language-specific structures to learn word vectors, for example subcharacter components of Chinese characters (Yu et al., 2017) and syntactic information (such as prefixes or post-fixes) derived from external sources for English (Cao and Lu, 2017).

For Korean, integrating Korean linguistic structure at the level of *jamo*, the consonants and vowels that are much more rigidly defined than English, is shown to be effective for sentence parsing (Stratos, 2017). Previous work has looked at improving the vector representations of Korean using the character-level decomposition (Choi et al., 2017), but there is room for further investigation because Korean characters can be decomposed to *jamos* which are smaller units than the characters.

In this paper, we propose a method to integrate Korean-specific subword information to learn Korean word vectors and show improvements over previous baselines methods for word similarity, analogy, and sentiment analysis. Our first contribution is the method to decompose the words into both character-level units and jamo-level units and train the subword vectors through the Skip-Gram model. Our second major contribution is the Korean evaluation datasets for word similarity and analogy tasks, a translation of the WS-353 with annotations by 14 Korean native speakers, and 10,000 items for semantic and syntactic analogies, developed with Korean linguistic expertise. Using those datasets, we show that our model improves performance over other baseline methods without relying on external resources for word decomposition.

2 Related Work

2.1 Language-specific features for NLP

Recent studies in NLP field flourish with development of various word vector models. Although such studies aim for universal usage, distinct characteristics of individual languages still remain as a barrier for a unified model. The aforementioned issue is even more prominent when it comes to languages that have rich morphology but lack resources for research (Berardi et al., 2015). Accordingly, various studies dealing with language specific NLP technique proposed considering linguistics traits in models.

A large portion of these papers was dedicated to Chinese. Since Chinese is a logosyllabic language, (Yu et al., 2017) relevant studies focused on incorporation of different subword level features on word embedding, such as word internal structure (Wang et al., 2017), subcharacter component,(Yu et al., 2017), syllable (Assylbekov et al., 2017), radicals (Yin et al., 2016), and sememe (Niu et al., 2017).

The Korean language is a member of the agglutinative languages (Song, 2006), so previous studies have tried fusing the complex internal structure into the model. For example, a grammatical composition called 'Josa' in combination with word embedding is utilized in semantic role labeling (Nam and Kim, 2016) and exploiting *jamo* to handle morphological variation (Stratos, 2017). Also considered in prior work to obtain the word vector presentations for Korean is the syllable (Choi et al., 2017).

2.2 Subword features for NLP

Applying subword features to various NLP tasks has become popular in the NLP field. Typically, character-level information is useful when combined with the neural network based models. (Vania and Lopez, 2017; Assylbekov et al., 2017; Cao and Lu, 2017) Previous papers showed performance enhancement in various tasks including language modeling (Bojanowski et al., 2017, 2015), machine translation (Ling et al., 2015), text classification (Zhang et al., 2015; Ling et al., 2015) and parsing (Yu and Vu, 2017). In addition, the character n-gram fused model was suggested as a solution for a small dataset due to its robustness against data sparsity (Cao and Lu, 2017).

3 Model

We introduce our model training Korean word vector representations based on a subword-level information Skip-Gram. First, we briefly explain the hierarchical composition structure of Korean words to show how we decompose a Korean word into a sequence of subword components (*jamo*). Then, we extract character and *jamo* n-grams from the decomposed sequence to compute word vectors as a mean of the extracted n-grams. We train the vectors by widely-used Skip-Gram model.

3.1 Decomposition of Korean Words

Korean words are formed by an explicit hierarchical structure which can be exploited for better modeling. Every word can be decomposed into a sequence of characters, which in turn can be decomposed into *jamo*s, the smallest lexicographic units representing the consonants and vowels of the language. Unlike English which has a more flexible sequences of consonants and vowels making up syllables (e.g., "straight"), a Korean "character" which is similar to a syllable in English has a rigid structure of three *jamo*s. They have names that reflect the position in a character: 1) chosung (syllable onset), 2) joongsung (syllable nucleus), and 3) jongsung (syllable coda). The prefix *cho* in *chosung* means "first", *joong* in *joongsung* means "middle", and *jong* in *jongsung* means "end" of a character. Each component indicates how the character should be pronounced. With the exception of empty consonants, chosung and jongsung are consonants while joongsung are vowels. The *jamo*s are written with the chosung on top, with joongsung on the right of or below chosung, and jongsung on the bottom (see Fig. 1).

As shown in the top of Fig. 1, some characters such as '해Sun' lack jongsung. In this case, we add

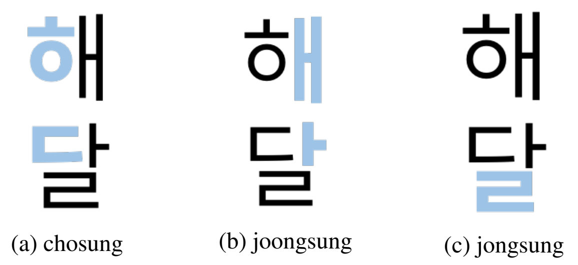


Figure 1: Example of the composition of a Korean character. Each character is comprised of 3 parts as shown in example of '달Moon'. On the other hand, as in the top case '해Sun', some characters lack the last component, 'jongsung'.

an empty jongsung symbol e such that a character always has three (*jamo*s). Thus, the character '달Moon' is decomposed into {ㄷ, ㅏ, ㄹ}, and '해Sun' into {ㅎ, ㅐ, e}.

When decomposing a word, we keep the order of the characters and the order of *jamo*s (chosung, joongsung, and jongsung) within the character. By following this rule, we ensure that a Korean word with N characters will have $3N$ *jamo*s in order. Lastly, the symbols for start of a word < and end of a word > are added to the sequence. For example, the word '강아지puppy' will be decomposed to a sequence of *jamo*s: {<, ㄱ, ㅏ, ㅇ, ㅇ, ㅏ, e, ㅈ, ㅣ, e, >}.

3.2 Extracting N-grams from *jamo* Sequence

We extract the following *jamo*-level and character-level n-grams from the decomposed Korean words: 1) character-level n-grams, and 2) inter-character *jamo*-level n-grams. These two levels of subword features can be successfully integrated into *jamo*-level n-grams by ensuring a character has three *jamo*s, adding empty jongsung symbol to the sequence. For better understanding, we start with the word '먹었다ate'.

Character-level n-grams. Since we add the empty jongsung symbol e when decomposing characters, we can find *jamo*-level trigrams representing a single character in the decomposed *jamo* sequence of a word. For example, there are three character-level unigrams in the word '먹었다ate':

{ㅁ, ㅓ, ㄱ},{ㅇ, ㅓ, ㅆ},{ㄷ, ㅏ, e}

Next, we find character-level n-grams by using the extracted unigrams. Adjacent unigrams are attached to construct n-grams. There are two character-level bigrams, and one trigram in the example:

{ㅁ, ㅓ, ㄱ, ㅇ, ㅓ, ㅆ},{ㅇ, ㅓ, ㅆ, ㄷ, ㅏ, e}
{ㅁ, ㅓ, ㄱ, ㅇ, ㅓ, ㅆ, ㅇ, ㅓ, ㅆ, ㄷ, ㅏ, e}

Lastly, we add the total *jamo* sequence of a word including < and > to the set of extracted character-level n-grams.

Inter-character *jamo*-level n-grams. Since Korean is a member of the agglutinative language, a syntactic character is attached to the semantic part in the word, and this generates many variations. These variations are often determined by *jamo*-level information. For example, usage of the subjective case '이' or '가' is determined by the existence of jongsung in the previous character. In order to learn these regularities, we consider *jamo*-level n-grams across adjacent characters as well. For instance, there are 6 inter-character *jamo*-level trigrams in the example:

{<, ㅁ, ㅓ},{ㅓ, ㄱ, ㅇ},{ㄱ, ㅇ, ㅓ},
{ㅆ, ㄷ, ㅏ},{ㅓ, ㅆ, ㄷ},{ㅏ, e, >}

3.3 Subword Information Skip-Gram

Suppose the training corpus contains a sequence of words $\{..., w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}, ...\}$, the Skip-Gram model maximizes the log probability of context word w_{t+j} under a target word w_t:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}^{2c} \log p(w_{t+j}|w_t) \tag{1}$$

where c is the size of context window, t is total number of words in the corpus. The original Skip-Gram model use softmax function outputs for $\log p(w_{t+j}|w_t)$ in Eq. 1, however, it requires large computational cost. To avoid computing softmax precisely, we approximately maximize the log probability by Noise Contrastive Estimation, and it can be simplified to the negative sampling using the binary logistic loss:

$$\log(1+e^{-s(w_{t+j},w_t)}) + \sum_{n=1}^{n_c}\log(1+e^{s(w_{t+j},w_n)}) \tag{2}$$

where n_c is the number of negative samples, and $s(w_{t+j}, w_t)$ is a scoring function. The function computes the dot product between the input of the target word vector w_t and the output of the context word vector w_{t+j}. In Skip-Gram (Mikolov et al., 2013a), an input of a word w_t is uniquely assigned over the training corpus; however, the vector in the Subword Information Skip-Gram model (Bojanowski et al., 2017) is the mean vector of the set of n-grams extracted from the word. Formally, the scoring function $s(w_t, w_{t+j})$ is:

$$\frac{1}{|G_t|}\sum_{g_t \in G_t}^{|G_t|} \mathbf{z}_{g_t}^{\mathrm{T}} \mathbf{v}_{t+j} \tag{3}$$

where the decomposed set of n-grams of w_t is G_t and its elements are g_t, $|G_t|$ is total number of elements of G_t. In general, the n-grams for $3 \leq n \leq 6$ is extracted from a word, regardless of the subword-level or compositionality of a word.

Similarly, we construct a vector representation of a Korean word by using the extracted two types of n-grams. We compute the sum of *jamo*-level n-grams, sum of character-level n-grams, and compute mean of the vectors. Let us denote character-level n-grams of w_t to G_{ct}, and inter-character *jamo*-level n-grams G_{jt}, then we obtain the scoring function $s(w_t, w_{t+j})$ as follows:

$$\frac{1}{N}\Big(\sum_{g_{ct} \in G_{ct}}^{|G_{ct}|} \mathbf{z}_{g_{ct}}^{\mathrm{T}} \mathbf{v}_{t+j} + \sum_{g_{jt} \in G_{jt}}^{|G_{jt}|} \mathbf{z}_{g_{jt}}^{\mathrm{T}} \mathbf{v}_{t+j}\Big) \tag{4}$$

where $\mathbf{z}_{g_{jt}}$ is the vector representation of the *jamo*-level n-gram g_{jt}, and $\mathbf{z}_{g_{ct}}$ is that of the character-level n-gram g_{ct}. N is sum of the number of character-level n-grams and the number of inter-character *jamo*-level n-grams $|G_{ct}| + |G_{jt}|$.

4 Experiments

4.1 Corpus

We collect a corpus of Korean documents from various sources to cover a wide context of word usages. The corpus used to train the models include: 1) Korean Wikipedia, 2) online news articles, and 3) Sejong Corpus. The corpus contains 0.12 billion tokens with 638,708 unique words. We discard words that occur fewer than ten times in the entire corpus. Details of the corpus are shown in Table 1.

Korean Wikipedia. First, we choose Korean Wikipedia articles[1] for training word vector representations. The corpus contains 0.4M articles, 3.3M sentences and 43.4M words.

Online News Articles. We collect online news articles of 5 major press from following sections: 1) society, 2) politics, 3) economics, 4) foreign, 5) culture, 6) digital. The articles were published from September to November, 2017. The corpus contains 3.2M sentences and 47.1M words.

[1]https://dumps.wikimedia.org/kowiki/20171103/

	# of words	# of sentences	# of unique words
Wikipedia	43.4M	3.3M	299,528
Online News	47.1M	3.2M	282,955
Sejong Corpus	31.4M	2.2M	231,332
Total	121.9M	8.8M	638,708

Table 1: Number of tokens, sentences and unique words of corpus used to train the word vector representations. We aggregate three sources to make the corpus containing 0.12 billions word tokens with 0.6M unique words.

Sejong Corpus. This data is a publicly available corpus[2] which is collected under a national research project named the “21st century Sejong Project”. The corpus was developed from 1998 to 2007, and contains formal text (newpapers, dictionaries, novels, etc) and informal text (transcriptions of TV shows and radio programs, etc). Thus, the corpus covers topics and context of language usage which could not be dealt with Wikipedia or news articles. We exclude some documents containing unnatural sentences such as POS-tagged sentences.

4.2 Evaluation Tasks and Datasets

We evaluate the performance of word vectors through word similarity task and word analogy task. However, to best of our knowledge, there is no Korean evaluation dataset for either task. Thus we first develop the evaluation datasets. We also test the word vectors for sentiment analysis.

4.2.1 Word Similarity Evaluation Dataset

Translating the test set. We develop a Korean version of the word similarity evaluation set. Two graduate students who speak Korean as native language translated the English word pairs in WS-353 (Finkelstein et al., 2001). Then, 14 Korean native speakers annotated the similarity between pairs by giving scores from 0 to 10 for the translated pairs, following written instructions. The original English instructions were translated into Korean as well. Among the 14 scores for each pair, we exclude the minimum and maximum scores and compute the mean of the rest of the scores. The correlation between the original scores and the annotated scores of the translated pairs is .82, which

[2]https://ithub.korean.go.kr/user/main.do

indicates that the translations are sufficiently reliable. We attribute the difference to the linguistic and cultural differences. We make the Korean version of WS-353 publicly available.[3]

4.2.2 Word Analogy Evaluation Dataset

We develop the word analogy test items to evaluate the performance of word vectors. The evaluation dataset consists of 10,000 items and includes 5,000 items for evaluating the semantic features and 5,000 for the syntactic features. We also release our word analogy evaluation dataset for future research.

Semantic Feature Evaluation To evaluate the semantic features of word vectors, we refer to the English version of the word analogy test sets. (Mikolov et al., 2013a; Gladkova et al., 2016). We cover the features in both sets and translated items into Korean. The items are clustered to five categories including miscellaneous items. Each category consists of 1,000 items.

- *Capital-Country (Capt.)* includes two word pairs representing the relation between the country name and its capital:
  아테네Athens : 그리스Greece = 바그다드Baghdad : 이라크Iraq
- *Male-Female (Gend.)* evaluates the relation between male and female:
  왕자prince:공주princess = 신사gentlemen:숙녀ladies
- *Name-Nationality (Name)* evaluates the relation between the name of celebrities or stars and their nationality:
  간디Gandhi : 인도India = 링컨Lincoln : 미국USA
- *Country-Language (Lang.)* evaluates the relation between the country name and its official language:
  아르헨티나Argentina : 스페인어Spanish = 미국USA : 영어English
- *Miscellaneous (Mics.)* includes various semantic features, such as pairs of a young animals, sound of animals, and Korean-specific color-words or regions, etc..
  개구리Frog : 올챙이tadpole = 말horse : 망아지pony
  닭chicken:꼬꼬댁cackling=호랑이tiger:으르렁growl
  파란blue:새파란bluish=노란yellow:샛노란yellowish
  부산Busan : 경상남도South Gyeongsang Province
  = 대구Daegu : 경상북도North Gyeongsang Province

Syntactic Feature Evaluation We define five representative syntactic categories and develop Korean-specific test items, rather than trying to cover the existing categories in the original sets (Mikolov et al., 2013a; Gladkova et al., 2016). This is because most of the syntactic features in these sets are not available in Korean.

We develop the test set with linguistic expert knowledge of Korean. The following case is a good example. In Korean, the subject marker is attached to the back of a word, and other case markers are also explicit at the word level. Here, word level refers to 'a phrase delimited by two whitespaces around it'. Unlike Korean, in English, subjects are determined by the position in a sentence (i.e., subject comes before the verb), so the case is not explicitly marked in the word. Similarly, there are other important and unique syntactic features of the Korean language, of which we choose the following five categories to evaluate the word vectors:

- *Case* contains various case markers attached to common nouns. This evaluates a case in Korean which is represented within a word-level:
  교수Professor : 교수가Professor+case가
  = 축구soccer : 축구가soccer+case가
- *Tense* includes a verb variation of two tenses, one of which is a present tense and a past tense for the other:
  싸우다fight : 싸웠다fought = 오다come : 왔다came
- *Voice* has a pair of verb voice, one for an active voice and a passive voice for the other. It evaluates the voice which is represented by a verbal suffix:
  팔았다sold : 팔렸다be sold
  = 평가했다evaluated : 평가됐다was evaluated
- *Verb ending form* includes various verb ending forms. The various forms are part of verbal inflection in Korean:
  가다go : 가고go+form고
  = 쓰다write : 쓰고write+form고
- *Honorific (Honr.)* evaluates a morphological variation for verbs in Korean. An honorific expression is one of the most distinctive feature in Korean compared to other languages. This test set introduces the honorific morpheme '-시-' which is used in verbs:
  도왔다helped : 도우셨다helped+honorific시
  = 됐다done : 되셨다done+honorific시

4.2.3 Sentiment Analysis

We perform a binary sentiment classification task for evaluation of word vectors. Given a sequence

[3]https://github.com/SungjoonPark/KoreanWordVectors

of words, the trained classifier should predict the sentiment from the inputs while maintaining the input word vectors fixed.

Dataset We choose Naver Sentiment Movie Corpus[4]. Scraped from Korean portal site Naver, the dataset contains 200K movie reviews. Each review is no longer than 140 characters and contain binary label according to its sentiment (1 for positive and 0 for negative). The number of samples in both sentiments is equal with 100K of positives and 100K of negatives in sum. We sample from the dataset for training (100K), validation (25K), and test set (25K). Again, each set's ratio of sentiment class is balanced. Although we apply simple preprocessing of stripping out punctuation and emoticon, the dataset is still noisy with typos, segmentation errors and abnormal word usage since its original source is raw comments from portal site.

Classifier In order to build sentiment classifier, we adopt single layer LSTM with 300 hidden units and 0.5 dropout rates. Given the final state of LSTM unit, sigmoid activation function is applied for output prediction. We use cross-entropy loss and optimize parameters through Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.001.

4.3 Comparison Models

We compare performance of our model to comparison models including word-level, character-level, and *jamo*-level Skip-Gram models trained by negative sampling. Hyperparameters of each models are tuned over word similarity task. We fix the number of training epochs 5.

Skip-Gram (SG) We first compare the performance with word-level Skip-Gram model (Mikolov et al., 2013a) where a unique vector is assigned for every unique words in the corpus. We set the number of dimensions as 300, number of negative samples to 5, and window size to 5.

Character-level Skip-Gram (SISG(ch)) splits words to character-level n-grams based on subword information skip-gram. (Bojanowski et al., 2017). We set the number of dimensions as 300, number of negative samples to 5, and window size to 5. The n was set to 2-4.

***Jamo*-level Skip-Gram with Empty Jongsung Symbol (SISG(jm))** splits words to *jamo*-level n-grams based on subword information skip-gram. (Bojanowski et al., 2017). In addition, if a character lacks jongsung, the symbol ⓔ is added. We set the number of dimensions as 300, number of negative samples to 5, and window size to 5. The n was set to 3-6. Note that setting n=3-6 and adding the jongsung symbol makes this model as a specific case of our model, containing *jamo*-level n-grams (n=3-6) and character-level n-grams (n=1-2) as well.

[4]https://github.com/e9t/nsmc

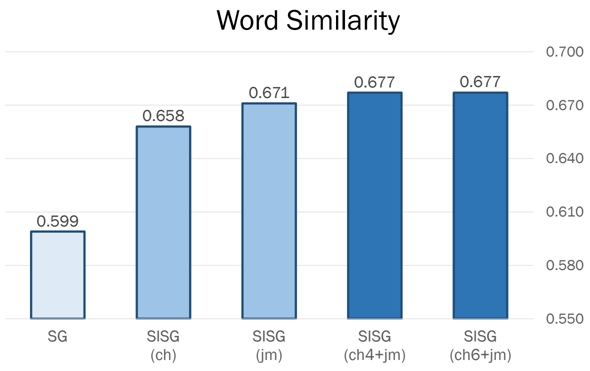


Figure 2: Spearman's correlation coefficient of word similarity task for each models. The results show higher consistency to human word similarity judgment on our method.

4.4 Optimization

In order to train our model, we apply stochastic gradient descent with linearly scheduled learning rate decay. Initial learning rate is set to .025. To speed up the training, we train the vectors in parallel with shared parameters, and they are updated asynchronously.

For our model, we set n of character n-grams to 1-4 or 1-6, and n of inter-character *jamo*-level n-grams to 3-5. We name both model as SISG(ch4+jm) and SISG(ch6+jm), respectively. The number of dimension is set to 300, window size to 5, and negative samples to 5. We train our model 5 epochs over training corpus.

5 Results

Word Similarity. We report Spearman's correlation coefficient between the human judgment and model's cosine similarity for the similarity of word pairs. Fig. 2 presents the results. For word-level skip-gram, Spearman's correlation is .599. If we decompose words into characters n-grams in order to construct word vectors (SISG(Ch)), performance is highly improved to .658. It indicates that decomposing words itself is helpful to learn good

Model	Analogy									
	Semantic					Syntactic				
	Capt	Gend	Name	Lang	Misc	Case	Tense	Voice	Form	Honr
SG	0.460	0.551	**0.537**	0.435	0.574	0.521	0.597	0.594	0.685	0.634
SISG(ch)	0.469	0.584	0.608	0.439	0.614	0.422	0.559	0.550	0.656	0.489
SISG(jm)	0.442	0.515	0.574	0.362	0.565	0.228	0.421	0.434	0.537	0.367
SISG(ch4+jm)	0.431	0.504	0.570	0.361	0.556	0.212	0.415	0.434	**0.501**	**0.364**
SISG(ch6+jm)	**0.425**	**0.498**	0.561	**0.354**	**0.554**	**0.210**	**0.414**	**0.426**	0.507	0.367

Table 2: Performance of our method and comparison models. Average cosine distance for each category in word analogy task are reported. Overall, our model outperforms comparison models, showing close distance between predicted vector $a + b - c$ and the target vector d (a:b=c:d). Specifically, performance is improved more in syntactic analogies.

Korean word vectors, which is morphologically rich language. Moreover, if the words are decomposed to deeper level (SISG(jm)), performance is further improved to .671.

Next, addition of an empty jongsung symbol ⓔ to *jamo* sequence, which reflects Korean-specific linguistic regularities, improves the quality of word vectors. SISG(jm), specific case of our model, shows higher correlation coefficient than the other baselines. Lastly, when we extend number of characters to learn in a word to 4 or 6, our models outperform others.

Word Analogy. In general, given an item a:b=c:d and corresponding word vectors u_a, u_b, u_c, u_d, the vector $u_a + u_b - u_c$ is used to compute cosine distances between the vector and the others. Then the vectors are ranked in terms of the distance by ascending order and if the vector u_d is found at the top, the item is counted as correct. Top 1 accuracy or error rate for each category is frequently used metric for this task, however, in this case these rank-based measures may not be an appropriate measure since the total number of unique n-grams (e.g., SISG) or unique words (e.g., SG) over the same corpus largely differ from each other. For fair comparison, we directly report cosine distances between the vector $u_a + u_b - u_c$ and u_d of each category, rather than evaluating ranks of the vectors. Formally, given an item a:b=c:d, we compute 3COSADD based metric:

$$1 - \cos(\mathbf{u}_a + \mathbf{u}_b - \mathbf{u}_c, \mathbf{u}_d) \qquad (5)$$

We report the average cosine distance between predicted vector $u_a + u_b - u_c$ and target vector u_d of each category.

In semantic analogies, decomposing word into character helps little for learning semantic features. However, *jamo*-level n-grams help representing overall semantic features and our model show higher performance compared to baseline models. One exception is Name-Nationality category since it mainly consists of items including proper nouns, and decomposing these nouns does not help learning the semantic feature of the word. For example, it is obvious that the semantic features of both words '간디Ghandi' and '인도India' could not be derived from that of characters or *jamo* n-grams comprising those words.

On the other hand, decomposing words does help to learn syntactic features for all categories, and decomposing a word to even deeper levels makes learning those features more effectively. Our model outperforms all other baselines, and the amount of decreased cosine distances compared to that of word-level Skip-Gram is larger than semantic categories. Korean language is agglutinative language that character-level syntactic affixes are attached to the root of the word, and the combination of them determines final form the word. Also, the form can be reduced with *jamo*-level transformation. This is the main reason that we can learn syntactic feature of Korean words if we decompose a word into character-level and *jamo*-level simultanously. We observe similar tendency when using 3COSMUL distance metric. (Levy and Goldberg, 2014)

Sentiment Analysis. We report accuracy, loss, precision, recall and f1 score for binary sentiment classification task over test set. Although overall performance is homogeneous, our method which decompose a word to 1-6 character n-grams and 3-5 *jamo* n-grams show slightly higher performance over comparison models. In addition, our approach show better results compared to character-

Model	Acc. (%)	Prc.	Rec.	F1
SG	76.15	.746	.792	.768
SISG(ch)	76.26	.774	.741	.757
SISG(jm)	76.53	**.790**	.722	.754
SISG(ch4+jm)	76.28	.755	.776	.765
SISG(ch6+jm)	**76.54**	.750	**.795**	**.772**

Table 3: Performance of sentiment classification task. 3-5 *jamo* n-grams and 1-6 chracter n-grams show slightly higher performance in terms of accuracy and f1-score over comparison models.

Word Sim.		# of chars			
		4	5	6	all
# of *jamo*s	2-4	0.660	0.655	0.659	0.651
	3-4	0.660	0.650	0.652	0.660
	3-5	**0.677**	0.672	**0.677**	0.675
	3-6	0.665	0.663	0.664	0.669

Table 4: Spearman's correlation coefficient of Word similarity task by n-gram of *jamo*s and characters. Performance are improved when the 3-5 gram of *jamo*s and 1-4 or 1-6 gram of characters.

level SISG or *jamo*-level SISG. On the other hand, word-level Skip-Gram show comparable F1-score to our model, and is even higher than other comparison models. This is because the dataset contains significant amount of proper nouns, such as movie or actor names, and these word's semantic representations are captured better by word-level representations, as shown in word analogy task.

Effect of Size n in both n-grams. Table. 4 shows performance of word similarity task for each number of inter-character *jamo*-level n-grams and character-level n-grams. For the n of *jamo*-level n-grams, including n=5,6 of n-grams and excluding bigrams show higher performance. Meanwhile, n of character-level n-grams, including all of the character n-grams while decomposing a word does not guarantee performance improvement. Since most of the Korean word consists of no more than 6 characters (97.2% of total corpus), it seems maximum number of n=6 in character n-gram is large enough to learn word vectors. In addition, words with no more than 4 characters takes 82.6% of total corpus, so that n=4 sufficient to learn character n-grams as well.

6 Conclusion and Discussions

In this paper, we present how to decompose a Korean character into a sequence of *jamo*s with empty jongsung symbols, then extract character-level n-grams and intercharacter *jamo*-level n-grams from that sequence. Both n-grams construct a word vector representation by computing the average of n-grams, and these vectors are trained by subword-level information Skip-Gram. Prior to evaluating the performance of the vectors, we developed test set for word similarity and word analogy tasks for Korean.

We demonstrated the effectiveness of the learned word vectors in capturing the semantic and syntactic information by evaluating these vectors with word similarity and word analogy tasks. Specifically, the vectors using both *jamo* and character-level information can represent syntactic features more precisely even in an agglutinative language. Furthermore, sentiment classification results of our work indicate that the representative power of the vectors positively contributes to downstream NLP task.

Decomposing Korean word into *jamo*-level or character unigram helps capturing syntactic information. For example, Korean words add a character to the root of the word (e.g., '-은' subjective case, '-었' for past tense '-시-' for honorific, '-히-' for voice, and '-고-' for verb ending form.) Then composed word can be reduced to have fewer characters by transforming *jamo*s, such as '되었다' to '됐다'. Hence, the inter-character *jamo*-level n-grams also help capture these features. On the other hand, larger n-grams such as character-level trigram will learn unique meaning of that word since those larger component of the word will mostly occur with that word. By leveraging both features, our method produces word vectors reflecting linguistic features effectively, and thus, outperforms previous word-level approaches.

Since Korean words are divisible once more into grapheme level, resulting in longer sequence of *jamo*s for a given word, we plan to explore potential applicability of deeper level of subword information in Korean. Meanwhile, we will further train our model over noisy data and investigate how it is dealing with noisy words. Generally, informal Korean text contains intentional typos ('맛잇다'delicious' with typo'), stand-alone *jamo* as a character, ('ㅋㅋlol') and segmentation errors. ('같이가다'go together' without space'). Since these errors occur frequently, it is important to apply the vectors in training NLP models over real-word data. We plan to apply these vectors for various neural network based NLP models, such as conversation modeling. Lastly, since our method can capture Korean syntactic features through *jamo* and character n-grams, we can apply the same idea to other tasks such as POS tagging and parsing.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01778, Development of Explainable Human-level Deep Machine Learning Inference Framework).

References

Zhenisbek Assylbekov, Rustem Takhanov, Bagdat Myrzakhmetov, and Jonathan N Washington. 2017. Syllable-aware neural language models: A failure to beat character-aware ones. In *Proc. of EMNLP*.

Giacomo Berardi, Andrea Esuli, and Diego Marcheggiani. 2015. Word embeddings go to italy: A comparison of models and training datasets. In *IIR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL* .

Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. 2015. Alternative structures for character-level rnns. *arXiv preprint arXiv:1511.06303* .

Shaosheng Cao and Wei Lu. 2017. Improving word embeddings with convolutional feature learning and subword information. In *Proc. of AAAI*.

Sanghyuk Choi, Taeuk Kim, Jinseok Seol, and Sanggoo Lee. 2017. A syllable-based technique for word embeddings of korean words. In *Proc. of the First Workshop on Subword and Character Level Models in NLP*. pages 36–40.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proc. of WWW*.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proc. of the NAACL Student Research Workshop*. pages 8–15.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proc. of AAAI*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the ACL* .

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth CoNLL*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. In *Proc. of ACL*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proc. of IJCAI*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Kyeong-Min Nam and Yu-Seop Kim. 2016. A word embedding and a josa vector for korean unsupervised semantic role induction. In *AAAI*. pages 4240–4241.

Masato Neishi, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda. 2017. A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proc. of AAAI*.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

Karl Stratos. 2017. A sub-character architecture for korean language processing. In *Proc. of EMNLP*. pages 721–726.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proc. of ACL*.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2017. Exploiting word internal structures for generic chinese sentence representation. In *Proc. of EMNLP*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proc. of NAACL*.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proc. of EMNLP*. pages 981–986.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proc. of EMNLP*. pages 286–291.

Xiang Yu and Ngoc Thang Vu. 2017. Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages. In *Proc. of ACL*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of NIPS*.

Incorporating Chinese Characters of Words for Lexical Sememe Prediction

Huiming Jin[1*†], **Hao Zhu**[2†], **Zhiyuan Liu**[2,3‡], **Ruobing Xie**[4],
Maosong Sun[2,3], **Fen Lin**[4], **Leyu Lin**[4]
[1] Shenyuan Honors College, Beihang University, Beijing, China
[2] Beijing National Research Center for Information Science and Technology,
State Key Laboratory of Intelligent Technology and Systems,
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]Jiangsu Collaborative Innovation Center for Language Ability,
Jiangsu Normal University, Xuzhou 221009 China
[4] Search Product Center, WeChat Search Application Department, Tencent, China

Abstract

Sememes are minimum semantic units of concepts in human languages, such that each word sense is composed of one or multiple sememes. Words are usually manually annotated with their sememes by linguists, and form linguistic common-sense knowledge bases widely used in various NLP tasks. Recently, the lexical sememe prediction task has been introduced. It consists of automatically recommending sememes for words, which is expected to improve annotation efficiency and consistency. However, existing methods of lexical sememe prediction typically rely on the external context of words to represent the meaning, which usually fails to deal with low-frequency and out-of-vocabulary words. To address this issue for Chinese, we propose a novel framework to take advantage of both internal character information and external context information of words. We experiment on HowNet, a Chinese sememe knowledge base, and demonstrate that our framework outperforms state-of-the-art baselines by a large margin, and maintains a robust performance even for low-frequency words. [i]

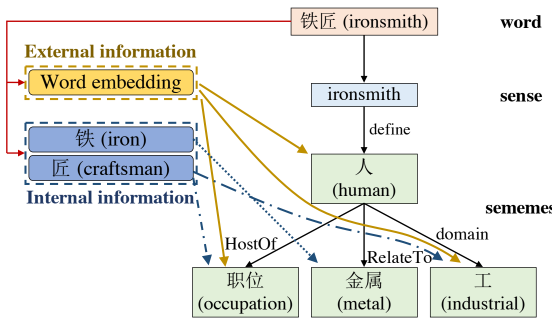


Figure 1: Sememes of the word “铁匠” (ironsmith) in HowNet, where *occupation*, *human* and *industrial* can be inferred by both external (contexts) and internal (characters) information, while *metal* is well-captured only by the internal information within the character “ 铁” (iron).

1 Introduction

A sememe is an indivisible semantic unit for human languages defined by linguists (Bloomfield, 1926). The semantic meanings of concepts (e.g., words) can be composed by a finite number of sememes. However, the sememe set of a word is not explicit, which is why linguists build knowledge bases (KBs) to annotate words with sememes manually.

HowNet is a classical widely-used sememe KB (Dong and Dong, 2006). In HowNet, linguists manually define approximately 2,000 sememes, and annotate more than 100,000 common words in Chinese and English with their relevant sememes in hierarchical structures. HowNet is well developed and has a wide range of applications in many NLP tasks, such as word sense disambiguation (Duan et al., 2007), sentiment analysis (Fu et al., 2013; Huang et al., 2014) and cross-lingual word similarity (Xia et al., 2011).

Since new words and phrases are emerging every day and the semantic meanings of existing concepts keep changing, it is time-consuming and work-intensive for human experts to annotate new

∗ Work done while doing internship at Tsinghua University.
† Equal contribution. Huiming Jin proposed the overall idea, designed the first experiment, conducted both experiments, and wrote the paper; Hao Zhu made suggestions on ensembling, proposed the second experiment, and spent a lot of time on proofreading the paper and making revisions. All authors helped shape the research, analysis and manuscript.
‡ Corresponding author: Z. Liu (liuzy@tsinghua.edu.cn)
i Code is available at `https://github.com/thunlp/Character-enhanced-Sememe-Prediction`

concepts and maintain consistency for large-scale sememe KBs. To address this issue, Xie et al. (2017) propose an automatic sememe prediction framework to assist linguist annotation. They assumed that words which have similar semantic meanings are likely to share similar sememes. Thus, they propose to represent word meanings as embeddings (Pennington et al., 2014; Mikolov et al., 2013) learned from a large-scale text corpus, and they adopt collaborative filtering (Sarwar et al., 2001) and matrix factorization (Koren et al., 2009) for sememe prediction, which are concluded as Sememe Prediction with Word Embeddings (SPWE) and Sememe Prediction with Sememe Embeddings (SPSE) respectively. However, those methods ignore the internal information within words (e.g., the characters in Chinese words), which is also significant for word understanding, especially for words which are of low-frequency or do not appear in the corpus at all. In this paper, we take Chinese as an example and explore methods of taking full advantage of both external and internal information of words for sememe prediction.

In Chinese, words are composed of one or multiple characters, and most characters have corresponding semantic meanings. As shown by Yin (1984), more than 90% of Chinese characters in modern Chinese corpora are morphemes. Chinese words can be divided into single-morpheme words and compound words, where compound words account for a dominant proportion. The meanings of compound words are closely related to their internal characters as shown in Fig. 1. Taking a compound word "铁匠" (ironsmith) for instance, it consists of two Chinese characters: "铁" (iron) and "匠" (craftsman), and the semantic meaning of "铁匠" can be inferred from the combination of its two characters (*iron* + *craftsman* $\rightarrow$ *ironsmith*). Even for some single-morpheme words, their semantic meanings may also be deduced from their characters. For example, both characters of the single-morpheme word "徘徊" (hover) represent the meaning of "hover" or "linger". Therefore, it is intuitive to take the internal character information into consideration for sememe prediction.

In this paper, we propose a novel framework for Character-enhanced Sememe Prediction (CSP), which leverages both internal character information and external context for sememe prediction. CSP predicts the sememe candidates for a target word from its word embedding and the corresponding character embeddings. Specifically, we follow SPWE and SPSE as introduced by Xie et al. (2017) to model external information and propose Sememe Prediction with Word-to-Character Filtering (SPWCF) and Sememe Prediction with Character and Sememe Embeddings (SPCSE) to model internal character information. In our experiments, we evaluate our models on the task of sememe prediction using HowNet. The results show that CSP achieves state-of-the-art performance and stays robust for low-frequency words.

To summarize, the key contributions of this work are as follows: (1) To the best of our knowledge, this work is the first to consider the internal information of characters for sememe prediction. (2) We propose a sememe prediction framework considering both external and internal information, and show the effectiveness and robustness of our models on a real-world dataset.

2 Related Work

Knowledge Bases. Knowledge Bases (KBs), aiming to organize human knowledge in structural forms, are playing an increasingly important role as infrastructural facilities of artificial intelligence and natural language processing. KBs rely on manual efforts (Bollacker et al., 2008), automatic extraction (Auer et al., 2007), manual evaluation (Suchanek et al., 2007), automatic completion and alignment (Bordes et al., 2013; Toutanova et al., 2015; Zhu et al., 2017) to build, verify and enrich their contents. WordNet (Miller, 1995) and BabelNet (Navigli and Ponzetto, 2012) are the representative of linguist KBs, where words of similar meanings are grouped to form thesaurus (Nastase and Szpakowicz, 2001). Apart from other linguistic KBs, sememe KBs such as HowNet (Dong and Dong, 2006) can play a significant role in understanding the semantic meanings of concepts in human languages and are favorable for various NLP tasks: information structure annotation (Gan and Wong, 2000), word sense disambiguation (Gan et al., 2002), word representation learning (Niu et al., 2017; Faruqui et al., 2015), and sentiment analysis (Fu et al., 2013) inter alia. Hence, lexical sememe prediction is an important task to construct sememe KBs.

Automatic Sememe Prediction. Automatic sememe prediction is proposed by Xie et al. (2017).

For this task, they propose SPWE and SPSE, which are inspired by collaborative filtering (Sarwar et al., 2001) and matrix factorization (Koren et al., 2009) respectively. SPWE recommends the sememes of those words that are close to the unlabelled word in the embedding space. SPSE learns sememe embeddings by matrix factorization (Koren et al., 2009) within the same embedding space of words, and it then recommends the most relevant sememes to the unlabelled word in the embedding space. In these methods, word embeddings are learned based on external context information (Pennington et al., 2014; Mikolov et al., 2013) on large-scale text corpus. These methods do not exploit internal information of words, and fail to handle low-frequency words and out-of-vocabulary words. In this paper, we propose to incorporate internal information for lexical sememe prediction.

Subword and Character Level NLP. Subword and character level NLP models the internal information of words, which is especially useful to address the out-of-vocabulary (OOV) problem. Morphology is a typical research area of subword level NLP. Subword level NLP has also been widely considered in many NLP applications, such as keyword spotting (Narasimhan et al., 2014), parsing (Seeker and Çetinoğlu, 2015), machine translation (Dyer et al., 2010), speech recognition (Creutz et al., 2007), and paradigm completion (Sutskever et al., 2014; Bahdanau et al., 2015; Cotterell et al., 2016a; Kann et al., 2017; Jin and Kann, 2017). Incorporating subword information for word embeddings (Bojanowski et al., 2017; Cotterell et al., 2016b; Chen et al., 2015; Wieting et al., 2016; Yin et al., 2016) facilitates modeling rare words and can improve the performance of several NLP tasks to which the embeddings are applied. Besides, people also consider character embeddings which have been utilized in Chinese word segmentation (Sun et al., 2014).

The success of previous work verifies the feasibility of utilizing internal character information of words. We design our framework for lexical sememe prediction inspired by these methods.

3 Background and Notation

In this section, we first introduce the organization of *sememes*, *senses* and *words* in HowNet. Then we offer a formal definition of lexical sememe prediction and develop our notation.

3.1 Sememes, Senses and Words in HowNet

HowNet provides sememe annotations for Chinese words, where each word is represented as a hierarchical tree-like sememe structure. Specifically, a word in HowNet may have various *senses*, which respectively represent the semantic meanings of the word in the real world. Each *sense* is defined as a hierarchical structure of *sememes*. For instance, as shown in the right part of Fig. 1, the word “铁匠” (ironsmith) has one sense, namely *ironsmith*. The sense *ironsmith* is defined by the sememe “人” (human) which is modified by sememe “职位” (occupation), “金属” (metal) and “工” (industrial). In HowNet, linguists use about $2,000$ sememes to describe more than $100,000$ words and phrases in Chinese with various combinations and hierarchical structures.

3.2 Formalization of the Task

In this paper, we focus on the relationships between the *words* and the *sememes*. Following the settings of Xie et al. (2017), we simply ignore the senses and the hierarchical structure of sememes, and we regard the sememes of all senses of a word together as the sememe set of the word.

We now introduce the notation used in this paper. Let $G = (W, S, T)$ denotes the sememe KB, where $W = \{w_1, w_2, \ldots, w_{|W|}\}$ is the set of words, S is the set of sememes, and $T \subseteq W \times S$ is the set of relation pairs between words and sememes. We denote the Chinese character set as C, with each word $w_i \in C^+$. Each word w has its sememe set $S_w = \{s|(w, s) \in T\}$. Take the word “铁匠” (ironsmith) for example, the sememe set $S_{\text{铁匠}\,(\textit{ironsmith})}$ consists of “人” (human), “职位” (occupation), “金属” (metal) and “工” (industrial).

Given a word $w \in C^+$, the task of lexical sememe prediction aims to predict the corresponding $P(s|w)$ of sememes in S to recommend them to w.

4 Methodology

In this section, we present our framework for lexical sememe prediction (SP). For each unlabelled word, our framework aims to recommend the most appropriate sememes based on the internal and external information. Because of introducing character information, our framework can work for both high-frequency and low-frequency words.

Our framework is the ensemble of two parts: sememe prediction with internal information (i.e., *internal* models), and sememe prediction with external information (i.e., *external* models). Explicitly, we adopt SPWE, SPSE, and their ensemble (Xie et al., 2017) as *external* models, and we take SPWCF, SPCSE, and their ensemble as *internal* models.

In the following sections, we first introduce SPWE and SPSE. Then, we show the details of SPWCF and SPCSE. Finally, we present the method of model ensembling.

4.1 SP with External Information

SPWE and SPSE are introduced by Xie et al. (2017) as the state of the art for sememe prediction. These methods represent word meanings with embeddings learned from external information, and apply the ideas of collaborative filtering and matrix factorization in recommendation systems for sememe predication.

SP with Word Embeddings (SPWE) is based on the assumption that similar words should have similar sememes. In SPWE, the similarity of words are measured by cosine similarity. The score function $P(s_j|w)$ of sememe s_j given a word w is defined as:

$$P(s_j|w) \sim \sum_{w_i \in W} \cos(\mathbf{w}, \mathbf{w_i}) \cdot \mathbf{M}_{ij} \cdot c^{r_i}, \quad (1)$$

where $\mathbf{w}$ and $\mathbf{w_i}$ are pre-trained word embeddings of words w and w_i. $\mathbf{M}_{ij} \in \{0, 1\}$ indicates the annotation of sememe s_j on word w_i, where $\mathbf{M}_{ij} = 1$ indicates the word $s_j \in S_{w_i}$ and otherwise is not. r_i is the descend cosine word similarity rank between w and w_i, and $c \in (0, 1)$ is a hyper-parameter.

SP with Sememe Embeddings (SPSE) aims to map sememes into the same low-dimensional space of the word embeddings to predict the semantic correlations of the sememes and the words. This method learns two embeddings $\mathbf{s}$ and $\bar{\mathbf{s}}$ for each sememe by solving matrix factorization with the loss function defined as:

$$\mathcal{L} = \sum_{w_i \in W, s_j \in S} \left(\mathbf{w}_i \cdot (\mathbf{s}_j + \bar{\mathbf{s}}_j) + \mathbf{b}_i + \mathbf{b}'_j - \mathbf{M}_{ij}\right)^2 + \lambda \sum_{s_j, s_k \in S} (\mathbf{s}_j \cdot \bar{\mathbf{s}}_k - \mathbf{C}_{jk})^2, \quad (2)$$

where $\mathbf{M}$ is the same matrix used in SPWE. $\mathbf{C}$ indicates the correlations between sememes, in which $\mathbf{C}_{jk}$ is defined as the point-wise mutual information $\text{PMI}(s_j, s_k)$. The sememe embeddings are learned by factorizing the word-sememe matrix $\mathbf{M}$ and the sememe-sememe matrix $\mathbf{C}$ synchronously with fixed word embeddings. $\mathbf{b}_i$ and $\mathbf{b}'_j$ denote the bias of w_i and s_j, and λ is a hyper-parameter. Finally, the score of sememe s_j given a word w is defined as:

$$P(s_j|w) \sim \mathbf{w} \cdot (\mathbf{s}_j + \bar{\mathbf{s}}_j). \quad (3)$$

4.2 SP with Internal Information

We design two methods for sememe prediction with only internal character information without considering contexts as well as pre-trained word embeddings.

4.2.1 SP with Word-to-Character Filtering (SPWCF)

Inspired by collaborative filtering (Sarwar et al., 2001), we propose to recommend sememes for an unlabelled word according to its similar words based on internal information. Instead of using pre-trained word embeddings, we consider words as *similar* if they contain the same characters at the same positions.

In Chinese, the meaning of a character may vary according to its position within a word (Chen et al., 2015). We consider three positions within a word: `Begin`, `Middle`, and `End`. For example, as shown in Fig. 2, the character at the `Begin` position of the word "火车站" (railway station) is "火" (fire), while "车" (vehicle) and "站" (station) are at the `Middle` and `End` position respectively. The character "站" usually means *station* when it is at the `End` position, while it usually means *stand* at the `Begin` position like in "站立" (stand), "站岗哨兵" (standing guard) and "站起来" (stand up).

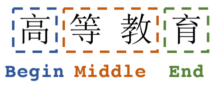


Figure 2: An example of the position of characters in a word.

Formally, for a word $w = c_1 c_2 ... c_{|w|}$, we define $\pi_B(w) = \{c_1\}$, $\pi_M(w) = \{c_2, ..., c_{|w-1|}\}$, $\pi_E(w) = \{c_{|w|}\}$, and

$$P_p(s_j|c) \sim \frac{\sum_{w_i \in W \wedge c \in \pi_p(w_i)} \mathbf{M}_{ij}}{\sum_{w_i \in W \wedge c \in \pi_p(w_i)} |S_{w_i}|}, \quad (4)$$

that represents the score of a sememe s_j given a character c and a position p, where π_p may be π_B, π_M, or π_E. $\mathbf{M}$ is the same matrix used in Eq. (1). Finally, we define the score function $P(s_j|w)$ of sememe s_j given a word w as:

$$P(s_j|w) \sim \sum_{p \in \{B,M,E\}} \sum_{c \in \pi_p(w)} P_p(s_j|c). \tag{5}$$

SPWCF is a simple and efficient method. It performs well because compositional semantics are pervasive in Chinese compound words, which makes it straightforward and effective to find similar words according to common characters.

4.2.2 SP with Character and Sememe Embeddings (SPCSE)

The method Sememe Prediction with Word-to-Character Filtering (SPWCF) can effectively recommend the sememes that have strong correlations with characters. However, just like SPWE, it ignores the relations between sememes. Hence, inspired by SPSE, we propose Sememe Prediction with Character and Sememe Embeddings (SPCSE) to take the relations between sememes into account. In SPCSE, we instead learn the sememe embeddings based on internal character information, then compute the semantic distance between sememes and words for prediction.

Inspired by GloVe (Pennington et al., 2014) and SPSE, we adopt matrix factorization in SPCSE, by decomposing the word-sememe matrix and the sememe-sememe matrix simultaneously. Instead of using pre-trained word embeddings in SPSE, we use pre-trained character embeddings in SPCSE. Since the ambiguity of characters is stronger than that of words, multiple embeddings are learned for each character (Chen et al., 2015). We select the most representative character and its embedding to represent the word meaning. Because low-frequency characters are much rare than those low-frequency words, and even low-frequency words are usually composed of common characters, it is feasible to use pre-trained character embeddings to represent rare words. During factorizing the word-sememe matrix, the character embeddings are fixed.

We set N_e as the number of embeddings for each character, and each character c has N_e embeddings $\mathbf{c}^1, ..., \mathbf{c}^{N_e}$. Given a word w and a sememe s, we select the embedding of a character of w closest to the sememe embedding by cosine distance as the representation of the word w,

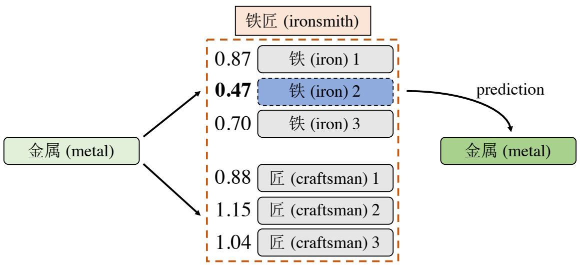


Figure 3: An example of adopting multiple-prototype character embeddings. The numbers are the cosine distances. The sememe "金属" (metal) is the closest to one embedding of "铁" (iron).

as shown in Fig. 3. Specifically, given a word $w = c_1...c_{|w|}$ and a sememe s_j, we define

$$\hat{k}, \hat{r} = \arg\min_{k,r} \left[1 - \cos(\mathbf{c}_k^r, (\mathbf{s}'_j + \bar{\mathbf{s}}'_j))\right], \tag{6}$$

where $\hat{k}$ and $\hat{r}$ indicate the indices of the character and its embedding closest to the sememe s_j in the semantic space. With the same word-sememe matrix $\mathbf{M}$ and sememe-sememe correlation matrix $\mathbf{C}$ in Eq. (2), we learn the sememe embeddings with the loss function:

$$\mathcal{L} = \sum_{w_i \in W, s_j \in S} \left(\mathbf{c}_{\hat{k}}^{\hat{r}} \cdot (\mathbf{s}'_j + \bar{\mathbf{s}}'_j) + \mathbf{b}_{\hat{k}}^c + \mathbf{b}''_j - \mathbf{M}_{ij}\right)^2 + \lambda' \sum_{s_j, s_q \in S} \left(\mathbf{s}'_j \cdot \bar{\mathbf{s}}'_q - \mathbf{C}_{jq}\right)^2, \tag{7}$$

where $\mathbf{s}'_j$ and $\bar{\mathbf{s}}'_j$ are the sememe embeddings for sememe s_j, and $\mathbf{c}_{\hat{k}}^{\hat{r}}$ is the embedding of the character that is the closest to sememe s_j within w_i. Note that, as the characters and the words are not embedded into the same semantic space, we learn new sememe embeddings instead of using those learned in SPSE, hence we use different notations for the sake of distinction. $\mathbf{b}_k^c$ and $\mathbf{b}''_j$ denote the biases of c_k and s_j, and λ' is the hyper-parameter adjusting the two parts. Finally, the score function of word $w = c_1...c_{|w|}$ is defined as:

$$P(s_j|w) \sim \mathbf{c}_{\hat{k}}^{\hat{r}} \cdot \left(\mathbf{s}'_j + \bar{\mathbf{s}}'_j\right). \tag{8}$$

4.3 Model Ensembling

SPWCF / SPCSE and SPWE / SPSE take different sources of information as input, which means that they have different characteristics: SPWCF / SPCSE only have access to internal information, while SPWE / SPSE can only make use of external information. On the other hand, just like the difference between SPWE and SPSE, SPWCF originates from collaborative filtering, whereas SPCSE uses matrix factorization. All of those methods have in common that they tend to recommend the sememes of *similar* words, but they diverge in their interpretation of *similar*.

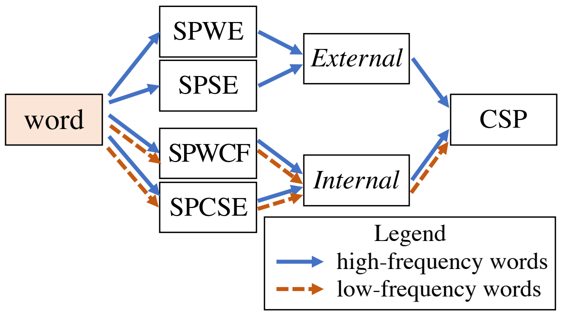


Figure 4: The illustration of model ensembling.

Hence, to obtain better prediction performance, it is necessary to combine these models. We denote the ensemble of SPWCF and SPCSE as the *internal* model, and we denote the ensemble of SPWE and SPSE as the *external* model. The ensemble of the *internal* and the *external* models is our novel framework CSP. In practice, for words with reliable word embeddings, i.e., high-frequency words, we can use the integration of the *internal* and the *external* models; for words with extremely low frequencies (e.g., having no reliable word embeddings), we can just use the *internal* model and ignore the *external* model, because the external information is noise in this case. Fig. 4 shows model ensembling in different scenarios. For the sake of comparison, we use the integration of SPWCF, SPCSE, SPWE, and SPSE as CSP in our all experiments. In this paper, two models are integrated by simple weighted addition.

5 Experiments

In this section, we evaluate our models on the task of sememe prediction. Additionally, we analyze the performance of different methods for various word frequencies. We also execute an elaborate case study to demonstrate the mechanism of our methods and the advantages of using internal information.

5.1 Dataset

We use the human-annotated sememe KB HowNet for sememe prediction. In HowNet, $103,843$ words are annotated with $212,539$ senses, and each sense is defined as a hierarchical structure of sememes. There are about $2,000$ sememes in HowNet. However, the frequencies of some sememes in HowNet are very low, such that we consider them unimportant and remove them. Our final dataset contains $1,400$ sememes. For learning the word and character embeddings, we use the Sogou-T corpus[ii] (Liu et al., 2012), which contains 2.7 billion words.

5.2 Experimental Settings

In our experiments, we evaluate SPWCF, SPCSE, and SPWCF + SPCSE which only use internal information, and the ensemble framework CSP which uses both internal and external information for sememe prediction. We use the state-of-the-art models from Xie et al. (2017) as our baselines. Additionally, we use the SPWE model with word embeddings learned by fastText (Bojanowski et al., 2017) that considers both internal and external information as a baseline.

For the convenience of comparison, we select $60,000$ high-frequency words in Sogou-T corpus from HowNet. We divide the $60,000$ words into train, dev, and test sets of size $48,000$, $6,000$, and $6,000$, respectively, and we keep them fixed throughout all experiments except for Section 5.4. In Section 5.4, we utilize the same train and dev sets, but use other words from HowNet as the test set to analyze the performance of our methods for different word frequency scenarios. We select the hyper-parameters on the dev set for all models including the baselines and report the evaluation results on the test set.

We set the dimensions of the word, sememe, and character embeddings to be 200. The word embeddings are learned by GloVe (Pennington et al., 2014). For the baselines, in SPWE, the hyper-parameter c is set to 0.8, and the model considers no more than $K = 100$ nearest words. We set the probability of decomposing zero elements in the word-sememe matrix in SPSE to be 0.5%. λ in Eq. (2) is 0.5. The model is trained for 20 epochs, and the initial learning rate is 0.01, which decreases through iterations. For fastText, we use skip-gram with hierarchical softmax to learn word embeddings, and we set the minimum length of character n-grams to be 1 and the maximum length

ii Sogou-T corpus is provided by Sogou Inc., a Chinese commercial search engine company. https://www.sogou.com/labs/resource/t.php of character n-grams to be 2. For model ensembling, we use $\frac{\lambda_{SPWE}}{\lambda_{SPSE}} = 2.1$ as the addition weight.

For SPCSE, we use Cluster-based Character Embeddings (Chen et al., 2015) to learn pre-trained character embeddings, and we set N_e to be 3. We set λ' in Eq. (7) to be 0.1, and the model is trained for 20 epochs. The initial learning rate is 0.01 and decreases during training as well. Since generally each character can relate to about 15 - 20 sememes, we set the probability of decomposing zero elements in the word-sememe matrix in SPCSE to be 2.5%. The ensemble weight of SPWCF and SPCSE $\frac{\lambda_{SPWCF}}{\lambda_{SPCSE}} = 4.0$. For better performance of the final ensemble model CSP, we set $\lambda = 0.1$ and $\frac{\lambda_{SPWE}}{\lambda_{SPSE}} = 0.3125$, though 0.5 and 2.1 are the best for SPSE and SPWE + SPSE. Finally, we choose $\frac{\lambda_{internal}}{\lambda_{external}} = 1.0$ to integrate the *internal* and *external* models.

5.3 Sememe Prediction

5.3.1 Evaluation Protocol

The task of sememe prediction aims to recommend appropriate sememes for unlabelled words. We cast this as a multi-label classification task, and adopt mean average precision (MAP) as the evaluation metric. For each unlabelled word in the test set, we rank all sememe candidates with the scores given by our models as well as baselines, and we report the MAP results. The results are reported on the test set, and the hyper-parameters are tuned on the dev set.

5.3.2 Experiment Results

The evaluation results are shown in Table 1. We can observe that:

Method	MAP
SPSE	0.411
SPWE	0.565
SPWE+SPSE	0.577
SPWCF	0.467
SPCSE	0.331
SPWCF + SPCSE	**0.483**
SPWE + fastText	0.531
CSP	**0.654**

Table 1: Evaluation results on sememe prediction. The result of SPWCF + SPCSE is bold for comparing with other methods (SPWCF and SPCSE) which use only internal information.

(1) Considerable improvements are obtained via model ensembling, and the CSP model achieves state-of-the-art performance. CSP combines the internal character information with the external context information, which significantly and consistently improves performance on sememe prediction. Our results confirm the effectiveness of a combination of internal and external information for sememe prediction; since different models focus on different features of the inputs, the ensemble model can absorb the advantages of both methods.

(2) The performance of SPWCF + SPCSE is better than that of SPSE, which means using only internal information could already give good results for sememe prediction as well. Moreover, in *internal* models, SPWCF performs much better than SPCSE, which also implies the strong power of collaborative filtering.

(3) The performance of SPWCF + SPCSE is worse than SPWE + SPSE. This indicates that it is still difficult to figure out the semantic meanings of a word without contextual information, due to the ambiguity and meaning vagueness of internal characters. Moreover, some words are not compound words (e.g., single-morpheme words or transliterated words), whose meanings can hardly be inferred directly by their characters. In Chinese, internal character information is just partial knowledge. We present the results of SPWCF and SPCSE merely to show the capability to use the internal information in isolation. In our case study, we will demonstrate that *internal* models are powerful for low-frequency words, and can be used to predict senses that do not appear in the corpus.

5.4 Analysis on Different Word Frequencies

To verify the effectiveness of our models on different word frequencies, we incorporate the remaining words in HowNet[iii] into the test set. Since the remaining words are low-frequency, we mainly focus on words with long-tail distribution. We count the number of occurrences in the corpus for each word in the test set and group them into eight categories by their frequency. The evaluation results are shown in Table 2, from which we can observe that:

iii In detail, we do not use the numeral words, punctuations, single-character words, the words do not appear in Sogou-T corpus (because they need to appear at least for one time to get the word embeddings), and foreign abbreviations.

word frequency occurrences	⩽ 50 8537	51– 100 4868	101 – 1,000 3236	1,001 – 5,000 2036	5,001 – 10,000 663	10,001 – 30,000 753	>30,000 686
SPWE	0.312	0.437	0.481	0.558	0.549	0.556	0.509
SPSE	0.187	0.273	0.339	0.409	0.407	0.424	0.386
SPWE + SPSE	0.284	0.414	0.478	0.556	0.548	0.554	0.511
SPWCF	0.456	0.414	0.400	0.443	0.462	0.463	0.479
SPCSE	0.309	0.291	0.286	0.312	0.339	0.353	0.342
SPWCF + SPCSE	0.467	0.437	0.418	0.456	0.477	0.477	0.494
SPWE + fastText	0.495	0.472	0.462	0.520	0.508	0.499	0.490
CSP	**0.527**	**0.555**	**0.555**	**0.626**	**0.632**	**0.641**	**0.624**

Table 2: MAP scores on sememe prediction with different word frequencies.

words	models	Top 5 sememes
钟表匠 (clockmaker)	internal	人**(human)**, 职位**(occupation)**, 部件(part), 时间**(time)**, 告诉**(tell)**
	external	人**(human)**, 专(ProperName), 地方(place), 欧洲(Europe), 政(politics)
	ensemble	人**(human)**, 职位**(occupation)**, 告诉**(tell)**, 时间**(time)**, 用具**(tool)**
奥斯卡 (Oscar)	internal	专**(ProperName)**, 地方(place), 市(city), 人(human), 国都(capital)
	external	奖励**(reward)**, 艺**(entertainment)**, 专**(ProperName)**, 用具(tool), 事情**(fact)**
	ensemble	专**(ProperName)**, 奖励**(reward)**, 艺**(entertainment)**, 著名(famous), 地方(place)

Table 3: Examples of sememe prediction. For each word, we present the top 5 sememes predicted by the *internal* model, *external* model and the final ensemble model (CSP). Bold sememes are correct.

(1) The performances of SPSE, SPWE, and SPWE + SPSE decrease dramatically with low-frequency words compared to those with high-frequency words. On the contrary, the performances of SPWCF, SPCSE, and SPWCF + SPCSE, though weaker than that on high-frequency words, is not strongly influenced in the long-tail scenario. The performance of CSP also drops since CSP also uses external information, which is not sufficient with low-frequency words. These results show that the word frequencies and the quality of word embeddings can influence the performance of sememe prediction methods, especially for *external* models which mainly concentrate on the word itself. However, the *internal* models are more robust when encountering long-tail distributions. Although words do not need to appear too many times for learning good word embeddings, it is still hard for *external models* to recommend sememes for low-frequency words. While since *internal* models do not use external word embeddings, they can still work in such scenario. As for the performance on high-frequency words, since these words are used widely, the ambiguity of high-frequency words is thus much stronger, while the *internal* models are still stable for high-frequency words.

(2) The results also indicate that even low-frequency words in Chinese are mostly composed of common characters, and thus it is possible to utilize internal character information for sememe prediction on words with long-tail distribution (even on those new words that never appear in the corpus). Moreover, the stability of the MAP scores given by our methods on various word frequencies also reflects the reliability and universality of our models in real-world sememe annotations in HowNet. We will give detailed analysis in our case study.

5.5 Case Study

The results of our main experiments already show the effectiveness of our models. In this case study, we further investigate the outputs of our models to confirm that character-level knowledge is truly incorporated into sememe prediction.

In Table 3, we demonstrate the top 5 sememes for "钟表匠" (clockmaker) and "奥斯卡" (Oscar, i.e., the Academy Awards). "钟表匠" (clockmaker) is a typical compound word, while "奥斯卡" (Oscar) is a transliterated word. For each word, the top 5 results generated by the internal model (SPWCF + SPCSE), the *external* model (SPWE + SPSE) and the ensemble model (CSP) are listed.

The word "钟表匠" (clockmaker) is composed of three characters: "钟" (bell, clock), "表" (clock, watch) and "匠" (craftsman). Humans can intuitively conclude that *clock* + *craftsman* $\rightarrow$ *clockmaker*. However, the *external* model does not perform well for this example. If we investigate the word embedding of “钟表匠” (clockmaker), we can know why this method recommends these unreasonable sememes. The closest 5 words in the train set to “钟表匠” (clockmaker) by cosine similarity of their embeddings are: “瑞士” (Switzerland), “卢梭” (Jean-Jacques Rousseau), “鞋匠” (cobbler), “发明家” (inventor) and “奥地利人” (Austrian). Note that none of these words are directly relevant to *bells*, *clocks* or *watches*. Hence, the sememes “时间” (time), “告诉” (tell), and “用具” (tool) cannot be inferred by those words, even though the correlations between sememes are introduced by SPSE. In fact, those words are related to *clocks* in an indirect way: Switzerland is famous for watch industry; Rousseau was born into a family that had a tradition of watchmaking; cobbler and inventor are two kinds of occupations as well. With the above reasons, those words usually co-occur with “钟表匠” (clockmaker), or usually appear in similar contexts as “钟表匠” (clockmaker). It indicates that related word embeddings as used in an *external* model do not always recommend related sememes.

The word “奥斯卡” (Oscar) is created by the pronunciation of *Oscar*. Therefore, the meaning of each character in “奥斯卡” (Oscar) is unrelated to the meaning of the word. Moreover, the characters “奥”, “斯”, and “卡” are common among transliterated words, thus the *internal* method recommends “专” (ProperName) and “地方” (place), etc., since many transliterated words are proper nouns or place names.

6 Conclusion and Future Work

In this paper, we introduced character-level internal information for lexical sememe prediction in Chinese, in order to alleviate the problems caused by the exclusive use of external information. We proposed a Character-enhanced Sememe Prediction (CSP) framework which integrates both internal and external information for lexical sememe prediction and proposed two methods for utilizing internal information. We evaluated our CSP framework on the classical manually annotated sememe KB HowNet. In our experiments, our methods achieved promising results and outperformed the state of the art on sememe prediction, especially for low-frequency words.

We will explore the following research directions in the future: (1) Concepts in HowNet are annotated with hierarchical structures of senses and sememes, but those are not considered in this paper. In the future, we will take structured annotations into account. (2) It would be meaningful to take more information into account for blending external and internal information and design more sophisticated methods. (3) Besides Chinese, many other languages have rich subword-level information. In the future, we will explore methods of exploiting internal information in other languages. (4) We believe that sememes are universal for all human languages. We will explore a general framework to recommend and utilize sememes for other NLP tasks.

Acknowledgments

This research is part of the NExT++ project, supported by the National Research Foundation, Prime Minister’s Office, Singapore under its IRC@Singapore Funding Initiative. This work is also supported by the National Natural Science Foundation of China (NSFC No. 61661146007 and 61572273) and the research fund of Tsinghua University-Tencent Joint Laboratory for Internet Innovation Technology. Hao Zhu is supported by Tsinghua University Initiative Scientific Research Program. We would like to thank Katharina Kann, Shen Jin, and the anonymous reviewers for their helpful comments.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of ISWC*, pages 722–735.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Leonard Bloomfield. 1926. A set of postulates for the science of language. *Language*, 2(3):153–164.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of IJCAI*, pages 1236–1242.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of SIGMORPHON*, pages 10–22.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016b. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*, pages 1651–1660.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages. In *Processings of HLT-NAACL*, pages 380–387.

Zhendong Dong and Qiang Dong. 2006. *HowNet and the computation of meaning*. World Scientific.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Word sense disambiguation through sememe labeling. In *Proceedings of IJCAI*, pages 1594–1599.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of HLT-NAACL*, pages 1606–1615.

Xianghua Fu, Liu Guo, Guo Yanyan, and Wang Zhiqiang. 2013. Multi-aspect sentiment analysis for Chinese online social reviews based on topic modeling and HowNet lexicon. *Knowledge-Based Systems*, 37:186–195.

Kok-Wee Gan, Chi-Yung Wang, and Brian Mak. 2002. Knowledge-based sense pruning using the HowNet: an alternative to word sense disambiguation. In *Proceedings of ISCSLP*.

Kok Wee Gan and Ping Wai Wong. 2000. Annotating information structures in Chinese texts using HowNet. In *Proceedings of The Second Chinese Language Processing Workshop*, pages 85–92.

Minlie Huang, Borui Ye, Yichen Wang, Haiqiang Chen, Junjun Cheng, and Xiaoyan Zhu. 2014. New word detection for sentiment analysis. In *Proceedings of ACL*, pages 531–541.

Huiming Jin and Katharina Kann. 2017. Exploring cross-lingual transfer of morphological knowledge in sequence-to-sequence models. In *Proceedings of SCLeM*, pages 70–75.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *Proceedings of ACL*, pages 1993–2003.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8).

Yiqun Liu, Fei Chen, Weize Kong, Huijia Yu, Min Zhang, Shaoping Ma, and Liyun Ru. 2012. Identifying web spam with the wisdom of the crowds. *ACM Transactions on the Web*, 6(1):2:1–2:30.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

George A Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *Proceedings of EMNLP*, pages 880–885.

Vivi Nastase and Stan Szpakowicz. 2001. Word sense disambiguation in Roget's thesaurus using WordNet. In *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Improved word representation learning with sememes. In *Proceedings of ACL*, pages 2049–2058.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW*, pages 285–295.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL*, 3:359–373.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of WWW*, pages 697–706.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced Chinese character embedding. In *Proceedings of ICONIP*, pages 279–286.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of EMNLP*, pages 1499–1509.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of EMNLP*, pages 1504–1515.

Yunqing Xia, Taotao Zhao, Jianmin Yao, and Peng Jin. 2011. Measuring Chinese-English cross-lingual word similarity with HowNet and parallel corpus. In *Proceedings of CICLing*, pages 221–233. Springer.

Ruobing Xie, Xingchi Yuan, Zhiyuan Liu, and Maosong Sun. 2017. Lexical sememe prediction via word embeddings and matrix factorization. In *Proceedings of IJCAI*, pages 4200–4206.

Binyong Yin. 1984. Quantitative research on Chinese morphemes. *Studies of the Chinese Language*, 5:338–347.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity Chinese word embedding. In *Proceedings of EMNLP*, pages 981–986.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of IJCAI*, pages 4258–4264.

SemAxis: A Lightweight Framework to Characterize Domain-Specific Word Semantics Beyond Sentiment

Jisun An[†] **Haewoon Kwak**[†] **Yong-Yeol Ahn**[§]
[†]Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar
[§]Indiana University, Bloomington, IN, USA
jisun.an@acm.org haewoon@acm.org yyahn@iu.edu

Abstract

Because word semantics can substantially change across communities and contexts, capturing domain-specific word semantics is an important challenge. Here, we propose SemAxis, a simple yet powerful framework to characterize word semantics using many semantic axes in word-vector spaces beyond sentiment. We demonstrate that SemAxis can capture nuanced semantic representations in multiple online communities. We also show that, when the sentiment axis is examined, SemAxis outperforms the state-of-the-art approaches in building domain-specific sentiment lexicons.

1 Introduction

In lexicon-based text analysis, a common, tacit assumption is that the meaning of each word does not change significantly across contexts. This approximation, however, falls short because context can strongly alter the meaning of words (Fischer, 1958; Eckert and McConnell-Ginet, 2013; Hovy, 2015; Hamilton et al., 2016b). For instance, the word *kill* may be used much more positively in the context of video games than it would be in a news story; the word *soft* may be used much more negatively in the context of sports than it is in the context of toy animals (Hamilton et al., 2016a). Thus, lexicon-based analysis exhibits a clear limitation when two groups with strongly dissimilar lexical contexts are compared.

Recent breakthroughs in vector-space representation, such as `word2vec` (Mikolov et al., 2013b), provide new opportunities to tackle this challenge of context-dependence, because in these approaches, the representation of each word is learned from its context. For instance, a recent study shows that a propagation method on the vector space embedding can infer context-dependent sentiment values of words (Hamilton et al., 2016a). Yet, it remains to be seen whether it is possible to generalize this idea to general word semantics other than sentiment.

In this work, we propose SemAxis, a lightweight framework to characterize domain-specific word semantics beyond sentiment. SemAxis characterizes word semantics with respect to many semantic perspectives in a domain-specific word-vector space. To systematically discover the manifold of word semantics, we induce 732 semantic axes based on the antonym pairs from ConceptNet (Speer et al., 2017). We would like to emphasize that, although some of the induced axes can be considered as an extended version of sentiment analysis, such as an axis of 'respectful' (positive) and 'disrespectful' (negative), some cannot be mapped to a positive and negative relationship, such as 'exogeneous' and 'endogeneous,' and 'loose' and 'monogamous.' Based on this rich set of semantic axes, SemAxis captures nuanced semantic representations across corpora. The key contributions of this paper are:

- We propose a general framework to characterize the manifold of domain-specific word semantics.
- We systematically identify semantic axes based on the antonym pairs in ConceptNet.
- We demonstrate that SemAxis can capture semantic differences between two corpora.
- We provide a systematic evaluation in comparison to the state-of-the-art, domain-specific sentiment lexicon construction methodologies.

Although the idea of defining a semantic axis and assessing the meaning of a word with a vector projection is not new, it has not been demonstrated that this simple method can effectively induce context-aware semantic lexicons. All of the inferred lexicons along with code for SEMAXIS and all methods evaluated are made available in the SEMAXIS package released with this paper[1].

2 Related Work

For decades, researchers have been developing computational techniques for text analysis, including: sentiment analysis (Pang and Lee, 2004), stance detection (Biber and Finegan, 1988), point of view (Wiebe, 1994), and opinion mining (Pang and Lee, 2004). The practice of creating and sharing large-scale annotated lexicons also accelerated the research (Stone et al., 1966; Bradley and Lang, 1999; Pennebaker et al., 2001; Dodds et al., 2011; Mohammad et al., 2016, 2017).

These approaches can be roughly grouped into two major categories: lexicon-based approach (Turney, 2002; Taboada et al., 2011) and classification-based approach (Pang et al., 2002; Kim, 2014; Socher et al., 2013). Although the recent advancement of neural networks increases the potential of the latter approach, the former has been widely used for its simplicity and transparency. ANEW (Bradley and Lang, 1999), LIWC (Pennebaker et al., 2001), SO-CAL (Taboada et al., 2011), SentiWordNet (Esuli and Sebastiani, 2006), and LabMT (Dodds et al., 2011) are well-known lexicons.

A clear limitation of the lexicon-based approach is that it overlooks the context-dependent semantic changes. Scholars reported that the meaning of a word can be altered by context, such as communities (Yang and Eisenstein, 2015; Hamilton et al., 2016a), diachronic changes (Hamilton et al., 2016b) or demographic (Rosenthal and McKeown, 2011; Eckert and McConnell-Ginet, 2013; Green, 2002; Hovy, 2015), geography (Trudgill, 1974), political and cultural attitudes (Fischer, 1958) and personal variations (Yang and Eisenstein, 2017). Recently, a few studies have shown the importance of taking such context into accounts. For example, Hovy *et al.* (2015) showed that, by including author demographics such as age and gender, the accuracy of sentiment analysis and topic classification can be improved. It was also shown that, without domain-specific lexicons, the performance of sentiment analysis can be significantly degraded (Hamilton et al., 2016a).

Building domain-specific sentiment lexicons through human input (crowdsourcing or experts) requires not only significant resources but also careful control of biases (Dodds et al., 2011; Mohammad and Turney, 2010). The challenge is exacerbated because 'context' is difficult to concretely operationalize and there can be numerous contexts of interest. For resource-scarce languages, such problems become even more critical (Hong et al., 2013). Automatically building lexicons from web-scale resources (Velikovich et al., 2010; Tang et al., 2014) may solve this problem but poses a severe risk of unintended biases (Loughran and McDonald, 2011).

Inducing domain-specific lexicons from the unlabeled corpora reduces the cost of dictionary building (Hatzivassiloglou and McKeown, 1997; Rothe et al., 2016). Although earlier research utilize syntactic (grammatical) structures (Hatzivassiloglou and McKeown, 1997; Widdows and Dorow, 2002), the approach of learning word-vector representations has gained a lot of momentum (Rothe et al., 2016; Hamilton et al., 2016a; Velikovich et al., 2010; Fast et al., 2016).

The most relevant work is SENTPROP (Hamilton et al., 2016a), which constructs domain-specific sentiment lexicons using graph propagation techniques (Velikovich et al., 2010; Rao and Ravichandran, 2009). In contrast to SENTPROP's sentiment-focused approach, we provide a framework to understand the semantics of words with respect to 732 semantic axes based on ConceptNet (Speer et al., 2017).

3 SEMAXIS Framework

Our framework, SEMAXIS, involves three steps: constructing a word embedding, defining a semantic axis, and mapping words onto the semantic axis. Although they seem straightforward, the complexities and challenges in each step can add up. In particular, we tackle the issues of treating small corpora and selecting pole words.

3.1 The Basics of SEMAXIS

3.1.1 Building word embeddings

The first step in our approach is to obtain word vectors from a given corpus. In principle, any standard method, such as Positive Pointwise Mutual Information (PPMI), Singular-Value Decomposition (SVD), or `word2vec`, can be used. Here, we use the `word2vec` model because `word2vec` is easier to train and is known to be more robust than

[1]`https://github.com/ghdi6758/SemAxis`

competing methods (Levy et al., 2015).

3.1.2 Defining a semantic axis and computing the semantic axis vector

A semantic axis is defined by the vector between two sets of 'pole' words that are antonymous to each other. For instance, a sentiment axis can be defined by a set of the most positive words on one end and the most negative words on the other end. Similarly, any antonymous word pair can be used to define a semantic axis (e.g., 'clean' vs. 'dirty' or 'respectful' vs. 'disrespectful').

Once two sets of pole words for the corresponding axis are chosen, we compute the average vector for each pole. Then, by subtracting one vector from the other, we obtain the semantic axis vector that encodes the antonymous relationship between two sets of words. More formally, let $S^{+}=\{v_1^{+}, v_2^{+}, ..., v_n^{+}\}$ and $S^{-}=\{v_1^{-}, v_2^{-}, ..., v_m^{-}\}$ be two sets of pole word vectors that have an antonym relationship. Then, the average vectors for each set are computed as $\mathbf{V}^{+}=\frac{1}{n}\sum_1^n v_i^{+}$ and $\mathbf{V}^{-}=\frac{1}{m}\sum_1^m v_j^{-}$. From the two average vectors, the semantic axis vector, $\mathbf{V}_{\text{axis}}$ (from S^{-} to S^{+}), can be defined as:

$$\mathbf{V}_{\text{axis}} = \mathbf{V}^{+} - \mathbf{V}^{-} \tag{1}$$

3.1.3 Projecting words onto a semantic axis

Once the semantic axis vector is obtained, we can compute the cosine similarity between the axis vector and a word vector. The resulting cosine similarity captures how closely the word is aligned to the semantic axis. Given a word vector v_w for a word w, the score of the word w along with the given semantic axis, $\mathbf{V}_{\text{axis}}$, is computed as:

$$\begin{aligned} score(w)_{\mathbf{V}_{\text{axis}}} &= \cos(v_w, \mathbf{V}_{\text{axis}}) \\ &= \frac{v_w \cdot \mathbf{V}_{\text{axis}}}{\parallel v_w \parallel \parallel \mathbf{V}_{\text{axis}} \parallel} \end{aligned} \tag{2}$$

A higher score means that the word w is more closely aligned to S^{+} than S^{-}. When $\mathbf{V}_{\text{axis}}$ is a sentiment axis, a higher score is corresponds to more positive sentiment.

3.2 SEMAXIS for Comparative Text Analysis

Although aforementioned steps are conceptually simple, there are two practical challenges: 1) dealing with small corpus and 2) finding good pole words for building a semantic axis.

3.2.1 Semantic relations encoded in word embeddings

Since semantic relations are particularly important in our method, we need to ensure that our word embedding maintains general semantic relations. This can be evaluated by analogy tasks. In particular, we use the Google analogy test dataset (Mikolov et al., 2013a), which contains 19,544 questions — 8,869 semantic and 10,675 syntactic questions — in 14 relation types, such as capital-world pairs and opposite relationships.

3.2.2 Dealing with small corpus

As in other machine learning tasks, the amount of data critically influences the performance of word embedding methods. However, the corpora of our interest are often too small to facilitate the learning of rich semantic relationships therein. To mitigate this issue, we propose to pre-train a word embedding using a background corpus and update with the target corpora. In doing so, we capture the semantic changes while maintaining general semantic relations offered by the large reference model.

The vector-space embedding drifts from the reference model as we train with the target corpus. If trained too much with the smaller target corpus, it will lose the 'good' initial embedding from the huge reference corpus. If trained too little, it will not be able to capture context-dependent semantic changes. Our goal is thus to minimize the loss in general semantic relations while maximizing the characteristic semantic relations in the new texts.

Consider a corpus of our interest C and a reference corpus R. The model M is pre-trained on R, and then we start training it on C. We use the superscript e to represent the e-th epoch of training. That is, M_C^e is the model after the e-th epoch trained on C. Then, we evaluate the model regarding two aspects: general semantic relations and context-dependent semantic relations. The former is measured by the overall accuracy of the analogy test (Mikolov et al., 2013a). The latter is measured by tracking the semantic changes of the top k words in the given corpus C. The semantic changes of the words are measured by the changes in their scores, Δ, on a certain axis; for instance, a sentiment axis, between consecutive epochs. We stop learning when two conditions are satisfied: (1) When the accuracy of the analogy test drops by α; and (2) When Δ is lower than β. In principle, the model can be updated with the target corpus as long as the accuracy does not drop. We then use β

to control the epochs. When Δ is low, the gain by updating the model becomes negligible compared to the cost and thus we can stop updating.

3.3 Identifying rich semantic axes for SEMAXIS

The primary advantage of SEMAXIS is that it can be easily extended to examine diverse perspectives of texts as it is lightweight. Although the axis can be defined by any pair of words in principle, we propose a systematic way to define the axes.

3.3.1 732 Pre-defined Semantic Axes

We begin with a pair of antonyms, called *initial pole words*. For instance, a sentiment axis, which is a basis of sentiment analyses, can be defined by a pair of sentiment antonyms, such as 'good' and 'bad.'

To build a comprehensive set of initial pole words, we compile a list of antonyms from ConceptNet 5.5, which is the latest release of a knowledge graph among concepts (Speer et al., 2017). We extract all the antonym concepts marked as '/r/Antonym' edges. Then, we filter out non-English concepts and multi-word concepts. In addition, we eliminate duplicated antonyms that involve synonyms. For instance, only one of the (empower, prohibit) and (empower, forbid) needs to be kept because 'prohibit' and 'forbid' are synonyms, marked as '/r/Synonym' in ConceptNet.

To further refine the antonym pairs, we create a crowdsourcing task on *Figure Eight*, formerly known as *CrowdFlower*. Specifically, we ask crowdworkers: *Do these two words have opposite meanings?* We include those word pairs that a majority of crowdworkers agree to have an opposite meaning. The word pairs that the majority of crowdsource workers disagree were mostly erroneous antonym pairs, such as '5' and '3', and 'have' and 'has.' We then filter out the antonyms that are highly similar to each other. For example, ('advisedly' and 'accidentally') and ('purposely' and 'accidentally') show the cosine similarity of 0.5148, while 'advisedly' and 'purposely' are not marked as synonyms in ConceptNet. Although we use the threshold of 0.4 in this work, a different threshold can be chosen depending on the purpose. Finally, we eliminate concepts that do not appear in the pre-trained Google News 100B word embeddings. As a result, we obtain 732 pairs of antonyms. Each pair of antonyms becomes initial pole words to define one of the diverse axes for SEMAXIS.

We assess the semantic diversity of the axes by computing cosine similarity between every possible pair of the axes. The absolute mean value of the cosine similarity is 0.062, and the standard deviation is 0.050. These low cosine similarity and standard deviation values indicate that the chosen axes have a variety of directions, covering diverse and distinct semantics.

3.3.2 Augmenting pole words

We then expand the two initial pole words to larger sets of pole words, called *expanded pole words*, to obtain more robust results. If we use only two initial pole words to define the corresponding axis, the result will be sensitive to the choice of those words. Since the initial pole words are not necessarily the best combinations possible, we would like to augment it so that it is more robust to the choice of the initial pole words.

To address this issue, we find the l closest words of each initial pole word in the word embedding. We then compute the geometric center (average vector) of l+1 words (including the initial pole word) and regard it as the vector representation of that pole of the axis. For instance, refining an axis representing a 'good' and 'bad' relation, we first find the l closest words for each of 'good' and 'bad' and then compute the geometric center of them. The newly computed geometric centers then become both ends of the axis representing a 'good' and 'bad' relation. We demonstrate how this approach improves the explanatory power of an axis describing a corresponding antonym relation in Section 4.3.

4 SEMAXIS Validation

In this section, we quantitatively evaluate our approach using the ground-truth data and by comparing our method against the standard baselines and state-of-the-art approaches. We reproduce the evaluation task introduced by Hamilton *et al.* (2016a), recreating Standard English and Twitter sentiment lexicons for evaluation. We then compare the accuracy of sentiment classification with three other methods that generate domain-specific sentiment lexicons.

It is worth noting that we validate SEMAXIS based on a sentiment axis mainly due to the availability of the well-established ground-truth data and evaluation process. Nevertheless, as the sentiment axis in the SEMAXIS framework is not specifically or manually designed but established based on the corresponding pole words, the validation based on the sentiment axis can be generalized to other axes that are similarly established based on other corresponding pole words.

Standard English: We use well-known General Inquirer lexicon (Stone et al., 1966) and continuous valence scores collected by Warriner *et al.* (2013) to evaluate the performance of SemAxis compared to other state-of-the-art methods. We test all of the methods by using the off-the-shelf Google news embedding constructed from 10^{11} tokens (Google, 2013).

Twitter: We evaluate our approach with the test dataset from the 2015 SemEval task 10E competition (Rosenthal et al., 2015) using the embedding constructed by Rothe *et al.* (2016).

Domain	Positive pole words	Negative pole words
Standard	good, lovely, excellent, fortunate, pleasant, delightful, perfect, loved, love, happy	bad, horrible, poor, unfortunate, unpleasant, disgusting, evil, hated, hate, unhappy
Twitter	love, loved, loves, awesome, nice, amazing, best, fantastic, correct, happy	hate, hated, hates, terrible, nasty, awful, worst, horrible, wrong, sad

Table 1: Manually selected pole words used for the evaluation task in (Hamilton et al., 2016a). These pole words are called seed words in (Hamilton et al., 2016a)

4.1 Evaluation Setup

We compare our method against state-of-the-art approaches that generate domain-specific sentiment lexicons.

State-of-the-art approaches: Our baseline for the standard English is a WordNet-based method, which performs label propagation over a WordNet-derived graph (San Vicente et al., 2014). For Twitter, we use Sentiment140, a distantly supervised approach that uses signals from emoticons (Mohammad and Turney, 2010). Moreover, on both datasets, we compare against two state-of-the-art sentiment induction methods: Densifier, a method that learns orthogonal transformations of word vectors (Rothe et al., 2016), and SentProp, a method with a label propagation approach on word embeddings (Hamilton et al., 2016a). Seed words, which are called pole words in our work, are listed in Table 1.

Evaluation metrics: We evaluate the aforementioned approaches according to (i) their binary classification accuracy (positive and negative), (ii) ternary classification performance (positive, neutral, and negative), and (iii) Kendall τ rank-correlation with continuous human-annotated polarity scores. Since all methods result in sentiment scores of words rather than assigning a class of sentiment, we label words as positive, neutral, or negative using the class-mass normalization method (Zhu et al., 2003). This normalization uses knowledge of the label distribution of a test dataset and simply assigns labels to best match this distribution. For the implementation of other methods, we directly use the source code without any modification or tuning (SocialSent, 2016) used in (Hamilton et al., 2016a).

4.2 Evaluation Results

Table 2 summarizes the performance. Surprisingly, SemAxis — the simplest approach — outperforms others on both Standard English and Twitter datasets across all measures.

Standard English			
Method	AUC	Ternary F1	Tau
SemAxis	**92.2**	**61.0**	**0.48**
Densifier	91.0	58.2	0.46
SentProp	88.4	56.1	0.41
WordNet	89.5	58.7	0.34
Twitter			
Method	AUC	Ternary F1	Tau
SemAxis	**90.0**	**59.2**	**0.57**
Densifier	88.5	58.8	0.55
SentProp	85.0	58.2	0.50
Sentiment140	86.2	57.7	0.51

Table 2: Evaluation results. Our method performs best on both Standard English and Twitter.

4.3 Sensitivity to Pole Words

As discussed in Section 3.3.2, because the axes are derived from pole words, the choice of the pole words can significantly affect the performance. We compare the robustness of three methods for selecting pole words: 1) using sentiment lexicons; 2) using two pole words only (initial pole words); and 3) using l closest words on the `word2vec` model as well as the two initial pole words (expanded pole words). For the first, we choose two sets of pole words that have the highest scores and the lowest scores in two widely used sentiment lexicons, ANEW (Bradley and Lang, 1999) and LabMT (Dodds et al., 2011). Then, for the two pole words, we match 1-of-10 positive pole words and 1-of-10 negative pole words in Table 1, resulting in 100 pairs of pole words. For these 100 pairs, in addition to the two initial pole words, we then use the l closest words ($l = 10$) of each of them to evaluate the third method.

We compare these three methods by quantifying how well SEMAXIS performs for the evaluation task. The average AUC for the two pole words method is 78.2. We find that one of the 100 pairs — 'good' and 'bad' — shows the highest AUC (92.4). However, another random pair ('happy' and 'evil') results in the worst performance with the AUC of 67.2. In other words, an axis defined by only two pole words is highly sensitive to the choice of the word pair. By contrast, when an axis is defined by aggregating l closest words, the average AUC increases to 80.6 (the minimum performance is above 71.2). Finally, using pre-established sentiment lexicons results in the worst performance (the AUC of 77.8 for ANEW and 67.5 for LabMT). These results show that identifying an axis is a crucial step in SEMAXIS, and using l closest words in addition to initial pole words is a more robust method to define the axis.

5 SEMAXIS in the Wild

We now demonstrate how SEMAXIS can be used in comparative text analysis to capture nuanced linguistic representations beyond the sentiment. As an example, we use Reddit (Reddit, 2005), one of the most popular online communities. Reddit is known to serve diverse sub-communities with different linguistic styles (Zhang et al., 2017). We focus on a few pairs of subreddits that are known to express different views. We also choose them to capture a wide range of topics from politics to religion, entertainment, and daily life to demonstrate the broad applicability of SEMAXIS.

5.1 Dataset, Pre-processing, Reference model, and Hyper-parameters

We use Reddit 2016 comment datasets that are publicly available (/u/Dewarim, 2017). We build a corpus from each subreddit by extracting all the comments posted on that subreddit. When the size of two corpora used for comparison is considerably different, we undersample the bigger corpus for a fair comparison. Every corpus then undergoes the same pre-processing, where we first remove punctuation and stop words, then replace URLs with its domain name.

Reference model for Reddit data As we discussed earlier, many datasets of our interest are likely too small to obtain good vector representations. For example, two popular subreddits, `/r/The_Donald` and `/r/SandersForPresident`[2], show only 59.8% and 42.1% in analogy test, respectively.[3] Therefore, as we proposed, we first create a pre-trained word embedding with a larger background corpus and perform additional training with target subreddits. We sample 1 million comments from each of the top 200 subreddits, resulting in 20 million comments. Using this sample, we build a word embedding, denoted as Reddit20M, using the CBOW model with a window size of five, a minimum word count of 10, the negative sampling, and down-sampling of frequent terms as suggested in (Levy et al., 2015). For the subsequent training with the target corpora, we train the model with a small starting learning rate, 0.005; Using different rates, such as 0.01 and 0.001, did not make much difference. We further tune the model with the dimension size of 300 and the number of the epoch of 100 using the analogy test results.

Category	Reddit20M	Google300D
World	28.34	70.2
family	94.58	90.06
Gram1-9	70.21	73.40
Total	67.88	77.08

Table 3: Results of analogy tests, comparing 20M sample texts from Reddit vs. Google 100B News.

Table 3 shows the results of the analogy test using our Reddit20M in comparison with Google300D, which is the Google News embedding used in previous sections. As one can expect, Reddit20M shows worse performance than Google300D. However, the four categories (capital-common-countries, capital-world, currency, and city-in-state denoted by **World**), which require some general knowledge on the world, drive the 10% decrease in overall accu-

[2] `/r/` is a common notation for indicating subreddits.

[3] For both corpora, continuous bag-of-words (CBOW) model with the dimension size of 300 achieves the highest accuracy in the analogy test.

racy. Other categories show comparable or even better accuracy. For example, Reddit20M outperforms Google300D by 4.52% in the family category. Since Reddit is a US-based online community, the Reddit model may not be able to properly capture semantic relationships in **World** category. By contrast, for the categories for testing grammar (denoted by **Gram1-9**), Reddit20M shows comparable performances with Google300D (70.21 vs. 73.4). In this study, we use Reddit20M as a reference model and update it with new target corpora.

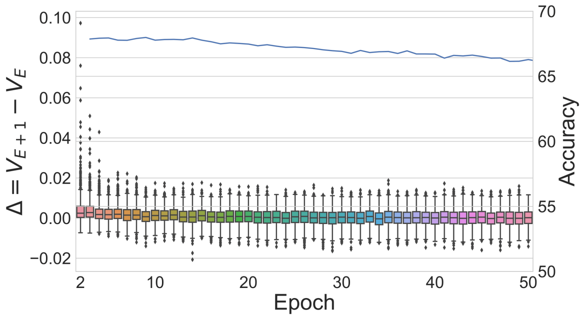


Figure 1: Changes of word semantics (box plot) and accuracy (line graph) over epoch for the model for `/r/SandersForPresident`

Updating the reference model As we explained in Section 3.2.2, we stop updating the reference model depending on the accuracy of the analogy test and semantic changes of the top 1000 words of the given corpus. In our experiments, we set $\alpha = 0.3$ and $\beta = 0.001$. Figure 1 shows the accuracy of the analogy test over epoch as a line plot and the semantic changes of words as a box plot for the model for `/r/SandersForPresident`. The model gradually loses general semantic relation over epochs, and the characteristic semantic changes stabilize after about 10 epochs. Given the α and β, we use the embedding after 10 epochs of training with the target subreddit data. We note that the results are consistent when epoch is greater than 10. We choose the number of epoch for other corpora based on the same tactic.

5.2 Confirming well-known language representations

Once we have word embeddings for given subreddits by updating the pre-trained model, we can compare the languages of two subreddits. As a case study, we compare supporters of Donald Trump (`/r/The_Donald`) and Bernie Sanders (`/r/SandersForPresident`)[4], and examine the semantic differences in diverse issues, such as gun and minority, based on different axes. This can be easily compared with our educated guess learned from the 2016 U.S. Election.

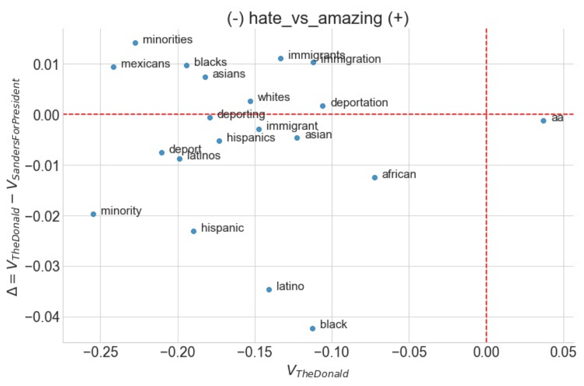
Starting from a topic word (e.g., 'gun') and its closest word (e.g., 'guns'), we compute the average vector of the two words. We then find the closest word from the computed average vector and repeat this process to collect 30 topical terms in each word embedding. Then, we remove words that have appeared less than n times in both corpora. The higher n leads to less coverage of topical terms but eliminate noise. We set $n = 100$ in the following experiments. We consider the remaining words as topic words.

Figure 2 compares how the minority-related terms are depicted in the two subreddits. Figure 2(a) and 2(b) show how minority issues are perceived in two communities with respect to 'Sentiment' and 'Respect' . The x-axis is the value for each word on the Sentiment axis for Trump supporters, and the y-axis is the difference between the value for Trump and Sanders supporters. If the y-value is greater than 0, then it means the word is more 'positive' among Trump supporters compared to that among Sanders supporters.

Some terms perceived more positively (e.g., 'immigration' and 'minorities') while other terms were perceived more negatively ('black', 'latino', 'hispanic') among Trump supporters (Figure 2(a)). As this positive perception on immigration and minorities is unexpected, we examine the actual comments. Through the manual inspection of relevant comments, we find that Trump supporters often mention that they 'agree' with or 'support' the idea of banning immigration, resulting in having a term 'immigration' as more positive than Sanders supporters. However, when examining those words on the 'Disrespect' vs. 'Respect' axis (Figure 2(b)), most of the minority groups are considered disrespectful by Trump supporters compared to Sanders supporters, demonstrating a benefit of examining multiple semantic axes that can reflect rich semantics beyond basic sentiments.

Then, one would expect that 'Gun' would be more positively perceived for Trump supporters compared to for Sanders supporters. Beyond the sentiment, we examine how 'gun' is perceived in

[4]Both subreddits have a policy of banning users who post content critical of the candidate. Thus, we assume most of the users in these subreddits are supporters of the candidate.


(a) Minority (Hate vs Amazing)

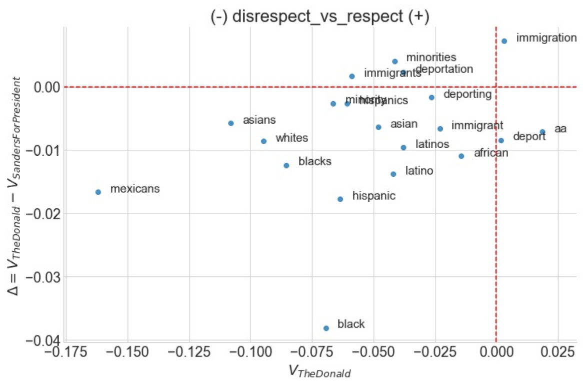

(b) Minority (Disrespect vs Respect)

Figure 2: Trump supporters vs Sanders supporters on Minority issue

two communities for 'Arousal' and 'Safety' axes. We find that Trump supporters are generally positive about gun-related issues, and Sanders supporters associate 'gun' with more arousal and danger.

We also examine two other subreddits: /r/atheism and /r/Christianity. As their names indicate, the former is the subreddit for atheists and the latter is the subreddit for Christians. We expect that the two groups would have different perspectives regarding 'god' and 'evolution.' When examining the four words 'god,' 'pray,' 'evolution,' and 'science' on the 'Unholy' vs. 'Holy' axis, 'god' and 'pray' appear to be more 'holy' in /r/Christianity while 'evolution' and 'science' appear more 'unholy' than in /r/Atheism, which fits in our intuition.

As another example, we examine the /r/PS4 and /r/NintendoSwitch subreddits. PS4 is a video game console released by Sony and Nintendo Switch is released by Nintendo. Although both video game consoles originated from Japan, Nintendo Switch targets more family (children) and casual gamers with more playful and easier games while the games for PS4 target adult and thus tend to be more violent and more difficult to play. We examine three terms ('Nintendo,' 'Mario,' and 'Zelda') from Nintendo Switch and three terms ('Sony,' 'Uncharted,' and 'Killzone') from Sony on the 'Casual' vs. 'Hardcore' axis.[5] We find that 'Mario' and 'Zelda' are perceived more casual in /r/PS4, and 'Uncharted' and 'Killzone' are more hardcore in /r/PS4 than /r/NintendoSwitch. Although both 'Nintendo' and 'Sony' have negative values, 'Nintendo' was considered more casual than 'Sony' in /r/PS4. Overall, our method effectively captures context-dependent semantic changes beyond the basic sentiments.

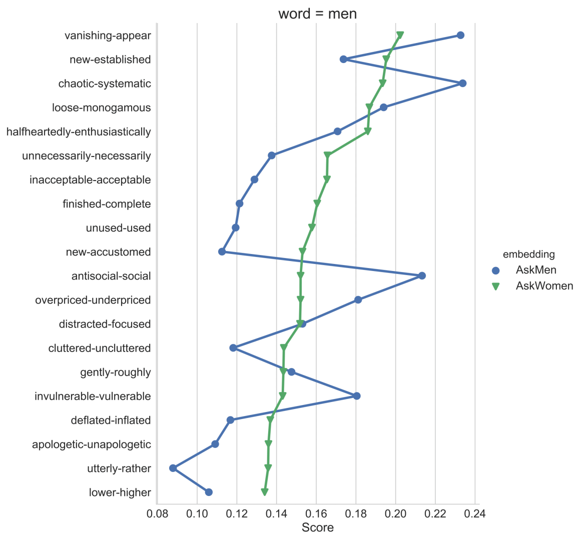
[5]Mario and Zelda are popular Nintendo Switch games, and Uncharted and Killzone are popular PS4 games.

5.3 Comparative Text Analysis with Diverse Axes

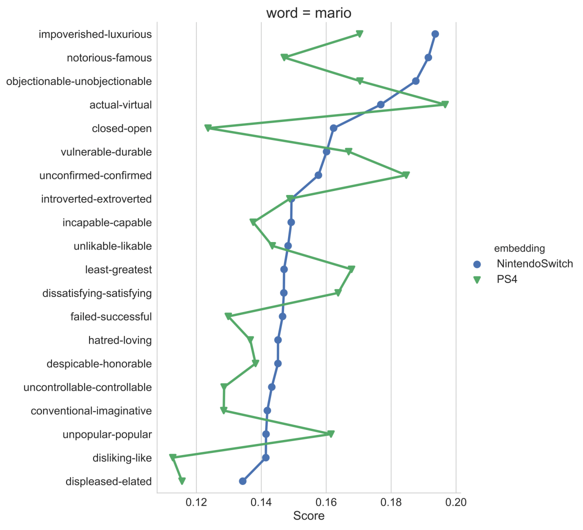
Let us show how SEMAXIS can find, for a given word, a set of the best axes that describe its semantic. We map the word on our predefined 732 axes, which are explained in Section 3.3.1, and rank the axes based on the projection values on the axes. In other words, the top axes describe the word with the highest strength.

Figure 3(a) shows the top 20 axes with the largest projection values for 'Men' in /r/AskWomen and /r/AskMen, which are the subreddits where people expect replies from each gender. In /r/AskWomen, compared with /r/AskMen, 'Men' seems to be perceived as more vanishing, more established, less systematic, less monogamous, more enthusiastic, less social, more uncluttered, less vulnerable, and more unapologetic. This observed perception of men from women's perspective seems to concur with the common gender stereotype, demonstrating strong potential of SEMAXIS.

Likewise, in Figure 3(b), we examine how a word 'Mario' is perceived in two subreddits /r/NintendoSwitch and /r/PS4. In /r/NintendoSwitch, 'Mario' is perceived, compared with /r/PS4, as more luxurious, famous, unobjectionable, open, capable, likable, successful, loving, honorable, and controllable. On the other hand, users in /r/PS4 consider 'Mario' to be more virtual, creative, durable,



(a) Men in /r/AskWomen and /r/AskMen



(b) Mario in /r/NintendoSwitch and /r/PS4

Figure 3: An example of comparative text analysis using SEMAXIS: (a) 'Men' in `/r/AskWomen` and `/r/AskMen` and (b) 'Mario' in `/r/NintendoSwitch` and `/r/PS4`

satisfying, popular, undetectable, and unstoppable. 'Mario' is perceived more positively in `/r/NintendoSwitch` than in `/r/PS4`, as expected. Furthermore, SEMAXIS reveals detailed and nuanced perceptions of different communities.

6 Discussion and Conclusion

We have proposed SEMAXIS to examine a nuanced representation of words based on diverse semantic axes. We have shown that SEMAXIS can construct good domain-specific sentiment lexicons by projecting words on the sentiment axis. We have also demonstrated that our approach can reveal nuanced context-dependence of words through the lens of numerous semantic axes.

There are two major limitations. First, we performed the quantitative evaluation only with the sentiment axis, even though we supplemented it with more qualitative examples. We used the sentiment axis because it is better studied and more methods exist, but ideally it would be better to perform evaluation across many semantic axes. We hope that SEMAXIS can facilitate research on other semantic axes so that we will have labeled datasets for other axes as well. Secondly, Gaffney and Matias (2018) recently reported the Reddit data used in this study is incomplete. The authors suggest using the data with caution, particularly when analyzing user interactions. Although our work examine communities in Reddit, we focus on the difference of the word semantics. Thus, we believe the effect of deleted comment would be marginal in our analyses.

Despite these limitations, we identify the following key implications. First, SEMAXIS offers a framework to examine texts on diverse *semantic axes* beyond the sentiment axis, through the 732 systematically induced semantic axes that capture common antonyms. Our study may facilitate further investigations on context-dependent text analysis techniques and applications. Second, the unsupervised nature of SEMAXIS provides a powerful way to build lexicons of any semantic axis, including the sentiment axis, for non-English languages, particularly the resource-scarce languages.

7 Acknowledgements

The authors thank Jaehyuk Park for his helpful comments. This research has been supported in part by Volkswagen Foundation and in part by the Defense Advanced Research Projects Agency (DARPA), W911NF-17-C-0094. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

Douglas Biber and Edward Finegan. 1988. Adverbial stance types in english. *Discourse processes*, 11(1):1–34.

Margaret M Bradley and Peter J Lang. 1999. Affective norms for english words (ANEW): Instruction manual and affective ratings. Technical report, Technical report C-1, the center for research in psychophysiology, University of Florida.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M Kloumann, Catherine A Bliss, and Christopher M Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *PLOS ONE*, 6(12):e26752.

Penelope Eckert and Sally McConnell-Ginet. 2013. *Language and gender*. Cambridge University Press.

A. Esuli and F. Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA).

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM.

John L. Fischer. 1958. Social influences on the choice of a linguistic variant. *WORD*, 14(1):47–56.

Devin Gaffney and J Nathan Matias. 2018. Caveat emptor, computational social science: Large-scale missing data in a widely-published reddit corpus. *arXiv preprint arXiv:1803.05046*.

Google. 2013. Google word2vec. `https://code.google.com/archive/p/word2vec/`. [Online; accessed February 23, 2018].

Lisa J Green. 2002. *African American English: a linguistic introduction*. Cambridge University Press.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016a. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605. Association for Computational Linguistics.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501. Association for Computational Linguistics.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *8th Conference of the European Chapter of the Association for Computational Linguistics*.

Yoonsung Hong, Haewoon Kwak, Youngmin Baek, and Sue Moon. 2013. Tower of babel: A crowdsourcing game building sentiment lexicons for resource-scarce languages. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, pages 549–556, New York, NY, USA. ACM.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 752–762. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41. Association for Computational Linguistics.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26:1–26:23.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 675–682. Association for Computational Linguistics.

Reddit. 2005. Reddit: the front page of the internet. `https://www.reddit.com/`. [Online; accessed February 23, 2018].

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre- and post-social media generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 763–772. Association for Computational Linguistics.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 451–463. Association for Computational Linguistics.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777. Association for Computational Linguistics.

Iñaki San Vicente, Rodrigo Agerri, and German Rigau. 2014. Simple, robust and (almost) unsupervised generation of polarity lexicons for multiple languages. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 88–97. Association for Computational Linguistics.

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

SocialSent. 2016. Code and data for inducing domain-specific sentiment lexicons. `https://github.com/williamleif/socialsent`. [Online; accessed February 23, 2018].

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451.

Philip J Stone, Dexter C Dunphy, and Marshall S Smith. 1966. *The general inquirer: A computer approach to content analysis.* MIT press.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2).

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 172–182. Dublin City University and Association for Computational Linguistics.

Peter Trudgill. 1974. Linguistic change and diffusion: Description and explanation in sociolinguistic dialect geography. *Language in society*, 3(2):215–246.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

/u/Dewarim. 2017. Reddit comment dataset. `https://files.pushshift.io/`. [Online; downloaded July 23, 2017].

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785. Association for Computational Linguistics.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Janyce M. Wiebe. 1994. Tracking point of view in narrative. *Computational Linguistics*, 20(2).

Yi Yang and Jacob Eisenstein. 2015. Putting things in context: Community-specific embedding projections for sentiment analysis. *CoRR*, abs/1511.06052.

Yi Yang and Jacob Eisenstein. 2017. Overcoming language variation in sentiment analysis with social attention. *Transactions of the Association for Computational Linguistics*, 5:295–307.

Justine Zhang, William L Hamilton, Cristian Danescu-Niculescu-Mizil, Dan Jurafsky, and Jure Leskovec. 2017. Community identity and user engagement in a multi-community landscape. In *Proceedings of The International Conference on Web and Social Media*.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919.

End-to-End Reinforcement Learning for Automatic Taxonomy Induction

Yuning Mao[1], **Xiang Ren**[2], **Jiaming Shen**[1], **Xiaotao Gu**[1], **Jiawei Han**[1]
[1]Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA
[2]Department of Computer Science, University of Southern California, CA, USA
[1]{yuningm2, js2, xiaotao2, hanj}@illinois.edu [2]xiangren@usc.edu

Abstract

We present a novel end-to-end reinforcement learning approach to automatic taxonomy induction from a set of terms. While prior methods treat the problem as a two-phase task (*i.e.*, detecting hypernymy pairs followed by organizing these pairs into a tree-structured hierarchy), we argue that such two-phase methods may suffer from error propagation, and cannot effectively optimize metrics that capture the holistic structure of a taxonomy. In our approach, the representations of term pairs are learned using multiple sources of information and used to determine *which* term to select and *where* to place it on the taxonomy via a policy network. All components are trained in an end-to-end manner with cumulative rewards, measured by a holistic tree metric over the training taxonomies. Experiments on two public datasets of different domains show that our approach outperforms prior state-of-the-art taxonomy induction methods up to 19.6% on ancestor F1. [1]

1 Introduction

Many tasks in natural language understanding (*e.g.*, information extraction (Demeester et al., 2016), question answering (Yang et al., 2017), and textual entailment (Sammons, 2012)) rely on lexical resources in the form of term taxonomies (cf. rightmost column in Fig. 1). However, most existing taxonomies, such as WordNet (Miller, 1995) and Cyc (Lenat, 1995), are manually curated and thus may have limited coverage or become unavailable in some domains and languages. Therefore, recent efforts have been focusing on *automatic taxonomy induction*, which aims to organize a set of terms into a taxonomy based on relevant resources such as text corpora.

[1]Code and data can be found at `https://github.com/morningmoni/TaxoRL`

Prior studies on automatic taxonomy induction (Gupta et al., 2017; Camacho-Collados, 2017) often divide the problem into two sequential subtasks: (1) *hypernymy detection* (*i.e.*, extracting term pairs of "is-a" relation); and (2) *hypernymy organization* (*i.e.*, organizing is-a term pairs into a tree-structured hierarchy). Methods developed for hypernymy detection either harvest new terms (Yamada et al., 2009; Kozareva and Hovy, 2010) or presume a vocabulary is given and study term semantics (Snow et al., 2005; Fu et al., 2014; Tuan et al., 2016; Shwartz et al., 2016). The hypernymy pairs extracted in the first subtask form a noisy hypernym graph, which is then transformed into a tree-structured taxonomy in the hypernymy organization subtask, using different graph pruning methods including maximum spanning tree (MST) (Bansal et al., 2014; Zhang et al., 2016), minimum-cost flow (MCF) (Gupta et al., 2017) and other pruning heuristics (Kozareva and Hovy, 2010; Velardi et al., 2013; Faralli et al., 2015; Panchenko et al., 2016).

However, these two-phase methods encounter two major limitations. First, most of them ignore the taxonomy structure when estimating the probability that a term pair holds the hypernymy relation. They estimate the probability of different term pairs independently and the learned term pair representations are fixed during hypernymy organization. In consequence, there is no feedback from the second phase to the first phase and possibly wrong representations cannot be rectified based on the results of hypernymy organization, which causes the error propagation problem. Secondly, some methods (Bansal et al., 2014; Zhang et al., 2016) do explore the taxonomy space by regarding the induction of taxonomy structure as inferring the conditional distribution of edges. In other words, they use the product of edge proba-

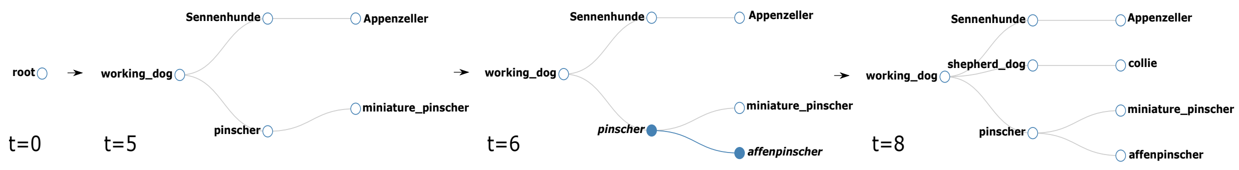


Figure 1: An illustrative example showing the process of taxonomy induction. The input vocabulary V_0 is {"*working dog*", "*pinscher*", "*shepherd dog*", ...}, and the initial taxonomy T_0 is empty. We use a virtual "*root*" node to represent T_0 at $t = 0$. At time $t = 5$, there are 5 terms on the taxonomy T_5 and 3 terms left to be attached: V_t = {"*shepherd dog*", "*collie*", "*affenpinscher*"}. Suppose the term "*affenpinscher*" is selected and put under "*pinscher*", then the remaining vocabulary V_{t+1} at next time step becomes {"*shepherd dog*", "*collie*"}. Finally, after $|V_0|$ time steps, all the terms are attached to the taxonomy and $V_{|V_0|} = V_8 = \{\}$. A full taxonomy is then constructed from scratch.

bilities to represent the taxonomy quality. However, the edges are treated equally, while in reality, they contribute to the taxonomy differently. For example, a high-level edge is likely to be more important than a bottom-out edge because it has much more influence on its descendants. In addition, these methods cannot *explicitly* capture the holistic taxonomy structure by optimizing global metrics.

To address the above issues, we propose to jointly conduct hypernymy detection and organization by learning term pair representations and constructing the taxonomy simultaneously. Since it is infeasible to estimate the quality of all possible taxonomies, we design an end-to-end reinforcement learning (RL) model to combine the two phases. Specifically, we train an RL agent that employs the term pair representations using multiple sources of information and determines *which* term to select and *where* to place it on the taxonomy via a policy network. The feedback from hypernymy organization is propagated back to the hypernymy detection phase, based on which the term pair representations are adjusted. All components are trained in an end-to-end manner with cumulative rewards, measured by a holistic tree metric over the training taxonomies. The probability of a full taxonomy is no longer a simple aggregated probability of its edges. Instead, we assess an edge based on how much it can contribute to the whole quality of the taxonomy.

We perform two sets of experiments to evaluate the effectiveness of our proposed approach. First, we test the end-to-end taxonomy induction performance by comparing our approach with the state-of-the-art two-phase methods, and show that our approach outperforms them significantly on the quality of constructed taxonomies. Second, we use the same (noisy) hypernym graph as the input of all compared methods, and demonstrate that our RL approach does better hypernymy organization through optimizing metrics that can capture holistic taxonomy structure.

Contributions. In summary, we have made the following contributions: (1) We propose a deep reinforcement learning approach to unify hypernymy detection and organization so as to induct taxonomies in an end-to-end manner. (2) We design a policy network to incorporate semantic information of term pairs and use cumulative rewards to measure the quality of constructed taxonomies holistically. (3) Experiments on two public datasets from different domains demonstrate the superior performance of our approach compared with state-of-the-art methods. We also show that our method can effectively reduce error propagation and capture global taxonomy structure.

2 Automatic Taxonomy Induction

2.1 Problem Definition

We define a taxonomy $T = (V, R)$ as a tree-structured hierarchy with term set V (*i.e.*, *vocabulary*), and edge set R (which indicates is-a relationship between terms). A term $v \in V$ can be either a unigram or a multi-word phrase. The task of *end-to-end taxonomy induction* takes a set of training taxonomies and related resources (*e.g.*, background text corpora) as input, and aims to learn a *model* to construct a full taxonomy T by adding terms from a given vocabulary V_0 onto an empty hierarchy T_0 *one at a time*. An illustration of the taxonomy induction process is shown in Fig. 1.

2.2 Modeling Hypernymy Relation

Determining which term to select from V_0 and where to place it on the current hierarchy requires understanding of the semantic relationships between the selected term and all the other terms. We consider multiple sources of information (*i.e.*, resources) for learning hypernymy relation representations of term pairs, including dependency path-based contextual embedding and distributional term embeddings (Shwartz et al., 2016).

Path-based Information. We extract the shortest dependency paths between each co-occurring term pair from sentences in the given background corpora. Each path is represented as a sequence of edges that goes from term x to term y in the dependency tree, and each edge consists of the word lemma, the part-of-speech tag, the dependency label and the edge direction between two contiguous words. The edge is represented by the concatenation of embeddings of its four components:

$$\mathbf{V}_e = [\mathbf{V}_l, , \mathbf{V}_{\text{pos}}, \mathbf{V}_{\text{dep}}, \mathbf{V}_{\text{dir}}].$$

Instead of treating the entire dependency path as a single feature, we encode the sequence of dependency edges $\mathbf{V}_{e_1}, \mathbf{V}_{e_2}, ..., \mathbf{V}_{e_k}$ using an LSTM so that the model can focus on learning from parts of the path that are more informative while ignoring others. We denote the final output of the LSTM for path p as $\mathbf{O}_p$, and use $\mathcal{P}(x, y)$ to represent the set of all dependency paths between term pair (x, y). A single vector representation of the term pair (x, y) is then computed as $\mathbf{P}_{\mathcal{P}(x,y)}$, the weighted average of all its path representations by applying an average pooling:

$$\mathbf{P}_{\mathcal{P}(x,y)} = \frac{\sum_{p \in \mathcal{P}(x,y)} c_{(x,y)}(p) \cdot \mathbf{O}_p}{\sum_{p \in \mathcal{P}(x,y)} c_{(x,y)}(p)},$$

where $c_{(x,y)}(p)$ denotes the frequency of path p in $\mathcal{P}(x, y)$. For those term pairs without dependency paths, we use a randomly initialized *empty path* to represent them as in Shwartz et al. (2016).

Distributional Term Embedding. The previous path-based features are only applicable when two terms co-occur in a sentence. In our experiments, however, we found that only about 17% of term pairs have sentence-level co-occurrences.[2] To alleviate the sparse co-occurrence issue, we concatenate the path representation $\mathbf{P}_{\mathcal{P}(x,y)}$ with the word embeddings of x and y, which capture the distributional semantics of two terms.

[2]In comparison, more than 70% of term pairs have sentence-level co-occurrences in BLESS (Baroni and Lenci, 2011), a standard hypernymy detection dataset.

Surface String Features. In practice, even the embeddings of many terms are missing because the terms in the input vocabulary may be multi-word phrases, proper nouns or named entities, which are likely not covered by the external pre-trained word embeddings. To address this issue, we utilize several surface features described in previous studies (Yang and Callan, 2009; Bansal et al., 2014; Zhang et al., 2016). Specifically, we employ *Capitalization*, *Ends with*, *Contains*, *Suffix match*, *Longest common substring* and *Length difference*. These features are effective for detecting hypernyms solely based on the term pairs.

Frequency and Generality Features. Another feature source that we employ is the hypernym candidates from TAXI[3] (Panchenko et al., 2016). These hypernym candidates are extracted by lexico-syntactic patterns and may be noisy. As only term pairs and the co-occurrence frequencies of them (under specific patterns) are available, we cannot recover the dependency paths between these terms. Thus, we design two features that are similar to those used in (Panchenko et al., 2016; Gupta et al., 2017). [4]

- *Normalized Frequency Diff.* For a hyponym-hypernym pair (x_i, x_j) where x_i is the hyponym and x_j is the hypernym, its normalized frequency is defined as $\text{freq}_n(x_i, x_j) = \frac{\text{freq}(x_i,x_j)}{\max_k \text{freq}(x_i,x_k)}$, where $\text{freq}(x_i, x_j)$ denotes the raw frequency of (x_i, x_j). The final feature score is defined as $\text{freq}_n(x_i, x_j) - \text{freq}_n(x_j, x_i)$, which down-ranks synonyms and co-hyponyms. Intuitively, a higher score indicates a higher probability that the term pair holds the hypernymy relation.

- *Generality Diff.* The generality $g(x)$ of a term x is defined as the logarithm of the number of its distinct hyponyms, *i.e.*, $g(x) = log(1+|\mathbf{hypo}|)$, where for any $hypo \in \mathbf{hypo}$, $(hypo, x)$ is a hypernym candidate. A high $g(x)$ of the term x implies that x is general since it has many distinct hyponyms. The generality of a term pair is defined as the difference in generality between x_j and x_i: $g(x_j) - g(x_i)$. This feature would

[3]http://tudarmstadt-lt.github.io/taxi/

[4]Since the features use additional resource, we wouldn't include them unless otherwise specified.

promote term pairs with the right level of generality and penalize term pairs that are either too general or too specific.

The surface, frequency, and generality features are binned and their embeddings are concatenated as a part of the term pair representation. In summary, the final term pair representation $\mathbf{R}_{xy}$ has the following form:

$$\mathbf{R}_{xy} = [\mathbf{P}_{\mathcal{P}(x,y)}, \mathbf{V}_{w_x}, \mathbf{V}_{w_y}, \mathbf{V}_{F(x,y)}],$$

where $\mathbf{P}_{\mathcal{P}(x,y)}$, $\mathbf{V}_{w_x}$, $\mathbf{V}_{w_y}$, $\mathbf{V}_{F(x,y)}$ denote the path representation, the word embedding of x and y, and the feature embeddings, respectively.

Our approach is general and can be flexibly extended to incorporate different feature representation components introduced by other relation extraction models (Zhang et al., 2017; Lin et al., 2016; Shwartz et al., 2016). We leave in-depth discussion of the design choice of hypernymy relation representation components as future work.

3 Reinforcement Learning for End-to-End Taxonomy Induction

We present the reinforcement learning (RL) approach to taxonomy induction in this section. The RL agent employs the term pair representations described in Section 2.2 as input, and explores how to generate a whole taxonomy by selecting one term at each time step and attaching it to the current taxonomy. We first describe the environment, including the actions, states, and rewards. Then, we introduce how to choose actions via a policy network.

3.1 Actions

We regard the process of building a taxonomy as making a sequence of actions. Specifically, we define that an action a_t at time step t is to (1) select a term x_1 from the remaining vocabulary V_t; (2) remove x_1 from V_t, and (3) attach x_1 as a hyponym of one term x_2 that is already on the current taxonomy T_t. Therefore, the size of action space at time step t is $|V_t| \times |T_t|$, where $|V_t|$ is the size of the remaining vocabulary V_t, and $|T_t|$ is the number of terms on the current taxonomy. At the beginning of each episode, the remaining vocabulary V_0 is equal to the input vocabulary and the taxonomy T_0 is empty. During the taxonomy induction process, the following relations always hold: $|V_t| = |V_{t-1}| - 1$, $|T_t| = |T_{t-1}| + 1$, and $|V_t| + |T_t| = |V_0|$. The episode terminates when all the terms are attached to the taxonomy, which makes the length of one episode equal to $|V_0|$.

A remaining issue is how to select the first term when no terms are on the taxonomy. One approach that we tried is to add a virtual node as root and consider it as if a real node. The *root embedding* is randomly initialized and updated with other parameters. This approach presumes that all taxonomies share a common root representation and expects to find the real root of a taxonomy via the term pair representations between the virtual root and other terms. Another approach that we explored is to postpone the decision of root by initializing T with a random term as current root at the beginning of one episode, and allowing the selection of *new root* by attaching one term as the hypernym of current root. In this way, it overcomes the lack of prior knowledge when the first term is chosen. The size of action space then becomes $|A_t| = |V_t| \times |T_t| + |V_t|$, and the length of one episode becomes $|V_0| - 1$. We compare the performance of the two approaches in Section 4.

3.2 States

The *state* **s** at time t comprises the current taxonomy T_t and the remaining vocabulary V_t. At each time step, the environment provides the information of current state, based on which the RL agent takes an action. Once a term pair (x_1, x_2) is selected, the position of the new term x_1 is automatically determined since the other term x_2 is already on the taxonomy and we can simply attach x_1 by adding an edge between x_1 and x_2.

3.3 Rewards

The agent takes a scalar *reward* as feedback of its actions to learn its policy. One obvious reward is to wait until the end of taxonomy induction, and then compare the predicted taxonomy with gold taxonomy. However, this reward is delayed and difficult to measure individual actions in our scenario. Instead, we use reward shaping, *i.e.*, giving intermediate rewards at each time step, to accelerate the learning process. Empirically, we set the reward r at time step t to be the difference of Edge-F1 (defined in Section 4.2 and evaluated by comparing the current taxonomy with the gold taxonomy) between current and last time step: $r_t = F1_{e_t} - F1_{e_{t-1}}$. If current Edge-F1 is better than that at last time step, the reward would be positive, and vice versa. The cumula-

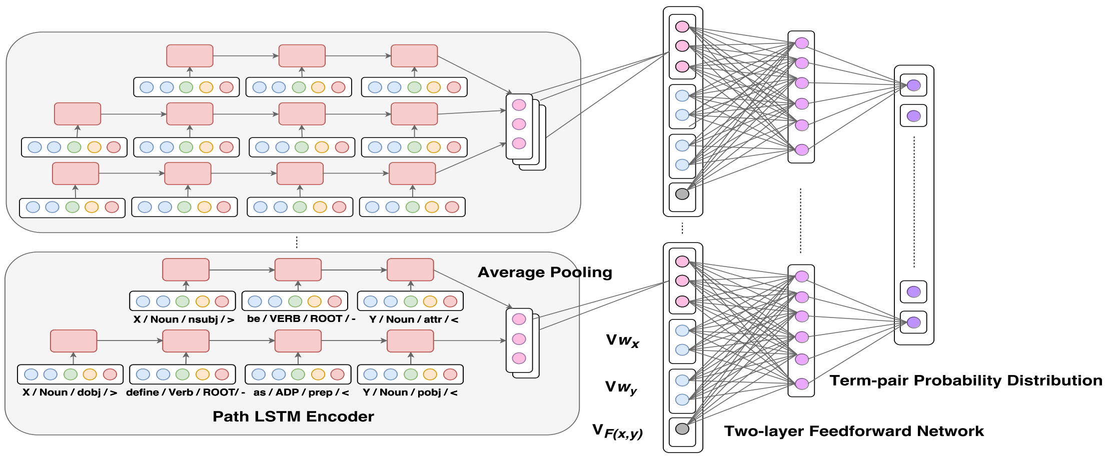


Figure 2: The architecture of the policy network. The dependency paths are encoded and concatenated with word embeddings and feature embeddings, and then fed into a two-layer feed-forward network.

tive reward from current time step to the end of an episode would cancel the intermediate rewards and thus reflect whether current action improves the overall performance or not. As a result, the agent would not focus on the selection of current term pair but have a long-term view that takes following actions into account. For example, suppose there are two actions at the same time step. One action attaches a leaf node to a high-level node, and the other action attaches a non-leaf node to the same high-level node. Both attachments form a wrong edge but the latter one is likely to receive a higher cumulative reward because its following attachments are more likely to be correct.

3.4 Policy Network

After we introduce the term pair representations and define the states, actions, and rewards, the problem becomes how to choose an action from the action space, *i.e.*, which term pair (x_1, x_2) should be selected given the current state? To solve the problem, we parameterize each action a by a policy network $\pi(\mathbf{a} \mid \mathbf{s}; \mathbf{W}_{RL})$. The architecture of our policy network is shown in Fig. 2. For each term pair, its representation is obtained by the path LSTM encoder, the word embeddings of both terms, and the embeddings of features. By stacking the term pair representations, we can obtain an action matrix $\mathbf{A}_t$ with size $(|V_t| \times |T_t|) \times dim(\mathbf{R})$, where $(|V_t| \times |T_t|)$ denotes the number of possible actions (term pairs) at time t and $dim(\mathbf{R})$ denotes the dimension of term pair representation $\mathbf{R}$. $\mathbf{A}_t$ is then fed into a two-layer feed-forward network followed by a softmax layer which outputs the probability distribution of actions.[5] Finally, an action a_t is sampled based on the probability distribution of the action space:

$$\begin{aligned}
\mathbf{H}_t &= \mathrm{ReLU}(\mathbf{W}^1_{RL}\mathbf{A}^T_t + \mathbf{b}^1_{RL}),\\
\pi(\mathbf{a} \mid \mathbf{s}; \mathbf{W}_{RL}) &= \mathrm{softmax}(\mathbf{W}^2_{RL}\mathbf{H}_t + \mathbf{b}^2_{RL}),\\
a_t &\sim \pi(\mathbf{a} \mid \mathbf{s}; \mathbf{W}_{RL}).
\end{aligned}$$

At the time of inference, instead of sampling an action from the probability distribution, we greedily select the term pair with the highest probability.

We use *REINFORCE* (Williams, 1992), one instance of the policy gradient methods as the optimization algorithm. Specifically, for each episode, the weights of the policy network are updated as follows:

$$\mathbf{W}_{RL} = \mathbf{W}_{RL} + \alpha \sum_{t=1}^{T} \nabla log\pi(\mathbf{a}_t \mid \mathbf{s}; \mathbf{W}_{RL}) \cdot v_t,$$

where $v_i = \sum_{t=i}^{T} \gamma^{t-i} r_t$ is the culmulative future reward at time i and $\gamma \in [0, 1]$ is a discounting factor of future rewards.

To reduce variance, 10 rollouts for each training sample are run and the rewards are averaged. Another common strategy for variance reduction is to use a baseline and give the agent the difference between the real reward and the baseline reward instead of feeding the real reward directly. We use a moving average of the reward as the baseline for simplicity.

[5]We tried to encode induction history by feeding representations of previously selected term pairs into an LSTM, and leveraging the output of the LSTM as history representation (concatenating it with current term pair representations or passing it to a feed-forward network). However, we didn't observe clear performance change.

3.5 Implementation Details

We use pre-trained GloVe word vectors (Pennington et al., 2014) with dimensionality 50 as word embeddings. We limit the maximum number of dependency paths between each term pair to be 200 because some term pairs containing general terms may have too many dependency paths. We run with different random seeds and hyperparameters and use the validation set to pick the best model. We use an Adam optimizer with initial learning rate 10^{-3}. We set the discounting factor γ to 0.4 as it is shown that using a smaller discount factor than defined can be viewed as regularization (Jiang et al., 2015). Since the parameter updates are performed at the end of each episode, we cache the term pair representations and reuse them when the same term pairs are encountered again in the same episode. As a result, the proposed approach is very time efficient – each training epoch takes less than 20 minutes on a single-core CPU using DyNet (Neubig et al., 2017).

4 Experiments

We design two experiments to demonstrate the effectiveness of our proposed RL approach for taxonomy induction. First, we compare our end-to-end approach with two-phase methods and show that our approach yields taxonomies with higher quality through reducing error propagation and optimizing towards holistic metrics. Second, we conduct a controlled experiment on hypernymy organization, where the same hypernym graph is used as the input of both our approach and the compared methods. We show that our RL method is more effective at hypernymy organization.

4.1 Experiment Setup

Here we introduce the details of our two experiments on validating that (1) the proposed approach can effectively reduce error propagation; and (2) our approach yields better taxonomies via optimizing metrics on holistic taxonomy structure.

Performance Study on End-to-End Taxonomy Induction. In the first experiment, we show that our joint learning approach is superior to two-phase methods. Towards this goal, we compare with TAXI (Panchenko et al., 2016), a typical two-phase approach, two-phase HypeNET, implemented by pairwise hypernymy detection and hypernymy organization using MST, and Bansal et al. (2014). The dataset we use in this experiment is from Bansal et al. (2014), which is a set of medium-sized full-domain taxonomies consisting of bottom-out full subtrees sampled from WordNet. Terms in different taxonomies are from various domains such as animals, general concepts, daily necessities. Each taxonomy is of height four (*i.e.*, 4 nodes from root to leaf) and contains (10, 50] nodes. The dataset contains 761 non-overlapped taxonomies in total and is partitioned by 70/15/15% (533/114/114) as training, validation, and test set, respectively.

Testing on Hypernymy Organization. In the second experiment, we show that our approach is better at hypernymy organization by leveraging the global taxonomy structure. For a fair comparison, we reuse the hypernym graph as in TAXI (Panchenko et al., 2016) and SubSeq (Gupta et al., 2017) so that the inputs of each model are the same. Specifically, we restrict the action space to be the same as the baselines by considering only term pairs in the hypernym graph, rather than all $|V| \times |T|$ possible term pairs. As a result, it is possible that at some point no more hypernym candidates can be found but the remaining vocabulary is still not empty. If the induction terminates at this point, we call it a *partial induction*. We can also continue the induction by restoring the original action space at this moment so that all the terms in V are eventually attached to the taxonomy. We call this setting a *full induction*. In this experiment, we use the English environment and science taxonomies in the SemEval-2016 task 13 (TExEval-2) (Bordea et al., 2016). Each taxonomy is composed of hundreds of terms, which is much larger than the WordNet taxonomies. The taxonomies are aggregated from existing resources such as WordNet, Eurovoc[6], and the Wikipedia Bitaxonomy (Flati et al., 2014). Since this dataset provides no training data, we train our model using the WordNet dataset in the first experiment. To avoid possible overlap between these two sources, we exclude those taxonomies constructed from WordNet.

In both experiments, we combine three public corpora – the latest Wikipedia dump, the UMBC web-based corpus (Han et al., 2013) and the One Billion Word Language Modeling Benchmark (Chelba et al., 2013). Only sentences where term pairs co-occur are reserved, which results in

[6]http://eurovoc.europa.eu/drupal/

Model	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$
TAXI	66.1	13.9	23.0	54.8	18.0	27.1
HypeNET	32.8	26.7	29.4	26.1	17.2	20.7
HypeNET+MST	33.7	41.1	37.0	29.2	29.2	29.2
TaxoRL (RE)	35.8	47.4	40.8	35.4	35.4	35.4
TaxoRL (NR)	41.3	49.2	**44.9**	35.6	35.6	**35.6**
Bansal et al. (2014)	48.0	55.2	51.4	-	-	-
TaxoRL (NR) + FG	52.9	58.6	**55.6**	43.8	43.8	**43.8**

Table 1: Results of the end-to-end taxonomy induction experiment. Our approach significantly outperforms two-phase methods (Panchenko et al., 2016; Shwartz et al., 2016; Bansal et al., 2014). Bansal et al. (2014) and TaxoRL (NR) + FG are listed separately because they use extra resources.

a corpus with size 2.6 GB for the WordNet dataset and 810 MB for the TExEval-2 dataset. Dependency paths between term pairs are extracted from the corpus via spaCy[7].

4.2 Evaluation Metrics

Ancestor-F1. It compares the ancestors ("is-a" pairs) on the predicted taxonomy with those on the gold taxonomy. We use P_a, R_a, $F1_a$ to denote the precision, recall, and F1-score, respectively:

$$P_a = \frac{|\text{is-a}_{\text{sys}} \wedge \text{is-a}_{\text{gold}}|}{|\text{is-a}_{\text{sys}}|}, R_a = \frac{|\text{is-a}_{\text{sys}} \wedge \text{is-a}_{\text{gold}}|}{|\text{is-a}_{\text{gold}}|}.$$

Edge-F1. It is more strict than *Ancestor-F1* since it only compares predicted edges with gold edges. Similarly, we denote edge-based metrics as P_e, R_e, and $F1_e$, respectively. Note that $P_e = R_e = F1_e$ if the number of predicted edges is the same as gold edges.

4.3 Results

Comparison on End-to-End Taxonomy Induction. Table 1 shows the results of the first experiment. HypeNET (Shwartz et al., 2016) uses additional surface features described in Section 2.2. HypeNET+MST extends HypeNET by first constructing a hypernym graph using HypeNET's output as weights of edges and then finding the MST (Chu, 1965) of this graph. TaxoRL (RE) denotes our RL approach which assumes a common *Root Embedding*, and TaxoRL (NR) denotes its variant that allows a *New Root* to be added.

We can see that TAXI has the lowest $F1_a$ while HypeNET performs the worst in $F1_e$. Both TAXI and HypeNET's $F1_a$ and $F1_e$ are lower than 30. HypeNET+MST outperforms HypeNET in both $F1_a$ and $F1_e$, because it considers the global taxonomy structure, although the two phases are performed independently. TaxoRL (RE) uses exactly the same input as HypeNET+MST and yet achieves significantly better performance, which demonstrates the superiority of combining the phases of hypernymy detection and hypernymy organization. Also, we found that presuming a shared root embedding for all taxonomies can be inappropriate if they are from different domains, which explains why TaxoRL (NR) performs better than TaxoRL (RE). Finally, after we add the frequency and generality features (TaxoRL (NR) + FG), our approach outperforms Bansal et al. (2014), even if a much smaller corpus is used.[8]

	Model	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$
Env	TAXI (DAG)	50.1	32.7	39.6	33.8	26.8	29.9
	TAXI (tree)	**67.5**	30.8	42.3	**41.1**	23.1	29.6
	SubSeq	-	-	-	-	-	22.4
	TaxoRL (Partial)	51.6	36.4	42.7	37.5	24.2	29.4
	TaxoRL (Full)	47.2	**54.6**	**50.6**	32.3	**32.3**	**32.3**
Sci	TAXI (DAG)	61.6	41.7	49.7	38.8	34.8	36.7
	TAXI (tree)	76.8	38.3	51.1	44.8	28.8	35.1
	SubSeq	-	-	-	-	-	39.9
	TaxoRL (Partial)	**84.6**	34.4	48.9	**56.9**	33.0	**41.8**
	TaxoRL (Full)	68.3	**52.9**	**59.6**	37.9	**37.9**	37.9

Table 2: Results of the hypernymy organization experiment. Our approach outperforms Panchenko et al. (2016); Gupta et al. (2017) when the same hypernym graph is used as input. The precision of partial induction in both metrics is high. The precision of full induction is relatively lower but its recall is much higher.

Analysis on Hypernymy Organization. Table 2 lists the results of the second experiment. TAXI (DAG) (Panchenko et al., 2016) denotes TAXI's original performance on the TExEval-2 dataset.[9] Since we don't allow DAG in our setting, we convert its results to trees (denoted by TAXI (tree)) by only keeping the first parent of each node. SubSeq (Gupta et al., 2017) also reuses TAXI's hypernym candidates. TaxoRL (Partial) and TaxoRL (Full) denotes partial induction and full induction, respectively. Our joint RL approach outperforms baselines in both domains substantially. TaxoRL (Partial) achieves higher precision in both ancestor-based and edge-based metrics but has rel-

[7]https://spacy.io/

[8]Bansal et al. (2014) use an unavailable resource (Brants and Franz, 2006) which contains one trillion tokens while our public corpus contains several billion tokens. The frequency and generality features are sparse because the vocabulary that TAXI (in the TExEval-2 competition) used for focused crawling and hypernymy detection was different.

[9]alt.qcri.org/semeval2016/task13/index.php?id=evaluation

atively lower recall since it discards some terms. In addition, it achieves the best $F1_e$ in science domain. TaxoRL (Full) has the highest recall in both domains and metrics, with the compromise of lower precision. Overall, TaxoRL (Full) performs the best in both domains in terms of $F1_a$ and achieves best $F1_e$ in environment domain.

5 Ablation Analysis and Case Study

In this section, we conduct ablation analysis and present a concrete case for better interpreting our model and experimental results.

Table 3 shows the ablation study of TaxoRL (NR) on the WordNet dataset. As one may find, different types of features are complementary to each other. Combining distributional and path-based features performs better than using either of them alone (Shwartz et al., 2016). Adding surface features helps model string-level statistics that are hard to capture by distributional or path-based features. Significant improvement is observed when more data is used, meaning that standard corpora (such as Wikipedia) might not be enough for complicated taxonomies like WordNet.

Fig. 3 shows the results of taxonomy about *filter*. We denote the selected term pair at time step t as (hypo, hyper, t). Initially, the term *water filter* is randomly chosen as the taxonomy root. Then, a wrong term pair (water filter, air filter, 1) is selected possibly due to the noise and sparsity of features, which makes the term *air filter* become the new root. (air filter, filter, 2) is selected next and the current root becomes *filter* that is identical to the real root. After that, term pairs such as (fuel filter, filter, 3), (coffee filter, filter, 4) are selected correctly, mainly because of the substring inclusion intuition. Other term pairs such as (colander, strainer, 13), (glass wool, filter, 16) are discovered later, largely by the information encoded in the dependency paths and embeddings. For those undiscovered relations, (filter tip, air filter) has no dependency path in the corpus. *sifter* is attached to the taxonomy before its hypernym *sieve*. There is no co-occurrence between *bacteria bed* (or *drain basket*) and other terms. In addition, it is hard to utilize the surface features since they "look different" from other terms. That is also why (bacteria bed, air filter, 17) and (drain basket, air filter, 18) are attached in the end: our approach prefers to select term pairs with high confidence first.

Model	P_a	R_a	$F1_a$	$F1_e$
Distributional Info	27.1	24.3	25.6	13.8
Path-based Info	27.8	48.5	33.7	27.4
D + P	36.6	39.4	37.9	28.3
D + P + Surface Features	41.3	49.2	44.9	35.6
D + P + S + FG	**52.9**	**58.6**	**55.6**	**43.8**

Table 3: Ablation study on the WordNet dataset (Bansal et al., 2014). P_e and R_e are omitted because they are the same as $F1_e$ for each model. We can see that our approach benefits from multiple sources of information which are complementary to each other.

6 Related Work

6.1 Hypernymy Detection

Finding high-quality hypernyms is of great importance since it serves as the first step of taxonomy induction. In previous works, there are mainly two categories of approaches for hypernymy detection, namely pattern-based and distributional methods. Pattern-based methods consider lexico-syntactic patterns between the joint occurrences of term pairs for hypernymy detection. They generally achieve high precision but suffer from low recall. Typical methods that leverage patterns for hypernym extraction include (Hearst, 1992; Snow et al., 2005; Kozareva and Hovy, 2010; Panchenko et al., 2016; Nakashole et al., 2012). Distributional methods leverage the contexts of each term separately. The co-occurrence of term pairs is hence unnecessary. Some distributional methods are developed in an unsupervised manner. Measures such as symmetric similarity (Lin et al., 1998) and those based on distributional inclusion hypothesis (Weeds et al., 2004; Chang et al., 2017) were proposed. Supervised methods, on the other hand, usually have better performance than unsupervised methods for hypernymy detection. Recent works towards this direction include (Fu et al., 2014; Rimell, 2014; Yu et al., 2015; Tuan et al., 2016; Shwartz et al., 2016).

6.2 Taxonomy Induction

There are many lines of work for taxonomy induction in the prior literature. One line of works (Snow et al., 2005; Yang and Callan, 2009; Shen et al., 2012; Jurgens and Pilehvar, 2015) aims to complete existing taxonomies by attaching new terms in an incremental way. Snow et al. (2005) enrich WordNet by maximizing the probability of an extended taxonomy given evidence of relations from text corpora. Shen et al. (2012) determine whether an entity is on the taxonomy and either attach it to the right category or link it to an existing one based on the results. Another line of works (Suchanek et al., 2007; Ponzetto and Strube, 2008; Flati et al., 2014) focuses on the taxonomy induction of existing encyclopedias (*e.g.*, Wikipedia), mainly by employing the nature that they are already organized into semi-structured data. To deal with the issue of incomplete coverage, some works (Liu et al., 2012; Dong et al., 2014; Panchenko et al., 2016; Kozareva and Hovy, 2010) utilize data from domain-specific resources or the Web. Panchenko et al. (2016) extract hypernyms by patterns from general purpose corpora and domain-specific corpora bootstrapped from the input vocabulary. Kozareva and Hovy (2010) harvest new terms from the Web by employing Hearst-like lexico-syntactic patterns and validate the learned is-a relations by a web-based concept positioning procedure.

Many works (Kozareva and Hovy, 2010; Anh et al., 2014; Velardi et al., 2013; Bansal et al., 2014; Zhang et al., 2016; Panchenko et al., 2016; Gupta et al., 2017) cast the task of hypernymy organization as a graph optimization problem. Kozareva and Hovy (2010) begin with a set of root terms and leaf terms and aim to generate intermediate terms by deriving the longest path from the root to leaf in a noisy hypernym graph. Velardi et al. (2013) induct a taxonomy from the hypernym graph via optimal branching and a weighting policy. Bansal et al. (2014) regard the induction of a taxonomy as a structured learning problem by building a factor graph to model the relations between edges and siblings, and output the MST found by the Chu-Liu/Edmond's algorithm (Chu, 1965). Zhang et al. (2016) propose a probabilistic Bayesian model which incorporates visual features (images) in addition to text features (words) to improve the performance. The optimal taxonomy is also found by the MST. Gupta et al. (2017) extract hypernym subsequences based on hypernym pairs, and regard the task of taxonomy induction as an instance of the minimum-cost flow problem.

7 Conclusion and Future Work

This paper presents a novel end-to-end reinforcement learning approach for automatic taxonomy induction. Unlike previous two-phase methods

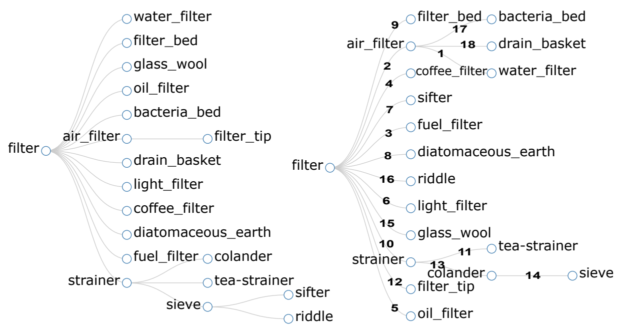


Figure 3: The gold taxonomy in WordNet is on the left. The predicted taxonomy is on the right. The numbers indicate the order of term pair selections. Term pairs with high confidence are selected first.

that treat term pairs independently or equally, our approach learns the representations of term pairs by optimizing a holistic tree metric over the training taxonomies. The error propagation between two phases is thus effectively reduced and the global taxonomy structure is better captured. Experiments on two public datasets from different domains show that our approach outperforms state-of-the-art methods significantly. In the future, we will explore more strategies towards term pair selection (*e.g.*, allow the RL agent to remove terms from the taxonomy) and reward function design. In addition, study on how to effectively encode induction history will be interesting.

Acknowledgments

Research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and grant 1U54GM-114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. We thank Mohit Bansal, Hao Zhang, and anonymous reviewers for valuable feedback.

References

Tuan Luu Anh, Jung-jae Kim, and See Kiong Ng. 2014. Taxonomy construction using syntactic contextual evidence. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 810–819.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *ACL (1)*, pages 1041–1051.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *SemEval-2016*, pages 1081–1091. Association for Computational Linguistics.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram corpus version 1.1. *Google Inc.*

Jose Camacho-Collados. 2017. Why we have switched from building full-fledged taxonomies to simply detecting hypernymy relations. *arXiv preprint arXiv:1703.04178*.

Haw-Shiuan Chang, ZiYun Wang, Luke Vilnis, and Andrew McCallum. 2017. Unsupervised hypernym detection by distributional inclusion vector embedding. *arXiv preprint arXiv:1710.00880*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted rule injection for relation embeddings. In *EMNLP*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Stefano Faralli, Giovanni Stilo, and Paola Velardi. 2015. Large scale homophily analysis in twitter using a twixonomy. In *IJCAI*, pages 2334–2340.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the wikipedia bitaxonomy project. In *ACL (1)*, pages 945–955.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1199–1209.

Amit Gupta, Rémi Lebret, Hamza Harkous, and Karl Aberer. 2017. Taxonomy induction using hypernym subsequences. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1329–1338. ACM.

Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc_ebiquity-core: Semantic textual similarity systems. In ** SEM@ NAACL-HLT*, pages 44–52.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. 2015. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189. International Foundation for Autonomous Agents and Multiagent Systems.

David Jurgens and Mohammad Taher Pilehvar. 2015. Reserating the awesometastic: An automatic extension of the wordnet taxonomy for novel terms. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1459–1465.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1110–1118. Association for Computational Linguistics.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Dekang Lin et al. 1998. An information-theoretic definition of similarity. In *Icml*, volume 98, pages 296–304. Citeseer.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL (1)*.

Xueqing Liu, Yangqiu Song, Shixia Liu, and Haixun Wang. 2012. Automatic taxonomy construction from keywords. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1433–1441. ACM.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cedrick Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, San Diego, CA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Simone Paolo Ponzetto and Michael Strube. 2008. Wikitaxonomy: A large scale knowledge resource. In *ECAI*, volume 178, pages 751–752.

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 511–519.

Mark Sammons. 2012. Recognizing textual entailment.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. A graph-based approach for ontology population with named entities. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 345–354. ACM.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2389–2398.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, pages 1297–1304.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Luu A Tuan, Yi Tay, Siu C Hui, and See K Ng. 2016. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the EMNLP conference*, pages 403–413.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1015. Association for Computational Linguistics.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ichiro Yamada, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, Masaki Murata, Stijn De Saeger, Francis Bond, and Asuka Sumida. 2009. Hypernym discovery based on distributional similarity and hierarchical structures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 929–937. Association for Computational Linguistics.

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 271–279. Association for Computational Linguistics.

Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions - a knowledge graph approach. In *AAAI*.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *IJCAI*, pages 1390–1397.

Hao Zhang, Zhiting Hu, Yuntian Deng, Mrinmaya Sachan, Zhicheng Yan, and Eric Xing. 2016. Learning concept taxonomies from multi-modal data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1791–1801.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Incorporating Glosses into Neural Word Sense Disambiguation

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang and **Zhifang Sui**
Key Laboratory of Computational Linguistics, Ministry of Education,
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{luofuli, tianyu0421, xql, chbb, szf}@pku.edu.cn

Abstract

Word Sense Disambiguation (WSD) aims to identify the correct meaning of polysemous words in the particular context. Lexical resources like WordNet which are proved to be of great help for WSD in the knowledge-based methods. However, previous neural networks for WSD always rely on massive labeled data (context), ignoring lexical resources like glosses (sense definitions). In this paper, we integrate the context and glosses of the target word into a unified framework in order to make full use of both labeled data and lexical knowledge. Therefore, we propose **GAS**: a **g**loss-**a**ugmented W**S**D neural network which jointly encodes the context and glosses of the target word. GAS models the semantic relationship between the context and the gloss in an improved memory network framework, which breaks the barriers of the previous supervised methods and knowledge-based methods. We further extend the original gloss of word sense via its semantic relations in WordNet to enrich the gloss information. The experimental results show that our model outperforms the state-of-the-art systems on several English all-words WSD datasets.

1 Introduction

Word Sense Disambiguation (WSD) is a fundamental task and long-standing challenge in Natural Language Processing (NLP). There are several lines of research on WSD. Knowledge-based methods focus on exploiting lexical resources to infer the senses of word in the context. Supervised methods usually train multiple classifiers with manual designed features. Although supervised methods can achieve the state-of-the-art performance (Raganato et al., 2017b,a), there are still two major challenges.

Firstly, supervised methods (Zhi and Ng, 2010; Iacobacci et al., 2016) usually train a dedicated classifier for each word individually (often called *word expert*). So it can not easily scale up to all-words WSD task which requires to disambiguate all the polysemous word in texts [1]. Recent neural-based methods (Kågebäck and Salomonsson, 2016; Raganato et al., 2017a) solve this problem by building a unified model for all the polysemous words, but they still can't beat the best *word expert* system.

Secondly, all the neural-based methods always only consider the local context of the target word, ignoring the lexical resources like WordNet (Miller, 1995) which are widely used in the knowledge-based methods. The gloss, which extensionally defines a word sense meaning, plays a key role in the well-known Lesk algorithm (Lesk, 1986). Recent studies (Banerjee and Pedersen, 2002; Basile et al., 2014) have shown that enriching gloss information through its semantic relations can greatly improve the accuracy of Lesk algorithm.

To this end, our goal is to incorporate the gloss information into a unified neural network for all of the polysemous words. We further consider extending the original gloss through its semantic relations in our framework. As shown in Figure 1, the glosses of hypernyms and hyponyms can enrich the original gloss information as well as help to build better a sense representation. Therefore, we integrate not only the original gloss but also the related glosses of hypernyms and hyponyms into the neural network.

[1] If there are N polysemous words in texts, they need to train N classifiers individually.

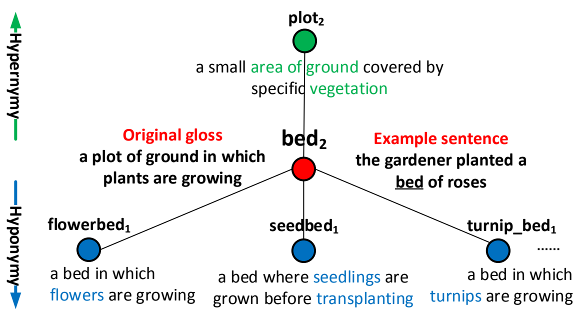

Hypernymy
Hyponymy
plot₂
a small area of ground covered by specific vegetation
Original gloss
a plot of ground in which plants are growing
bed₂
Example sentence
the gardener planted a bed of roses
flowerbed₁
a bed in which flowers are growing
seedbed₁
a bed where seedlings are grown before transplanting
turnip_bed₁
......
a bed in which turnips are growing

Figure 1: The hypernym (green node) and hyponyms (blue nodes) for the 2nd sense bed_2 of bed, which means *a plot of ground in which plants are growing*, rather than the bed for sleeping in. The figure shows that bed_2 is a kind of $plot_2$, and bed_2 includes $flowerbed_1$, $seedbed_1$, etc.

In this paper, we propose a novel model **GAS**: a gloss-augmented WSD neural network which is a variant of the memory network (Sukhbaatar et al., 2015b; Kumar et al., 2016; Xiong et al., 2016). GAS jointly encodes the context and glosses of the target word and models the semantic relationship between the context and glosses in the memory module. In order to measure the inner relationship between glosses and context more accurately, we employ multiple passes operation within the memory as the re-reading process and adopt two memory updating mechanisms.

The main contributions of this paper are listed as follows:

- To the best of our knowledge, our model is the first to incorporate the glosses into an end-to-end neural WSD model. In this way, our model can benefit from not only massive labeled data but also rich lexical knowledge.
- In order to model semantic relationship of context and glosses, we propose a gloss-augmented neural network (GAS) in an improved memory network paradigm.
- We further expand the gloss through its semantic relations to enrich the gloss information and better infer the context. We extend the gloss module in GAS to a hierarchical framework in order to mirror the hierarchies of word senses in WordNet.
- The experimental results on several English all-words WSD benchmark datasets show that our model outperforms the state-of-the-art systems.

2 Related Work

Knowledge-based, supervised and neural-based methods have already been applied to WSD task (Navigli, 2009).

Knowledge-based WSD methods mainly exploit two kinds of knowledge to disambiguate polysemous words: **1)** The gloss, which defines a word sense meaning, is mainly used in Lesk algorithm (Lesk, 1986) and its variants. **2)** The structure of the semantic network, whose nodes are synsets [2] and edges are semantic relations, is mainly used in graph-based algorithms (Agirre et al., 2014; Moro et al., 2014).

Supervised methods (Zhi and Ng, 2010; Iacobacci et al., 2016) usually involve each target word as a separate classification problem (often called *word expert*) and train classifiers based on manual designed features.

Although *word expert* supervised WSD methods perform best in terms of accuray, they are less flexible than knowledge-based methods in the all-words WSD task (Raganato et al., 2017a). To deal with this problem, recent neural-based methods aim to build a unified classifier which shares parameters among all the polysemous words. Kågebäck and Salomonsson (2016) leverages the bidirectional long short-term memory network which shares model parameters among all the polysemous words. Raganato et al. (2017a) transfers the WSD problem into a neural sequence labeling task. However, none of the neural-based methods can totally beat the best *word expert* supervised methods on English all-words WSD datasets.

What's more, all of the previous supervised methods and neural-based methods rarely take the lexical resources like WordNet (Fellbaum, 1998) into consideration. Recent studies on sense embeddings have proved that lexical resources are helpful. Chen et al. (2015) trains word sense embeddings through learning sentence level embeddings from glosses using a convolutional neural networks. Rothe and Schütze (2015) extends word embeddings to sense embeddings by using the constraints and semantic relations in WordNet. They achieve an improvement of more than 1% in WSD performance when using sense embeddings as WSD features for SVM classifier. This work shows that integrating structural information of lexical resources can help to *word expert* supervised methods. However, sense embeddings

[2] A synset is a set of words that denote the same sense.

can only indirectly help to WSD (as SVM classifier features). Raganato et al. (2017a) shows that the coarse-grained semantic labels in WordNet can help to WSD in a multi-task learning framework. As far as we know, there is no study directly integrates glosses or semantic relations of the WordNet into an end-to-end model.

In this paper, we focus on how to integrate glosses into a unified neural WSD system. Memory network (Sukhbaatar et al., 2015b; Kumar et al., 2016; Xiong et al., 2016) is initially proposed to solve question answering problems. Recent researches show that memory network obtains the state-of-the-art results in many NLP tasks such as sentiment classification (Li et al., 2017) and analysis (Gui et al., 2017), poetry generation (Zhang et al., 2017), spoken language understanding (Chen et al., 2016), etc. Inspired by the success of memory network used in many NLP tasks, we introduce it into WSD. We make some adaptations to the initial memory network in order to incorporate glosses and capture the inner relationship between the context and glosses.

3 Incorporating Glosses into Neural Word Sense Disambiguation

In this section, we first give an overview of the proposed model **GAS**: a gloss-augmented WSD neural network which integrates the context and the glosses of the target word into a unified framework. After that, each individual module is described in detail.

3.1 Architecture of GAS

The overall architecture of the proposed model is shown in Figure 2. It consists of four modules:

- **Context Module**: The context module encodes the local context (a sequence of surrounding words) of the target word into a distributed vector representation.

- **Gloss Module**: Like the context module, the gloss module encodes all the glosses of the target word into a separate vector representations of the same size. In other words, we can get $|s_t|$ word sense representations according to $|s_t|$ [3] senses of the target word, where $|s_t|$ is the sense number of the target word w_t .

[3] s_t is the sense set $\{s_t^1, s_t^2, \ldots, s_t^p\}$ corresponding to the target word x_t

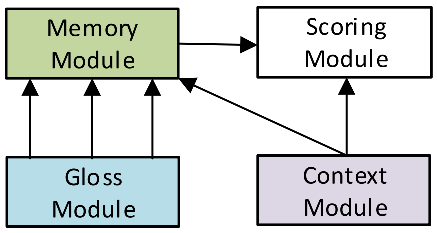


Figure 2: Overview of Gloss-augmented Memory Network for Word Sense Disambiguation.

- **Memory Module**: The memory module is employed to model the semantic relationship between the context embedding and gloss embedding produced by context module and gloss module respectively.

- **Scoring Module**: In order to benefit from both labeled contexts and gloss knowledge, the scoring module takes the context embedding from context module and the last step result from the memory module as input. Finally it generates a probability distribution over all the possible senses of the target word.

Detailed architecture of the proposed model is shown in Figure 3. The next four sections will show detailed configurations in each module.

3.2 Context Module

Context module encodes the context of the target word into a vector representation, which is also called context embedding in this paper.

We leverage the bidirectional long short-term memory network (Bi-LSTM) for taking both the preceding and following words of the target word into consideration. The input of this module $[x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_{T_x}]$ is a sequence of words surrounding the target word x_t, where T_x is the length of the context. After applying a lookup operation over the pre-trained word embedding matrix $\mathrm{M} \in \mathbb{R}^{D \times V}$, we transfer a one hot vector x_i into a D-dimensional vector. Then, the forward LSTM reads the segment $(x_1, \ldots, x_{t-1})$ on the left of the target word x_t and calculates a sequence of *forward hidden states* $(\overrightarrow{h}_1, \ldots, \overrightarrow{h}_{t-1})$. The backward LSTM reads the segment $(x_{T_x}, \ldots, x_{t+1})$ on the right of the target word x_t and calculates a sequence of *backward hidden states* $(\overleftarrow{h}_{T_x}, \ldots, \overleftarrow{h}_{t+1})$. The context vector c is finally concatenated as

$$c = [\overrightarrow{h}_{t-1} : \overleftarrow{h}_{t+1}] \qquad (1)$$

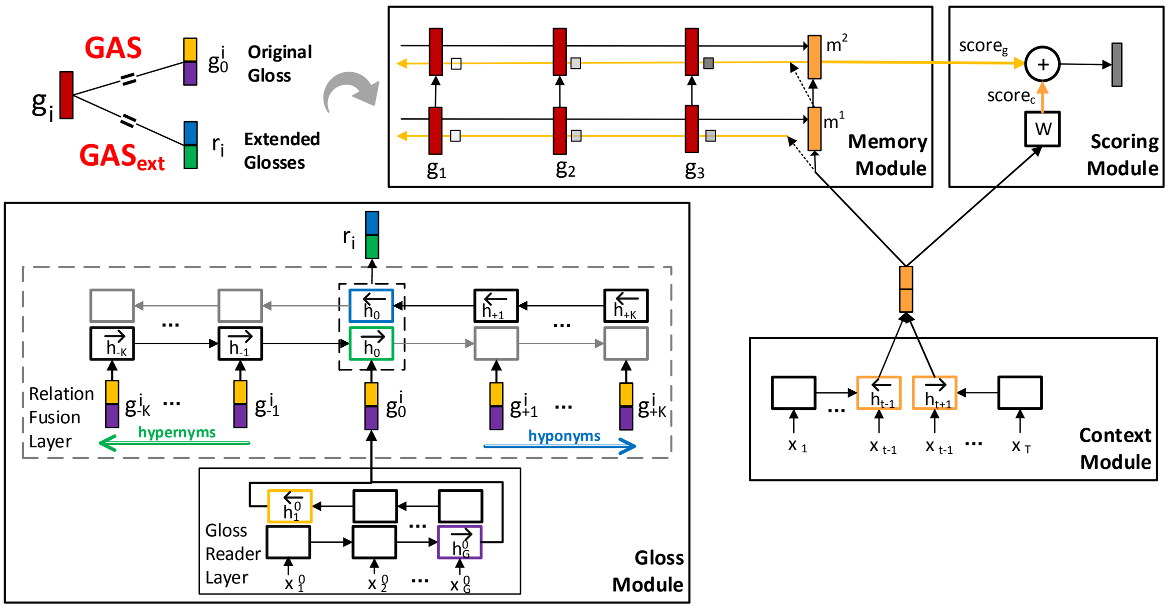


Figure 3: Detailed architecture of our proposed model, which consists of a context module, a gloss module, a memory module and a scoring module. The context module encodes the adjacent words surrounding the target word into a vector c. The gloss module encodes the original gloss or extended glosses into a vector g_i. In the memory module, we calculate the inner relationship (as attention) between context c and each gloss g_i and then update the memory as m_i at pass i. In the scoring module, we make final predictions based on the last pass attention of memory module and the context vector c. Note that GAS only uses the original gloss, while GAS$_{ext}$ uses the entended glosses through hypernymy and hyponymy relations. In other words, the relation fusion layer (grey dotted box) only belongs to GAS$_{ext}$.

where : is the concatenation operator.

3.3 Gloss Module

The gloss module encodes each gloss of the target word into a fixed size vector like the context vector c, which is also called gloss embedding. We further enrich the gloss information by taking semantic relations and their associated glosses into consideration.

This module contains a gloss reader layer and a relation fusion layer. Gloss reader layer generates a vector representations for a gloss. Relation fusion layer aims at modeling the semantic relations of each gloss in the expanded glosses list which consists of related glosses of the original gloss. Our model GAS with extended glosses is denoted as GAS$_{ext}$. GAS only encodes the original gloss, while GAS$_{ext}$ encodes the expanded glosses from hypernymy and hyponymy relations (details in Figure 3).

3.3.1 Gloss Reader Layer

Gloss reader layer contains two parts: gloss expansion and gloss encoder. Gloss expansion is to enrich the original gloss information through its hypernymy and hyponymy relations in WordNet. Gloss encoder is to encode each gloss into a vector representation.

Gloss Expansion: We only expand the glosses of nouns and verbs via their corresponding hypernyms and hyponyms. There are two reasons: One is that most of polysemous words (about 80%) are nouns and verbs; the other is that the most frequent relations among word senses for nouns and verbs are the hypernymy and hyponymy relations [4].

The original gloss is denoted as g_0. Breadth-first search method with a limited depth K is employed to extract the related glosses. The glosses of hypernyms within K depth are denoted as $[g_{-1}, g_{-2}, \ldots, g_{-L_1}]$. The glosses of hyponyms within K depth are denoted as $[g_{+1}, g_{+2}, \ldots, g_{+L_2}]$ [5]. Note that g_{+1} and g_{-1} are the glosses of the nearest word sense.

Gloss Encoder: We denote the j-th [6] gloss in

[4]In WordNet, more than 95% of relations for nouns and 80% for verbs are hypernymy and hyponymy relations.

[5]Since one synset has one or more direct hypernyms and hyponyms, $L_1 >= K$ and $L_2 >= K$.

[6]Since GAS don't have gloss expansion, j is always 0 and $g_i = g_0^i$. See more in Figure 3.

the expanded glosses list for i_{th} sense of the target word as a sequence of G words. Like the context encoder, the gloss encoder also leverages Bi-LSTM units to process the words sequence of the gloss. The gloss representation g_j^i is computed as the concatenation of the last hidden states of the *forward* and *backward* LSTM.

$$g_j^i = [\overrightarrow{h}_G^{i,j} : \overleftarrow{h}_1^{i,j}] \tag{2}$$

where $j \in [-L_1, \dots, -1, 0, +1, \dots, +L_2]$ and : is the concatenation operator .

3.3.2 Relation Fusion Layer

Relation fusion layer models the hypernymy and hyponymy relations of the target word sense. A forward LSTM is employed to encode the hypernyms' glosses of i_{th} sense $(g_{-L_1}^i, \cdots, g_{-1}^i, g_0^i)$ as a sequence of *forward hidden states* $(\overrightarrow{h}_{-L_1}^i, \dots, \overrightarrow{h}_{-1}^i, \overrightarrow{h}_0^i)$. A backward LSTM is employed to encode the hyponyms' glosses of i_{th} sense $(g_{+L_2}^i, \dots, g_{+1}^i, g_0^i)$ as a sequence of *backward hidden states* $(\overleftarrow{h}_{+L_2}^i, \dots, \overleftarrow{h}_{+1}^i, \overleftarrow{h}_0^i)$. In order to highlight the original gloss g_0^i, the enhanced i_{th} sense representation is concatenated as the final state of the forward and backward LSTM.

$$g_i = [\overrightarrow{h}_0^i : \overleftarrow{h}_0^i] \tag{3}$$

3.4 Memory Module

The memory module has two inputs: the context vector c from the context module and the gloss vectors $\{g_1, g_2, \dots, g_{|s_t|}\}$ from the gloss module, where $|s_t|$ is the number of word senses. We model the inner relationship between the context and glosses by attention calculation. Since one-pass attention calculation may not fully reflect the relationship between the context and glosses (details in Section 4.4.2), the memory module adopts a repeated deliberation process. The process repeats reading gloss vectors in the following passes, in order to highlight the correct word sense for the following scoring module by a more accurate attention calculation. After each pass, we update the memory to refine the states of the current pass. Therefore, memory module contains two phases: attention calculation and memory update.

Attention Calculation: For each pass k, the attention e_i^k of gloss g_i is generally computed as

$$e_i^k = f(g_i, m^{k-1}, c) \tag{4}$$

where m^{k-1} is the memory vector in the $(k-1)$-th pass while c is the context vector. The scoring function f calculates the semantic relationship of the gloss and context, taking the vector set (g_i, m^{k-1}, c) as input. In the first pass, the attention reflects the similarity of context and each gloss. In the next pass, the attention reflects the similarity of adapted memory and each gloss. A dot product is applied to calculate the similarity of each gloss vector and context (or memory) vector. We treat c as m^0. So, the attention α_i^k of gloss g_i at pass k is computed as a dot product of g_i and m^{k-1}:

$$e_i^k = g_i \cdot m^{k-1} \tag{5}$$

$$\alpha_i^k = \frac{\exp(e_i^k)}{\sum_{j=1}^{|s_t|} \exp(e_i^j)} \tag{6}$$

Memory Update: After calculating the attention, we store the memory state in u^k which is a weighted sum of gloss vectors and is computed as

$$u^k = \sum_{i=1}^{n} \alpha_i^k g_i \tag{7}$$

where n is the hidden size of LSTM in the context module and gloss module. And then, we update the memory vector m^k from last pass memory m^{k-1}, context vector c, and memory state u^k. We propose two memory update methods:

- *Linear*: we update the memory vector m^k by a linear transformation from m^{k-1}

$$m^k = Hm^{k-1} + u^k \tag{8}$$

  where $H \in \mathbb{R}^{2n \times 2n}$.

- *Concatenation*: we get a new memory for k-th pass by taking both the gloss embedding and context embedding into consideration

$$m^k = ReLU(W[m^{k-1} : u^k : c] + b) \tag{9}$$

  where : is the concatenation operator, $W \in \mathbb{R}^{n \times 6n}$ and $b \in \mathbb{R}^{2n}$.

3.5 Scoring Module

The scoring module calculates the scores for all the related senses $\{s_t^1, s_t^2, \dots, s_t^p\}$ corresponding to the target word x_t and finally outputs a sense probability distribution over all senses.

The overall score for each word sense is determined by gloss attention $\alpha_i^{T_M}$ from the last pass in the memory module, where T_M is the number of passes in the memory module. The e^{T_M} (α^{T_M} without Softmax) is regarded as the gloss score.

$$score_g = e^{T_M} \tag{10}$$

Meanwhile, a fully-connected layer is employed to calculate the context score.

$$score_c = W_{x_t} c + b_{x_t} \tag{11}$$

where $W_{x_t} \in \mathbb{R}^{|s_t| \times 2n}$, $b_{x_t} \in \mathbb{R}^{|s_t|}$, $|s_t|$ is the number of senses for the target word x_t and n is the number of hidden units in the LSTM.

It's noteworthy that in Equation 11, each ambiguous word x_t has its corresponding weight matrix W_{x_t} and bias b_{x_t} in the scoring module.

In order to balance the importance of background knowledge and labeled data, we introduce a parameter $\lambda \in \mathbb{R}^N$ [7] in the scoring module which is jointly learned during the training process. The probability distribution $\hat{y}$ over all the word senses of the target word is calculated as:

$$\hat{y} = Softmax(\lambda_{x_t} score_c + (1 - \lambda_{x_t}) score_g)$$

where λ_{x_t} is the parameter for word x_t, and $\lambda_{x_t} \in [0, 1]$.

During training, all model parameters are jointly learned by minimizing a standard cross-entropy loss between $\hat{y}$ and the true label y.

4 Experiments and Evaluation

4.1 Dataset

Evaluation Dataset: we evaluate our model on several English all-words WSD datasets. For fair comparison, we use the benchmark datasets proposed by Raganato et al. (2017b) which includes five standard all-words fine-grained WSD datasets from the Senseval and SemEval competitions. They are Senseval-2 (**SE2**), Senseval-3 task 1 (**SE3**), SemEval-07 task 17 (**SE7**), SemEval-13 task 12 (**SE13**), and SemEval-15 task 13 (**SE15**). Following by Raganato et al. (2017a), we choose SE7, the smallest test set as the development (validation) set, which consists of 455 labeled instances. The last four test sets consist of 6798 labeled instances with four types of target words, namely nouns, verbs, adverbs and adjectives. We extract word sense glosses from WordNet3.0 because Raganato et al. (2017b) maps all the sense annotations [8] from its original version to 3.0.

Training Dataset: We choose SemCor 3.0 as the training set, which was also used by Raganato et al. (2017a), Raganato et al. (2017b), Iacobacci et al. (2016), Zhi and Ng (2010), etc. It consists of 226,036 sense annotations from 352 documents, which is the largest manually annotated corpus for WSD. Note that all the systems listed in Table 1 are trained on SemCor 3.0.

4.2 Implementation Details

We use the validation set (SE7) to find the optimal settings of our framework: the hidden state size n, the number of passes $|T_M|$, the optimizer, etc. We use pre-trained word embeddings with 300 dimensions[9], and keep them fixed during the training process. We employ 256 hidden units in both the gloss module and the context module, which means n=256. Orthogonal initialization is used for weights in LSTM and random uniform initialization with range [-0.1, 0.1] is used for others. We assign gloss expansion depth K the value of 4. We also experiment with the number of passes $|T_M|$ from 1 to 5 in our framework, finding $|T_M| = 3$ performs best. We use Adam optimizer (Kingma and Ba, 2014) in the training process with 0.001 initial learning rate. In order to avoid overfitting, we use dropout regularization and set drop rate to 0.5. Training runs for up to 100 epochs with early stopping if the validation loss doesn't improve within the last 10 epochs.

4.3 Systems to be Compared

In this section, we describe several knowledge-based methods, supervised methods and neural-based methods which perform well on the English all-words WSD datasets for comparison.

4.3.1 Knowledge-based Systems

- **Lesk**$_{ext+emb}$: Basile et al. (2014) is a variant of Lesk algorithm (Lesk, 1986) by using a word similarity function defined on a distributional semantic space to calculate the gloss-context overlap. This work shows that glosses are important to WSD and enriching

[7] N is the number of polysemous words in the training corpora.

[8] The original WordNet version of SE2, SE3, SE7, SE13, SE15 are 1.7, 1.7.1, 2.1, 3.0 and 3.0, respectively.

[9] We download the pre-trained word embeddings from `https://github.com/stanfordnlp/GloVe`, and we select the smaller Wikipedia 2014 + Gigaword 5.

	Test Datasets				Concatenation of Test Datasets				
System	SE2	SE3	SE13	SE15	Noun	Verb	Adj	Adv	**All**
MFS baseline	65.6	66.0	63.8	67.1	67.7	49.8	73.1	80.5	65.5
Lesk$_{ext+emb}$ (Basile et al., 2014)	63.0	63.7	66.2	64.6	70.0	51.1	51.7	80.6	64.2
Babelfy (Moro et al., 2014)	67.0	63.5	66.4	70.3	68.9	50.7	73.2	79.8	66.4
IMS (Zhi and Ng, 2010)	70.9	69.3	65.3	69.5	70.5	55.8	75.6	82.9	68.9
IMS$_{+emb}$ (Iacobacci et al., 2016)	72.2	70.4	65.9	71.5	71.9	56.6	75.9	84.7	70.1
Bi-LSTM (Kågebäck and Salomonsson, 2016)	71.1	68.4	64.8	68.3	69.5	55.9	76.2	82.4	68.4
Bi-LSTM$_{+att.+LEX}$ (Raganato et al., 2017a)*	72.0	69.4	66.4	72.4	71.6	57.1	75.6	83.2	69.9
Bi-LSTM$_{+att.+LEX+POS}$ (Raganato et al., 2017a)*	72.0	69.1	66.9	71.5	71.5	57.5	75.0	83.8	69.9
GAS (Linear)*	72.0	70.0	66.7	71.6	71.7	57.4	76.5	83.5	70.1
GAS (Concatenation)*	72.1	70.2	67.0	71.8	72.1	57.2	76.0	84.4	70.3
GAS$_{ext}$ (Linear)*	**72.4**	70.1	67.1	72.1	71.9	**58.1**	76.4	84.7	70.4
GAS$_{ext}$ (Concatenation)*	72.2	**70.5**	**67.2**	**72.6**	**72.2**	57.7	**76.6**	**85.0**	**70.6**

Table 1: F1-score (%) for fine-grained English all-words WSD on the test sets. **Bold** font indicates best systems. The * represents the neural network models using external knowledge. The fives blocks list the MFS baseline, two knowledge-based systems, two supervised systems (feature-based), three neural-based systems and our models, respectively.

gloss information via its semantic relations can help to WSD.

- **Babelfy**: Moro et al. (2014) exploits the semantic network structure from BabelNet and builds a unified graph-based architecture for WSD and Entity Linking.

4.3.2 Supervised Systems

The supervised systems mentioned in this paper refers to traditional feature-based systems which train a dedicated classifier for every word individually (*word expert*).

- **IMS**: Zhi and Ng (2010) selects a linear Support Vector Machine (SVM) as its classifier and makes use of a set of features surrounding the target word within a limited window, such as POS tags, local words and local collocations.

- **IMS$_{+emb}$**: Iacobacci et al. (2016) selects IMS as the underlying framework and makes use of word embeddings as features which makes it hard to beat in most of WSD datasets.

4.3.3 Neural-based Systems

Neural-based systems aim to build an end-to-end unified neural network for all the polysemous words in texts.

- **Bi-LSTM**: Kågebäck and Salomonsson (2016) leverages a bidirectional LSTM network which shares model parameters among all words. Note that this model is equivalent to our model if we remove the gloss module and memory module of GAS.

- **Bi-LSTM$_{+att.+LEX}$** and its variant **Bi-LSTM$_{+att.+LEX+POS}$**: Raganato et al. (2017a) transfers WSD into a sequence learning task and propose a multi-task learning framework for WSD, POS tagging and coarse-grained semantic labels (LEX). These two models have used the external knowledge, for the LEX is based on lexicographer files in WordNet.

Moreover, we introduce **MFS** baseline, which simply selects the most frequent sense in the training data set.

4.4 Results and Discussion

4.4.1 English all-words results

In this section, we show the performance of our proposed model in the English all-words task. Table 1 shows the F1-score results on the four test sets mentioned in Section 4.1. The systems in the first four blocks are implemented by Raganato et al. (2017a,b) except for the single Bi-LSTM model. The last block lists the performance of our proposed model GAS and its variant GAS$_{ext}$ which extends the gloss module in GAS.

GAS and GAS$_{ext}$ achieves the state-of-the-art performance on the concatenation of all test datasets. Although there is no one system always performs best on all the test sets [10], we can find that GAS$_{ext}$ with *concatenation* memory updating strategy achieves the best results **70.6** on the concatenation of the four test datasets. Compared with other three neural-based methods in the

[10] Because the source of the four datasets are extremely different which belongs to different domains.

Context: He **plays** a pianist in the film					
Glosses	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
g_1: participate in games or sport	🟦				
g_2: perform music on a instrument	🟦	🟦	🟦	🟦	🟦
g_3: act a role or part	🟦	🟦	🟦	🟦	🟦

Table 2: An example of attention weights in the memory module within 5 passes. Darker colors mean that the attention weight is higher. Case studies show that the proposed multi-pass operation can recognize the correct sense by enlarging the attention gap between correct senses and incorrect ones.

Pass	**SE2**	**SE3**	**SE13**	**SE15**	**ALL**
1	71.6	70.3	67.0	72.5	70.3
2	71.9	70.2	67.1	**72.8**	70.4
3	**72.2**	**70.5**	**67.2**	72.6	**70.6**
4	72.1	70.4	**67.2**	72.4	70.5
5	72.0	70.4	67.1	71.5	70.3

Table 3: F1-score (%) of different passes from 1 to 5 on the test data sets. It shows that appropriate number of passes can boost the performance as well as avoid over-fitting of the model.

fourth block, we can find that our best model outperforms the previous best neural network models (Raganato et al., 2017a) on every individual test set. The IMS$_{+emb}$, which trains a dedicated classifier for each word individually (*word expert*) with massive manual designed features including word embeddings, is hard to beat for neural networks models. However, our best model can also beat IMS$_{+emb}$ on the SE3, SE13 and SE15 test sets.

Incorporating glosses into neural WSD can greatly improve the performance and extending the original gloss can further boost the results. Compared with the Bi-LSTM baseline which only uses labeled data, our proposed model greatly improves the WSD task by **2.2%** F1-score with the help of gloss knowledge. Furthermore, compared with the GAS which only uses original gloss as the background knowledge, GAS$_{ext}$ can further improve the performance with the help of the extended glosses through the semantic relations. This proves that incorporating extended glosses through its hypernyms and hyponyms into the neural network models can boost the performance for WSD.

4.4.2 Multiple Passes Analysis

To better illustrate the influence of multiple passes, we give an example in Table 2. Consider the situation that we meet an unknown word **x** [11], we look up from the dictionary and find three word senses and their glosses corresponding to **x**.

We try to figure out the correct meaning of **x** according to its context and glosses of different word senses by the proposed memory module. In the first pass, the first sense is excluded, for there are no relevance between the context and g_1. But the g_2 and g_3 may need repeated deliberation, for word *pianist* is similar to the word *music* and *role* in the two glosses. By re-reading the context and gloss information of the target word in the following passes, the correct word sense g_3 attracts much more attention than the other two senses. Such re-reading process can be realized by multi-pass operation in the memory module.

Furthermore, Table 3 shows the effectiveness of multi-pass operation in the memory module. It shows that multiple passes operation performs better than one pass, though the improvement is not significant. The reason of this phenomenon is that for most target words, one main word sense accounts for the majority of their appearances. Therefore, in most circumstances, one-pass inference can lead to the correct word senses. Case studies in Table 2 show that the proposed multi-pass inference can help to recognize the infrequent senses like the third sense for word *play*. In Table 3, with the increasing number of passes, the F1-score increases. However, when the number of passes is larger than 3, the F1-score stops increasing or even decreases due to over-fitting. It shows that appropriate number of passes can boost the performance as well as avoid over-fitting of the model.

5 Conclusions and Future Work

In this paper, we seek to address the problem of integrating the glosses knowledge of the ambiguous word into a neural network for WSD. We further extend the gloss information through its semantic relations in WordNet to better infer the context. In

[11]**x** refers to word *play* in reality.

this way, we not only make use of labeled context data but also exploit the background knowledge to disambiguate the word sense. Results on four English all-words WSD data sets show that our best model outperforms the existing methods.

There is still one challenge left for the future. We just extract the gloss, missing the structural properties or graph information of lexical resources. In the next step, we will consider integrating the rich structural information into the neural network for Word Sense Disambiguation.

Acknowledgments

We thank the Lei Sha, Jiwei Tan, Jianmin Zhang and Junbing Liu for their instructive suggestions and invaluable help. The research work is supported by the National Science Foundation of China under Grant No. 61772040 and No. 61751201. The contact authors are Baobao Chang and Zhifang Sui.

References

Eneko Agirre, Oier Lpez De Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics* 40(1):57–84.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 136–145.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Ijcai*. volume 3, pages 805–810.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Roceedings of COLING 2014, the International Conference on Computational Linguistics: Technical Papers*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A unified multilingual semantic representation of concepts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 741–751.

T. Chen, R. Xu, Y. He, and X. Wang. 2015. Improving distributed representation of word sense via wordnet gloss composition and context clustering. *Atmospheric Measurement Techniques* 4(3):5211–5251.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Conference on Empirical Methods in Natural Language Processing*. pages 1025–1035.

Yun Nung Chen, Dilek Hakkani-Tr, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *The Meeting of the International Speech Communication Association*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Qin Lu, and Jiachen Du. 2017. A question answering approach to emotion cause extraction. *arXiv preprint arXiv:1708.05482* .

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *The Meeting of the Association for Computational Linguistics*.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries:how to tell a pine cone from an ice cream cone. In *Acm Special Interest Group for Design of Communication*. pages 24–26.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Learning word sense embeddings from word sense definitions .

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*. pages 2237–2243.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation:a survey. *Acm Computing Surveys* 41(2):1–69.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *Conference on Empirical Methods in Natural Language Processing*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proc. of EACL*. pages 99–110.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127* .

Lei Sha, Feng Qian, and Zhifang Sui. 2017. Will repeated reading benefit natural language understanding? In *National CCF Conference on Natural Language Processing and Chinese Computing*. pages 366–379.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015a. End-to-end memory networks. *Computer Science* .

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015b. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding .

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. pages 2397–2406.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory pages 1364–1373.

Zhong Zhi and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*. pages 78–83.

Bilingual Sentiment Embeddings: Joint Projection of Sentiment Across Languages

Jeremy Barnes, Roman Klinger, and **Sabine Schulte im Walde**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
{barnesjy,klinger,schulte}@ims.uni-stuttgart.de

Abstract

Sentiment analysis in low-resource languages suffers from a lack of annotated corpora to estimate high-performing models. Machine translation and bilingual word embeddings provide some relief through cross-lingual sentiment approaches. However, they either require large amounts of parallel data or do not sufficiently capture sentiment information. We introduce Bilingual Sentiment Embeddings (BLSE), which jointly represent sentiment information in a source and target language. This model only requires a small bilingual lexicon, a source-language corpus annotated for sentiment, and monolingual word embeddings for each language. We perform experiments on three language combinations (Spanish, Catalan, Basque) for sentence-level cross-lingual sentiment classification and find that our model significantly outperforms state-of-the-art methods on four out of six experimental setups, as well as capturing complementary information to machine translation. Our analysis of the resulting embedding space provides evidence that it represents sentiment information in the resource-poor target language without any annotated data in that language.

1 Introduction

Cross-lingual approaches to sentiment analysis are motivated by the lack of training data in the vast majority of languages. Even languages spoken by several million people, such as Catalan, often have few resources available to perform sentiment analysis in specific domains. We therefore aim to harness the knowledge previously collected in resource-rich languages.

Previous approaches for cross-lingual sentiment analysis typically exploit machine translation based methods or multilingual models. Machine translation (MT) can provide a way to transfer sentiment information from a resource-rich to resource-poor languages (Mihalcea et al., 2007; Balahur and Turchi, 2014). However, MT-based methods require large parallel corpora to train the translation system, which are often not available for under-resourced languages.

Examples of multilingual methods that have been applied to cross-lingual sentiment analysis include domain adaptation methods (Prettenhofer and Stein, 2011), delexicalization (Almeida et al., 2015), and bilingual word embeddings (Mikolov et al., 2013; Hermann and Blunsom, 2014; Artetxe et al., 2016). These approaches however do not incorporate enough sentiment information to perform well cross-lingually, as we will show later.

We propose a novel approach to incorporate sentiment information in a model, which does not have these disadvantages. *Bilingual Sentiment Embeddings* (BLSE) are embeddings that are jointly optimized to represent both (a) semantic information in the source and target languages, which are bound to each other through a small bilingual dictionary, and (b) sentiment information, which is annotated on the source language only. We only need three resources: (i) a comparably small bilingual lexicon, (ii) an annotated sentiment corpus in the resource-rich language, and (iii) monolingual word embeddings for the two involved languages.

We show that our model outperforms previous state-of-the-art models in nearly all experimental settings across six benchmarks. In addition, we offer an in-depth analysis and demonstrate that our model is aware of sentiment. Finally, we provide a qualitative analysis of the joint bilingual sentiment space. Our implementation is publicly available at https://github.com/jbarnesspain/blse.

2 Related Work

Machine Translation: Early work in cross-lingual sentiment analysis found that machine translation (MT) had reached a point of maturity that enabled the transfer of sentiment across languages. Researchers translated sentiment lexicons (Mihalcea et al., 2007; Meng et al., 2012) or annotated corpora and used word alignments to project sentiment annotation and create target-language annotated corpora (Banea et al., 2008; Duh et al., 2011; Demirtas and Pechenizkiy, 2013; Balahur and Turchi, 2014).

Several approaches included a multi-view representation of the data (Banea et al., 2010; Xiao and Guo, 2012) or co-training (Wan, 2009; Demirtas and Pechenizkiy, 2013) to improve over a naive implementation of machine translation, where only the translated data is used. There are also approaches which only require parallel data (Meng et al., 2012; Zhou et al., 2016; Rasooli et al., 2017), instead of machine translation.

All of these approaches, however, require large amounts of parallel data or an existing high quality translation tool, which are not always available. A notable exception is the approach proposed by Chen et al. (2016), an adversarial deep averaging network, which trains a joint feature extractor for two languages. They minimize the difference between these features across languages by learning to fool a language discriminator, which requires no parallel data. It does, however, require large amounts of unlabeled data.

Bilingual Embedding Methods: Recently proposed bilingual embedding methods (Hermann and Blunsom, 2014; Chandar et al., 2014; Gouws et al., 2015) offer a natural way to bridge the language gap. These particular approaches to bilingual embeddings, however, require large parallel corpora in order to build the bilingual space, which are not available for all language combinations.

An approach to create bilingual embeddings that has a less prohibitive data requirement is to create monolingual vector spaces and then learn a projection from one to the other. Mikolov et al. (2013) find that vector spaces in different languages have similar arrangements. Therefore, they propose a linear projection which consists of learning a rotation and scaling matrix. Artetxe et al. (2016, 2017) improve upon this approach by requiring the projection to be orthogonal, thereby preserving the monolingual quality of the original word vectors.

Given source embeddings S, target embeddings T, and a bilingual lexicon L, Artetxe et al. (2016) learn a projection matrix W by minimizing the square of Euclidean distances

$$\arg\min_{W} \sum_{i} ||S'W - T'||_F^2 \,, \tag{1}$$

where $S' \in S$ and $T' \in T$ are the word embedding matrices for the tokens in the bilingual lexicon L. This is solved using the Moore-Penrose pseudoinverse $S'^{+} = (S'^{T} S')^{-1} S'^{T}$ as $W = S'^{+} T'$, which can be computed using SVD. We refer to this approach as ARTETXE.

Gouws and Søgaard (2015) propose a method to create a pseudo-bilingual corpus with a small task-specific bilingual lexicon, which can then be used to train bilingual embeddings (BARISTA). This approach requires a monolingual corpus in both the source and target languages and a set of translation pairs. The source and target corpora are concatenated and then every word is randomly kept or replaced by its translation with a probability of 0.5. Any kind of word embedding algorithm can be trained with this pseudo-bilingual corpus to create bilingual word embeddings.

These last techniques have the advantage of requiring relatively little parallel training data while taking advantage of larger amounts of monolingual data. However, they are not optimized for sentiment.

Sentiment Embeddings: Maas et al. (2011) first explored the idea of incorporating sentiment information into semantic word vectors. They proposed a topic modeling approach similar to latent Dirichlet allocation in order to collect the semantic information in their word vectors. To incorporate the sentiment information, they included a second objective whereby they maximize the probability of the sentiment label for each word in a labeled document.

Tang et al. (2014) exploit distantly annotated tweets to create Twitter sentiment embeddings. To incorporate distributional information about tokens, they use a hinge loss and maximize the likelihood of a true n-gram over a corrupted n-gram. They include a second objective where they classify the polarity of the tweet given the true n-gram. While these techniques have proven useful, they are not easily transferred to a cross-lingual setting.

Zhou et al. (2015) create bilingual sentiment embeddings by translating all source data to the target language and vice versa. This requires the existence of a machine translation system, which is a prohibitive assumption for many under-resourced languages, especially if it must be open and freely accessible. This motivates approaches which can use smaller amounts of parallel data to achieve similar results.

3 Model

In order to project not only semantic similarity and relatedness but also sentiment information to our target language, we propose a new model, namely *Bilingual Sentiment Embeddings* (BLSE), which jointly learns to predict sentiment and to minimize the distance between translation pairs in vector space. We detail the projection objective in Section 3.1, the sentiment objective in Section 3.2, and the full objective in Section 3.3. A sketch of the model is depicted in Figure 1.

3.1 Cross-lingual Projection

We assume that we have two precomputed vector spaces $S = \mathbb{R}^{v \times d}$ and $T = \mathbb{R}^{v' \times d'}$ for our source and target languages, where v (v') is the length of the source vocabulary (target vocabulary) and d (d') is the dimensionality of the embeddings. We also assume that we have a bilingual lexicon L of length n which consists of word-to-word translation pairs $L = \{(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)\}$ which map from source to target.

In order to create a mapping from both original vector spaces S and T to shared sentiment-informed bilingual spaces $\mathbf{z}$ and $\hat{\mathbf{z}}$, we employ two linear projection matrices, M and M'. During training, for each translation pair in L, we first look up their associated vectors, project them through their associated projection matrix and finally minimize the mean squared error of the two projected vectors. This is very similar to the approach taken by Mikolov et al. (2013), but includes an additional target projection matrix.

The intuition for including this second matrix is that a single projection matrix does not support the transfer of sentiment information from the source language to the target language. Without M', any signal coming from the sentiment classifier (see Section 3.2) would have no affect on the target embedding space T, and optimizing M to predict sentiment and projection would only be detrimental to classification of the target language. We analyze this further in Section 6.3. Note that in this configuration, we do not need to update the original vector spaces, which would be problematic with such small training data.

The projection quality is ensured by minimizing the mean squared error[1][2]

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{z_i} - \hat{\mathbf{z}}_\mathbf{i})^2 , \qquad (2)$$

where $\mathbf{z_i} = S_{s_i} \cdot M$ is the dot product of the embedding for source word s_i and the source projection matrix and $\hat{\mathbf{z}}_\mathbf{i} = T_{t_i} \cdot M'$ is the same for the target word t_i.

3.2 Sentiment Classification

We add a second training objective to optimize the projected source vectors to predict the sentiment of source phrases. This inevitably changes the projection characteristics of the matrix M, and consequently M' and encourages M' to learn to predict sentiment without any training examples in the target language.

To train M to predict sentiment, we require a source-language corpus $C_{\text{source}} = \{(x_1, y_1), (x_2, y_2), \dots, (x_i, y_i)\}$ where each sentence x_i is associated with a label y_i.

For classification, we use a two-layer feed-forward averaging network, loosely following Iyyer et al. (2015)[3]. For a sentence x_i we take the word embeddings from the source embedding S and average them to $\mathbf{a}_i \in \mathbb{R}^d$. We then project this vector to the joint bilingual space $\mathbf{z}_i = \mathbf{a}_i \cdot M$. Finally, we pass $\mathbf{z}_i$ through a softmax layer P to get our prediction $\hat{y}_i = \text{softmax}(\mathbf{z}_i \cdot P)$.

To train our model to predict sentiment, we minimize the cross-entropy error of our predictions

$$H = -\sum_{i=1}^{n} y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i) \, . \quad (3)$$

3.3 Joint Learning

In order to jointly train both the projection component and the sentiment component, we combine the two loss functions to optimize the parameter

[1] We omit parameters in equations for better readability.

[2] We also experimented with cosine distance, but found that it performed worse than Euclidean distance.

[3] Our model employs a linear transformation after the averaging layer instead of including a non-linearity function. We choose this architecture because the weights M and M' are also used to learn a linear cross-lingual projection.

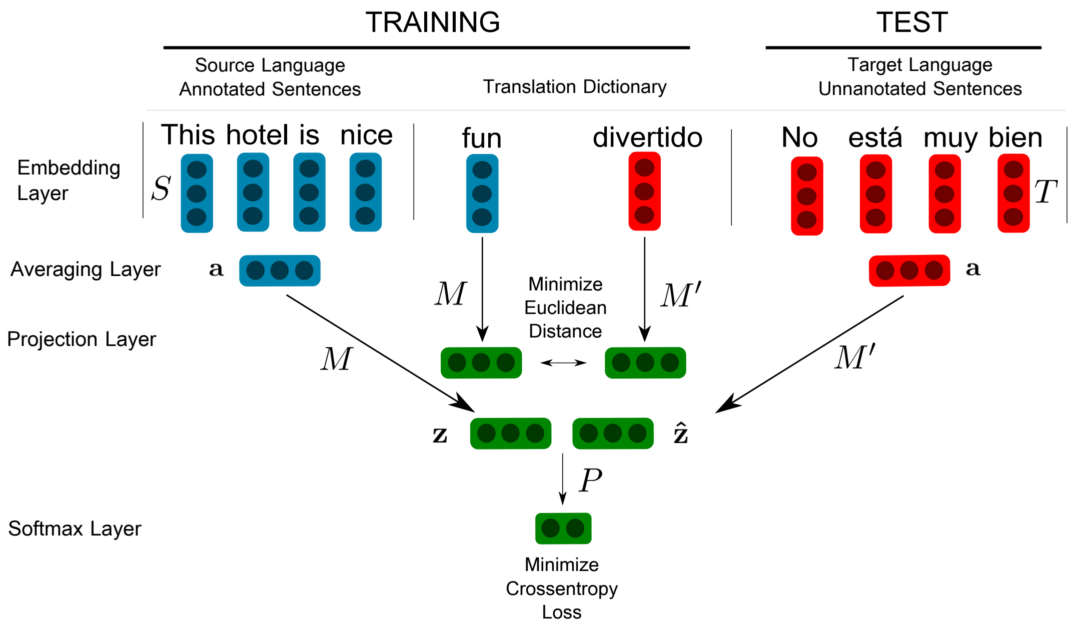


Figure 1: Bilingual Sentiment Embedding Model (BLSE)

		EN	ES	CA	EU
Binary	$+$	1258	1216	718	956
	$-$	473	256	467	173
	Total	1731	1472	1185	1129
4-class	$++$	379	370	256	384
	$+$	879	846	462	572
	$-$	399	218	409	153
	$--$	74	38	58	20
	Total	1731	1472	1185	1129

Table 1: Statistics for the OpeNER English (EN) and Spanish (ES) as well as the MultiBooked Catalan (CA) and Basque (EU) datasets.

matrices M, M', and P by

$$J = \sum_{(x,y)\in C_{\text{source}}} \sum_{(s,t)\in L} \alpha H(x,y) + (1-\alpha)\cdot \text{MSE}(s,t)\,, \tag{4}$$

where α is a hyperparameter that weights sentiment loss vs. projection loss.

3.4 Target-language Classification

For inference, we classify sentences from a target-language corpus C_{target}. As in the training procedure, for each sentence, we take the word embeddings from the target embeddings T and average them to $\mathbf{a}_i \in \mathbb{R}^d$. We then project this vector to the joint bilingual space $\hat{\mathbf{z}}_i = \mathbf{a}_i \cdot M'$. Finally, we pass $\hat{\mathbf{z}}_i$ through a softmax layer P to get our prediction $\hat{y}_i = \text{softmax}(\hat{\mathbf{z}}_i \cdot P)$.

	Spanish	Catalan	Basque
Sentences	23 M	9.6 M	0.7 M
Tokens	610 M	183 M	25 M
Embeddings	0.83 M	0.4 M	0.14 M

Table 2: Statistics for the Wikipedia corpora and monolingual vector spaces.

4 Datasets and Resources

4.1 OpeNER and MultiBooked

To evaluate our proposed model, we conduct experiments using four benchmark datasets and three bilingual combinations. We use the OpeNER English and Spanish datasets (Agerri et al., 2013) and the MultiBooked Catalan and Basque datasets (Barnes et al., 2018). All datasets contain hotel reviews which are annotated for aspect-level sentiment analysis. The labels include *Strong Negative* ($--$), *Negative* ($-$), *Positive* ($+$), and *Strong Positive* ($++$). We map the aspect-level annotations to sentence level by taking the most common label and remove instances of mixed polarity. We also create a binary setup by combining the strong and weak classes. This gives us a total of six experiments. The details of the sentence-level datasets are summarized in Table 1. For each of the experi-

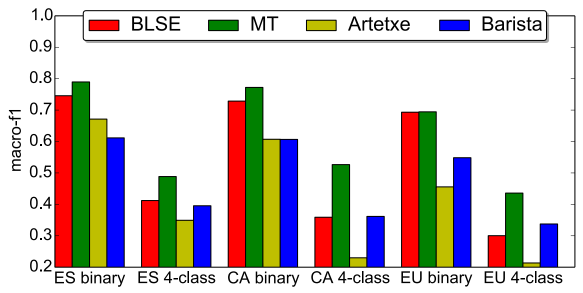


Figure 2: Binary and four class macro F_1 on Spanish (ES), Catalan (CA), and Basque (EU).

ments, we take 70 percent of the data for training, 20 percent for testing and the remaining 10 percent are used as development data for tuning.

4.2 Monolingual Word Embeddings

For BLSE, ARTETXE, and MT, we require monolingual vector spaces for each of our languages. For English, we use the publicly available GoogleNews vectors[4]. For Spanish, Catalan, and Basque, we train skip-gram embeddings using the Word2Vec toolkit[4] with 300 dimensions, subsampling of 10^{-4}, window of 5, negative sampling of 15 based on a 2016 Wikipedia corpus[5] (sentence-split, tokenized with IXA pipes (Agerri et al., 2014) and lowercased). The statistics of the Wikipedia corpora are given in Table 2.

4.3 Bilingual Lexicon

For BLSE, ARTETXE, and BARISTA, we also require a bilingual lexicon. We use the sentiment lexicon from Hu and Liu (2004) (to which we refer in the following as Bing Liu) and its translation into each target language. We translate the lexicon using Google Translate and exclude multi-word expressions.[6] This leaves a dictionary of 5700 translations in Spanish, 5271 in Catalan, and 4577 in Basque. We set aside ten percent of the translation pairs as a development set in order to check that the distances between translation pairs not seen during training are also minimized during training.

[4]https://code.google.com/archive/p/word2vec/
[5]http://attardi.github.io/wikiextractor/
[6]Note that we only do that for convenience. Using a machine translation service to generate this list could easily be replaced by a manual translation, as the lexicon is comparably small.

5 Experiments

5.1 Setting

We compare BLSE (Sections 3.1–3.3) to ARTETXE (Section 2) and BARISTA (Section 2) as baselines, which have similar data requirements and to machine translation (MT) and monolingual (MONO) upper bounds which request more resources. For all models (MONO, MT, ARTETXE, BARISTA), we take the average of the word embeddings in the source-language training examples and train a linear SVM[7]. We report this instead of using the same feed-forward network as in BLSE as it is the stronger upper bound. We choose the parameter c on the target language development set and evaluate on the target language test set.

Upper Bound MONO. We set an empirical upper bound by training and testing a linear SVM on the target language data. As mentioned in Section 5.1, we train the model on the averaged embeddings from target language training data, tuning the c parameter on the development data. We test on the target language test data.

Upper Bound MT. To test the effectiveness of machine translation, we translate all of the sentiment corpora from the target language to English using the Google Translate API[8]. Note that this approach is not considered a baseline, as we assume not to have access to high-quality machine translation for low-resource languages of interest.

Baseline ARTETXE. We compare with the approach proposed by Artetxe et al. (2016) which has shown promise on other tasks, such as word similarity. In order to learn the projection matrix W, we need translation pairs. We use the same word-to-word bilingual lexicon mentioned in Section 3.1. We then map the source vector space S to the bilingual space $\hat{S} = SW$ and use these embeddings.

Baseline BARISTA. We also compare with the approach proposed by Gouws and Søgaard (2015). The bilingual lexicon used to create the pseudo-bilingual corpus is the same word-to-word bilingual lexicon mentioned in Section 3.1. We follow the authors' setup to create the pseudo-bilingual corpus. We create bilingual embeddings by training skip-gram embeddings using the Word2Vec toolkit on the pseudo-bilingual corpus using the same parameters from Section 4.2.

[7]LinearSVC implementation from scikit-learn.
[8]https://translate.google.com

Our method: BLSE. We implement our model BLSE in Pytorch (Paszke et al., 2016) and initialize the word embeddings with the pretrained word embeddings S and T mentioned in Section 4.2. We use the word-to-word bilingual lexicon from Section 4.3, tune the hyperparameters α, training epochs, and batch size on the target development set and use the best hyperparameters achieved on the development set for testing. ADAM (Kingma and Ba, 2014) is used in order to minimize the average loss of the training batches.

			Binary			4-class		
			ES	CA	EU	ES	CA	EU
Upper Bounds	MONO	P	75.0	79.0	74.0	55.2	50.0	48.3
		R	72.3	79.6	67.4	42.8	50.9	46.5
		F_1	73.5	79.2	69.8	45.5	49.9	47.1
	MT	P	**82.3**	78.0	75.6	51.8	**58.9**	43.6
		R	76.6	76.8	66.5	48.5	50.5	45.2
		F_1	79.0	77.2	69.4	48.8	52.7	43.6
	BLSE	P	72.1	_**72.8_	_**67.5_	**_**60.0_**	_38.1_	_*42.5_
		R	**_**80.1_**	_**73.0_	**_**72.7_**	_*43.4_	38.1	_37.4_
		F_1	_**74.6_	_**72.9_	_**69.3_	_*41.2_	35.9	30.0
Baselines	Artetxe	P	75.0	60.1	42.2	40.1	21.6	30.0
		R	64.3	61.2	49.5	36.9	29.8	35.7
		F_1	67.1	60.7	45.6	34.9	23.0	21.3
	Barista	P	64.7	65.3	55.5	44.1	36.4	34.1
		R	59.8	61.2	54.5	37.9	38.5	34.3
		F_1	61.2	60.1	54.8	39.5	36.2	33.8
Ensemble	Artetxe	P	65.3	63.1	70.4	43.5	46.5	50.1
		R	61.3	63.3	64.3	44.1	48.7	50.7
		F_1	62.6	63.2	66.4	43.8	47.6	49.9
	Barista	P	60.1	63.4	50.7	48.3	52.8	50.8
		R	55.5	62.3	50.4	46.6	53.7	49.8
		F_1	56.0	62.5	49.8	47.1	53.0	47.8
	BLSE	P	79.5	**84.7**	**80.9**	49.5	54.1	**50.3**
		R	**78.7**	**85.5**	69.9	**51.2**	**53.9**	**51.4**
		F_1	**80.3**	**85.0**	**73.5**	**50.3**	**53.9**	**50.5**

Table 3: Precision (P), Recall (R), and macro F_1 of four models trained on English and tested on Spanish (ES), Catalan (CA), and Basque (EU). The **bold** numbers show the best results for each metric per column and the *highlighted* numbers show where BLSE is better than the other projection methods, ARTETXE and BARISTA (** $p < 0.01$, * $p < 0.05$).

Ensembles We create an ensemble of MT and each projection method (BLSE, ARTETXE, BARISTA) by training a random forest classifier on the predictions from MT and each of these approaches. This allows us to evaluate to what extent each projection model adds complementary information to the machine translation approach.

5.2 Results

In Figure 2, we report the results of all four methods. Our method outperforms the other projection methods (the baselines ARTETXE and BARISTA) on four of the six experiments substantially. It performs only slightly worse than the more resource-costly upper bounds (MT and MONO). This is especially noticeable for the binary classification task, where BLSE performs nearly as well as machine translation and significantly better than the other methods. We perform approximate randomization tests (Yeh, 2000) with 10,000 runs and highlight the results that are statistically significant (**$p <$ 0.01, *$p < 0.05$) in Table 3.

In more detail, we see that MT generally performs better than the projection methods (79–69 F_1 on binary, 52–44 on 4-class). BLSE (75–69 on binary, 41–30 on 4-class) has the best performance of the projection methods and is comparable with MT on the binary setup, with no significant difference on binary Basque. ARTETXE (67–46 on binary, 35–21 on 4-class) and BARISTA (61–55 on binary, 40–34 on 4-class) are significantly worse than BLSE on all experiments except Catalan and Basque 4-class. On the binary experiment, ARTETXE outperforms BARISTA on Spanish (67.1 vs. 61.2) and Catalan (60.7 vs. 60.1) but suffers more than the other methods on the four-class experiments, with a maximum F_1 of 34.9. BARISTA

Model		voc	mod	neg	know	other	*total*
MT	bi	49	26	19	14	5	**113**
	4	147	94	19	21	12	**293**
ARTETXE	bi	80	44	27	14	7	**172**
	4	182	141	19	24	19	**385**
BARISTA	bi	89	41	27	20	7	**184**
	4	191	109	24	31	15	**370**
BLSE	bi	67	45	21	15	8	**156**
	4	146	125	29	22	19	**341**

Table 4: Error analysis for different phenomena. See text for explanation of error classes.

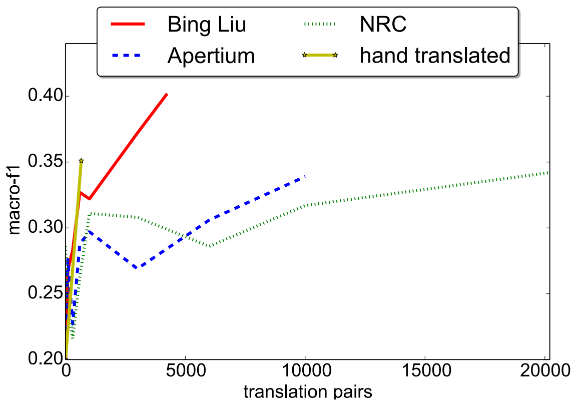


Figure 3: Macro F_1 for translation pairs in the Spanish 4-class setup.

is relatively stable across languages.

ENSEMBLE performs the best, which shows that BLSE adds complementary information to MT. Finally, we note that all systems perform successively worse on Catalan and Basque. This is presumably due to the quality of the word embeddings, as well as the increased morphological complexity of Basque.

6 Model and Error Analysis

We analyze three aspects of our model in further detail: (i) where most mistakes originate, (ii) the effect of the bilingual lexicon, and (iii) the effect and necessity of the target-language projection matrix M'.

6.1 Phenomena

In order to analyze where each model struggles, we categorize the mistakes and annotate all of the test phrases with one of the following error classes: vocabulary (voc), adverbial modifiers (mod), negation (neg), external knowledge (know) or other. Table 4 shows the results.

Vocabulary: The most common way to express sentiment in hotel reviews is through the use of polar adjectives (as in "the room was great) or the mention of certain nouns that are desirable ("it had a pool"). Although this phenomenon has the largest total number of mistakes (an average of 71 per model on binary and 167 on 4-class), it is mainly due to its prevalence. MT performed the best on the test examples which according to the annotation require a correct understanding of the vocabulary (81 F_1 on binary /54 F_1 on 4-class), with BLSE (79/48) slightly worse. ARTETXE (70/35) and BARISTA (67/41) perform significantly worse. This suggests that BLSE is better ARTETXE and BARISTA at transferring sentiment of the most important sentiment bearing words.

Negation: Negation is a well-studied phenomenon in sentiment analysis (Pang et al., 2002; Wiegand et al., 2010; Zhu et al., 2014; Reitan et al., 2015). Therefore, we are interested in how these four models perform on phrases that include the negation of a key element, for example "In general, this hotel isn't bad". We would like our models to recognize that the combination of two negative elements "isn't" and "bad" lead to a *Positive* label.

Given the simple classification strategy, all models perform relatively well on phrases with negation (all reach nearly 60 F_1 in the binary setting). However, while BLSE performs the best on negation in the binary setting (82.9 F_1), it has more problems with negation in the 4-class setting (36.9 F_1).

Adverbial Modifiers: Phrases that are modified by an adverb, *e. g.*, the food was *incredibly* good, are important for the four-class setup, as they often differentiate between the base and *Strong* labels. In the binary case, all models reach more than 55 F_1. In the 4-class setup, BLSE only achieves 27.2 F_1 compared to 46.6 or 31.3 of MT and BARISTA, respectively. Therefore, presumably, our model does currently not capture the semantics of the target adverbs well. This is likely due to the fact that it assigns too much sentiment to functional words (see Figure 6).

External Knowledge Required: These errors are difficult for any of the models to get correct. Many of these include numbers which imply positive or negative sentiment (350 meters from the beach is *Positive* while 3 kilometers from the beach is *Negative*). BLSE performs the best (63.5 F_1) while MT performs comparably well (62.5). BARISTA performs the worst (43.6).

Binary vs. 4-class: All of the models suffer when moving from the binary to 4-class setting; an average of 26.8 in macro F_1 for MT, 31.4 for ARTETXE, 22.2 for BARISTA, and for 36.6 BLSE. The two vector projection methods (ARTETXE and BLSE) suffer the most, suggesting that they are currently more apt for the binary setting.

6.2 Effect of Bilingual Lexicon

We analyze how the number of translation pairs affects our model. We train on the 4-class Spanish setup using the best hyper-parameters from the previous experiment.

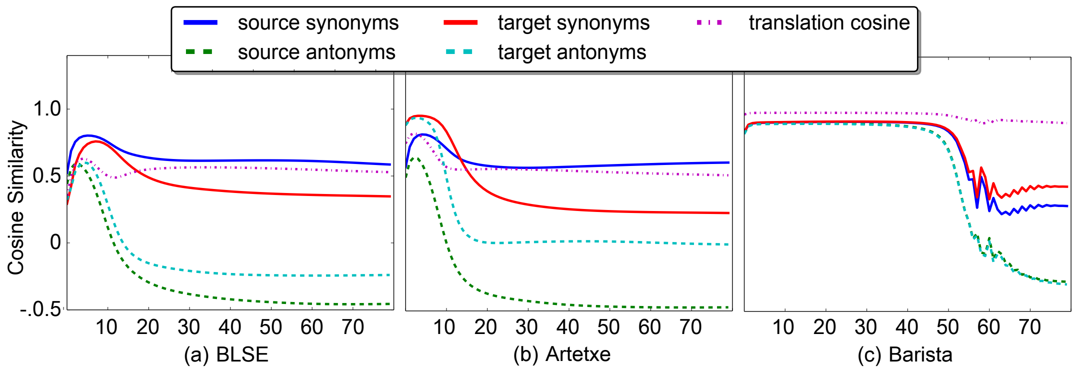


Figure 4: Average cosine similarity between a subsample of translation pairs of same polarity ("sentiment synonyms") and of opposing polarity ("sentiment antonyms") in both target and source languages in each model. The x-axis shows training epochs. We see that BLSE is able to learn that sentiment synonyms should be close to one another in vector space and sentiment antonyms should not.

Research into projection techniques for bilingual word embeddings (Mikolov et al., 2013; Lazaridou et al., 2015; Artetxe et al., 2016) often uses a lexicon of the most frequent 8–10 thousand words in English and their translations as training data. We test this approach by taking the 10,000 word-to-word translations from the Apertium English-to-Spanish dictionary[9]. We also use the Google Translate API to translate the NRC hashtag sentiment lexicon (Mohammad et al., 2013) and keep the 22,984 word-to-word translations. We perform the same experiment as above and vary the amount of training data from 0, 100, 300, 600, 1000, 3000, 6000, 10,000 up to 20,000 training pairs. Finally, we compile a small hand translated dictionary of 200 pairs, which we then expand using target language morphological information, finally giving us 657 translation pairs[10]. The macro F_1 score for the Bing Liu dictionary climbs constantly with the increasing translation pairs. Both the Apertium and NRC dictionaries perform worse than the translated lexicon by Bing Liu, while the expanded hand translated dictionary is competitive, as shown in Figure 3.

While for some tasks, *e. g.*, bilingual lexicon induction, using the most frequent words as translation pairs is an effective approach, for sentiment analysis, this does not seem to help. Using a translated sentiment lexicon, even if it is small, gives better results.

[9]http://www.meta-share.org

[10]The translation took approximately one hour. We can extrapolate that hand translating a sentiment lexicon the size of the Bing Liu lexicon would take no more than 5 hours.

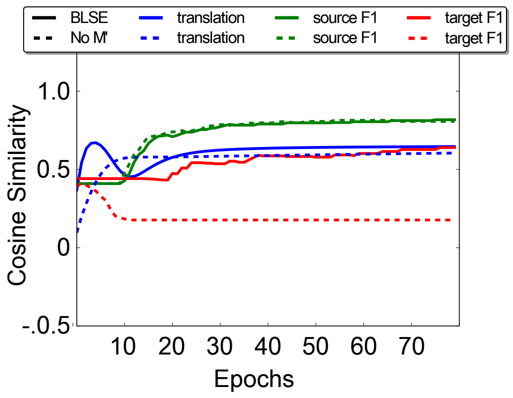


Figure 5: BLSE model (solid lines) compared to a variant without target language projection matrix M' (dashed lines). "Translation" lines show the average cosine similarity between translation pairs. The remaining lines show F_1 scores for the source and target language with both varints of BLSE. The modified model cannot learn to predict sentiment in the target language (red lines). This illustrates the need for the second projection matrix M'.

6.3 Analysis of M'

The main motivation for using two projection matrices M and M' is to allow the original embeddings to remain stable, while the projection matrices have the flexibility to align translations and separate these into distinct sentiment subspaces. To justify this design decision empirically, we perform an experiment to evaluate the actual need for the target language projection matrix M': We create a simplified version of our model without M', using M to project from the source to target and then P to classify sentiment.

The results of this model are shown in Figure 5. The modified model does learn to predict in the source language, but not in the target language. This confirms that M' is necessary to transfer sentiment in our model.

7 Qualitative Analyses of Joint Bilingual Sentiment Space

In order to understand how well our model transfers sentiment information to the target language, we perform two qualitative analyses. First, we collect two sets of 100 positive sentiment words and one set of 100 negative sentiment words. An effective cross-lingual sentiment classifier using embeddings should learn that two positive words should be closer in the shared bilingual space than a positive word and a negative word. We test if BLSE is able to do this by training our model and after every epoch observing the mean cosine similarity between the sentiment synonyms and sentiment antonyms after projecting to the joint space.

We compare BLSE with ARTETXE and BARISTA by replacing the Linear SVM classifiers with the same multi-layer classifier used in BLSE and observing the distances in the hidden layer. Figure 4 shows this similarity in both source and target language, along with the mean cosine similarity between a held-out set of translation pairs and the macro F_1 scores on the development set for both source and target languages for BLSE, BARISTA, and ARTETXE. From this plot, it is clear that BLSE is able to learn that sentiment synonyms should be close to one another in vector space and antonyms should have a negative cosine similarity. While the other models also learn this to some degree, jointly optimizing both sentiment and projection gives better results.

Secondly, we would like to know how well the projected vectors compare to the original space. Our hypothesis is that some relatedness and similarity information is lost during projection. Therefore, we visualize six categories of words in t-SNE (Van der Maaten and Hinton, 2008): positive sentiment words, negative sentiment words, functional words, verbs, animals, and transport.

The t-SNE plots in Figure 6 show that the positive and negative sentiment words are rather clearly separated after projection in BLSE. This indicates that we are able to incorporate sentiment information into our target language without any labeled data in the target language. However, the downside of this is that functional words and transportation words are highly correlated with positive sentiment.

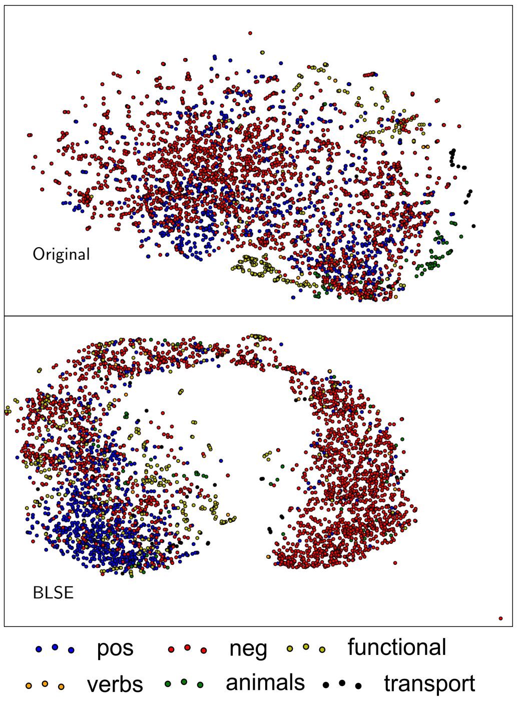


Figure 6: t-SNE-based visualization of the Spanish vector space before and after projection with BLSE. There is a clear separation of positive and negative words after projection, despite the fact that we have used no labeled data in Spanish.

8 Conclusion

We have presented a new model, BLSE, which is able to leverage sentiment information from a resource-rich language to perform sentiment analysis on a resource-poor target language. This model requires less parallel data than MT and performs better than other state-of-the-art methods with similar data requirements, an average of 14 percentage points in F_1 on binary and 4 pp on 4-class cross-lingual sentiment analysis. We have also performed a phenomena-driven error analysis which showed that BLSE is better than ARTETXE and BARISTA at transferring sentiment, but assigns too much sentiment to functional words. In the future, we will extend our model so that it can project multi-word phrases, as well as single words, which could help with negations and modifiers.

Acknowledgements

We thank Sebastian Padó, Sebastian Riedel, Eneko Agirre, and Mikel Artetxe for their conversations and feedback.

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. 2014. Ixa pipeline: Efficient and ready to use multilingual nlp tools. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. pages 3823–3828.

Rodrigo Agerri, Montse Cuadros, Sean Gaines, and German Rigau. 2013. OpeNER: Open polarity enhanced named entity recognition. *Sociedad Española para el Procesamiento del Lenguaje Natural* 51(Septiembre):215–218.

Mariana S. C. Almeida, Claudia Pinto, Helena Figueira, Pedro Mendes, and André F. T. Martins. 2015. Aligning opinions: Cross-lingual opinion mining with dependencies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pages 408–418.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 451–462.

Alexandra Balahur and Marco Turchi. 2014. Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language* 28(1):56–75.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2010. Multilingual subjectivity: Are more languages better? In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. pages 28–36.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. pages 127–135.

Jeremy Barnes, Patrik Lambert, and Toni Badia. 2018. Multibooked: A corpus of basque and catalan hotel reviews annotated for aspect-level sentiment classification. In *Proceedings of 11th Language Resources and Evaluation Conference (LREC'18)*.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 1853–1861.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Q. Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *CoRR* abs/1606.01614. http://arxiv.org/abs/1606.01614.

Erkin Demirtas and Mykola Pechenizkiy. 2013. Cross-lingual polarity detection with machine translation. *Proceedings of the International Workshop on Issues of Sentiment Discovery and Opinion Mining - WISDOM '13* pages 9:1–9:8.

Kevin Duh, Akinori Fujino, and Masaaki Nagata. 2011. Is machine translation ripe for cross-lingual sentiment classification? *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers* 2:429–433.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. *Proceedings of The 32nd International Conference on Machine Learning* pages 748–756.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1386–1390.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 58–68.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*. pages 168–177.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daume III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1681–1691.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* .

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: delving into cross-space mapping for zero-shot learning. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* pages 270–280.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 142–150.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 572–581. http://www.aclweb.org/anthology/P12-1060.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. pages 976–983.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *CoRR* abs/1309.4168. http://arxiv.org/abs/1309.4168.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 79–86.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2016. Pytorch deeplearning framework. http://pytorch.org. Accessed: 2017-08-10.

Peter Prettenhofer and Benno Stein. 2011. Cross-lingual adaptation using structural correspondence learning. *ACM Transactions on Intelligent Systems and Technology* 3(1):1–22.

Mohammad Sadegh Rasooli, Noura Farra, Axinia Radeva, Tao Yu, and Kathleen McKeown. 2017. Cross-lingual sentiment transfer with limited resources. *Machine Translation* .

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. pages 99–108.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1555–1565.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9:2579–2605.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. pages 235–243.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*. pages 60–68.

Min Xiao and Yuhong Guo. 2012. Multi-view adaboost for multilingual subjectivity analysis. In *Proceedings of COLING 2012*. pages 2851–2866.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational linguistics (COLING)*. pages 947–953.

Guangyou Zhou, Zhiyuan Zhu, Tingting He, and Xiaohua Tony Hu. 2016. Cross-lingual sentiment classification with stacked autoencoders. *Knowledge and Information Systems* 47(1):27–44.

HuiWei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pages 430–440.

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 304–313.

Learning Domain-Sensitive and Sentiment-Aware Word Embeddings*

Bei Shi[1], Zihao Fu[1], Lidong Bing[2] and Wai Lam[1]
[1]Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong
[2]Tencent AI Lab, Shenzhen, China
{bshi,zhfu,wlam}@se.cuhk.edu.hk
lyndonbing@tencent.com

Abstract

Word embeddings have been widely used in sentiment classification because of their efficacy for semantic representations of words. Given reviews from different domains, some existing methods for word embeddings exploit sentiment information, but they cannot produce domain-sensitive embeddings. On the other hand, some other existing methods can generate domain-sensitive word embeddings, but they cannot distinguish words with similar contexts but opposite sentiment polarity. We propose a new method for learning domain-sensitive and sentiment-aware embeddings that simultaneously capture the information of sentiment semantics and domain sensitivity of individual words. Our method can automatically determine and produce domain-common embeddings and domain-specific embeddings. The differentiation of domain-common and domain-specific words enables the advantage of data augmentation of common semantics from multiple domains and capture the varied semantics of specific words from different domains at the same time. Experimental results show that our model provides an effective way to learn domain-sensitive and sentiment-aware word embeddings which benefit sentiment classification at both sentence level and lexicon term level.

* This work was partially done when Bei Shi was an intern at Tencent AI Lab. This project is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

1 Introduction

Sentiment classification aims to predict the sentiment polarity, such as "positive" or "negative", over a piece of review. It has been a long-standing research topic because of its importance for many applications such as social media analysis, e-commerce, and marketing (Liu, 2012; Pang et al., 2008). Deep learning has brought in progress in various NLP tasks, including sentiment classification. Some researchers focus on designing RNN or CNN based models for predicting sentence level (Kim, 2014) or aspect level sentiment (Li et al., 2018; Chen et al., 2017; Wang et al., 2016). These works directly take the word embeddings pre-trained for general purpose as initial word representations and may conduct fine tuning in the training process. Some other researchers look into the problem of learning task-specific word embeddings for sentiment classification aiming at solving some limitations of applying general pre-trained word embeddings. For example, Tang et al. (2014b) develop a neural network model to convey sentiment information in the word embeddings. As a result, the learned embeddings are **sentiment-aware** and able to distinguish words with similar syntactic context but opposite sentiment polarity, such as the words "good" and "bad". In fact, sentiment information can be easily obtained or derived in large scale from some data sources (e.g., the ratings provided by users), which allows reliable learning of such sentiment-aware embeddings.

Apart from these words (e.g. "good" and "bad") with consistent sentiment polarity in different contexts, the polarity of some sentiment words is **domain-sensitive**. For example, the word "lightweight" usually connotes a positive sentiment in the electronics domain since a lightweight device is easier to carry. In contrast, in the movie domain, the word "lightweight" usually connotes a negative opinion describing movies that do not invoke deep thoughts among the audience. This observation motivates the study of learning domain-sensitive word representations (Yang et al., 2017; Bollegala et al., 2015, 2014). They basically learn separate embeddings of the same word for different domains. To bridge the semantics of individual embedding spaces, they select a subset of words that are likely to be domain-insensitive and align the dimensions of their embeddings. However, the sentiment information is not exploited in these methods although they intend to tackle the task of sentiment classification.

In this paper, we aim at learning word embeddings that are both domain-sensitive and sentiment-aware. Our proposed method can jointly model the sentiment semantics and domain specificity of words, expecting the learned embeddings to achieve superior performance for the task of sentiment classification. Specifically, our method can automatically determine and produce domain-common embeddings and domain-specific embeddings. Domain-common embeddings represent the fact that the semantics of a word including its sentiment and meaning in different domains are very similar. For example, the words "good" and "interesting" are usually domain-common and convey consistent semantic meanings and positive sentiments in different domains. Thus, they should have similar embeddings across domains. On the other hand, domain-specific word embeddings represent the fact that the sentiments or meanings across domains are different. For example, the word "lightweight" represents different sentiment polarities in the electronics domain and the movie domain. Moreover, some polysemous words have different meanings in different domains. For example, the term "apple" refers to the famous technology company in the electronics domain or a kind of fruit in the food domain.

Our model exploits the information of sentiment labels and context words to distinguish domain-common and domain-specific words. If a word has similar sentiments and contexts across domains, it indicates that the word has common semantics in these domains, and thus it is treated as domain-common. Otherwise, the word is considered as domain-specific. The learning of domain-common embeddings can allow the advantage of data augmentation of common semantics of multiple domains, and meanwhile, domain-specific embeddings allow us to capture the varied semantics of specific words in different domains. Specifically, for each word in the vocabulary, we design a distribution to depict the probability of the word being domain-common. The inference of the probability distribution is conducted based on the observed sentiments and contexts. As mentioned above, we also exploit the information of sentiment labels for the learning of word embeddings that can distinguish words with similar syntactic context but opposite sentiment polarity.

To demonstrate the advantages of our domain-sensitive and sentiment-aware word embeddings, we conduct experiments on four domains, including books, DVSs, electronics, and kitchen appliances. The experimental results show that our model can outperform the state-of-the-art models on the task of sentence level sentiment classification. Moreover, we conduct lexicon term sentiment classification in two common sentiment lexicon sets to evaluate the effectiveness of our sentiment-aware embeddings learned from multiple domains, and it shows that our model outperforms the state-of-the-art models on most domains.

2 Related Works

Traditional vector space models encode individual words using the one-hot representation, namely, a high-dimensional vector with all zeroes except in one component corresponding to that word (Baeza-Yates et al., 1999). Such representations suffer from the curse of dimensionality, as there are many components in these vectors due to the vocabulary size. Another drawback is that semantic relatedness of words cannot be modeled using such representations. To address these shortcomings, Rumelhart et al. (1988) propose to use distributed word representation instead, called word embeddings. Several techniques for generating such representations have been investigated. For example, Bengio et al. propose a neural network architecture for this purpose (Bengio et al., 2003; Bengio, 2009). Later, Mikolov et al. (2013) propose two methods that are considerably more efficient, namely skip-gram and CBOW. This work has made it possible to learn word embeddings from large data sets, which has led to the current popularity of word embeddings. Word embedding models have been applied to many tasks, such as named entity recognition (Turian et al., 2010), word sense disambiguation (Collobert et al., 2011; Iacobacci et al., 2016; Zhang and Hasan, 2017; Dave et al., 2018), parsing (Roth and Lapata, 2016), and document classification (Tang et al., 2014a,b; Shi et al., 2017).

Sentiment classification has been a long-standing research topic (Liu, 2012; Pang et al., 2008; Chen et al., 2017; Moraes et al., 2013). Given a review, the task aims at predicting the sentiment polarity on the sentence level (Kim, 2014) or the aspect level (Li et al., 2018; Chen et al., 2017). Supervised learning algorithms have been widely used in sentiment classification (Pang et al., 2002). People usually use different expressions of sentiment semantics in different domains. Due to the mismatch between domain-specific words, a sentiment classifier trained in one domain may not work well when it is directly applied to other domains. Thus cross-domain sentiment classification algorithms have been explored (Pan et al., 2010; Li et al., 2009; Glorot et al., 2011). These works usually find common feature spaces across domains and then share learned parameters from the source domain to the target domain. For example, Pan et al. (2010) propose a spectral feature alignment algorithm to align words from different domains into unified clusters. Then the clusters can be used to reduce the gap between words of the two domains, which can be used to train sentiment classifiers in the target domain. Compared with the above works, our model focuses on learning both domain-common and domain-specific embeddings given reviews from all the domains instead of only transferring the common semantics from the source domain to the target domain.

Some researchers have proposed some methods to learn task-specific word embeddings for sentiment classification (Tang et al., 2014a,b). Tang et al. (2014b) propose a model named SSWE to learn sentiment-aware embedding via incorporating sentiment polarity of texts in the loss functions of neural networks. Without the consideration of varied semantics of domain-specific words in different domains, their model cannot learn sentiment-aware embeddings across multiple domains. Some works have been proposed to learn word representations considering multiple domains (Yang et al., 2017; Bach et al., 2016; Bollegala et al., 2015). Most of them learn separate embeddings of the same word for different domains. Then they choose pivot words according to frequency-based statistical measures to bridge the semantics of individual embedding spaces. A regularization formulation enforcing that word representations of pivot words should be similar in different domains is added into the original word embedding framework. For example, Yang et al. (2017) use Sørensen-Dice coefficient (Sørensen, 1948) for detecting pivot words and learn word representations across domains. Even though they evaluate the model via the task of sentiment classification, sentiment information associated with the reviews are not considered in the learned embeddings. Moreover, the selection of pivot words is according to frequency-based statistical measures in the above works. In our model, the domain-common words are jointly determined by sentiment information and context words.

3 Model Description

We propose a new model, named DSE, for learning **D**omain-sensitive and **S**entiment-aware word **E**mbeddings. For presentation clarity, we describe DSE based on two domains. Note that it can be easily extended for more than two domains, and we remark on how to extend near the end of this section.

3.1 Design of Embeddings

We assume that the input consists of text reviews of two domains, namely $\mathcal{D}^p$ and $\mathcal{D}^q$. Each review r in $\mathcal{D}^p$ and $\mathcal{D}^q$ is associated with a sentiment label y which can take on the value of 1 and 0 denoting that the sentiment of the review is positive and negative respectively.

In our DSE model, each word w in the whole vocabulary Λ is associated with a domain-common vector U_w^c and two domain-specific vectors, namely U_w^p specific to the domain p and U_w^q specific to the domain q. The dimension of these vectors is d. The design of U_w^c, U_w^p and U_w^q reflects one characteristic of our model: allowing a word to have different semantics across different domains. The semantic of each word includes not only the semantic meaning but also the sentiment orientation of the word. If the semantic of w is consistent in the domains p and q, we use the vector U_w^c for both domains. Otherwise, w is represented by U_w^p and U_w^q for p and q respectively.

In traditional cross-domain word embedding methods (Yang et al., 2017; Bollegala et al., 2015, 2016), each word is represented by different vectors in different domains without differentiation of domain-common and domain-specific words. In contrast to these methods, for each word w, we use a latent variable z_w to depict its domain commonality. When $z_w = 1$, it means that w is common in both domains. Otherwise, w is specific to the domain p or the domain q.

In the standard skip-gram model (Mikolov et al., 2013), the probability of predicting the context words is only affected by the relatedness with the target words. In our DSE model, predicting the context words also depends on the domain-commonality of the target word, i.e z_w. For example, assume that there are two domains, e.g. the electronics domain and the movie domain. If $z_w = 1$, it indicates a high probability of generating some domain-common words such as "good", "bad" or "satisfied". Otherwise, the domain-specific words are more likely to be generated such as "reliable", "cheap" or "compacts" for the electronics domain. For a word w, we assume that the probability of predicting the context word w_t is formulated as follows:

$$p(w_t|w) = \sum_{k\in\{0,1\}} p(w_t|w, z_w = k)p(z_w = k) \tag{1}$$

If w is a domain-common word without differentiating p and q, the probability of predicting w_t can be defined as:

$$p(w_t|w, z_w = 1) = \frac{\exp(U_w^c \cdot V_{w_t})}{\sum_{w'\in\Lambda} \exp(U_w^c \cdot V_{w'})} \tag{2}$$

where Λ is the whole vocabulary and $V_{w'}$ is the output vector of the word w'.

If w is a domain-specific word, the probability of $p(w_t|w, z_w = 0)$ is specific to the occurrence of w in $\mathcal{D}^p$ or $\mathcal{D}^q$. For individual training instances, the occurrences of w in $\mathcal{D}^p$ or $\mathcal{D}^q$ have been established. Then the probability of $p(w_t|w, z_w = 0)$ can be defined as follows:

$$p(w_t|w, z_w = 0) = \begin{cases} \frac{\exp(U_w^p \cdot V_{w_t})}{\sum_{w'\in\Lambda} \exp(U_w^p \cdot V_{w'})}, \text{if } w \in \mathcal{D}^p \\ \frac{\exp(U_w^q \cdot V_{w_t})}{\sum_{w'\in\Lambda} \exp(U_w^q \cdot V_{w'})}, \text{if } w \in \mathcal{D}^q \end{cases} \tag{3}$$

3.2 Exploiting Sentiment Information

In our DSE model, the prediction of review sentiment depends on not only the text information but also the domain-commonality. For example, the domain-common word "good" has high probability to be positive in different reviews across multiple domains. However, for the word "lightweight", it would be positive in the electronics domain, but negative in the movie domain. We define the polarity y_w of each word w to be consistent with the sentiment label of the review: if we observe that a review is associated with a positive label, the words in the review are associated with a positive label too. Then, the probability of predicting the sentiment for the word w can be defined as:

$$p(y_w|w) = \sum_{k\in\{0,1\}} p(y_w|w, z_w = k)p(z_w = k) \tag{4}$$

If $z_w = 1$, the word w is a domain-common word. The probability $p(y_w = 1|w, z_w = 1)$ can be defined as:

$$p(y_w = 1|w, z_w = 1) = \sigma(U_w^c \cdot \mathbf{s}) \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function and the vector s with dimension d represents the boundary of the sentiment. Moreover, we have:

$$p(y_w = 0|w, z_w = 1) = 1 - p(y_w = 1|w, z_w = 1) \tag{6}$$

If w is a domain-specific word, similarly, the probability $p(y_w = 1|w, z_w = 0)$ is defined as:

$$p(y_w = 1|w, z_w = 0) = \begin{cases} \sigma(U_w^p \cdot \mathbf{s}) & \text{if } w \in \mathcal{D}^p \\ \sigma(U_w^q \cdot \mathbf{s}) & \text{if } w \in \mathcal{D}^q \end{cases} \tag{7}$$

3.3 Inference Algorithm

We need an inference method that can learn, given $\mathcal{D}^p$ and $\mathcal{D}^q$, the values of the model parameters, namely, the domain-common embedding U_w^c, and the domain-specific embeddings U_w^p and U_w^q, as well as the domain-commonality distribution $p(z_w)$ for each word w. Our inference method combines the Expectation-Maximization (EM) method with a negative sampling scheme. It is summarized in Algorithm 1. In the E-step, we use the Bayes rule to evaluate the posterior distribution of z_w for each word and derive the objective function. In the M-step, we maximize the objective function with the gradient descent method and

Algorithm 1 EM negative sampling for DSE

```
1:  Initialize $U_w^c$, $U_w^p$, $U_w^q$, $V$, s, $p(z_w)$
2:  for $iter = 1$ to $Max\_iter$ do
3:      for each review $r$ in $\mathcal{D}^p$ and $\mathcal{D}^q$ do
4:          for each word $w$ in $r$ do
5:              Sample negative instances from the distribution P.
6:              Update $p(z_w|w, c_w, y_w)$ by Eq. 11 and Eq. 15 respectively.
7:          end for
8:      end for
9:      Update $p(z_w)$ using Eq. 13
10:     Update $U_w^c$, $U_w^p$, $U_w^q$, $V$, s via Maximizing Eq. 14
11: end for
```

update the corresponding embeddings U_w^c, U_w^p and U_w^q.

With the input of $\mathcal{D}^p$ and $\mathcal{D}^q$, the likelihood function of the whole training set is:

$$\mathcal{L} = \mathcal{L}^p + \mathcal{L}^q \tag{8}$$

where $\mathcal{L}^p$ and $\mathcal{L}^q$ are the likelihood of $\mathcal{D}^p$ and $\mathcal{D}^q$ respectively.

For each review r from $\mathcal{D}^p$, to learn domain-specific and sentiment-aware embeddings, we wish to predict the sentiment label and context words together. Therefore, the likelihood function is defined as follows:

$$\mathcal{L}^p = \sum_{r\in\mathcal{D}^p}\sum_{w\in r}\log p(y_w, c_w|w) \tag{9}$$

where y_w is the sentiment label and c_w is the set of context words of w. For the simplification of the model, we assume that the sentiment label y_w and the context words c_w of the word w are conditionally dependent. Then the likelihood $\mathcal{L}^p$ can be rewritten as:

$$\begin{aligned}\mathcal{L}^p = &\sum_{r\in\mathcal{D}^p}\sum_{w\in r}\sum_{w_t\in c_w}\log p(w_t|w)+\\ &\sum_{r\in\mathcal{D}^p}\sum_{w\in r}\log p(y_w|w)\end{aligned} \tag{10}$$

where $p(w_t|w)$ and $p(y_w|w)$ are defined in Eq. 1 and Eq. 4 respectively. The likelihood of the reviews from $\mathcal{D}^q$, i.e $\mathcal{L}^q$, is defined similarly.

For each word w in the review r, in the E-step, the posterior probability of z_w given c_w and y_w is:

$$p(z_w = k|w, c_w, y_w) = \frac{p(z_w = k)p(y_w|w, z_w = k)\prod_{w_t\in c_w} p(w_t|w, z_w = k)}{\sum_{k'\in\{0,1\}} p(z_w = k')p(y_w|w, z_w = k')\prod_{w_t\in c_w} p(w_t|w, z_w = k')} \tag{11}$$

In the M-step, given the posterior distribution of z_w in Eq. 11, the goal is to maxmize the following Q function:

$$\begin{aligned}\mathbf{Q} = &\sum_{r\in\{\mathcal{D}^p,\mathcal{D}^q\}}\sum_{w\in r}\sum_{z_w} p(z_w|w, y_w, w_{t+j})\\ &\times\log(p(z_w)p(c_w, y|z, w_t))\\ = &\sum_{r\in\{\mathcal{D}^p,\mathcal{D}^q\}}\sum_{w\in r}\sum_{z_w} p(z_w|w, y_w, c_w)\\ &[\log p(z_w) + \log(y_w|z, w)+\\ &\sum_{w_t\in c_w}\log p(w_t|z_w, w)]\end{aligned} \tag{12}$$

Using the Lagrange multiplier, we can obtain the update rule of $p(z_w)$, satisfying the normalization constraints that $\sum_{z_w\in 0,1} p(z_w) = 1$ for each word w:

$$p(z_w) = \frac{\sum_{r\in\{\mathcal{D}^p,\mathcal{D}^q\}}\sum_{w\in r} p(z_w|w, y_w, c_w)}{\sum_{r\in\{\mathcal{D}^p,\mathcal{D}^q\}} n(w, r)} \tag{13}$$

where $n(w, r)$ is the number of occurrence of the word w in the review r.

To obtain U_w^c, U_w^p and U_w^q, we collect the related items in Eq. 12 as follows:

$$\begin{aligned}\mathbf{Q_U} = &\sum_{r\in\{\mathcal{D}^p,\mathcal{D}^q\}}\sum_{w\in r}\sum_{z_w} p(z_w|w, y_w, w_{t+j})\\ &[\log(y_w|z_w, w) + \sum_{w_t\in c_w}\log p(w_t|z_w, w)]\end{aligned} \tag{14}$$

Note that computing the value $p(w_t|w, z_w)$ based on Eq. 2 and Eq. 3 is not feasible in practice, given that the computation cost is proportional to the size of Λ. However, similar to the skip-gram model, we can rely on negative sampling to address this issue. Therefore we estimate the probability of predicting the context word $p(w_t|w, z_w = 1)$ as follows:

$$\begin{aligned}\log p(w_t|w, z_w = 1) \propto &\log\sigma(U_w^c\cdot V_{w_t})\\ &+\sum_{i=1}^{n}\mathbf{E}_{w_i\sim P}[\log\sigma(-U_w^c\cdot V_{w_i})]\end{aligned} \tag{15}$$

where w_i is a negative instance which is sampled from the word distribution $P(.)$. Mikolov et al. (2013) have investigated many choices for $P(w)$ and found that the best $P(w)$ is equal to the unigram distribution $Unigram(w)$ raised to the $3/4rd$ power. We adopt the same setting. The probability $p(w_t|w, z_w = 0)$ in Eq. 3 can be approximated in a similar manner.

After the substitution of $p(w_t|w, z_w)$, we use the Stochastic Gradient Descent method to maximize Eq. 14, and obtain the update of U^c_w, U^p_w and U^q_w.

3.4 More Discussions

In our model, for simplifying the inference algorithm and saving the computational cost, we assume that the target word w_t in the context and the sentiment label y_w of the word w are conditionally independent. Such technique has also been used in other popular models such as the bi-gram language model. Otherwise, we need to consider the term $p(w_t|w, y_w)$, which complicates the inference algorithm.

We define the formulation of the term $p(w_t|w, z)$ to be similar to the original skip-gram model instead of the CBOW model. The CBOW model averages the context words to predict the target word. The skip-gram model uses pairwise training examples which are much easier to integrate with sentiment information.

Note that our model can be easily extended to more than two domains. Similarly, we use a domain-specific vector for each word in each domain and each word is also associated with a domain-common vector. We just need to extend the probability distribution of z_w from Bernoulli distribution to Multinomial distribution according to the number of domains.

4 Experiment

4.1 Experimental Setup

We conducted experiments on the Amazon product reviews collected by Blitzer et al. (2007). We use four product categories: books (**B**), DVDs (**D**), electronic items (**E**), and kitchen appliances (**K**). A category corresponds to a domain. For each domain, there are 17,457 unlabeled reviews on average associated with rating scores from 1.0 to 5.0 for each domain. We use unlabeled reviews with rating score higher than 3.0 as positive reviews and unlabeled reviews with rating score lower than 3.0 as negative reviews for embedding learning. We first remove reviews whose length is less than 5 words. We also remove punctuations and the stop words. We also stem each word to its root form using Porter Stemmer (Porter, 1980). Note that this review data is used for embedding learning, and the learned embeddings are used as feature vectors of words to conduct the experiments in the later two subsections.

Given the reviews from two domains, namely, $\mathcal{D}^p$ and $\mathcal{D}^q$, we compare our results with the following baselines and state-of-the-art methods:

SSWE The SSWE model[1] proposed by Tang et al. (2014b) can learn sentiment-aware word embeddings from tweets. We employ this model on the combined reviews from $\mathcal{D}^p$ and $\mathcal{D}^q$ and then obtain the embeddings.

Yang's Work Yang et al. (2017) have proposed a method[2] to learn domain-sensitive word embeddings. They choose pivot words and add a regularization item into the original skip-gram objective function enforcing that word representations of pivot words for the source and target domains should be similar. The method trains the embeddings of the source domain first and then fixes the learned embedding to train the embedding of the target domain. Therefore, the learned embedding of the target domain benefits from the source domain. We denote the method as Yang in short.

EmbeddingAll We learn word embeddings from the combined unlabeled review data of $\mathcal{D}^p$ and $\mathcal{D}^q$ using the skip-gram method (Mikolov et al., 2013).

EmbeddingCat We learn word embeddings from the unlabeled reviews of $\mathcal{D}^p$ and $\mathcal{D}^q$ respectively. To represent a word for review sentiment classification, we concatenate its learned word embeddings from the two domains.

EmbeddingP and EmbeddingQ In EmbeddingP, we use the original skip-gram method (Mikolov et al., 2013) to learn word

[1]We use the implementation from `https://github.com/attardi/deepnl/wiki/Sentiment-Specific-Word-Embeddings`.

[2]We use the implementation from `http://statnlp.org/research/lr/`.

	B & D		B & E		B & K		D & E		D & K		E & K	
	Acc.	F1	*Acc.*	F1	*Acc.*	F1	*Acc.*	F1	*Acc.*	F1	*Acc.*	F1
BOW	0.680	0.653	0.738	0.720	0.734	0.725	0.705	0.685	0.706	0.689	0.739	0.715
EmbeddingP	0.753	0.740	0.752	0.745	0.742	0.741	0.740	0.746	0.707	0.702	0.761	0.760
EmbeddingQ	0.736	0.732	0.697	0.697	0.706	0.701	0.762	0.759	0.758	0.759	0.783	0.780
EmbeddingCat	0.769	0.731	0.768	0.763	0.763	0.763	0.787	0.773	0.770	0.770	0.807	0.803
EmbeddingAll	0.769	0.759	0.765	0.740	0.775	0.767	0.783	0.779	0.779	0.776	0.819	0.815
Yang	0.767	0.752	0.775	0.766	0.760	0.755	0.791	0.785	0.762	0.760	0.805	0.804
SSWE	0.783	0.772	0.791	0.780	**0.801**	0.792	0.825	0.815	0.795	0.790	0.835	0.824
DSE$_c$	0.773	0.750	0.783	0.781	0.775	0.773	0.797	0.792	0.784	0.776	0.806	0.800
DSE$_w$	**0.794**$^{\dagger\natural}$	**0.793**$^{\dagger\natural}$	**0.806**$^{\dagger\natural}$	**0.802**$^{\dagger\natural}$	0.797†	**0.793**†	**0.843**$^{\dagger\natural}$	**0.832**$^{\dagger\natural}$	**0.829**$^{\dagger\natural}$	**0.827**$^{\dagger\natural}$	**0.856**$^{\dagger\natural}$	**0.853**$^{\dagger\natural}$

Table 1: Results of review sentiment classification. The markers † and $^{\natural}$ refer to p-value < 0.05 when comparing with Yang and SSWE respectively.

embeddings only from the unlabeled reviews of $\mathcal{D}^p$. Similarly, we only adopt the unlabeled reviews from $\mathcal{D}^q$ to learn embeddings in EmbeddingQ.

BOW We use the traditional bag of words model to represent each review in the training data.

For our DSE model, we have two variants to represent each word. The first variant DSE$_c$ represents each word via concatenating the domain-common vector and the domain-specific vector. The second variant DSE$_w$ concatenates domain-common word embeddings and domain-specific word embeddings by considering the domain-commonality distribution $p(z_w)$. For individual review instances, the occurrences of w in $\mathcal{D}_p$ or $\mathcal{D}_q$ have been established. The representation of w is specific to the occurrence of w in $\mathcal{D}_p$ or $\mathcal{D}_q$. Specifically, each word w can be represented as follows:

$$U_w = \begin{cases} \text{if } w \in \mathcal{D}^p \\ \quad U_w^c \times p(z_w) \oplus U_w^p \times (1.0 - p(z_w)) \\ \text{if } w \in \mathcal{D}^q \\ \quad U_w^c \times p(z_w) \oplus U_w^q \times (1.0 - p(z_w)) \end{cases} \tag{16}$$

where $\oplus$ denotes the concatenation operator.

For all word embedding methods, we set the dimension to 200. For the skip-gram based methods, we sample 5 negative instances and the size of the windows for each target word is 3. For our DSE model, the number of iterations for the whole reviews is 100 and the learning rate is set to 1.0.

4.2 Review Sentiment Classification

For the task of review sentiment classification, we use 1000 positive and 1000 negative sentiment reviews labeled by Blitzer et al. (2007) for each domain to conduct experiments. We randomly select 800 positive and 800 negative labeled reviews from each domain as training data, and the remaining 200 positive and 200 negative labeled reviews as testing data. We use the SVM classifier (Fan et al., 2008) with linear kernel to train on the training reviews for each domain, with each review represented as the average vector of its word embeddings.

We use two metrics to evaluate the performance of sentiment classification. One is the standard accuracy metric. The other one is Macro-F1, which is the average of F1 scores for both positive and negative reviews.

We conduct multiple trials by selecting every possible two domains from books (**B**), DVDs (**D**), electronic items (**E**) and kitchen appliances (**K**). We use the average of the results of each two domains. The experimental results are shown in Table 1.

From Table 1, we can see that compared with other baseline methods, our DSE$_w$ model can achieve the best performance of sentiment classification across most combinations of the four domains. Our statistical t-tests for most of the combinations of domains show that the improvement of our DSE$_w$ model over Yang and SSWE is statistically significant respectively (p-value < 0.05) at 95% confidence level. It shows that our method can capture the domain-commonality and sentiment information at the same time.

Even though both of the SSWE model and our DSE model can learn sentiment-aware word embeddings, our DSE$_w$ model can outperform SSWE. It demonstrates that compared with general sentiment-aware embeddings, our learned domain-common and domain-specific word em-

	B & D		B & E		B & K		D & E		D & K		E & K	
	HL	MPQA	HL	MPQA	HL	MPQA	HL	MPQA	HL	MPQA	HL	MPQA
EmbeddingP	0.740	0.733	0.742	0.734	0.747	0.735	0.744	0.701	0.745	0.709	0.628	0.574
EmbeddingQ	0.743	0.701	0.627	0.573	0.464	0.453	0.621	0.577	0.462	0.450	0.465	0.453
EmbeddingCat	0.780	0.772	0.773	0.756	0.772	0.751	0.744	0.728	0.755	0.702	0.683	0.639
EmbeddingAll	0.777	0.769	0.773	0.730	0.762	0.760	0.712	0.707	0.749	0.724	0.670	0.658
Yang	0.780	0.775	0.789	0.762	0.781	0.770	0.762	0.736	0.756	0.713	0.634	0.614
SSWE	**0.816**	**0.801**	0.831	0.817	0.822	**0.808**	**0.826**	0.785	0.784	0.772	0.707	0.659
DSE	0.802	0.788	**0.833**	**0.828**	**0.832**	0.799	0.804	**0.797**	**0.796**	**0.786**	**0.725**	**0.683**

Table 2: Results of lexicon term sentiment classification.

beddings can capture semantic variations of words across multiple domains.

Compared with the method of Yang which learns cross-domain embeddings, our DSE$_w$ model can achieve better performance. It is because we exploit sentiment information to distinguish domain-common and domain-specific words during the embedding learning process. The sentiment information can also help the model distinguish the words which have similar contexts but different sentiments.

Compared with EmbeddingP and EmbeddingQ, the methods of EmbeddingAll and EmbeddingCat can achieve better performance. The reason is that the data augmentation from other domains helps sentiment classification in the original domain. Our DSE model also benefits from such kind of data augmentation with the use of reviews from $\mathcal{D}^p$ and $\mathcal{D}^q$.

We observe that our DSE$_w$ variant performs better than the variant of DSE$_c$. Compared with DSE$_c$, our DSE$_w$ variant adds the item of $p(z_w)$ as the weight to combine domain-common embeddings and domain-specific embeddings. It shows that the domain-commonality distribution in our DSE model, i.e $p(w_z)$, can effectively model the domain-sensitive information of each word and help review sentiment classification.

4.3 Lexicon Term Sentiment Classification

To further evaluate the quality of the sentiment semantics of the learned word embeddings, we also conduct lexicon term sentiment classification on two popular sentiment lexicons, namely **HL** (Hu and Liu, 2004) and **MPQA** (Wilson et al., 2005). The words with neutral sentiment and phrases are removed. The statistics of **HL** and **MPQA** are shown in Table 3.

We conduct multiple trials by selecting every possible two domains from books (**B**), DVDs (**D**), electronics items (**E**) and kitchen appliances (**K**).

Lexicon	Positive	Negative	Total
HL	1,331	2,647	3,978
MPQA	1,932	2,817	3,075

Table 3: Statistics of the sentiment lexicons.

For each trial, we learn the word embeddings. For our DSE model, we only use the domain-common part to represent each word because the lexicons are usually not associated with a particular domain. For each lexicon, we select 80% to train the SVM classifier with linear kernel and the remaining 20% to test the performance. The learned embedding is treated as the feature vector for the lexicon term. We conduct 5-fold cross validation on all the lexicons. The evaluation metric is Macro-F1 of positive and negative lexicons.

Table 2 shows the experimental results of lexicon term sentiment classification. Our DSE method can achieve competitive performance among all the methods. Compared with SSWE, our DSE is still competitive because both of them consider the sentiment information in the embeddings. Our DSE model outperforms other methods which do not consider sentiments such as Yang, EmbeddingCat and EmbeddingAll. Note that the advantage of domain-sensitive embeddings would be insufficient for this task because the sentiment lexicons are not domain-specific.

5 Case Study

Table 4 shows the probabilities of "lightweight", "die", "mysterious", and "great" to be domain-common for different domain combinations. For "lightweight", its domain-common probability for the books domain and the DVDs domain ("B & D") is quite high, i.e. $p(z = 1) = 0.999$, and the review examples in the last column show that the word "lightweight" expresses the meaning of lacking depth of content in books or movies. Note that most reviews of DVDs are about movies.

Term	Domain	$p(z=1)$	Sample Reviews
“lightweight”	B & D	0.999	- I find Seth Godin’s books incredibly **lightweight**. There is really nothing of any substance here.(`B`) - I love the fact that it’s small and **lightweight** and fits into a tiny pocket on my camera case so I never lose track of it.(`E`) - These are not ”**lightweight**” actors. (`D`) - This vacuum does a pretty good job. It is **lightweight** and easy to use.(`K`)
	B & E	0.404	
	B & K	0.241	
	D & E	0.380	
	D & K	0.013	
	E & K	0.696	
“die”	B & E	0.435	- I’m glad Brando lived long enough to get old and fat, and that he didn’t **die** tragically young like Marilyn, JFK, or Jimi Hendrix.(`B`) - Like many others here, my CD-changer **died** after a couple of weeks and it wouldn’t read any CD.(`E`) - I had this toaster for under 3 years when I came home one day and it smoked and **died**. (`K`)
	B & K	0.492	
	E & K	0.712	
“mysterious”	B & E	0.297	- This novel really does cover the gamut: stunning twists, genuine love, beautiful settings, desire for riches, **mysterious** murders, detective investigations, false accusations, and self vindication.(`B`) - Caller ID functionality for Vonage **mysteriously** stopped working even though this phone’s REN is rated at 0.1b. (`E`)
“great”	B & D	0.760	- This is a **great** book for anyone learning how to handle dogs.(`B`) - This is a **great** product, and you can get it, along with any other products on Amazon up to $500 Free!(`E`) - I grew up with drag racing in the 50s, 60s & 70s and this film gives a **great** view of what it was like.(`D`) - This is a **great** mixer its a little loud but worth it for the power you get.(`K`)
	B & E	0.603	
	B & K	0.628	
	D & E	0.804	
	D & K	0.582	
	E & K	0.805	

Table 4: Learned domain-commonality for some words. $p(z = 1)$ denotes the probability that the word is domain-common. The letter in parentheses indicates the domain of the review.

In the electronics domain and the kitchen appliances domain (“E & K”), “lightweight” means light material or weighing less than average, thus the domain-common probability for these two domains is also high, 0.696. In contrast, for the other combinations, the probability of “lightweight” to be domain-common is much smaller, which indicates that the meaning of “lightweight” varies. Similarly, “die” in the domains of electronics and kitchen appliances (“E & K”) means that something does not work any more, thus, we have $p(z = 1) = 0.712$. While for the books domain, it conveys meaning that somebody passed away in some stories. The word “mysterious” conveys a positive sentiment in the books domain, indicating how wonderful a story is, but it conveys a negative sentiment in the electronics domain typically describing that a product breaks down unpredictably. Thus, its domain-common probability is small. The last example is the word “great”, and it usually has positive sentiment in all domains, thus has large values of $p(z = 1)$ for all domain combinations.

6 Conclusions

We propose a new method of learning domain-sensitive and sentiment-aware word embeddings. Compared with existing sentiment-aware embeddings, our model can distinguish domain-common and domain-specific words with the consideration of varied semantics across multiple domains. Compared with existing domain-sensitive methods, our model detects domain-common words according to not only similar context words but also sentiment information. Moreover, our learned embeddings considering sentiment information can distinguish words with similar syntactic context but opposite sentiment polarity. We have conducted experiments on two downstream sentiment classification tasks, namely review sentiment classification and lexicon term sentiment classification. The experimental results demonstrate the advantages of our approach.

References

Ngo Xuan Bach, Vu Thanh Hai, and Tu Minh Phuong. 2016. Cross-domain sentiment classification with word embeddings and canonical correlation analysis. In *Proceedings of the Seventh Symposium on Information and Communication Technology*. ACM, pages 159–166.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern Information Retrieval*, volume 463. ACM press New York.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and Trends in Machine Learning* 2(1):1–127.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. pages 440–447.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. pages 730–740.

Danushka Bollegala, Tingting Mu, and John Yannis Goulermas. 2016. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge and Data Engineering* 28(2):398–410.

Danushka Bollegala, David Weir, and John Carroll. 2014. Learning to predict distributions of words across domains. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 613–623.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 452–461.

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

Vachik Dave, Baichuan Zhang, Pin-Yu Chen, and Mohammad Al Hasan. 2018. Neural-brane: Neural bayesian personalized ranking for attributed network embedding. *arXiv preprint arXiv:1804.08774* .

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*. pages 513–520.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 168–177.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 897–907.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1746–1751.

Tao Li, Vikas Sindhwani, Chris Ding, and Yi Zhang. 2009. Knowledge transformation for cross-domain sentiment classification. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 716–717.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies* 5(1):1–167.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Rodrigo Moraes, JoãO Francisco Valiati, and Wilson P GaviãO Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications* 40(2):621–633.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web*. pages 751–760.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. pages 79–86.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1–2):1–135.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 1192–1202.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5(3):1.

Bei Shi, Wai Lam, Shoaib Jameel, Steven Schockaert, and Kwun Ping Lai. 2017. Jointly learning word embeddings and latent topics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 375–384.

Thorvald Sørensen. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biologiske Skrifter* 5:1–34.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon: A representation learning approach. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 172–182.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 1555–1565.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. pages 384–394.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 606–615.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*. pages 347–354.

Wei Yang, Wei Lu, and Vincent Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2898–2904.

Baichuan Zhang and Mohammad Al Hasan. 2017. Name disambiguation in anonymized graphs using network embedding. In *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management*. pages 1239–1248.

Cross-Domain Sentiment Classification with Target Domain Specific Information

Minlong Peng, Qi Zhang[*], Yu-gang Jiang, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{mlpeng16,qz,ygj,xjhuang}@fudan.edu.cn

Abstract

The task of adopting a model with good performance to a target domain that is different from the source domain used for training has received considerable attention in sentiment analysis. Most existing approaches mainly focus on learning representations that are domain-invariant in both the source and target domains. Few of them pay attention to domain specific information, which should also be informative. In this work, we propose a method to simultaneously extract domain specific and invariant representations and train a classifier on each of the representation, respectively. And we introduce a few target domain labeled data for learning domain-specific information. To effectively utilize the target domain labeled data, we train the domain-invariant representation based classifier with both the source and target domain labeled data and train the domain-specific representation based classifier with only the target domain labeled data. These two classifiers then boost each other in a co-training style. Extensive sentiment analysis experiments demonstrated that the proposed method could achieve better performance than state-of-the-art methods.

1 Introduction

Sentiment classification aims to automatically predict sentiment polarity of user generated sentiment data like movie reviews. The exponential increase in the availability of online reviews and recommendations makes it an interesting topic in research and industrial areas. However, reviews can span so many different domains that it is difficult to gather annotated training data for all of them. This has motivated much research on cross-domain sentiment classification which transfers the knowledge from label rich domain (source domain) to the label few domain (target domain).

[*]Corresponding author.

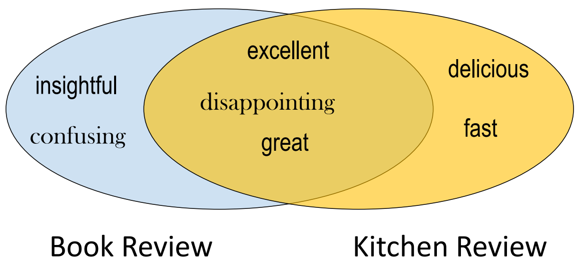


Figure 1: Top indicators extracted with logistic regression for Book and Kitchen domains. The overlap between the two ellipses denotes the shared features between these two domains.

In recent years, the most popular cross-domain sentiment classification approach is to extract domain invariant features, whose distribution in the source domain is close to that in the target domain. (Glorot et al., 2011; Fernando et al., 2013; Kingma and Welling, 2013; Aljundi et al., 2015; Baochen Sun, 2015; Long et al., 2015; Ganin et al., 2016; Zellinger et al., 2017). And based on this representation, it trains a classifier with the source rich labeled data. Specifically, for data of the source domain $\mathbf{X}_s$ and data of the target domain $\mathbf{X}_t$, it trains a feature generator $G(\cdot)$ with restriction $P(G(\mathbf{X}_s)) \approx P(G(\mathbf{X}_t))$. And the classifier is trained on $G(\mathbf{X}_s)$ with the source labels $\mathbf{Y}_s$. The main difference of these approaches is the mechanism to incorporate the restriction on $G(\cdot)$ into the system. The major limitation of this framework is that it losses the domain specific information. As depicted in Figure1, even if it can perfectly extract the domain invariant features (e.g., excellent), it will loss some strong indicators (e.g., delicious, fast) of the target Kitchen domain. We believe that it can achieve greater improvement if it can effectively make use of this information.

Thus, in this work, we try to explore a path to use the target domain specific information with as few as possible target labeled data. Specifically, we first introduce a novel method to extract the domain invariant and domain specific features of target domain data. Then, we treat these two representations as two different views of the target domain data and accordingly train a domain invariant classifier and a target domain specific classifier, respectively. Because the domain invariant representation is compatible with both source data and target data, we train the domain invariant classifier with both source and target labeled data. And for the target domain specific classifier, we train it with target labeled data only. Based on these two classifiers, we perform co-training on target unlabeled data, which can further improve the usage of target data in a bootstrap style.

In summary, the contributions of this paper include: (i) This is the first work to explore the usage of target domain specific information in cross-domain sentiment classification task. (ii) We propose a novel to extract the domain specific representation of target domain data, which encodes the individual characteristics of the target domain.

2 Related Work

Domain adaptation aims to generalize a classifier that is trained on a source domain, for which typically plenty of labeled data is available, to a target domain, for which labeled data is scarce. In supervised domain adaptation, cross-domain classifiers are learnt by using labeled source samples and a small number of labeled target samples (Hoffman et al., 2014). A common practice is training the cross-domain classifiers with the labeled source data and then fine-tuning the classifier with the target labeled data (Pan and Yang, 2010). Meanwhile, some unsupervised and semi-supervised cross domain methods (Ganin et al., 2016; Louizos et al., 2015; Zellinger et al., 2017) are proposed by combining the transfer of classifiers with the match of distributions. These methods focus on extracting the domain-invariant features with the help of unlabeled data. Specifically, Ganin et al., (2016) incorporated an adversarial framework to perform this task. It trained the feature generator to minimize the classification loss and simutaneously deceive the discriminator, which is trained to distinguish the domain of the input data coming from. Louizos et al., (2015) used the Maximum Mean Discrepancy (Borgwardt et al., 2006) regularizer to constrain the feature generator to extract the domain invariant features. And similarly, Zellinger et al., (2017) proposed the central moment discrepancy (CMD) metric for the role of domain regularizer. The above methods either treat it no difference between domain specific information and domain invariant information or just ignore the domain specific information during in the process of learning adaptive classifiers.

One of the most related work is the DSN model (Bousmalis et al., 2016). It proposed to extract the domain specific and the domain invariant representations, simultaneously. However, It does not explored the usage of the domain specific information. Its classifier was still only trained on the domain invariant representation. This work differs from it in the following two aspects. First, we make use of the source and target unlabeled data to extract domain specific information, instead of relying on the orthogonality constraint between the extra representation and the domain invariant counterpart. It is achieved by forcing the distribution of the source examples and that of the target examples in the domain specific space to be different. We argue that this can avoid the potential problem of the orthogonality constraint in that the domain specific representation can be well predicted by the domain invariant representation, while simultaneously meeting the orthogonality constraint. For example, let $\boldsymbol{X} = (\boldsymbol{0}, \boldsymbol{Z})$ be the domain invariant representation and $\boldsymbol{Y} = (\boldsymbol{Z}, \boldsymbol{0})$ be the domain specific representation, then $\boldsymbol{X}$ can be uniquely determined by $\boldsymbol{Y}$, while in the meanwhile $\boldsymbol{X} \perp \boldsymbol{Y}$. Second, we apply a co-training framework to make use of the domain specific representation, rather than simply treating it as a regularizer for extracting the domain invariant representation.

Another related work is the CODA model (Chen et al., 2011). It also applied a co-training framework for semi-supervised domain adaptation. However, instead of dividing the feature space into domain invariant and domain specific

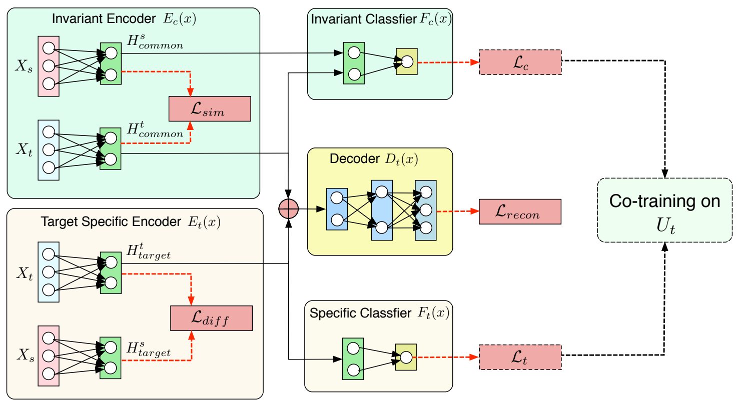


Figure 2: The general architecture of the proposed model. The source data $\boldsymbol{X}_s$ and target data $\boldsymbol{X}_t$ are mapped to a domain invariant representation and a target domain specific representation by feature maps E_c and E_t, respectively. In the space of the domain invariant representation, the distributions of source data $\boldsymbol{H}^s_{inv}$ and target data $\boldsymbol{H}^t_{common}$ are forced to be similar by minimizing a certain distance $\mathcal{L}_{sim}$. In contrast, in the space of the target domain specific representation, the distributions of source data $\boldsymbol{H}^s_{spec}$ and target data $\boldsymbol{H}^t_{spec}$ are forced to be different by minimizing the distance $\mathcal{L}_{diff}$. Based on the domain invariant representation, a classifier F_c is trained with the source rich labeled data and some of the target labeled data. In addition, based on the target domain specific representation, a classifier F_t is trained with the target labeled data only. These two classifiers teach each other in a co-training framework based on the target unlabeled data $\boldsymbol{U}_t$.

parts, it randomly separated the features space.

3 Approach

We consider the following domain adaptation setting. The source domain consists of a set of n_s fully labeled points $D_s = \{(\mathbf{x}^s_1, \mathbf{y}^s_1), \cdots, (\mathbf{x}^s_{n_s}, \mathbf{y}^s_{n_s})\} \subset R^d \times \mathcal{Y}$ drawn from the distribution $P_s(\mathbf{X}, \mathbf{Y})$. And the target data is divided into n_l ($n_l \ll n_s$) labeled points $D^l_t = \{(\mathbf{x}^t_1, \mathbf{y}^t_1), \cdots, (\mathbf{x}^t_{n_l}, \mathbf{y}^t_{n_l}\} \subset R^d \times \mathcal{Y}$ from the distribution $P_t(\mathbf{X}, \mathbf{Y})$ and n_u ($n_u \gg n_l$) unlabeled points $D^u_t = \{(\mathbf{x}^t_{n_l+1}, \mathbf{y}^t_{n_l+1}), \cdots, (\mathbf{x}^t_{n_l+n_u}, \mathbf{y}^t_{n_l+n_u}\} \subset R^d$ from the marginal distribution $P_t(\mathbf{X})$. The goal is to build a classifier for the target domain data using the source domain data and a few labeled target domain data.

In the following section, we first introduce the CMD metric, which is used to measure the probability distribution discrepancy between two random variables. Then, we describe our method to extract the domain specific and domain invariant representations of target domain examples, using the CMD-based regularizer. Finally, we show how to combine these two representations using a co-training framework.

3.1 Central Moment Discrepancy (CMD)

The CMD metric was proposed by Zellinger et al.(2017) to measure the discrepancy between the probability distributions of two (high-dimensional) random variables. It is one of the state-of-the-art metrics and is used as a domain regularizer for domain adaptation. Here, we introduce its definition as a domain regularizer.

Definition 1 *(CMD regularizer). Let X and Y be bounded random samples with respective probability distributions p and q on the interval $[a, b]^N$. The CMD regularizer CMD_K is defined by*

$$CMD_K(X,Y) = \frac{1}{|b-a|} \parallel E(X) - E(Y) \parallel_2 + \frac{1}{|b-a|^k} \sum_{k=2}^{K} \parallel C_k(X) - C_k(Y) \parallel_2, \tag{1}$$

where $E(X) = \frac{1}{|X|}\sum_{x \in X} x$ is the empirical expectation vector computed on the sample X and

$$C_k(X) = \left(E(\prod_{i=1}^{N}(X_i - E(X_i))^{r_i} \right)_{r_i \geq 0, \sum_i^N r_i = k},$$

is the vector of all k^{th} order sample central moments of the coordinates of X.

An intuitive understanding of this metric is that if two probability distributions are similar, their central moment of each order should be close.

3.2 Extract Domain Invariant and Domain Specific Representations

In this work, we aim to extract a domain invariant representation, as well as a domain specific counterpart, for each target example. This makes our work different from most of the existing works, which only focus on the domain invariant representation. The general architecture of the proposed model is illustrated in Figure 2. Data are mapped into a domain invariant hidden space and target domain specific hidden space using two different mappers E_t and E_c, respectively:

$$\begin{aligned} \boldsymbol{H}^s_{spec} &= E_t(\boldsymbol{X}_s; \boldsymbol{\theta}^t_e) \\ \boldsymbol{H}^t_{spec} &= E_t(\boldsymbol{X}_t; \boldsymbol{\theta}^t_e) \\ \boldsymbol{H}^s_{inv} &= E_c(\boldsymbol{X}_s; \boldsymbol{\theta}^c_e) \\ \boldsymbol{H}^t_{inv} &= E_c(\boldsymbol{X}_t; \boldsymbol{\theta}^c_e). \end{aligned} \tag{2}$$

Here, E_t refers to the domain invariant mapper and E_c is the target domain specific mapper. $\boldsymbol{\theta}^t_e$ and $\boldsymbol{\theta}^c_e$ denote their corresponding parameters. The subscript e denotes *encode*. Based on the hidden presentations $\boldsymbol{H}^t_{inv}$ and $\boldsymbol{H}^t_{spec}$, we build an auto-encoder for the target domain examples:

$$\hat{\boldsymbol{X}}_t = D_t(\boldsymbol{H}^t_{inv}, \boldsymbol{H}^t_{spec}; \boldsymbol{\theta}^t_d), \tag{3}$$

with respect to parameters $\boldsymbol{\theta}^t_d$, where the subscript d denotes *decode*. The corresponding reconstruction loss is defined by the mean square error:

$$\mathcal{L}_{recon} = \frac{1}{n_t} \sum_{i}^{n_t} \frac{1}{k} ||\boldsymbol{X}^i_t - \hat{\boldsymbol{X}}_t^{\,i}||^2_2, \tag{4}$$

where k is the dimension of the input feature vector, and $\boldsymbol{X}^i_t$ denotes the i^{th} example of the target domain data. Note that in this work, only target examples are passed to the auto-encoder because we only want to extract target domain specific information.

For E_c, we hope that it only encodes features shared by both the source and target domains. From the distribution view, we hope that the distributions of the mapped outputs, by E_c, of source and target data are similar. To this end, we apply the CMD regularizer onto the hidden representation of source data $\boldsymbol{H}^s_{inv}$ and that of target data $\boldsymbol{H}^t_{inv}$. The corresponding loss is defined by:

$$\mathcal{L}_{sim} = CMD_K(\boldsymbol{H}^s_{inv}, \boldsymbol{H}^t_{inv}). \tag{5}$$

Minimizing this loss will force the distribution of $\boldsymbol{H}^s_{inv}$ and $\boldsymbol{H}^t_{inv}$ to be similar, which in turn encourages E_c to encode domain invariant features.

And for the domain specific encoder E_t, we hope that it only encodes features dominated by the target domain. Ideally, these features should commonly appear in the target domain while hardly appear in the source domain. We argue that this can also be obtained by forcing the distribution of these features in the target domain to differ from that in the source domain, because the target specific auto-encoder D_t should filter out features that hardly appear in the target domain while commonly appear in the source domain. Based on this intuition, we apply a signal flipped CMD regularizer onto the mapped representation of source data $\boldsymbol{H}^s_{spec}$ and that of target data $\boldsymbol{H}^t_{spec}$. The corresponding loss is defined by:

$$\mathcal{L}_{diff} = -CMD_K(\boldsymbol{H}^s_{spec}, \boldsymbol{H}^t_{spec}). \tag{6}$$

Minimizing this loss encourages the distribution of $\boldsymbol{H}^s_{spec}$ to differ from that of $\boldsymbol{H}^t_{spec}$, which in turn encourage E_t to encode domain specific features.

3.3 Co-Training with Domain Invariant and Domain Specific Representations

The co-training algorithm assumes that the data set is presented in two separate views, and two classifiers are trained for each view. In each iteration, some unlabeled examples that are confidently predicted according to exactly one of the two classifiers are moved to the training set. In this way, one classifier provides the predicted labels to the unlabeled examples, on which the other classifier may be uncertain.

In this work, we treat the domain invariant representation and the domain specific representation as the two separate views of target domain examples. Based on the domain invariant representation, we train a domain invariant classifier, F_c, with respect to parameters $\boldsymbol{\theta}_c$. In addition, based on the domain specific representation, we train a domain specific classifier, F_t, with respect to parameters $\boldsymbol{\theta}_t$.

Because the distribution of the source examples is compatible with that of the target examples in

```
input:
        $L_s$: labeled source domain examples
        $L_t$: labeled target domain examples
        $U_t$: unlabeled target domain examples
        $H^s_{inv}$: Invariant representation of $L_s$
        $H^t_{inv}$: Invariant representation of $L_t$
        $H^t_{spec}$: Specific representation of $L_t$
repeat
    Train classifier $F_c$ with $L_s$ and $L_t$ based on $H^s_{inv}$ and $H^t_{inv}$;
    Apply classifier $F_c$ to label $U_t$;
    Select $p$ positive and $n$ negative the most confidently predicted examples $U^c_t$ from $U_t$;
    Train classifier $F_t$ with $L_t$ based on $H^t_{spec}$;
    Apply classifier $F_t$ to label $U_t$;
    Select $p$ positive and $n$ negative the most confidently predicted examples $U^t_t$ from $U_t$;
    Remove examples $U^c_t \cup U^t_t$ from $U_t$;
    Add examples $U^c_t \cup U^t_t$ and their corresponding labels to $L_t$;
until best performance obtained on the developing data set;
```

Algorithm 1: Co-Training for Domain Adaptation

the domain invariant hidden space, we use both the source rich labels and a few target labels to train the classifier F_c. To train the classifier F_t, only the target labels are used. The entire procedure is described in Algorithm 1.

3.4 Model Learning

The training of this model is divided into two parts with one for the domain invariant classifier, F_c, and another one for the domain specific classifier, F_t. For F_c, the goal of training is to minimize the following loss with respect to parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_d, \boldsymbol{\theta}_c\}$:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{recon}(\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_e, \boldsymbol{\theta}^t_d) + \alpha \mathcal{L}_c(\boldsymbol{\theta}^c_e, \boldsymbol{\theta}_c) \\ + \gamma \mathcal{L}_{sim}(\boldsymbol{\theta}^c_e) + \lambda \mathcal{L}_{diff}(\boldsymbol{\theta}^t_e),\end{aligned} \tag{7}$$

where α, γ, and λ are weights that control the interaction of the loss terms. $\mathcal{L}(\theta)$ means that loss, $\mathcal{L}$, is optimized on the parameters θ during training. And $\mathcal{L}_c$ denotes the classification loss on the domain invariant representation, which is defined by the negative log-likelihood of the ground truth class for examples of both source and target domains:

$$\begin{aligned}\mathcal{L}_c = \frac{1}{n_s + l_t} \sum_{i=1}^{n_s} -Y^i_s \log F_c(Y^i_s | E_c(L^i_s)) \\ + \frac{1}{n_s + l_t} \sum_{i=1}^{l_t} -Y^i_t \log F_c(Y^i_t | E_c(L^i_t)),\end{aligned} \tag{8}$$

where Y^i_s is the one-hot encoding of the class label for the i^{th} source example, Y^i_t is that for the i^{th} labeled target example, and l_t denotes the dynamic number of target labeled data in each iteration.

For F_t, the goal of training is to minimize the following loss with respect to parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_e, \boldsymbol{\theta}^t_d, \boldsymbol{\theta}_t\}$:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{recon}(\boldsymbol{\theta}^c_e, \boldsymbol{\theta}^t_e, \boldsymbol{\theta}^t_d) + \beta \mathcal{L}_t(\boldsymbol{\theta}^t_e, \boldsymbol{\theta}_t) \\ + \gamma \mathcal{L}_{sim}(\boldsymbol{\theta}^c_e) + \lambda \mathcal{L}_{diff}(\boldsymbol{\theta}^t_e),\end{aligned} \tag{9}$$

where γ and λ are the same weights as those for the classifier F_c, and β is the weight that controls the portion of classification loss, $\mathcal{L}_t$, on the domain specific representation, which is defined by the negative log-likelihood of the ground truth class for examples of the target domain only:

$$\mathcal{L}_t = \frac{1}{l_t} \sum_{i=1}^{l_t} -Y^i_t \log F_t(Y^i_t | E_t(L^i_t)) \tag{10}$$

4 Experiment

4.1 Dataset

Domain adaptation for sentiment classification has been widely studied in the NLP community. The major experiments were performed on the benchmark made of reviews of Amazon products gathered by Blitzer et al. (2007). This data set[1] contains Amazon product reviews from four different domains: Books, DVD, Electronics, and Kitchen appliances from Amazon.com. Each review was originally associated with a rating of 1-5 stars. For simplicity, we are only concerned with whether or not a review is positive (higher than 3 stars) or negative (3 stars or lower). Reviews are encoded in 5,000 dimensional *tf-idf* feature vectors of bag-of-words unigrams and bigrams. From this data, we constructed 12 cross-domain binary classification tasks. Each domain adaptation task consists of 2,000 labeled source examples,

[1]https://www.cs.jhu.edu/ mdredze/datasets/sentiment/

S→T	Supervised Learning		Unsupervised Transfer		Semi-supervised Transfer			
	SO	ST	CMD	DSN	CMD-ft	DSN-ft	CODA	CoCMD (p-value)
B→D	81.7±0.2	81.6±0.4	82.6±0.3	82.8±0.4	82.7±0.1	82.7±0.6	81.9±0.4	**83.1**±0.1(.003)
B→E	74.0±0.6	75.8±0.2	81.5±0.6	81.9±0.5	82.4±0.6	82.3±0.8	77.5±2.0	**83.0**±0.6(.061)
B→K	76.4±1.0	78.2±0.6	84.4±0.3	84.4±0.6	84.7±0.5	84.8±0.9	80.4±0.8	**85.3**±0.7(.039)
D→B	79.5±0.3	80.0±0.4	80.7±0.6	80.1±1.3	81.0±0.7	81.1±1.2	80.6±0.3	**81.8**±0.5(.022)
D→E	75.6±0.7	77.0±0.3	82.2±0.5	81.4±1.1	82.5±0.7	81.3±1.2	79.4±0.7	**83.4**±0.6(.019)
D→K	79.5±0.4	80.4±0.6	84.8±0.2	83.3±0.7	84.5±0.9	83.8±0.8	82.4±0.5	**85.5**±0.8(.055)
E→B	72.3±1.5	74.7±0.4	74.9±0.6	75.1±0.4	76.2±0.6	76.3±1.4	73.6±0.7	**76.9**±0.6(.094)
E→D	74.2±0.6	75.4±0.4	77.4±0.3	77.1±0.3	77.7±0.7	77.1±1.1	75.9±0.2	**78.3**±0.1(.079)
E→K	85.6±0.6	85.7±0.7	86.4±0.9	87.2±0.7	86.7±0.3	87.1±0.9	86.1±0.4	**87.3**±0.4(.093)
K→B	73.1±0.1	73.8±0.3	75.8±0.3	76.4±0.5	76.4±0.5	76.2±0.3	74.3±1.0	**77.2**±0.4(.016)
K→D	75.2±0.7	76.6±0.9	77.7±0.4	78.0±1.4	78.8±0.4	78.5±0.5	77.5±0.4	**79.6**±0.5(.039)
K→E	85.4±1.0	85.3±1.6	86.7±0.6	86.7±0.7	**87.3**±0.3	87.2±0.4	86.4±0.5	87.2±0.4(.512)

Table 1: Average prediction accuracy with 5 runs on target domain testing data set. The left group of models refer to previous state-of-the-art methods and the right group of models refer to the proposed model and some of its variants. We list the p-values of the T-test between CoCMD and CMD-ft for more intuitive understanding.

2,000 unlabeled target examples, and 50 labeled target examples for training. To fine-tune the hyper-parameters, we randomly select 500 target examples as developing data set, leaving 2,500-5,500 examples for testing. All of the compared methods and CoCMD share this setting.

4.2 Compared Methods

CoCMD is systematically compared with: 1) neural network classifier without any domain adaptation trained on labeled source data only (SO); 2) neural network classifier without any domain adaptation trained on the union of labeled source and target data (ST); 3) unsupervised central moment discrepancy trained with labeled source data only (CMD) (Zellinger et al., 2017); 4) unsupervised domain separation network (DSN) (Bousmalis et al., 2016); 5) semi-supervised CMD trained on labeled source data and then fine-tuned on labeled target data (CMD-ft); 6) semi-supervised DSN trained on labeled source data and then fine-tuned on labeled target data (DSN-ft); 7) semi-supervised Co-training for domain adaptation (CODA) (Chen et al., 2011).

4.3 Implementation Detail

CoCMD was imeplented with a similar architecture to that of Ganin et al., (2016) and Zellinger et al., (2017), with one dense hidden layer with 50 hidden nodes and sigmoid activation functions. The classifiers consist of a softmax layer with two dimensional outputs. And the decoder was implemented with a multilayer perceptron (MLP) with one dense hidden layer, tanh activation functions, and relu output functions.

Model optimization was performed using the RmsProp (Tieleman and Hinton, 2012) update rule with learning rate set to 0.005 for all of the tasks.Hyper-parameter K of the CMD regularizer was set to 3 for all of the tasks, according to the experiment result of Zellinger et al. (2017). For the hyper-parameters α, β, γ, and λ, we took the values that achieve the best performance on the developing data set via a grid search $\{0.01, 0.1, 1, 10, 100\}$. However, instead of building grids on α, β, γ, and λ all at the same time, we first fine-tuned the values of α and β with the values of γ and λ fixed at 1. After that, we fine-tuned the values of γ and λ with α and β fixed at the best values obtained at last step. Though, this practice may miss the best combination of these hyper-parameters, it can greatly reduce the time consuming for fine-tuning and still obtain acceptable results. And for each iteration of the co-training, we set $p = n = 5$.

4.4 Result

Table 1 shows the average classification accuracy of our proposed model and the baselines over all 12 domain adaptation tasks. We can first observe that the proposed model CoCMD outperforms the compared methods over almost all of the

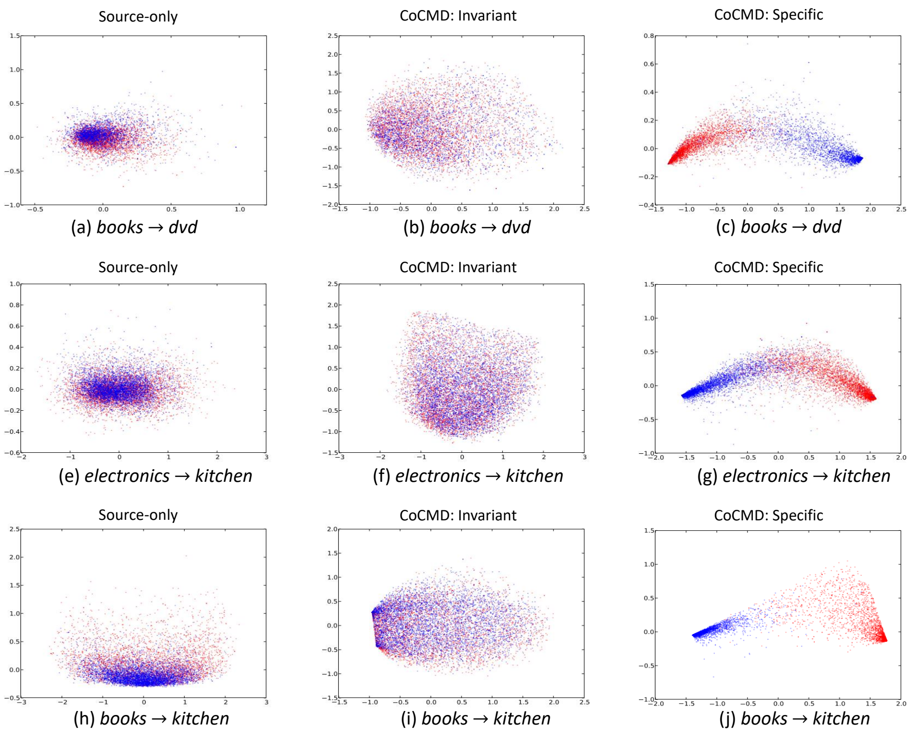


Figure 3: The distribution of source and target data in the hidden space of different representations. The red points denote the source examples and the blue ones denote the target examples. The pictures of each row correspond to the $B{\rightarrow}D$, $E{\rightarrow}K$, and $B{\rightarrow}K$ task. The pictures of each column correspond to the hidden space, H_c^e, of the source-only model, the domain invariant representation, and the target specific representation of the proposed model.

12 tasks except for the K→E task. And by comparing the results of CMD-based methods and DSN-based methods, we can find out that just extracting the domain specific information but not making further usage does not offer much improvement to the adaptation performance for sentiment classification task. This approves the necessary to explore the usage of domain specific information.

If organizing the domain B and D into a group and organizing the domain E and K into another group, we can observe that the domain adaptation methods achieve greater improvement on the standard classifiers over cross-group tasks (e.g., B $\rightarrow$ K) than over within-group tasks (e.g., B $\rightarrow$ D). Similar observation can also be observed by comparing ST with SO, CMD with CMD-ft, and DSN with DSN-ft. The possible explanation is that domains within the same group are more close. Thus adapting over within group tasks is easier than adapting over cross group tasks, if without any domain adaptation regularizer. In addition, we can also observe that CoCMD achieve relatively greater improvement on CMD baseline over the cross-group tasks that over the within-group tasks. We argue that this is because domains in the same group contain relatively less domain individual characteristic. While for domains cross the groups, the domain specific information usually takes a larger share of all of the information. Because the additional part of our proposed method compared to the CMD baseline, is built on the domain specific information, the improvement should be relatively less for within-group tasks. Further analysis of the proposed model in the next section empirically proves this explanation.

4.5 Model Analysis

In this section, we look into how similar two domains are to each other in the space of domain invariant representation and domain specific representation.

A-distance Study: Some of previous works proposed to make use of a proxy of the $\mathcal{A}$-distance (Ben-David et al., 2007) to measure

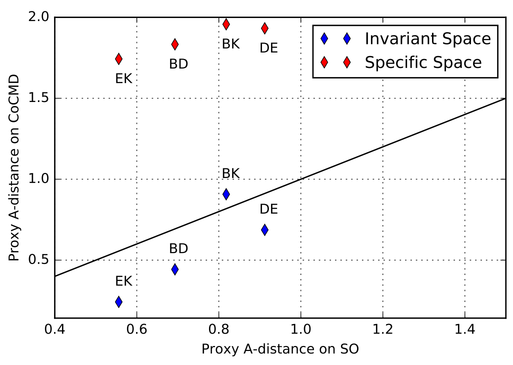


Figure 4: Proxy A-distance between domains of the Amazon benchmark for the 4 different tasks.

the distance of two domains. The proxy was defined by $2(1-2\epsilon)$, where ϵ is the generalization error of a linear SVM classifier trained on the binary classification problem to distinguish inputs between the source and target domains. Figure 4 shows the results of each pair of domains. We observe several trends: Firstly, the proxy A-distance of within-group domain pairs (i.e., BD and EK) is consistently smaller than that of the cross-group domain pairs (i.e., BK and DE) on all of the hidden spaces. Secondly, the proxy A-distance on the domain specific space is consistently larger than its corresponding value on the hidden space of SO model, as expected. While the proxy A-distance value on domain invariant space is generally smaller than its corresponding value on the hidden space of SO model, except for BK domain pair. A possible explanation is that the balance of classification loss and domain discrepancy loss makes there is still some target domain specific information in the domain invariant space, introduced by the target unlabeled data.

Visualization: For more intuitive understanding of the behaviour of the proposed model, we further perform a visualization of the domain invariant representation and the domain specific representation, respectively. For this purpose, we reduce the dimension of the hidden space to 2 using principle component analysis (PCA) (Wold et al., 1987). Due to space constraints we choose three tasks: two within-group tasks ($B \rightarrow D$ and $E \rightarrow K$) and a cross-group task ($B \rightarrow K$). For comparison, we also display the distribution of each domain in the hidden space of the SO model. The results are shown in Figure 3.

Pictures of the first column in Figure 3 show the original distribution of the source and target examples in the hidden space of SO model. As can be seen, there is a great overlap between the distributions of the domain B and the domain D domains and between the distributions of the domain E and the domain K. While there is quite a gap between the distribution of the domain B and the domain K. This strengthens our argument that within-group domains share relatively more common information than cross-group domains. Pictures of the second column show the distribution of the source and target examples in the domain invariant hidden space of the proposed model. From these pictures we can see that the distributions of the source and target data are quite similar in this presentation. This demonstrates the effectiveness of the CMD regularizer for extracting domain invariant representation. Pictures of the third column show the distribution of the source and target examples in the domain specific hidden space of the proposed model. As can be seen from these pictures, examples of the source and target domains are separated very well. This demonstrates the effectiveness of our proposed method for extracting domain specific information.

5 Conclusion

In this work, we investigated the importance of domain specific information for domain adaptation. In contrast with most of the previous methods, which pay more attention to domain invariant information, we showed that domain specific information could also be beneficially used in the domain adaptation task with a small amount of in-domain labeled data. Specifically, we proposed a novel method, based on the CMD metric, to simultaneously extract domain invariant feature and domain specific feature for target domain data. With these two different features, we performed co-training with labeled data from the source domain and a small amount of labeled data from the target domain. Sentiment analysis experiments demonstrated the effectiveness of this method.

6 Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural

Science Foundation of China (No. 61751201, 61532011, 61473092, and 61472088), and STCSM (No.16JC1420401,17JC1420200).

References

Rahaf Aljundi, Rémi Emonet, Damien Muselet, and Marc Sebban. 2015. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 56–63.

Jiashi Feng Kate Saenko Baochen Sun. 2015. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*. pages 137–144.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. volume 7, pages 440–447.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics* 22(14):e49–e57.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 343–351. http://papers.nips.cc/paper/6254-domain-separation-networks.pdf.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *Advances in neural information processing systems*. pages 2456–2464.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. 2013. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2960–2967.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

Judy Hoffman, Erik Rodner, Jeff Donahue, Brian Kulis, and Kate Saenko. 2014. Asymmetric and category invariant feature transformations for domain adaptation. *International journal of computer vision* 109(1-2):28–41.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Mingsheng Long, Jianmin Wang, Jiaguang Sun, and S Yu Philip. 2015. Domain invariant transfer kernel learning. *IEEE Transactions on Knowledge and Data Engineering* 27(6):1519–1532.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2015. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830* .

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2(1-3):37–52.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811* .

Aspect Based Sentiment Analysis with Gated Convolutional Networks

Wei Xue and **Tao Li**
School of Computing and Information Sciences
Florida International University, Miami, FL, USA
{wxue004, taoli}@cs.fiu.edu

Abstract

Aspect based sentiment analysis (ABSA) can provide more detailed information than general sentiment analysis, because it aims to predict the sentiment polarities of the given aspects or entities in text. We summarize previous approaches into two subtasks: aspect-category sentiment analysis (ACSA) and aspect-term sentiment analysis (ATSA). Most previous approaches employ long short-term memory and attention mechanisms to predict the sentiment polarity of the concerned targets, which are often complicated and need more training time. We propose a model based on convolutional neural networks and gating mechanisms, which is more accurate and efficient. First, the novel Gated Tanh-ReLU Units can selectively output the sentiment features according to the given aspect or entity. The architecture is much simpler than attention layer used in the existing models. Second, the computations of our model could be easily parallelized during training, because convolutional layers do not have time dependency as in LSTM layers, and gating units also work independently. The experiments on SemEval datasets demonstrate the efficiency and effectiveness of our models. [1]

1 Introduction

Opinion mining and sentiment analysis (Pang and Lee, 2008) on user-generated reviews can provide valuable information for providers and consumers. Instead of predicting the overall sentiment polarity, fine-grained aspect based sentiment analysis (ABSA) (Liu and Zhang, 2012) is proposed to better understand reviews than traditional sentiment analysis. Specifically, we are interested in the sentiment polarity of aspect categories or target entities in the text. Sometimes, it is coupled with aspect term extractions (Xue et al., 2017). A number of models have been developed for ABSA, but there are two different subtasks, namely aspect-category sentiment analysis (ACSA) and aspect-term sentiment analysis (ATSA). The goal of ACSA is to predict the sentiment polarity with regard to the given aspect, which is one of a few predefined categories. On the other hand, the goal of ATSA is to identify the sentiment polarity concerning the target entities that appear in the text instead, which could be a multi-word phrase or a single word. The number of distinct words contributing to aspect terms could be more than a thousand. For example, in the sentence "*Average to good Thai food, but terrible delivery.*", ATSA would ask the sentiment polarity towards the entity *Thai food*; while ACSA would ask the sentiment polarity toward the aspect *service*, even though the word *service* does not appear in the sentence.

Many existing models use LSTM layers (Hochreiter and Schmidhuber, 1997) to distill sentiment information from embedding vectors, and apply attention mechanisms (Bahdanau et al., 2014) to enforce models to focus on the text spans related to the given aspect/entity. Such models include Attention-based LSTM with Aspect Embedding (ATAE-LSTM) (Wang et al., 2016b) for ACSA; Target-Dependent Sentiment Classification (TD-LSTM) (Tang et al., 2016a), Gated Neural Networks (Zhang et al., 2016) and Recurrent Attention Memory Network (RAM) (Chen et al., 2017) for ATSA. Attention mechanisms has been successfully used in many

[1]The code and data is available at `https://github.com/wxue004cs/GCAE`

NLP tasks. It first computes the alignment scores between context vectors and target vector; then carry out a weighted sum with the scores and the context vectors. However, the context vectors have to encode both the aspect and sentiment information, and the alignment scores are applied across all feature dimensions regardless of the differences between these two types of information. Both LSTM and attention layer are very time-consuming during training. LSTM processes one token in a step. Attention layer involves exponential operation and normalization of all alignment scores of all the words in the sentence (Wang et al., 2016b). Moreover, some models needs the positional information between words and targets to produce weighted LSTM (Chen et al., 2017), which can be unreliable in noisy review text. Certainly, it is possible to achieve higher accuracy by building more and more complicated LSTM cells and sophisticated attention mechanisms; but one has to hold more parameters in memory, get more hyper-parameters to tune and spend more time in training. In this paper, we propose a fast and effective neural network for ACSA and ATSA based on convolutions and gating mechanisms, which has much less training time than LSTM based networks, but with better accuracy.

For ACSA task, our model has two separate convolutional layers on the top of the embedding layer, whose outputs are combined by novel gating units. Convolutional layers with multiple filters can efficiently extract n-gram features at many granularities on each receptive field. The proposed gating units have two nonlinear gates, each of which is connected to one convolutional layer. With the given aspect information, they can selectively extract aspect-specific sentiment information for sentiment prediction. For example, in the sentence "*Average to good Thai food, but terrible delivery.*", when the aspect *food* is provided, the gating units automatically ignore the negative sentiment of aspect *delivery* from the second clause, and only output the positive sentiment from the first clause. Since each component of the proposed model could be easily parallelized, it has much less training time than the models based on LSTM and attention mechanisms. For ATSA task, where the aspect terms consist of multiple words, we extend our model to include another convolutional layer for the target expressions. We evaluate our models on the SemEval datasets, which contains restaurants and laptops reviews with labels on aspect level. To the best of our knowledge, no CNN-based model has been proposed for aspect based sentiment analysis so far.

2 Related Work

We present the relevant studies into following two categories.

2.1 Neural Networks

Recently, neural networks have gained much popularity on sentiment analysis or sentence classification task. Tree-based recursive neural networks such as Recursive Neural Tensor Network (Socher et al., 2013) and Tree-LSTM (Tai et al., 2015), make use of syntactic interpretation of the sentence structure, but these methods suffer from time inefficiency and parsing errors on review text. Recurrent Neural Networks (RNNs) such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014) have been used for sentiment analysis on data instances having variable length (Tang et al., 2015; Xu et al., 2016; Lai et al., 2015). There are also many models that use convolutional neural networks (CNNs) (Collobert et al., 2011; Kalchbrenner et al., 2014; Kim, 2014; Conneau et al., 2016) in NLP, which also prove that convolution operations can capture compositional structure of texts with rich semantic information without laborious feature engineering.

2.2 Aspect based Sentiment Analysis

There is abundant research work on aspect based sentiment analysis. Actually, the name ABSA is used to describe two different subtasks in the literature. We classify the existing work into two main categories based on the descriptions of sentiment analysis tasks in SemEval 2014 Task 4 (Pontiki et al., 2014): Aspect-Term Sentiment Analysis and Aspect-Category Sentiment Analysis.

Aspect-Term Sentiment Analysis. In the first category, sentiment analysis is performed toward the aspect terms that are labeled in the given sentence. A large body of literature tries to utilize the relation or position between the target words and the surrounding context words either by using the tree structure of dependency or by simply counting the number of words between them as a relevance information (Chen et al., 2017).

Recursive neural networks (Lakkaraju et al., 2014; Dong et al., 2014; Wang et al., 2016a) rely on external syntactic parsers, which can be very inaccurate and slow on noisy texts like tweets and reviews, which may result in inferior performance.

Recurrent neural networks are commonly used in many NLP tasks as well as in ABSA problem. TD-LSTM (Tang et al., 2016a) and gated neural networks (Zhang et al., 2016) use two or three LSTM networks to model the left and right contexts of the given target individually. A fully-connected layer with gating units predicts the sentiment polarity with the outputs of LSTM layers.

Memory network (Weston et al., 2014) coupled with multiple-hop attention attempts to explicitly focus only on the most informative context area to infer the sentiment polarity towards the target word (Tang et al., 2016b; Chen et al., 2017). Nonetheless, memory network simply bases its knowledge bank on the embedding vectors of individual words (Tang et al., 2016b), which makes itself hard to learn the opinion word enclosed in more complicated contexts. The performance is improved by using LSTM, attention layer and feature engineering with word distance between surrounding words and target words to produce target-specific memory (Chen et al., 2017).

Aspect-Category Sentiment Analysis. In this category, the model is asked to predict the sentiment polarity toward a predefined aspect category. Attention-based LSTM with Aspect Embedding (Wang et al., 2016b) uses the embedding vectors of aspect words to selectively attend the regions of the representations generated by LSTMs.

3 Gated Convolutional Network with Aspect Embedding

In this section, we present a new model for ACSA and ATSA, namely Gated Convolutional network with Aspect Embedding (GCAE), which is more efficient and simpler than recurrent network based models (Wang et al., 2016b; Tang et al., 2016a; Ma et al., 2017; Chen et al., 2017). Recurrent neural networks sequentially compose hidden vectors $\mathbf{h}_i = f(\mathbf{h}_{i-1})$, which does not enable parallelization over inputs. In the attention layer, softmax normalization also has to wait for all the alignment scores computed by a similarity function. Hence, they cannot take advantage of highly-parallelized modern hardware and libraries. Our model is built on convolutional layers and gating units. Each convolutional filter computes n-gram features at different granularities from the embedding vectors at each position individually. The gating units on top of the convolutional layers at each position are also independent from each other. Therefore, our model is more suitable to parallel computing. Moreover, our model is equipped with two kinds of effective filtering mechanisms: the gating units on top of the convolutional layers and the max pooling layer, both of which can accurately generate and select aspect-related sentiment features.

We first briefly review the vanilla CNN for text classification (Kim, 2014). The model achieves state-of-the-art performance on many standard sentiment classification datasets (Le et al., 2017).

The CNN model consists of an embedding layer, a one-dimension convolutional layer and a max-pooling layer. The embedding layer takes the indices $w_i \in \{1, 2, \ldots, V\}$ of the input words and outputs the corresponding embedding vectors $\boldsymbol{v}_i \in \mathbb{R}^D$. D denotes the dimension size of the embedding vectors. V is the size of the word vocabulary. The embedding layer is usually initialized with pre-trained embeddings such as GloVe (Pennington et al., 2014), then they are fine-tuned during the training stage. The one-dimension convolutional layer convolves the inputs with multiple convolutional kernels of different widths. Each kernel corresponds a linguistic feature detector which extracts a specific pattern of n-gram at various granularities (Kalchbrenner et al., 2014). Specifically, the input sentence is represented by a matrix through the embedding layer, $\mathbf{X} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L]$, where L is the length of the sentence with padding. A convolutional filter $\mathbf{W}_c \in \mathbb{R}^{D\times k}$ maps k words in the receptive field to a single feature c. As we slide the filter across the whole sentence, we obtain a sequence of new features $\mathbf{c} = [c_1, c_2, \ldots, c_L]$.

$$c_i = f(\mathbf{X}_{i:i+K} * \mathbf{W}_c + b_c) \quad , \qquad (1)$$

where $b_c \in \mathbb{R}$ is the bias, f is a non-linear activation function such as tanh function, $*$ denotes convolution operation. If there are n_k filters of the same width k, the output features form a matrix $\mathbf{C} \in \mathbb{R}^{n_k \times L_k}$. For each convolutional filter, the max-over-time pooling layer takes the maximal value among the generated convolutional features, resulting in a fixed-size vector whose size is equal to the number of filters n_k. Finally, a softmax layer uses the vector to predict the sentiment polarity of the input sentence.

Figure 1 illustrates our model architecture. The

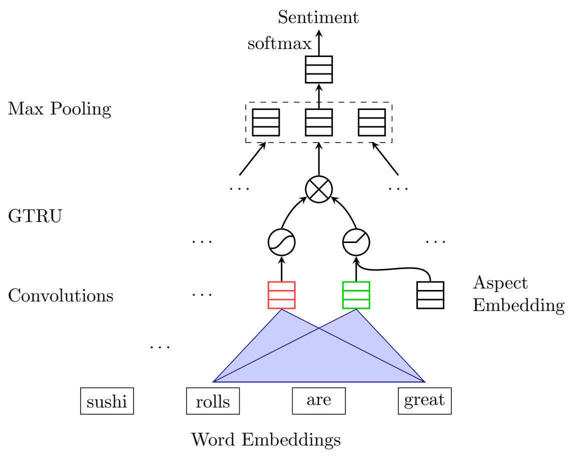


Figure 1: Illustration of our model GCAE for ACSA task. A pair of convolutional neuron computes features for a pair of gates: tanh gate and ReLU gate. The ReLU gate receives the given aspect information to control the propagation of sentiment features. The outputs of two gates are element-wisely multiplied for the max pooling layer.

Gated Tanh-ReLU Units (GTRU) with aspect embedding are connected to two convolutional neurons at each position t. Specifically, we compute the features c_i as

$$a_i = \text{relu}(\mathbf{X}_{i:i+k} * \mathbf{W}_a + \mathbf{V}_a \boldsymbol{v}_a + b_a) \quad (2)$$

$$s_i = \tanh(\mathbf{X}_{i:i+k} * \mathbf{W}_s + b_s) \quad (3)$$

$$c_i = s_i \times a_i \quad , \quad (4)$$

where $\boldsymbol{v}_a$ is the embedding vector of the given aspect category in ACSA or computed by another CNN over aspect terms in ATSA. The two convolutions in Equation 2 and 3 are the same as the convolution in the vanilla CNN, but the convolutional features a_i receives additional aspect information $\boldsymbol{v}_a$ with ReLU activation function. In other words, s_i and a_i are responsible for generating sentiment features and aspect features respectively. The above max-over-time pooling layer generates a fixed-size vector $\boldsymbol{e} \in \mathbb{R}^{d_k}$, which keeps the most salient sentiment features of the whole sentence. The final fully-connected layer with softmax function uses the vector $\boldsymbol{e}$ to predict the sentiment polarity $\hat{y}$. The model is trained by minimizing the cross-entropy loss between the ground-truth y and the predicted value $\hat{y}$ for all data samples.

$$\mathcal{L} = -\sum_i \sum_j y_i^j \log \hat{y}_i^j \quad , \quad (5)$$

where i is the index of a data sample, j is the index of a sentiment class.

4 Gating Mechanisms

The proposed Gated Tanh-ReLU Units control the path through which the sentiment information flows towards the pooling layer. The gating mechanisms have proven to be effective in LSTM. In aspect based sentiment analysis, it is very common that different aspects with different sentiments appear in one sentence. The ReLU gate in Equation 2 does not have upper bound on positive inputs but strictly zero on negative inputs. Therefore, it can output a similarity score according to the relevance between the given aspect information $\boldsymbol{v}_a$ and the aspect feature a_i at position t. If this score is zero, the sentiment features s_i would be blocked at the gate; otherwise, its magnitude would be amplified accordingly. The max-over-time pooling further removes the sentiment features which are not significant over the whole sentence.

In language modeling (Dauphin et al., 2017; Kalchbrenner et al., 2016; van den Oord et al., 2016; Gehring et al., 2017), Gated Tanh Units (GTU) and Gated Linear Units (GLU) have shown effectiveness of gating mechanisms. GTU is represented by $\tanh(\mathbf{X} * \mathbf{W} + b) \times \sigma(\mathbf{X} * \mathbf{V} + c)$, in which the sigmoid gates control features for predicting the next word in a stacked convolutional block. To overcome the gradient vanishing problem of GTU, GLU uses $(\mathbf{X} * \mathbf{W} + b) \times \sigma(\mathbf{X} * \mathbf{V} + c)$ instead, so that the gradients would not be downscaled to propagate through many stacked convolutional layers. However, a neural network that has only one convolutional layer would not suffer from gradient vanish problem during training. We show that on text classification problem, our GTRU is more effective than these two gating units.

5 GCAE on ATSA

ATSA task is defined to predict the sentiment polarity of the aspect terms in the given sentence. We simply extend GCAE by adding a small convolutional layer on aspect terms, as shown in Figure 2. In ACSA, the aspect information controlling the flow of sentiment features in GTRU is from one aspect word; while in ATSA, such information is provided by a small CNN on aspect terms $[w_i, w_{i+1}, \ldots, w_{i+k}]$. The additional CNN extracts the important features from multiple words

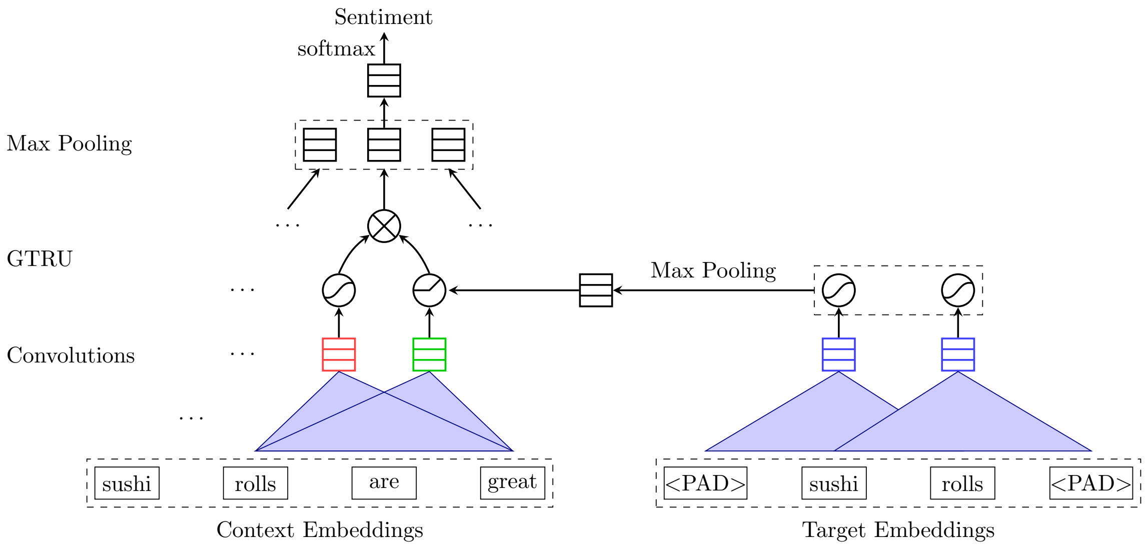


Figure 2: Illustration of model GCAE for ATSA task. It has an additional convolutional layer on aspect terms.

while retains the ability of parallel computing.

6 Experiments

6.1 Datasets and Experiment Preparation

We conduct experiments on public datasets from SemEval workshops (Pontiki et al., 2014), which consist of customer reviews about restaurants and laptops. Some existing work (Wang et al., 2016b; Ma et al., 2017; Chen et al., 2017) removed “conflict” labels from four sentiment labels, which makes their results incomparable to those from the workshop report (Kiritchenko et al., 2014). We reimplemented the compared methods, and used hyper-parameter settings described in these references.

The sentences which have different sentiment labels for different aspects or targets in the sentence are more common in review data than in standard sentiment classification benchmark. The sentence in Table 1 shows the reviewer's different attitude towards two aspects: food and delivery. Therefore, to access how the models perform on review sentences more accurately, we create small but difficult datasets, which are made up of the sentences having opposite or different sentiments on different aspects/targets. In Table 1, the two identical sentences but with different sentiment labels are both included in the dataset. If a sentence has 4 aspect targets, this sentence would have 4 copies in the data set, each of which is associated with different target and sentiment label.

For ACSA task, we conduct experiments on restaurant review data of SemEval 2014 Task 4. There are 5 aspects: food, price, service, ambience, and misc; 4 sentiment polarities: positive, negative, neutral, and conflict. By merging restaurant reviews of three years 2014 - 2016, we obtain a larger dataset called “Restaurant-Large”. Incompatibilities of data are fixed during merging. We replace conflict labels with neutral labels in the 2014 dataset. In the 2015 and 2016 datasets, there could be multiple pairs of “aspect terms” and “aspect category” in one sentence. For each sentence, let p denote the number of positive labels minus the number of negative labels. We assign a sentence a positive label if $p > 0$, a negative label if $p < 0$, or a neutral label if $p = 0$. After removing duplicates, the statistics are show in Table 2. The resulting dataset has 8 aspects: restaurant, food, drinks, ambience, service, price, misc and location.

For ATSA task, we use restaurant reviews and laptop reviews from SemEval 2014 Task 4. On each dataset, we duplicate each sentence n_a times, which is equal to the number of associated aspect categories (ACSA) or aspect terms (ATSA) (Ruder et al., 2016b,a). The statistics of the datasets are shown in Table 2.

The sizes of hard data sets are also shown in Table 2. The test set is designed to measure whether a model can detect multiple different sentiment polarities in one sentence toward different entities. Without such sentences, a classifier for overall sentiment classification might be good enough

Sentence	aspect category/term	sentiment label
Average to good Thai food, but terrible delivery.	food	positive
Average to good Thai food, but terrible delivery.	delivery	negative

Table 1: Two example sentences in one hard test set of restaurant review dataset of SemEval 2014.

for the sentences associated with only one sentiment label.

In our experiments, word embedding vectors are initialized with 300-dimension GloVe vectors which are pre-trained on unlabeled data of 840 billion tokens (Pennington et al., 2014). Words out of the vocabulary of GloVe are randomly initialized with a uniform distribution $U(-0.25, 0.25)$. We use Adagrad (Duchi et al., 2011) with a batch size of 32 instances, default learning rate of $1e-2$, and maximal epochs of 30. We only fine tune early stopping with 5-fold cross validation on training datasets. All neural models are implemented in PyTorch.

6.2 Compared Methods

To comprehensively evaluate the performance of GCAE, we compare our model against the following models.

NRC-Canada (Kiritchenko et al., 2014) is the top method in SemEval 2014 Task 4 for ACSA and ATSA task. SVM is trained with extensive feature engineering: various types of n-grams, POS tags, and lexicon features. The sentiment lexicons improve the performance significantly, but it requires large scale labeled data: 183 thousand Yelp reviews, 124 thousand Amazon laptop reviews, 56 million tweets, and 3 sentiment lexicons labeled manually.

CNN (Kim, 2014) is widely used on text classification task. It cannot directly capture aspect-specific sentiment information on ACSA task, but it provides a very strong baseline for sentiment classification. We set the widths of filters to 3, 4, 5 with 100 features each.

TD-LSTM (Tang et al., 2016a) uses two LSTM networks to model the preceding and following contexts of the target to generate target-dependent representation for sentiment prediction.

ATAE-LSTM (Wang et al., 2016b) is an attention-based LSTM for ACSA task. It appends the given aspect embedding with each word embedding as the input of LSTM, and has an attention layer above the LSTM layer.

IAN (Ma et al., 2017) stands for interactive attention network for ATSA task, which is also based on LSTM and attention mechanisms.

RAM (Chen et al., 2017) is a recurrent attention network for ATSA task, which uses LSTM and multiple attention mechanisms.

GCN stands for gated convolutional neural network, in which GTRU does not have the aspect embedding as an additional input.

6.3 Results and Analysis

6.3.1 ACSA

Following the SemEval workshop, we report the overall accuracy of all competing models over the test datasets of restaurant reviews as well as the hard test datasets. Every experiment is repeated five times. The mean and the standard deviation are reported in Table 4.

LSTM based model ATAE-LSTM has the worst performance of all neural networks. Aspect-based sentiment analysis is to extract the sentiment information closely related to the given aspect. It is important to separate aspect information and sentiment information from the extracted information of sentences. The context vectors generated by LSTM have to convey the two kinds of information at the same time. Moreover, the attention scores generated by the similarity scoring function are for the entire context vector.

GCAE improves the performance by 1.1% to 2.5% compared with ATAE-LSTM. First, our model incorporates GTRU to control the sentiment information flow according to the given aspect information at each dimension of the context vectors. The element-wise gating mechanism works at fine granularity instead of exerting an alignment score to all the dimensions of the context vectors in the attention layer of other models. Second, GCAE does not generate a single context vector, but two vectors for aspect and sentiment features respectively, so that aspect and sentiment information is unraveled. By comparing the performance on the hard test datasets against CNN, it is easy to see the convolutional layer of GCAE is able to differentiate the sentiments of multiple entities.

Convolutional neural networks CNN and GCN

	Positive		Negative		Neutral		Conflict	
	Train	Test	Train	Test	Train	Test	Train	Test
Restaurant-Large	2710	1505	1198	680	757	241	-	-
Restaurant-Large-Hard	182	92	178	81	107	61	-	-
Restaurant-2014	2179	657	839	222	500	94	195	52
Restaurant-2014-Hard	139	32	136	26	50	12	40	19

Table 2: Statistics of the datasets for ACSA task. The hard dataset is only made up of sentences having multiple aspect labels associated with multiple sentiments.

	Positive		Negative		Neutral		Conflict	
	Train	Test	Train	Test	Train	Test	Train	Test
Restaurant	2164	728	805	196	633	196	91	14
Restaurant-Hard	379	92	323	62	293	83	43	8
Laptop	987	341	866	128	460	169	45	16
Laptop-Hard	159	31	147	25	173	49	17	3

Table 3: Statistics of the datasets for ATSA task.

are not designed for aspect based sentiment analysis, but their performance exceeds that of ATAE-LSTM.

The performance of SVM (Kiritchenko et al., 2014) depends on the availability of the features it can use. Without the large amount of sentiment lexicons, SVM perform worse than neural methods. With multiple sentiment lexicons, the performance is increased by 7.6%. This inspires us to work on leveraging sentiment lexicons in neural networks in the future.

The hard test datasets consist of replicated sentences with different sentiments towards different aspects. The models which cannot utilize the given aspect information such as CNN and GCN perform poorly as expected, but GCAE has higher accuracy than other neural network models. GCAE achieves 4% higher accuracy than ATAE-LSTM on Restaurant-Large and 5% higher on SemEval-2014 on ACSA task. However, GCN, which does not have aspect modeling part, has higher score than GCAE on the original restaurant dataset. It suggests that GCN performs better than GCAE when there is only one sentiment label in the given sentence, but not on the hard test dataset.

6.3.2 ATSA

We apply the extended version of GCAE on ATSA task. On this task, the aspect terms are marked in the sentences and usually consist of multiple words. We compare IAN (Ma et al., 2017), RAM (Chen et al., 2017), TD-LSTM (Tang et al., 2016a), ATAE-LSTM (Wang et al., 2016b), and our GCAE model in Table 5. The models other than GCAE is based on LSTM and attention mechanisms. IAN has better performance than TD-LSTM and ATAE-LSTM, because two attention layers guides the representation learning of the context and the entity interactively. RAM also achieves good accuracy by combining multiple attentions with a recurrent neural network, but it needs more training time as shown in the following section. On the hard test dataset, GCAE has 1% higher accuracy than RAM on restaurant data and 1.7% higher on laptop data. GCAE uses the outputs of the small CNN over aspect terms to guide the composition of the sentiment features through the ReLU gate. Because of the gating mechanisms and the convolutional layer over aspect terms, GCAE outperforms other neural models and basic SVM. Again, large scale sentiment lexicons bring significant improvement to SVM.

6.4 Training Time

We record the training time of all models until convergence on a validation set on a desktop machine with a 1080 Ti GPU, as shown in Table 6. LSTM based models take more training time than convolutional models. On ATSA task, because of multiple attention layers in IAN and RAM, they need even more time to finish the training. GCAE is much faster than other neural models, because neither convolutional operation nor GTRU has time dependency compared with LSTM and attention layer. Therefore, it is easier for hardware and libraries to parallel the comput-

Models	Restaurant-Large		Restaurant 2014	
	Test	Hard Test	Test	Hard Test
SVM*	-	-	75.32	-
SVM + lexicons*	-	-	**82.93**	-
ATAE-LSTM	83.91±0.49	66.32±2.28	78.29±0.68	45.62±0.90
CNN	84.28±0.15	50.43±0.38	79.47±0.32	44.94±0.01
GCN	84.48±0.06	50.08±0.31	**79.67±0.35**	44.49±1.52
GCAE	**85.92±0.27**	**70.75±1.19**	79.35±0.34	**50.55±1.83**

Table 4: The accuracy of all models on test sets and on the subsets made up of test sentences that have multiple sentiments and multiple aspect terms. Restaurant-Large dataset is created by merging all the restaurant reviews of SemEval workshops within three years. '*': the results with SVM are retrieved from NRC-Canada (Kiritchenko et al., 2014).

Models	Restaurant		Laptop	
	Test	Hard Test	Test	Hard Test
SVM*	77.13	-	63.61	-
SVM + lexicons*	**80.16**	-	**70.49**	-
TD-LSTM	73.44±1.17	56.48±2.46	62.23±0.92	46.11±1.89
ATAE-LSTM	73.74±3.01	50.98±2.27	64.38±4.52	40.39±1.30
IAN	76.34±0.27	55.16±1.97	68.49±0.57	44.51±0.48
RAM	76.97±0.64	55.85±1.60	68.48±0.85	45.37±2.03
GCAE	**77.28±0.32**	**56.73±0.56**	**69.14±0.32**	**47.06±2.45**

Table 5: The accuracy of ATSA subtask on SemEval 2014 Task 4. '*': the results with SVM are retrieved from NRC-Canada (Kiritchenko et al., 2014)

Model	ATSA
ATAE	25.28
IAN	82.87
RAM	64.16
TD-LSTM	19.39
GCAE	3.33

Table 6: The time to converge in seconds on ATSA task.

Gates	Restaurant-Large		Restaurant 2014	
	Test	Hard Test	Test	Hard Test
GTU	84.62	60.25	79.31	**51.93**
GLU	84.74	59.82	79.12	50.80
GTRU	**85.92**	**70.75**	**79.35**	50.55

Table 7: The accuracy of different gating units on restaurant reviews on ACSA task.

ing process. Since the performance of SVM is retrieved from the original paper, we are not able to compare the training time of SVM.

6.5 Gating Mechanisms

In this section, we compare GLU $(\mathbf{X} * \mathbf{W} + b) \times \sigma(\mathbf{X} * \mathbf{W}_a + \mathbf{V}\boldsymbol{v}_a + b_a)$ (Dauphin et al., 2017),

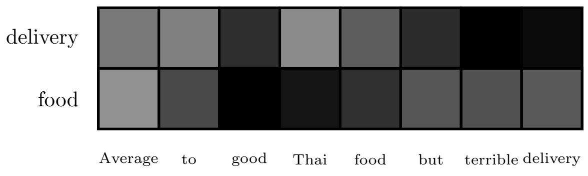


Figure 3: The outputs of the ReLU gates in GTRU.

GTU $\tanh(\mathbf{X} * \mathbf{W} + b) \times \sigma(\mathbf{X} * \mathbf{W}_a + \mathbf{V}\boldsymbol{v}_a + b_a)$ (van den Oord et al., 2016), and GTRU used in GCAE. Table 7 shows that all of three gating units achieve relatively high accuracy on restaurant datasets. GTRU outperforms the other gates. It has a convolutional layer generating aspect features via ReLU activation function, which controls the magnitude of the sentiment signals according to the given aspect information. On the other hand, the sigmoid function in GTU and GLU has the upper bound $+1$, which may not be able to distill sentiment features effectively.

7 Visualization

In this section, we take a concrete review sentence as an example to illustrate how the proposed gate GTRU works. It is more difficult to visualize the weights generated by the gates than the attention weights in other neural networks. The attention weight score is a global score over the words and the vector dimensions; whereas in our model, there are $N_{\text{word}} \times N_{\text{filter}} \times N_{\text{dimension}}$ gate outputs. Therefore, we train a small model with only one filter which is only three word wide. Then, for each word, we sum the $N_{\text{dimension}}$ outputs of the ReLU gates. After normalization, we plot the values on each word in Figure 3. Given different aspect targets, the ReLU gates would control the magnitude of the outputs of the tanh gates.

8 Conclusions and Future Work

In this paper, we proposed an efficient convolutional neural network with gating mechanisms for ACSA and ATSA tasks. GTRU can effectively control the sentiment flow according to the given aspect information, and two convolutional layers model the aspect and sentiment information separately. We prove the performance improvement compared with other neural models by extensive experiments on SemEval datasets. How to leverage large-scale sentiment lexicons in neural networks would be our future work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, pages CoRR abs–1409.0473.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent Attention Network on Memory for Aspect Sentiment Analysis. In *EMNLP*, pages 463–472.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. 2016. Very Deep Convolutional Networks for Text Classification. In *EACL*, pages 1107–1116.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *ICML*, pages 933–941.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In *ACL*, pages 49–54.

John C Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, pages 2121–2159.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*, pages 1243–1252.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural Machine Translation in Linear Time . *CoRR*, abs/1610.10099.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*, pages 1746–1751.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif M. Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *SemEval@COLING*, pages 437–442, Stroudsburg, PA, USA. Association for Computational Linguistics.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. *AAAI*, pages 2267–2273.

Himabindu Lakkaraju, Richard Socher, and Chris Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*.

Hoa T Le, Christophe Cerisara, and Alexandre Denis. 2017. Do Convolutional Networks need to be Deep for Text Classification ? *CoRR*, abs/1707.04108.

Bing Liu and Lei Zhang. 2012. A Survey of Opinion Mining and Sentiment Analysis. *Mining Text Data*, (Chapter 13):415–463.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive Attention Networks for Aspect-Level Sentiment Classification. In *IJCAI*, pages 4068–4074. International Joint Conferences on Artificial Intelligence Organization.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional Image Generation with PixelCNN Decoders. In *NIPS*, pages 4790–4798.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieva*, 2:1–135.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543.

Maria Pontiki, Dimitrios Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval@COLING*, pages 27–35, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016a. A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis. In *EMNLP*, pages 999–1005.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016b. INSIGHT-1 at SemEval-2016 Task 5 - Deep Learning for Multilingual Aspect-based Sentiment Analysis. In *SemEval@NAACL-HLT*, pages 330–336.

Richard Socher, Alex Perelygin, Jean Y Wu, and Jason Chuang. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *ACL*, pages 1556–1566.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective LSTMs for Target-Dependent Sentiment Classification. In *COLING*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *EMNLP*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect Level Sentiment Classification with Deep Memory Network. In *EMNLP*, pages 214–224.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive Neural Conditional Random Fields for Aspect-based Sentiment Analysis. In *EMNLP*, pages 616–626.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. Attention-based LSTM for Aspect-level Sentiment Classification. In *EMNLP*, pages 606–615.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory Networks. In *ICLR*, pages CoRR abs–1410.3916.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification. In *EMNLP*, pages 1660–1669.

Wei Xue, Wubai Zhou, Tao Li, and Qing Wang. 2017. Mtna: A neural multi-task model for aspect category classification and aspect term extraction on restaurant reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 151–156.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated Neural Networks for Targeted Sentiment Analysis. In *AAAI*, pages 3087–3093.

A Helping Hand: Transfer Learning for Deep Sentiment Analysis

Xin Dong
Rutgers University
New Brunswick, NJ, USA
xd48@rutgers.edu

Gerard de Melo
Rutgers University
New Brunswick, NJ, USA
gdm@demelo.org

Abstract

Deep convolutional neural networks excel at sentiment polarity classification, but tend to require substantial amounts of training data, which moreover differs quite significantly between domains. In this work, we present an approach to feed generic cues into the training process of such networks, leading to better generalization abilities given limited training data. We propose to induce sentiment embeddings via supervision on extrinsic data, which are then fed into the model via a dedicated memory-based component. We observe significant gains in effectiveness on a range of different datasets in seven different languages.

1 Introduction

Over the past decades, sentiment analysis has grown from an academic endeavour to an essential analytics tool. Across the globe, people are voicing their opinion in online social media, product review sites, booking platforms, blogs, etc. Hence, it is important to keep abreast of ongoing developments in all pertinent markets, accounting for different domains as well as different languages. In recent years, deep neural architectures based on convolutional or recurrent layers have become established as the preeminent models for supervised sentiment polarity classification. At the same time, it is also frequently observed that deep neural networks tend to be particularly data-hungry. This is a problem in many real-world settings, where large amounts of training examples may be too costly to obtain for every target domain. A model trained on movie reviews, for instance, will fare very poorly on the task of assessing restaurant or hotel reviews, let alone tweets about politicians.

In this paper, we investigate how extrinsic signals can be incorporated into deep neural networks for sentiment analysis. Numerous papers have found the use of regular pre-trained word vector representations to be beneficial for sentiment analysis (Socher et al., 2013; Kim, 2014; dos Santos and de C. Gatti, 2014). In our paper, we instead consider word embeddings specifically specialized for the task of sentiment analysis, studying how they can lead to stronger and more consistent gains, despite the fact that the embeddings were obtained using out-of-domain data.

An intuitive solution would be to concatenate regular embeddings, which provide semantic relatedness cues, with sentiment polarity cues that are captured in additional dimensions. We instead propose a bespoke convolutional neural network architecture with a separate memory module dedicated to the sentiment embeddings. Our empirical study shows that the sentiment embeddings can lead to consistent gains across different datasets in a diverse set of domains and languages if a suitable neural network architecture is used.

2 Approach

2.1 Sentiment Embedding Computation

Our goal is to incorporate external cues into a deep neural network such that the network is able to generalize better even when training data is scarce. While in computer vision, weights pre-trained on ImageNet are often used for transfer learning, the most popular way to incorporate external information into deep neural networks for text is to draw on word embeddings trained on vast amounts of word context information (Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2018). Indeed, the semantic relatedness signals provided by such representations often lead to slightly improved results in polarity classification tasks (Socher et al., 2013; Kim, 2014; dos Santos and de C. Gatti, 2014).

However, the co-occurrence-based objectives of word2vec and GloVe do not consider sentiment specifically. We thus seek to examine how complementary sentiment-specific information from an external source can give rise to further gains.

Transfer Learning. To this end, our goal is to induce sentiment embeddings that capture sentiment polarity signals in multiple domains and hence may be useful across a range of different sentiment analysis tasks. The multi-domain nature of these distinguish them from the kinds of generic polarity scores captured in sentiment polarity lexicons. We achieve this via transfer learning from trained models, benefiting from supervision on a series of sentiment polarity tasks from different domains. Given a training collection consisting of n binary classification tasks (e.g., with documents in n different domains), we learn n corresponding polarity prediction models. From these, we then extract token-level scores that are tied to specific prediction outcomes. Specifically, we train n linear models $f_i(\mathbf{x}) = \mathbf{w}_i^\intercal \mathbf{x} + b_i$ for tasks $i = 1, \ldots, n$. Then, each vocabulary word index j is assigned a new n-dimensional word vector $\mathbf{x_j} = (w_{1,j}, \cdots, w_{n,j})$ that incorporates the linear coefficients for that word across the different linear models.

A minor challenge is that naïvely using bag-of-word features can lead to counter-intuitive weights. If a word such as "*pleased*" in one domain mainly occurs after the word "*not*", while the reviews in another domain primarily used "*pleased*" in its unnegated form, then "*pleased*" would be assessed as possessing opposite polarities in different domains. To avoid this, we assume that features are preprocessed to better reflect whether words occur in a negated context. In our experiments, we simply treat occurrences of "*not* ⟨word⟩" as a single feature "*not_*⟨word⟩". Of course, one can replace this heuristic with much more sophisticated techniques that fully account for the scope of a wider range of negation constructions.

Graph-Based Extension. Most sentiment-related resources are available for the English language. To produce vectors for other languages in our experiments, we rely on cross-lingual projection via graph-based propagation (de Melo, 2015; de Melo, 2017; Dong and de Melo, 2018). At this point, we have a set of initial sentiment embedding vectors $\tilde{\mathbf{v}}_x \in \mathbb{R}^n$ for words $x \in V_0$. We assume that we have a lexical knowledge graph $G_\mathrm{L} = (V, A_\mathrm{L})$ with a node set consisting of an extended multilingual vocabulary $V \supseteq V_0$ and a set of weighted directed arcs $A_\mathrm{L} = \{(x_1, x'_1, w_1), \ldots, (x_m, x'_m, w_m)\}$. Each such arc reflects a weighted semantic connection between two vocabulary items $x, x' \in V$, where vocabulary items are words labeled with their respective language. Typically, many of the arcs in the G_L would reflect translational equivalence, but in our experiments, we also include monolingual links between semantically related words. Given this data, we aim to minimize

$$-\sum_{x \in V} \mathbf{v}_x^\intercal \left[\frac{1}{\sum\limits_{(x,x',w) \in A_\mathrm{L}} w} \sum_{(x,x',w) \in A_\mathrm{L}} w \mathbf{v}_{x'} \right] + C \sum_{x \in V_0} \|\mathbf{v}_x - \tilde{\mathbf{v}}_x\|_2 \qquad (1)$$

The first component of this objective seeks to ensure that sentiment embeddings of words accord with those of their connected words, in terms of the dot product. The second part ensures that the deviation from any available initial word vectors $\tilde{\mathbf{v}}_x$ is minimal (for some very high constant C). For optimization, we preinitialize $\mathbf{v}_x = \tilde{\mathbf{v}}_x$ for all $x \in V_0$, and then rely on stochastic gradient descent steps.

2.2 Dual-Module Memory based CNNs

To feed this sentiment information into our architecture, we propose a Dual-Module Memory based Convolutional Neural Network (DM-MCNN) approach, which incorporates a dedicated memory module to process the sentiment embeddings, as illustrated in Fig. 1. While the module with regular word embeddings enables the model to learn salient patterns and harness the nearest neighbour and linear substructure properties of word embeddings, we conjecture that a separate sentiment memory module allows for better exploiting the information brought to the table by the sentiment embeddings.

Convolutional Module Inputs and Filters. The Convolutional Module input of the DM-MCNN is a sentence matrix $\mathbf{S} \in \mathbb{R}^{s \times d}$, the rows of which represent the words of the input sentence after tokenization. In the case of $\mathbf{S}$, i.e., in the regular module, each word is represented by its conventional word vector representation. Here, s refers to the length of a sentence, and d represents the dimensionality of the regular word vectors.

We perform convolutional operations on these matrices via linear filters. Given rows representing discrete words, we rely on weight matrices $\mathbf{W} \in \mathbb{R}^{h \times d}$ with region size h. We use the notation $\mathbf{S_{i:j}}$ to denote the sub-matrix of $\mathbf{S}$ from row i to row

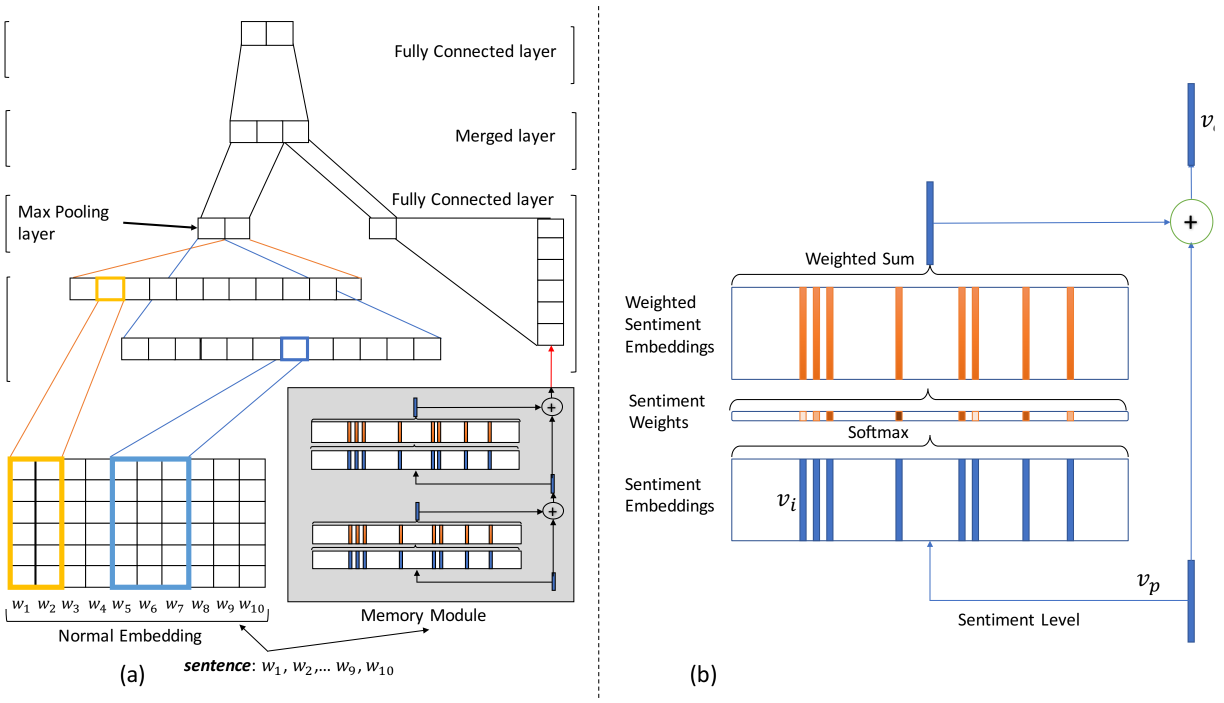


Figure 1: (a) Dual-Module Memory based Convolutional Neural Network architecture. (b) Single layer in Memory Module

j. Supposing that the weight matrix has a filter width of h, a wide convolution (Kalchbrenner et al., 2014) is induced such that out-of-range submatrix values $S_{i,j}$ with $i < 1$ or $i > s$ are taken to be zero. Thus, applying the filter on sub-matrices of $\mathbf{S}$ yields the output sequence $\mathbf{o} \in \mathbb{R}^{s+h-1}$ as

$$o_i = \mathbf{W} \odot \mathbf{S_{i:i+h-1}}, \tag{2}$$

where the $\odot$ operator provides the sum of an element-wise multiplication. Wide convolutions ensure that filters can cover words at the margins of the normal weight matrix.

Next, the c_i in feature maps $\mathbf{c} \in \mathbb{R}^{s+h-1}$ are computed as: $c_i = f(o_i + b)$, where $i = 1, \ldots, s + h - 1$, the parameter $b \in \mathbb{R}$ is a bias term, and f is an activation function.

Multiple Layers in Memory Module. The memory module obtains as input a sequence of sentiment embedding vectors for the input, and attempts to draw conclusions about the overall sentiment polarity of the entire input sequence. Given a set of sentence words $S = \{w_1, w_2, w_3, \ldots, w_n\}$, each word is mapped to its sentiment embedding vector of dimension d_s and we denote this set of vectors as V_s. The preliminary sentiment level $\mathbf{v}_\mathrm{p}$ is also a vector of dimensionality d_s. We take the mean of all sentiment vectors $\mathbf{v}_i$ for words $w_i \in S$ to initialize $\mathbf{v}_\mathrm{p}$. Next, we compute a vector $\mathbf{s}$ of similarities s_i between $\mathbf{v}_\mathrm{p}$ and each sentiment word vector $\mathbf{v}_i$, by taking the inner product, followed by ℓ_2-normalization and a softmax:

$$s_i = \frac{\exp \frac{\mathbf{v}_\mathrm{p}^\intercal \mathbf{v}_i}{\|\mathbf{v}_\mathrm{p}^\intercal \mathbf{v}_i\|_2}}{\sum_i \exp \frac{\mathbf{v}_\mathrm{p}^\intercal \mathbf{v}_i}{\|\mathbf{v}_\mathrm{p}^\intercal \mathbf{v}_i\|_2}} \tag{3}$$

As the sentiment embeddings used in our paper are generated from a linear model, the degree of correspondence between $\mathbf{v}_\mathrm{p}$ and $\mathbf{v}_i$ can adequately be assessed by the inner product. The resulting vector of scores $\mathbf{s}$ can be regarded as yielding sentiment weights for each word in the sentence. We apply ℓ_2-normalization to ensure a more balanced weight distribution. The output sentiment level vector $\mathbf{v}_\mathrm{o}$ is then a sum over the sentiment inputs $\mathbf{v}_i$ weighted by the ℓ_2-normalized vector of similarities:

$$\mathbf{v}_\mathrm{o} = \sum_i \frac{s_i}{\|\mathbf{s}\|_2} \mathbf{v}_i \tag{4}$$

This processing can be repeated in multiple passes, akin to how end-to-end memory networks for question answering often perform multiple hops (Sukhbaatar et al., 2015). While in the first iteration, $\mathbf{v}_\mathrm{p}$ was set to the mean sentiment vector, subsequent passes may allow us to iteratively refine this vector. Assuming that $\mathbf{v}_{\mathrm{o}}^{k}$ has been produced by the k-th pass, then the subsequent level $\mathbf{v}_{\mathrm{p}}^{k+1}$ in the next pass is:

$$\mathbf{v}_{\mathrm{p}}^{k+1} = \mathbf{v}_{\mathrm{o}}^{k} + \mathbf{v}_{\mathrm{p}}^{k} \tag{5}$$

The intuition here is that multiple passes can enable the model to adaptively retrieve iterative sentiment level statistics beyond the initial average sentiment information.

Merging Layer and Prediction. Subsequently, for the convolutional module, 1d-max pooling is applied to $\mathbf{c}$, which ought to capture the most prominent signals. In the memory module, the final sentiment vector is modulated by a weight matrix $\mathbf{W}_{\mathrm{s}} \in \mathbb{R}^{l \times d_{\mathrm{s}}}$ to form a feature vector of dimensionality l. In general, we can use multiple filters to obtain several features in the convolutional module, while the memory module allows for adjusting the number of passes over the memory.

Finally, the outputs of these two modules are concatenated to form a fixed-length vector, which is passed to a fully connected softmax layer to obtain the final output probabilities.

Loss Function and Training. Our loss function is the cross-entropy function

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{c \in C} y_{i,c} \ln \hat{y}_{i,c}, \tag{6}$$

where n is the number of training examples, C is the set of (two) classes, $y_{i,c}$ are ground truth labels for a given training example and class c, and $\hat{y}_{i,c}$ are corresponding label probabilities predicted by the model, as emitted by the softmax layer. We train our model using Adam optimization (Kingma and Ba, 2014) for better robustness across different datasets. Further details about our training regime follow in the Experiments section.

3 Experiments

We now turn to our extensive empirical evaluation, which assesses the effectiveness of our novel architecture with sentiment word vectors.

3.1 Experimental Setup

Datasets. For evaluation, we use real world datasets for 7 different languages, taken from a range of different sources that cover several domains. These are summarized in Table 1, with ISO 639-3 language codes. In our experimental setup, these are all cast as binary polarity classification tasks, for which we use accuracy as our evaluation metric.

Table 1: Dataset Descriptions

Language	Source	Domain	train	test
en	SST	Movies	6,920	1,821
	AFF	Food	5,945	1,189
es	SE16-T5	Restaurants	2,070	881
ru	TA	Hotels	2,387	682
de	TA	Restaurants	1,687	481
cs	TA	Restaurants	1,722	491
it	TA	Hotels	3,437	982
ja	TA	Restaurants	1,435	411

- The Stanford Sentiment Treebank (**SST**) dataset (Socher et al., 2013) consists of movie reviews taken from the Rotten Tomatoes website, including binary labels. We only used sentence-level data in our experiment.
- The SemEval-2016 Task 5 (**SE16-T5**) dataset (Pontiki et al., 2016) provides Spanish reviews of restaurants. It targeted aspect-based sentiment analysis, so we converted the entity-level annotations to sentence-level polarity labels via voting. As the number of entities per sentence is often one or very low, this process is reasonably precise. In any case, it enables us to compare the ability of different model variants to learn to recognize pertinent words.
- From TripAdvisor (**TA**), we crawled German, Russian, Italian, Czech, and Japanese reviews of restaurants and hotels. We removed three-star reviews, as these can be regarded as neutral ones, so reviews with a rating < 3 are considered negative, while those with a rating > 3 were deemed positive.
- The Amazon Fine Food Reviews **AFF** (McAuley and Leskovec, 2013) dataset provides food reviews left on Amazon. We chose a random subset of it with preprocessing as for TripAdvisor.

As there was no test set provided for TripAdvisor or for the Amazon Fine Food Reviews data, we randomly partitioned this data into training, validation, and test splits with a 80%/10%/20% ratio. Additionally, 10% of the training sets from SE16-T5 were randomly extracted and reserved for validation, while SST provides its own validation set. The new datasets are available from `http://gerard.demelo.org/sentiment/`.

Embeddings. The standard pre-trained word vectors used for English are the GloVe (Pennington et al., 2014) ones trained on 840 billion tokens of

Table 2: DM-MCNN Model Parameter Settings.

(a) General configuration.

Description		Values
Convol. Module	filter region size	(3,4,5)
	feature maps	100
	pooling	1d-max pooling
Memory Module	# passes (k)	2
	feature vector size	100
dropout rate		0.5
optimizer		Adam
activation function		ReLU
batch size		50

(b) Learning rate α used in DM-MCNN under 7 languages.

	en	es	de	ru
α	0.0004	0.0008	0.003	0.003
	cs	it	ja	
α	0.003	0.003	0.003	

Common Crawl data[1], while for other languages, we rely on the Facebook fastText Wikipedia embeddings (Bojanowski et al., 2016) as input representations. All of these are 300-dimensional. The vectors are either fed to the CNN, or to the convolutional module of the DM-MCNN during initialization, while unknown words are initialized with zeros. All words, including the unknown ones, are fine-tuned during the training process.

For our transfer learning approach, our experiments rely on the multi-domain sentiment dataset by Blitzer et al. (2007), collected from Amazon customers reviews. This dataset includes 25 categories of products and is used to generate our sentiment embeddings using linear models. Specifically, we train linear SVMs using scikit-learn to extract word coefficients in each domain and also for the union of all domains together, yielding a 26-dimensional sentiment embedding.

For comparison and analysis, we also consider several alternative forms of infusing external cues. Firstly, lexicon-driven methods have often been used for domain-independent sentiment analysis. We consider a recent sentiment lexicon called VADER (Hutto and Gilbert, 2014). The polarity scores assigned to words by the lexicon are taken as the components of a set of 1-dimensional word vectors (dividing the original scores by the difference between max and min polarity scores for normalization). Secondly, as another particularly strong alternative, we consider the SocialSent Reddit community-specific lexicons mined by the Stanford NLP group (Hamilton et al., 2016). These contain separate domain-specific scores for 250 different Reddit communities, and hence result in 250-dimensional embeddings.

For cross-lingual projection, we extract links between words from a 2017 dump of the English edition of Wiktionary. We restrict the vocabulary link set to include the languages in Table 1, mining corresponding translation, synonymy, derivation, and etymological links from Wiktionary.

Neural Network Details. For CNNs, we make use of the well-known CNN-non-static architecture and hyperparameters proposed by Kim (2014), with a learning rate of 0.0006, obtained by tuning on the validation data. For our DM-MCNN models, the configuration of the convolutional module is the same as for CNNs, and the remaining hyperparameter values were as well tuned on the validation sets. An overview of the relevant network parameter values is given in Table 2.

For greater efficiency and better convergence properties, the training relies on mini-batches. Our implementation considers the maximal sentence length in each mini-batch and zero-pads all other sentences to this length under convolutional module, thus enabling uniform and fast processing of each mini-batch. All neural network architectures are implemented using the PyTorch framework[2].

3.2 Results and Analysis

Baseline Results. Our main results are summarized in Table 3. We compare both regular CNNs and our dual-module alternative DM-MCNNs under a variety of settings. A common approach is to use a CNN with randomly initialized word vectors. Comparing this to CNNs with GloVe/fastText embeddings, where GloVe is used for English, and fastText is used for all other languages, we observe substantial improvements across all datasets. This shows that word vectors do tend to convey pertinent word semantics signals that enable models to generalize better. Note also that the accuracy using GloVe on the English movies review dataset is consistent with numbers reported in previous work (Zhang and Wallace, 2015).

Dual-Module Architecture. Next, we consider our DM-MCNNs with their dual-module mechanism to take advantage of transfer learning. We observe fairly consistent and sometimes quite substan-

[1]https://nlp.stanford.edu/projects/glove/

[2]http://pytorch.org

Table 3: Accuracy on several different English and non-English datasets from different domains, comparing our architecture against CNNs. Rest.: restaurants domain.

Approach	d	en		es	ru	de	cs	it	ja
		Movies	Food	Rest.	Hotels	Rest.	Rest.	Hotels	Rest.
CNN									
— Random Init.	300	80.78	86.63	81.50	90.18	88.09	90.00	93.18	78.59
— Word Vec. Init.	300	85.72	87.97	85.13	92.82	92.10	92.46	96.20	77.62
Our Approach									
— With fine-tuning	300/26	**86.99**	**90.08**	85.02	93.40	93.14	93.08	95.50	**85.40**
— No fine-tuning	300/26	86.38	88.81	**85.70**	**94.87**	**94.59**	**93.48**	96.20	77.62
CNN with Concatenated Sentiment Embeddings									
— VADER	301	85.89	88.39	84.90	92.31	88.36	93.08	96.34	77.62
— SocialSent	550	84.90	88.48	82.63	92.23	91.48	86.56	94.51	76.64
— Our Embeddings	326	86.05	89.07	84.56	92.72	93.56	91.24	95.78	77.62
Our Model with Alternative Sentiment Embeddings									
— Random	300/26	86.16	87.97	85.24	93.99	93.14	92.67	96.20	80.29
— VADER	300/1	86.33	88.39	84.45	94.18	92.31	92.87	96.48	75.43
— SocialSent	300/250	86.38	87.89	83.09	93.40	92.31	93.28	**96.62**	81.02

tial gains over CNNs with just the GloVe/fastText vectors. We see that the sentiment embeddings provide important complementary signals beyond what is provided in regular word embeddings, and that our dual-module approach succeeds at exploiting these signals across a range of different domains and languages. Our transfer learning approach leads to sentiment embeddings that capture signals from multiple domains. The model successfully picks the pertinent parts of this signal for datasets from domains as different as movie reviews and food reviews.

We report results for two different training conditions. In the first condition (with fine-tuning), the sentiment embedding matrix is preinitialized using the data from our transfer learning procedure, but the model is then able to modify these arbitrarily via backpropagation. In the second condition (no fine-tuning), we simply use our sentiment embedding matrix as is, and do not update it. Instead, the model is able to update its various other parameters, particularly its various weight matrices and bias vectors. While both training conditions outperform the CNN baseline, there is no obvious winner among the two. When the training data set is very small and hence there is a significant risk of overfitting, one may be best advised to forgo fine-tuning. In contrast, when it is somewhat larger (as for our English datasets, which each have over 5,000 training instances) or when the language is particularly idiosyncratic or not covered sufficiently well by our cross-lingual projection procedure (such as perhaps for Japanese), then fine-tuning is recommended. In this case, fine-tuning may allow the model to adjust the embeddings to cater to domain-specific meanings and corpus-specific correlations, while also overcoming possible sparsity of the cross-lingual vectors resulting from a lack of coverage of the translation dictionary.

It is important to note that many of the results in Table 3 stem from embeddings that were created automatically using cross-lingual projection. Our transfer learning embeddings were induced from entirely English data. Although the automatically projected cross-lingual embeddings are very noisy and limited in their coverage, particularly with respect to inflected forms, our model succeeds in exploiting them to obtain substantial gains in several different languages and domains.

Alternative Embedding Methods. For a more detailed analysis, we conducted additional experiments with alternative embedding conditions. In particular, as a simpler means of achieving gains over standard CNNs, we propose to use CNNs with word vectors augmented with sentiment cues. Given that regular word embeddings appear to be useful for capturing semantics, one may conjecture that extending these word vectors with additional dimensions to capture sentiment information can lead to improved results. For this, we simply concatenate the regular word embeddings with different forms of sentiment embeddings that we have obtained, including those from the sentiment lexicon VADER, from the Stanford SocialSent project, and from our transfer learning procedure via Amazon reviews. To conduct these experiments, we also produced cross-lingual projections of the VADER and SocialSent embedding data.

The results of using these embeddings as opposed to regular ones are somewhat mixed. Concatenating the VADER embeddings or our transfer learning ones leads to minor improvements on English, and our cross-lingual projection of them leads to occasional gains, but the results are far from consistent. Even on English, adding the 250-dimensional SocialSent embedding seems to degrade the effectiveness of the CNN, although all input information that was previously there continues to be provided to it. This suggests that a simple concatenation may harm the model's ability to harness the semantic information carried by regular word vectors. This risk seems more pronounced for larger-dimensional sentiment embeddings.

In contrast, with our DM-MCNNs approach, the sentiment information is provided to the model in a separate memory module that makes multiple passes over this data before combining it with the regular CNN module's signals. Thus, the model can exploit the two kinds of information independently, and learn a suitable way to aggregate them to produce an overall output classification.

This hence demonstrates not only that the sentiment embeddings tend to provide important complementary signals but also that a dual-module approach is best-suited to incorporate such signals into deep neural models.

We also analysed our DM-MCNNs with alternative embeddings. When we feed random sentiment embeddings into them, not unexpectedly, in many cases the results do not improve much. This is because our memory module has been designed to leverage informative prior information and to reweight its signals based on this assumption. Hence, it is important to feed genuine sentiment cues into the memory module. Yet, on some languages, we nevertheless note improvements over the CNN baseline. In these cases, even if similarities between pairs of sentiment vectors initially do not carry any significance, backpropagation may have succeeded in updating the sentiment embedding matrix such that eventually the memory module becomes able to discern salient patterns in the data.

We also considered our DM-MCNNs when feeding the VADER or SocialSent embeddings into the memory module. In this case, it also mostly succeeded in outperforming the CNN baseline. In fact, on the Italian TripAdvisor dataset, the SocialSent embeddings yielded the overall strongest results. In all other cases, however, our transfer learning embeddings proved more effective. We believe that this is because they are obtained in a data-driven manner based on an objective that directly seeks to optimize for classification accuracy.

Influence of Training Set Size. To look into the effect of our approach with restricted training data, we first consider the SST dataset as an instructive example. We set the training set size to 20%, 50%, 100% of its original size and compared our full dual module model with sentiment embeddings against state-of-the-art methods.

The results are given in Table 4. Our dual module CNN has a sizeable lead over other methods when only using 20% of SST training set. Given that we study how to incorporate extrinsic cues into a deep neural model, we consider CNN-Rule-q (Hu et al., 2016) and Gumbel Tree-LSTM (Choi et al., 2017) as the relevant baseline methods. The CNN-Rule-q method used an iterative distillation method that exploits structured information from logical rules, which for SST is based on the word *but* to determine the weights in the neural network. The Gumbel Tree-LSTM approach incorporates a Straight-Through Gumbel-Softmax into a tree-structured LSTM architecture that learns how to compose task-specific tree structures starting from plain raw text. They all require a large amount of data to pick up sufficient information during training, while our method is able to efficiently capture sentiment information from our transfer learning even though the data is scarce.

For further analysis, we also artificially reduce the training set sizes to 50% of the original sizes given in Table 1 for our multilingual datasets. The results are plotted in Fig. 2. We compare: 1) the CNN model baseline, 2) the CNN model but concatenating our sentiment embeddings from transfer learning, and 3) our full dual module model with these sentiment embeddings. We already saw in Table 3 that we obtain reasonable gains over generic embeddings when using the full training sets.

In Fig. 2, we additionally observe that the gains are overall much more remarkable on smaller training sets. This shows that the sentiment embeddings are most useful when they are of high quality and domain-specific training data is scarce, although a modest amount of training data is still needed for the model to be able to adapt to the target domain.

Inspection of the DM-MCNN-learned Deep Sentiment Information. To further investigate what the model is learning, we examine the changes of weights of words on the English SST dataset when using the VADER sentiment embeddings

Table 4: Accuracy on SST with increasing training sizes

Model	*20%*	*50%*	*100%*
CNN (Kim, 2014)	83.14	84.29	85.72
CNN-Rule-q (Hu et al., 2016)	83.75	85.45	86.49
Gumbel Tree-LSTM (Choi et al., 2017)	84.04	84.83	86.80
DC-MCNN (ours)	**85.06**	**86.16**	**86.99**

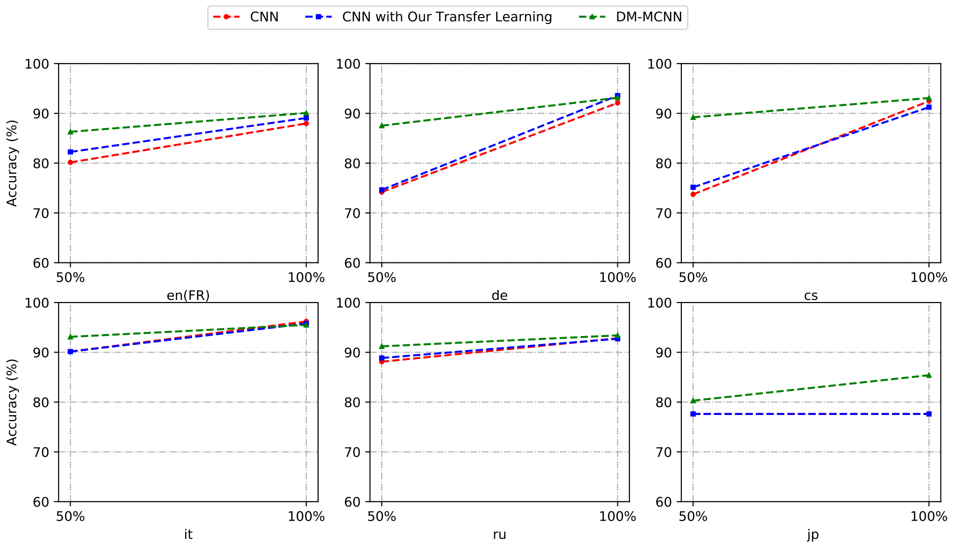


Figure 2: Effectiveness of three embedding alternatives on 6 languages at a reduced training size (comparing 50% and 100%).

with DM-MCNNs. Although these are not as powerful as our transfer learning embeddings, the VADER embeddings are the most easily interpretable here, since they are one-dimensional, and thus can be regarded as word-specific weights. The result is visualized in Fig. 3. Here, the dark-shaded segments (in blue) refer to the original weights, while the light-shaded segments (in red) refer to the adjusted weights after training. The medium-shaded segments (in purple) reflect the overlap between the two. Hence, whenever we observe a dark (blue) segment above a medium (purple) segment in a bar, we can infer that the fine-tuned weight for a word (e.g., for "*plays*" in Fig. 3) was lower than the original weight of that word. Conversely, whenever we observe a light (red) segment at the top, the weight increased during training (e.g., for *hilarious*). Generally, dark (blue) segments reflect decreased weight magnitudes and light (red) ones reflect increased weight magnitudes, both on the positive and on the negative side.

We consider in Fig. 3 the top 50 weight changes only of words that were already covered by the original VADER sentiment embeddings. Here, it is worth noting that the weight magnitudes of positive words such as "*laugh*", "*appealing*" and negative words such as "*lack*", "*missing*" increase further, while words such as "*damn*", "*interest*", "*war*" see decreases in magnitude, presumably due to their ambiguity and context (e.g., "*damn good*", "*lost the interest*", descriptions of war movies). Hence, the figure confirms that our DM-MCNN approach is able to exploit and customize the provided sentiment weights for the target domain. However, unlike the VADER data, our transfer learning approach results in multi-dimensional sentiment embeddings that can more easily capture multiple domains right from the start, thus making it possible to use them even without further fine-tuning.

4 Related Work

Sentiment Mining and Embeddings. There is a long history of work on collecting word polarity scores manually (Hu and Liu, 2004) or via graph-based propagation from seeds (Kim and Hovy, 2004; Baccianella et al., 2010). Maas et al. (2011) present a probabilistic topic model that exploits sentiment supervision during training, leading to rep-

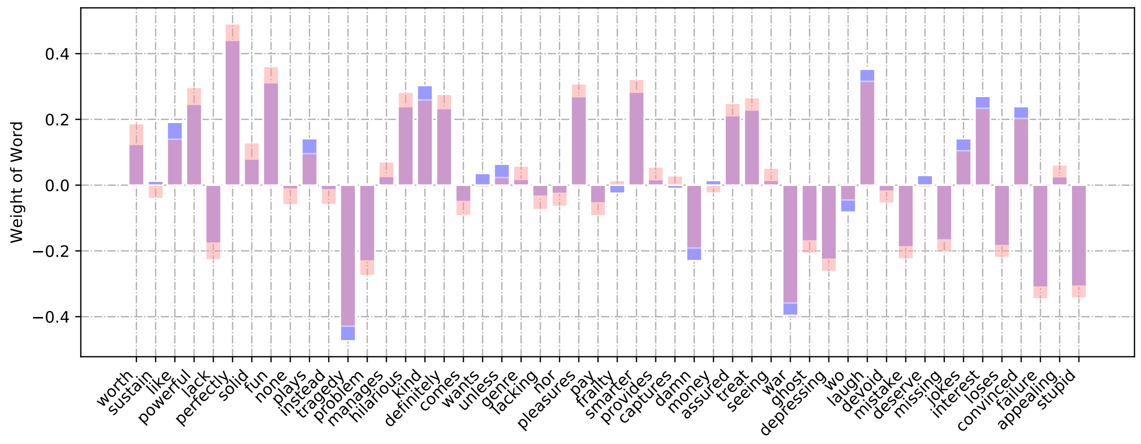


Figure 3: Top 50 weight changes of words fine-tuned by the sentiment memory module of the DM-MCNN, using the one-dimensional VADER embeddings, but considering only words with non-zero values in the original VADER data. Here, the dark shade (blue) refers to the original value of word vectors, while the light shade (red) refers to their fine-tuned values after training. The medium intensity (purple) corresponds to the overlap between the original and fine-tuned word vectors.

resentations that include sentiment signals. However, in their experiments, the semantic-only models mostly outperform the corresponding full models with extra sentiment signals. Tang et al. (2014) showed that one can acquire sentiment information by learning from millions of training examples via distant supervision. While prior work used such signals for rule-based sentiment analysis or for feature engineering in SVMs and other shallow models, our study examines how they are best be incorporated into deep neural models, as the baseline of naïvely feeding them into the model does not work sufficiently well.

Neural Architectures. In terms of architectures, deep recursive neural networks (Socher et al., 2013) were soon outperformed by deep convolutional and recurrent neural networks (İrsoy and Cardie, 2014; Kim, 2014). Recent work has investigated more involved models, with ingredients such as Tree-LSTMs (Tai et al., 2015; Looks et al., 2017), hierarchical attention (Yang et al., 2016), user and product attention (Chen et al., 2016), aspect-specific modeling (Wang et al., 2015), and part of speech-specific transition functions (Huang et al., 2017). Large ensemble models also tend to outperform individually trained sentiment analysis models (Looks et al., 2017). The goal of our study is not necessarily to devise the most sophisticated state-of-the-art neural architecture, but to demonstrate how external sentiment cues can be incorporated such architectures. Our initial explorations relied on a simple dual-channel convolutional neural network (Dong and de Melo, 2018). The present work proposes a more sophisticated approach, drawing on ideas from attention mechanisms in machine translation (Bahdanau et al., 2014) as well as from memory networks (Weston et al., 2014) and iterative attention (Kumar et al., 2015), which have proven useful for tasks such as question answering. We incorporate these ideas into a separate memory module that operates alongside the regular convolutional module.

5 Conclusions

Deep neural networks are widely used in sentiment polarity classification, but suffer from their dependence on very large annotated training corpora. In this paper, we study how to incorporate extrinsic cues into the network, beyond just generic word embeddings. We have found that this is best achieved using a dual-module approach that encourages the learning of models with favourable generalization abilities. Our experiments show that this can lead to gains across a number of different languages and domains. Our embeddings and multilingual datasets are freely available from `http://gerard.demelo.org/sentiment/`.

Acknowledgments

This research is funded in part by ARO grant no. W911NF-17-C-0098 as part of the DARPA SocialSim program.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*. European Language Resources Association.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659, Austin, Texas. Association for Computational Linguistics.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *arXiv preprint arXiv:1707.02786*.

Gerard de Melo. 2015. Wiktionary-based word embeddings. In *Proceedings of MT Summit XV*.

Xin Dong and Gerard de Melo. 2018. Cross-lingual propagation for deep sentiment analysis. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD 2004: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA. ACM.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Minlie Huang, Qiao Qian, and Xiaoyan Zhu. 2017. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Trans. Inf. Syst.*, 35(3):26:1–26:27.

C.J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proc. ICWSM-14*.

Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of Coling 2004*, pages 1367–1373, Geneva, Switzerland. COLING.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. *CoRR*, abs/1702.02181.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908. ACM.

Gerard de Melo. 2017. Inducing conceptual embedding spaces from Wikipedia. In *Proceedings of WWW 2017*. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. *ArXiv e-prints*.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. *Proceedings of SemEval*, pages 19–30.

Cícero Nogueira dos Santos and Maíra A. de C. Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING 2014*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland. Association for Computational Linguistics.

Linlin Wang, Kang Liu, Zhu Cao, Jun Zhao, and Gerard de Melo. 2015. Sentiment-aspect extraction based on Restricted Boltzmann Machines. In *Proceedings of ACL 2015*, pages 616–625.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Cold-Start Aware User and Product Attention for Sentiment Classification

Reinald Kim Amplayo and **Jihyeok Kim** and **Sua Sung** and **Seung-won Hwang**
Yonsei University
Seoul, South Korea
{rktamplayo, zizi1532, dormouse, seungwonh}@yonsei.ac.kr

Abstract

The use of user/product information in sentiment analysis is important, especially for *cold-start* users/products, whose number of reviews are very limited. However, current models do not deal with the cold-start problem which is typical in review websites. In this paper, we present Hybrid Contextualized Sentiment Classifier (HCSC), which contains two modules: (1) a fast word encoder that returns word vectors embedded with short and long range dependency features; and (2) Cold-Start Aware Attention (CSAA), an attention mechanism that considers the existence of cold-start problem when attentively pooling the encoded word vectors. HCSC introduces **shared** vectors that are constructed from similar users/products, and are used when the original **distinct** vectors do not have sufficient information (i.e. cold-start). This is decided by a frequency-guided selective gate vector. Our experiments show that in terms of RMSE, HCSC performs significantly better when compared with on famous datasets, despite having less complexity, and thus can be trained much faster. More importantly, our model performs significantly better than previous models when the training data is sparse and has cold-start problems.

1 Introduction

Sentiment classification is the fundamental task of sentiment analysis (Pang et al., 2002), where we are to classify the sentiment of a given text. It is widely used on online review websites as they contain huge amounts of review data that can be classified a sentiment. In these websites, a sentiment is usually represented as an intensity (e.g. 4 out of 5). The reviews are written by users who have bought a product. Recently, sentiment analysis research has focused on personalization (Zhang, 2015) to recommend product to users, and vise versa.

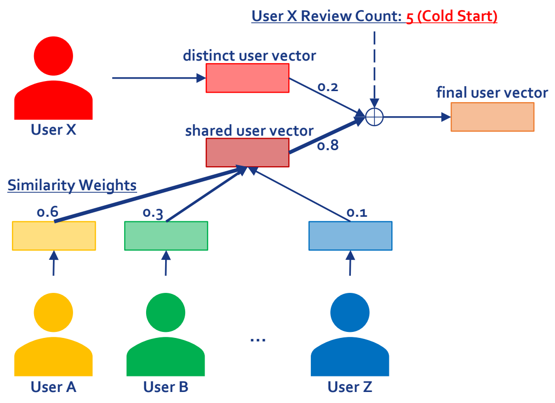


Figure 1: Conceptual schema of HCSC applied to users. The same idea can be applied to products.

To this end, many have used user and product information not only to develop personalization but also to improve the performance of the classification model (Tang et al., 2015). Indeed, these information are important in two ways. First, some expressions are user-specific for a certain sentiment intensity. For example, the phrase "*very salty*" may have different sentiments for a person who likes salty food and a person who likes otherwise. This is also apparent in terms of products. Second, these additional contexts help mitigate data sparsity and cold-start problems. Cold-start is a problem when the model cannot draw useful information from users/products where data is insufficient. User and product information can help by introducing a frequent user/product with similar attributes to the cold-start user/product.

Thanks to the promising results of deep neural networks to the sentiment classification task

(Glorot et al., 2011; Tang et al., 2014), more recent models incorporate user and product information to convolutional neural networks (Tang et al., 2015) and deep memory networks (Dou, 2017), and have shown significant improvements. The current state-of-the-art model, NSC (Chen et al., 2016a), introduced an attention mechanism called UPA which is based on user and product information and applied this to a hierarchical LSTM. The main problem with current models is that they use user and product information naively as an ordinary additional context, not considering the possible existence of cold-start problems. This makes NSC more problematic than helpful in reality since majority of the users in review websites have very few number of reviews.

To this end, we propose the idea shown in Figure 1. It can be described as follows: If the model does not have enough information to create a user/product vector, then we use a vector computed from other user/product vectors that are similar. We introduce a new model called Hybrid Contextualized Sentiment Classifier (HCSC), which consists of two modules. First, we build a fast yet effective word encoder that accepts word vectors and outputs new encoded vectors that are contextualized with short- and long-range contexts. Second, we combine these vectors into one pooled vector through a novel attention mechanism called Cold-Start Aware Attention (CSAA). The CSAA mechanism has three components: (a) a user/product-specific **distinct vector** derived from the original user/product information of the review, (b) a user/product-specific **shared vector** derived from other users/products, and (c) a frequency-guided **selective gate** which decides which vector to use. Multiple experiments are conducted with the following results: In the original non-sparse datasets, our model performs significantly better than the previous state-of-the-art, NSC, in terms of RMSE, despite being less complex. In the sparse datasets, HCSC performs significantly better than previous competing models.

2 Related work

Previous studies have shown that using additional contexts for sentiment classification helps improve the performance of the classifier. We survey several competing baseline models that use user and product information and other models using other kinds of additional context.

Baselines: Models with user and product information User and product information are helpful to improve the performance of a sentiment classifier. This argument was verified by Tang et al. (2015) through the observation at the consistency between user/product information and the sentiments and expressions found in the text. Listed below are the following models that employ user and product information:

- **JMARS** (Diao et al., 2014) jointly models the aspects, ratings, and sentiments of a review while considering the user and product information using collaborative filtering and topic modeling techniques.
- **UPNN** (Tang et al., 2015) uses a CNN-based classifier and extends it to incorporate user- and product-specific text preference matrix in the word level which modifies the word meaning.
- **TLFM+PRC** (Song et al., 2017) is a text-driven latent factor model that unifies user- and product-specific latent factor models represented using the consistency assumption by Tang et al. (2015).
- **UPDMN** (Dou, 2017) uses an LSTM classifier as the document encoder and modifies the encoded vector using a deep memory network with other documents of the user/product as the memory.
- **TUPCNN** (Chen et al., 2016b) extends the CNN-based classifier by adding temporal user and product embeddings, which are obtained from a sequential model and learned through the temporal order of reviews.
- **NSC** (Chen et al., 2016a) is the current state-of-the-art model that utilizes a hierarchical LSTM model (Yang et al., 2016) and incorporates user and product information in the attention mechanism.

Models with other additional contexts Other additional contexts used previously are spatial (Yang et al., 2017) and temporal (Fukuhara et al., 2007) features which help contextualize the sentiment based on the location where and the time when the text is written. Inferred contexts were also used as additional contexts for sentiment classifiers, such as latent topics (Lin and He, 2009) and aspects (Jo and Oh, 2011) from a topic model, argumentation features (Wachsmuth et al., 2015), and more recently, latent review clusters (Amplayo and Hwang, 2017). These additional con-

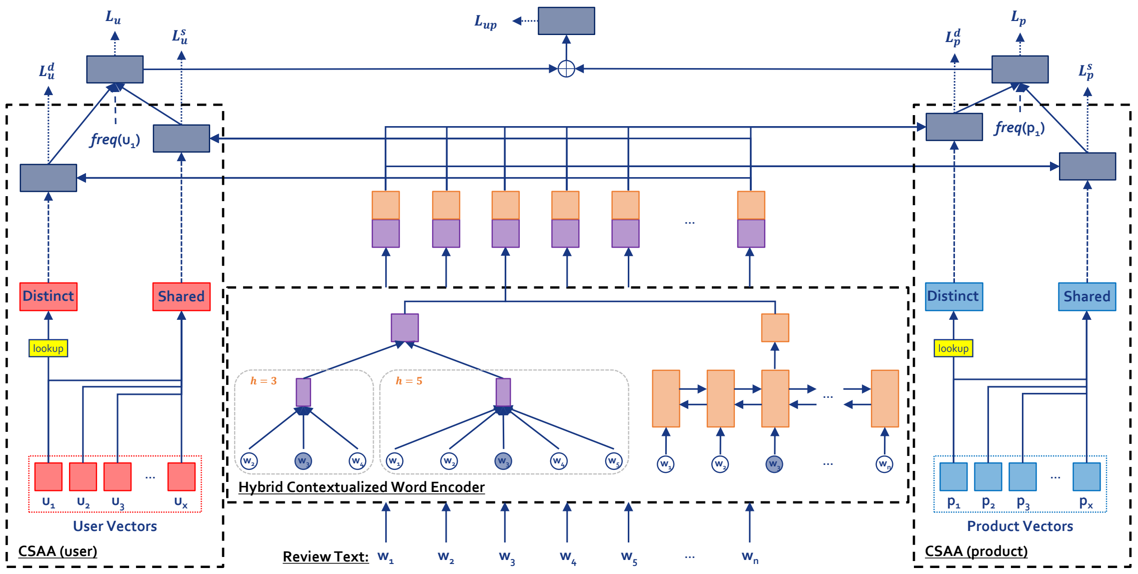


Figure 2: Full architecture of HCSC, which consists of the Hybrid Contextualized Word Encoder (middle), and user-specific (left) and product-specific (right) Cold-Start Aware Attention (CSAA).

texts were especially useful when data is sparse, i.e. number of instances is small or there exists cold-start entities.

Our model differs from the baseline models mainly because we consider the possible existence of the data sparsity problem. Through this, we are able to construct more effective models that are comparably powerful yet more efficient complexity-wise than the state-of-the-art, and are better when the training data is sparse. Ultimately, our goal is to demonstrate that, similar to other additional contexts, user and product information can be used to effectively mitigate the problem caused by cold-start users and products.

3 Our model

In this section, we present our model, **Hybrid Contextualized Sentiment Classifier (HCSC)**[1] which consists of a fast hybrid contextualized word encoder and an attention mechanism called Cold-Start Aware Attention (CSAA). The word encoder returns word vectors with both local and global contexts to cover both short and long range dependency relationship between words. The CSAA then incorporates user and product information to the contextualized words through an attention mechanism that considers the possible existence of cold-start problems. The full architecture of the model is presented in Figure 2. We describe the subparts of the model below.

3.1 Hybrid contextualized word encoder

The base model is a word encoder that transforms vectors of words $\{w_i\}$ in the text to new word vectors. In this paper, we present a fast yet very effective word encoder based on two different off-the-shelf classifiers.

The first part of HCWE is based on a CNN model which is widely used in text classification (Kim, 2014). This encoder contextualizes words based on local context words to capture short range relationships between words. Specifically, we do the convolution operation using filter matrices $W_f \in \mathbb{R}^{h \times d}$ with filter size h to a window of h words. We do this for different sizes of h. This produces new feature vectors $c_{i,h}$ as shown below, where $f(.)$ is a non-linear function:

$$c_{i,h} = f([w_{i-(h-1)/2}; ...; w_{i+(h-1)/2}]^\top W_f + b_f)$$

The convolution operation reduces the number of words differently depending on the filter size h. To prevent loss of information and to produce the same amount of feature vectors $c_{i,h}$, we pad the texts dynamically such that when the filter size is h, the number of paddings on each side is $(h-1)/2$. This requires the filter sizes to be odd numbers. Finally, we concatenate all feature vectors of different h's for each i as the new word vector:

$$w_{cnn_i} = [c_{i,h_1}; c_{i,h_2}; ...]$$

[1]The data and code used in this paper are available here: `https://github.com/rktamplayo/HCSC`.

The second part of HCWE is based on an RNN model which is used when texts are longer and include word dependencies that may not be captured by the CNN model. Specifically, we use a bidirectional LSTM and concatenate the forward and backward hidden state vectors as the new word vector, as shown below:

$$\begin{aligned}\overrightarrow{h}_i &= LSTM(w_i, \overrightarrow{h}_{i-1}) \\ \overleftarrow{h}_i &= LSTM(w_i, \overleftarrow{h}_{i+1}) \\ w_{rnn_i} &= [\overrightarrow{h}_i; \overleftarrow{h}_i]\end{aligned}$$

The answer to the question whether to use local or global context to encode words for sentiment classification is still unclear, and both CNN and RNN models have previous empirical evidence that they perform better than the other (Kim, 2014; McCann et al., 2017). We believe that both short and long range relationships, captured by CNN and RNN respectively, are useful for sentiment classification. There are already previous attempts to intricately combine both CNN and RNN (Zhou et al., 2016), resulting to a slower model. On the other hand, HCWE resorts to combine them by simply concatenating the word vectors encoded from both CNN and RNN encoders, i.e. $w_i = [w_{cnn_i}; w_{rnn_i}]$. This straightforward yet fast alternative outputs a word vector with semantics contextualized from both local and global contexts. Moreover, they perform as well as complex hierarchical structured models (Yang et al., 2016; Chen et al., 2016a) which train very slow.

3.2 Cold-start aware attention

Incorporating the user and product information of the text as context vectors u and p to attentively pool the word vectors, i.e. $e(w_i, u, p) = v^\top tanh(W_w w_i + W_u u + W_p p + b)$, has been proven to improve the performance of sentiment classifiers (Chen et al., 2016a). However, this method assumes that the user and product vectors are always present. This is not the case in real world settings where a user/product may be new and has just got its first review. In this case, the vectors u and p are rendered useless and may also contain noisy signals that decrease the overall performance of the models.

To this end, we present an attention mechanism called Cold-Start Aware Attention (CSAA). CSAA operates on the idea that a cold-start user/product can use the information of other similar users/products with sufficient number of reviews. CSAA separates the construction of pooled vectors for user and for product, unlike previous methods that use both user/product information to create a single pooled vector. Constructing a user/product-specific pooled vector consists of three parts: the distinct pooled vector created using the original user/product, the shared pooled vector created using similar users/products, and the selective gate to select between the distinct and shared vectors. Finally, the user- and product-specific pooled vectors are combined into one final pooled vector.

In the following paragraphs, we discuss the step-by-step process on how the user-specific pooled vector is constructed. A similar process is done to construct the product-specific pooled vector, but is not presented here for conciseness.

The user-specific **distinct pooled vector** v^d_u is created using a method similar to the additive attention mechanism (Bahdanau et al., 2014), i.e. $v^d_u = att(\{w_i\}, u)$, where the context vector is the distinct vector of user u, as shown in the equation below. An equivalent method is used to create the distinct product-specific pooled vector v^d_p.

$$\begin{aligned}e^d_u(w_i, u) &= v^{d\top} tanh(W^d_w w_i + W^d_u u + b^d) \\ a^d_{u_i} &= \frac{exp(e^d_u(w_i, u))}{\sum_j exp(e^d_u(w_j, u))} \\ v^d_u &= \sum_i a^d_{u_i} \times w_i\end{aligned}$$

The user-specific **shared pooled vector** v^s_u is created using the same method above, but using a shared context vector u'. The shared context vector u' is constructed using the vectors of other users and weighted based on a similarity weight. Similarity is defined as how similar the word usages of two users are. This means that if a user u_k uses words similarly to the word usage of the original user u, then u_k receives a high similarity weight. The similarity weight $a^s_{u_k}$ is calculated as the softmax of the product of $\mu(\{w_i\})$ and u_k with a project matrix in the middle, where $\mu(\{w_i\})$ is the average of the word vectors. The similarity weights are used to create u', as shown below. Similar method is used for the shared product-specific pooled vector v^s_p.

$$
\begin{aligned}
e^s_u(\mu(\{w_i\}), u_k) &= \mu(\{w_i\}) W^s_u u_k \\
a^s_{u_k} &= \frac{exp(e^s_u(w_i, u_k))}{\sum_j exp(e^s_u(w_i, u_j))} \\
u' &= \sum_k a^s_{u_k} \times u_k \\
v^s_u &= att(\{w_i\}, u')
\end{aligned}
$$

We select between the user-specific distinct and shared pooled vector, v^d_u and v^s_u, into one user-specific pooled vector v_u through a gate vector g_u. The vector g_u should put more weight to the distinct vector when user u is not cold-start and to the shared vector when u is otherwise. We use a **frequency-guided selective gate** that utilizes the frequency, i.e. the number of reviews user u has written. The challenge is that we do not know how many reviews should be considered cold-start or not. This is automatically learned through a two-parameter Weibull cumulative distribution where given the review frequency of the user $f(u)$, a learned shape vector k_u and a learned scale vector λ_u, a probability vector is sampled and is used as the gate vector g_u to create v_u, according to the equation below. We normalized $f(u)$ by dividing it to the average user review frequency. The relu function ensures that both k_u and λ_u are non-negative vectors. The final product-specific pooled vector v_p is created in a similar manner.

$$
\begin{aligned}
g_u &= 1 - exp\Bigg(- \Big(\frac{f(u)}{relu(\lambda_u)} \Big)^{relu(k_u)} \Bigg) \\
v_u &= g_u \times v^d_u + (1 - g_u) \times v^s_u
\end{aligned}
$$

Finally, we combine both the user- and product-specific pooled vector, v_u and v_p, into one pooled vector v_{up}. This is done by using a gate vector g_{up} created using a sigmoidal transformation of the concatenation of v_u and v_p, as illustrated in the equation below.

$$
\begin{aligned}
g_{up} &= \sigma(W_g[v_u; v_p] + b_g) \\
v_{up} &= g_{up} \times v_u + (1 - g_{up}) \times v_p
\end{aligned}
$$

We note that our attention mechanism can be applied to any word encoders, including the basic bag of words (BoW) to more recent models such as CNN and RNN. Later (in Section 4.2), we show that CSAA improves the performance of simpler models greatly.

3.3 Training objective

Normally, a sentiment classifier transforms the final vector v_{up}, usually in a linear fashion, into a vector with a dimension equivalent to the number of classes C. A softmax layer is then used to obtain a probability distribution y' over the sentiment classes. Finally, the full model uses a cross-entropy over all training documents D as objective function L during training, where y is the gold probability distribution:

$$
\begin{aligned}
y' &= softmax(W v_{up} + b) \\
L &= - \sum_{d \in D} \sum_{c \in C} y^{(d)}_c \cdot \log(y'^{(d)}_c)
\end{aligned}
$$

However, HCSC has a nice architecture which can be used to improve the training. It contains seven pooled vectors $\mathbb{V} = \{v^d_u, v^d_p, v^s_u, v^s_p, v_u, v_p, v_{up}\}$ that are essentially in the same vector space. This is because these vectors are created using weighted sums of either the encoded word vectors through attention or the parent pooled vectors through the selective gates. Therefore, we can train separate classifiers for each pooled vectors using the same parameters W and b. Specifically, for each $v \in \mathbb{V}$, we calculate the loss L_v using the above formulas. The final loss is then the sum of all the losses, i.e. $L = \sum_{v \in \mathbb{V}} L_v$.

4 Experiments

In this section, we present our experiments and the corresponding results. We use the models described in Section 2 as baseline models: **JMARS** (Diao et al., 2014), **UPNN** (Tang et al., 2015), **TLFM+PRC** (Song et al., 2017), **UPDMN** (Dou, 2017), **TUPCNN** (Chen et al., 2016b), and **NSC** (Chen et al., 2016a), where **NSC** is the model with state-of-the-art results.

4.1 Experimental settings

Implementation We set the size of the word, user, and product vectors to 300 dimensions. We use pre-trained GloVe embeddings[2] (Pennington et al., 2014) to initialize our word vectors. We simply set the parameters for both BiLSTMs and CNN to produce an output with 300 dimensions: For the BiLSTMs, we set the state sizes of the LSTMs to 75 dimensions, for a total of 150 dimensions. For CNN, we set $h = 3, 5, 7$, each with 50

[2] `https://nlp.stanford.edu/projects/glove/`

Datasets	Classes	Train			Dev			Test		
		#docs	#users	#prods	#docs	#users	#prods	#docs	#users	#prods
IMDB	10	67426	1310	1635	8381	1310	1574	9112	1310	1578
Yelp 2013	5	62522	1631	1633	7773	1631	1559	8671	1631	1577

Datasets	Classes	Sparse20			Sparse50			Sparse80		
		#docs	#users	#prods	#docs	#users	#prods	#docs	#users	#prods
IMDB	10	44261	1042	1323	17963	659	840	2450	250	312
Yelp 2013	5	38687	1301	1288	16058	818	823	2406	352	304

Table 1: Dataset statistics

feature maps, for a total of 150 dimensions. These two are concatenated to create a 300-dimension encoded word vectors. We use dropout (Srivastava et al., 2014) on all non-linear connections with a dropout rate of 0.5. We set the batch size to 32. Training is done via stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012), with l_2 constraint (Hinton et al., 2012) of 3. We perform early stopping using a subset of the given development dataset. Training and experiments are all done using a NVIDIA GeForce GTX 1080 Ti graphics card.

Additionally, we also implement two versions of our model where the word encoder is a subpart of HCSC, i.e. (a) the CNN-based model (**CNN+CSAA**) and (b) the RNN-based model (**RNN+CSAA**). For the CNN-based model, we use 100 feature maps for each of the filter sizes $h = 3, 5, 7$, for a total of 300 dimensions. For the RNN-based model, we set the state sizes of the LSTMs to 150, for a total of 300 dimensions.

Datasets and evaluation We evaluate and compare our models with other competing models using two widely used sentiment classification datasets with available user and product information: IMDB and Yelp 2013. Both datasets are curated by Tang et al. (2015), where they are divided into train, dev, and test sets using a 8:1:1 ratio, and are tokenized and sentence-splitted using Stanford CoreNLP (Manning et al., 2014). In addition, we create three subsets of the train dataset to test the robustness of the models on sparse datasets. To create these datasets, we randomly remove all the reviews of $x\%$ of all users and products, where $x = 20, 50, 80$. These datasets are not only more sparse than the original datasets, but also have smaller number of users and products, introducing cold-start users and products. All datasets are summarized in Table 1. Evaluation is done using two metrics: the Accuracy which measures the overall sentiment classification performance and the RMSE which measures the divergence between predicted and ground truth classes. We notice very minimal differences among performances of different runs.

Models	IMDB		Yelp 2013	
	Acc.	RMSE	Acc.	RMSE
JMARS	-	1.773^*	-	0.985^*
UPNN	0.435^*	1.602^*	0.596^*	0.784^*
TLFM+PRC	-	1.352^*	-	0.716^*
UPDMN	0.465^*	1.351^*	0.639^*	0.662
TUPCNN	0.488^*	1.451^*	0.639^*	0.694^*
NSC	0.533	1.281^*	0.650	0.692^*
CNN+CSAA	0.522^*	1.256^*	0.654	0.665
RNN+CSAA	0.527^*	1.237^*	0.654	0.667
HCSC	**0.542**	**1.213**	**0.657**	**0.660**

Table 2: Accuracy and RMSE values of competing models on the original non-sparse datasets. An asterisk indicates that HCSC is significantly better than the model ($p < 0.01$).

4.2 Comparisons on original datasets

We report the results on the original datasets in Table 2. On both datasets, HCSC outperforms all previous models based on both accuracy and RMSE. Based on accuracy, HCSC performs significantly better than all previous models except NSC, where it performs slightly better with 0.9% and 0.7% increase on IMDB and Yelp 2013 datasets. Based on RMSE, HCSC performs significantly better than all previous models, except when compared with UPDMN on the Yelp 2013 datasets, where it performs slightly better. We note that RMSE is a better metric because it measures how close the wrongly predicted sentiment and the ground truth sentiment are. Although NSC performs as well as HCSC based on accuracy, it performs worse based on RMSE, which means that its predictions deviate far from the original sentiment.

It is also interesting to note that when CSAA is used as attentive pooling, both simple CNN and RNN models perform just as well as NSC, despite NSC being very complex and modeling the documents with compositionality (Chen et al., 2016a). This is especially true when com-

Models	Sparse20	Sparse50	Sparse80
NSC(LA)	0.469	0.428	0.309
NSC	0.497	0.408	0.292
CNN+CSAA	0.497	0.444	0.343
RNN+CSAA	**0.505**	0.455	0.364
HCSC	**0.505**	**0.456**	**0.368**

(a) IMDB Datasets

Models	Sparse20	Sparse50	Sparse80
NSC(LA)	0.624	0.590	0.523
NSC	0.626	0.592	0.511
CNN+CSAA	0.626	0.605	0.522
RNN+CSAA	0.633	0.603	0.527
HCSC	**0.636**	**0.608**	**0.538**

(b) Yelp 2013 Datasets

Table 3: Accuracy values of competing models when the training data used is sparse. **Bold-faced** values are the best accuracies in the column, while red values are accuracies worse than NSC(LA).

pared using RMSE, where both CNN+CSAA and RNN+CSAA perform significantly better ($p < 0.01$) than NSC. This proves that CSAA is an effective use of the user and product information for sentiment classification.

4.3 Comparisons on sparse datasets

Table 3 shows the accuracy of NSC (Chen et al., 2016a) and our models CNN+CSAA, RNN+CSAA, and HCSC on the sparse datasets. As shown in the table, on all datasets with different levels of sparsity, HCSC performs the best among the competing models. The difference between the accuracy of HCSC and NSC increases as the level of sparsity intensifies: While the HCSC only gains 0.8% and 1.0% over NSC on the less sparse Sparse20 IMDB and Yelp 2013 datasets, it improves over NSC significantly with 7.6% and 2.7% increase on the more sparse Sparse80 IMDB and Yelp 2013 datasets, respectively.

We also run our experiments using NSC without user and product information, i.e. NSC(LA) which reduces the model into a hierarchical LSTM model (Yang et al., 2016). Results show that although the use of user and product information in NSC improves the model on less sparse datasets (as also shown in the original paper (Chen et al., 2016a)), it decreases the performance of the model on more sparse datasets: It performs 2.0%, 1.7%, and 1.2% worse than NSC(LA) on Sparse50 IMDB, Sparse80 IMDB, and Sparse80 Yelp 2013 datasets. We argue that this is because NSC does not consider the existence of cold-start problems, which makes the additional user and product information more noisy than helpful.

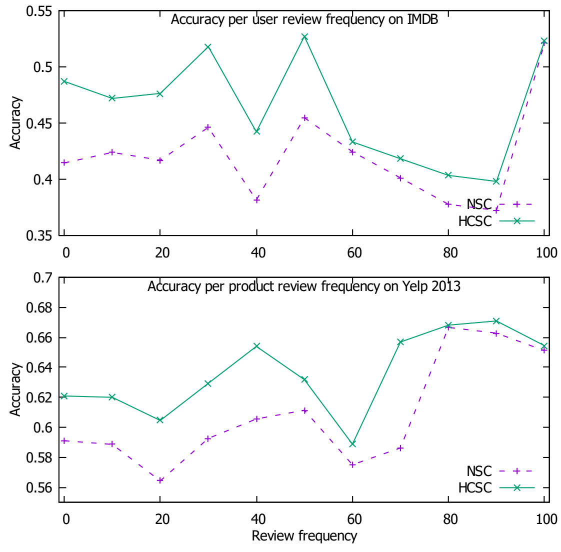


Figure 3: Accuracy per user/product review frequency on both datasets. The review frequency value f represents the frequencies in the range $[f, f + 10)$, except when $f = 100$, which represents the frequencies in the range $[f, \infty)$.

5 Analysis

In this section, we show further interesting analyses of the properties of HCSC. We use the Sparse50 datasets and the corresponding results of several models as the experimental data.

Performance per review frequency We investigate the performance of the model over users/products with different number of reviews. Figure 3 shows plots of accuracy of both NSC and HCSC over (a) different user review frequency on IMDB dataset and (b) different product review frequency on Yelp 2013 dataset. On both plots, we observe that when the review frequency is small, the performance gain of HCSC over NSC is very large. However, as the review frequency becomes larger, the performance gain of HCSC over NSC decreases to a very marginal increase. This means that HCSC finds its improvements over NSC from cold-start users and products, in which NSC does not consider explicitly.

How few is cold-start? One intriguing question is when do we say that a user/product is cold-start or not. Obviously, users/products with no previous reviews at all should be considered cold-start, however the cut-off point between cold-start and non-cold-start entities is vague. Although we

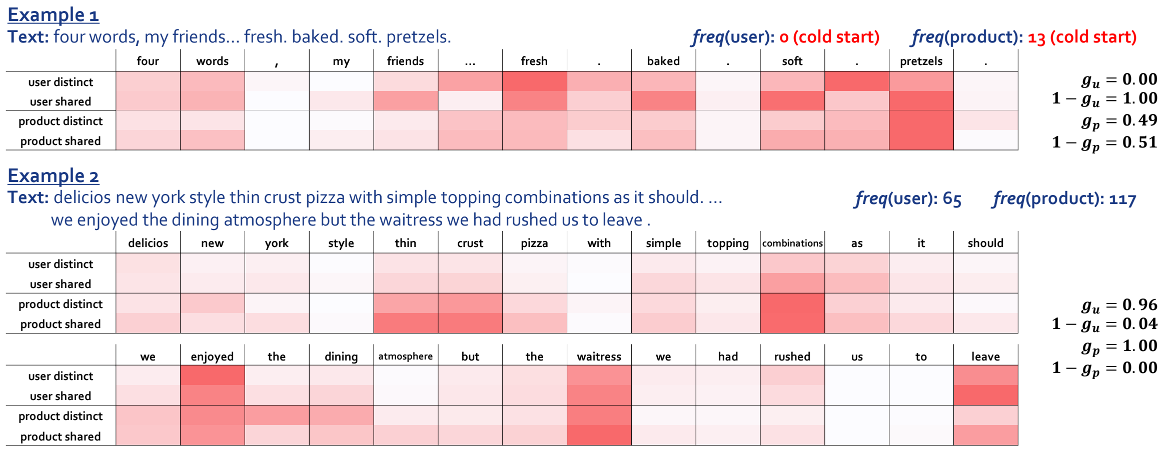


Figure 4: Visualization of attention and gate values of two examples from the Yelp 2013 dataset. Example 2 is truncated, leaving only the important parts. Gate values g's are the average of the values in the original gate vector.

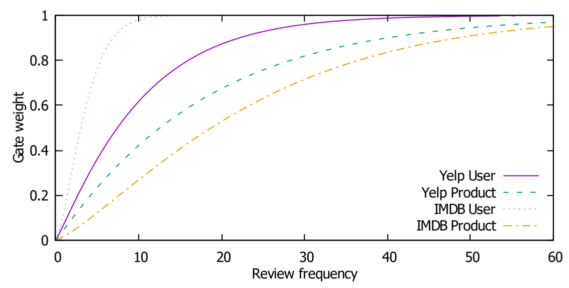


Figure 5: Graph of the user/product-specific Weibull cumulative distribution on both datasets.

Models	IMDB	Yelp 2013
NSC	7331	6569
CNN+CSAA	256 (28.6x)	146 (45.0x)
RNN+CSAA	968 (7.6x)	561 (11.7x)
HCSC	1110 (6.6x)	615 (10.7x)

Table 4: Time (in seconds) to process the first 100 batches of competing models for each dataset. The numbers in the parenthesis are the speedup of time when compared to NSC.

cannot provide an exact answer to this question, HCSC is able to provide a nice visualization by reducing the shape and scale vectors, k and λ, of the frequency-guided selective gate into their averages and draw a Weibull cumulative distribution graph, as shown in Figure 5. The figure provides us these observations: First, users have a more lenient cold-start cut-off point compared to products; in the IMDB dataset, a user only needs approximately at least five reviews to use at least 80% of its own information (i.e. distinct vector). On the other hand, products tend to need more reviews to be considered sufficient and not cold start; in the IMDB dataset, a product needs approximately 40 reviews to use at least 80% of its own information. This explains the marginal increase in performance of previous models when only product information is used as additional context, as reported by previous papers (Tang et al., 2015; Chen et al., 2016a).

On the different pooled vectors We visualize the attention and gate values of two example results from HCSC in Figure 4 to investigate on how user/product vectors, and distinct/shared vectors work. In the first example, both user and product are cold-start. The user distinct vector focuses its attention to wrong words, since it is not able to use any useful information from the user at all. In this case, HCSC uses the user shared vector by using a gate vector $g_u = 0$. The user shared vector correctly attends to important words such as *fresh*, *baked*, *soft*, and *pretzels*. In the second example, both user and product are not cold-start. In this case, the distinct vectors are used almost entirely by setting the gates close to 1. Still, the corresponding shared vectors are similar to the distinct vectors, proving that HCSC is able to create useful user/product-specific context from similar users/products. Finally, we look at the differing attention values of users and products. We observe that user vectors focus on words that describe the product or express their emotions (e.g. *fresh* and *enjoyed*). On the other hand, product vectors focus more on words pertaining to the products/services (e.g. *pretzels* and *waitress*).

On the time complexity of models Finally, we report the time in seconds to run 100 batches of data of the models NSC, CNN+CSAA, RNN+CSAA, and HCSC in Figure 4. NSC takes too long to train, needing at least 6500 seconds to process 100 batches of data. This is because it uses two non-parallelizable LSTMs on top of each other. Our models, on the other hand, only use one (or none in the case of CNN+CSAA) level of BiLSTM. This results to at least 6.6x speedup on the IMDB datasets, and at least 10.7x speedup on the Yelp 2013 datasets. This means that HCSC does not sacrifice a lot of time complexity to obtain better results.

6 Conclusion

We propose Hybrid Contextualized Sentiment Classifier (HCSC) with a fast word encoder which contextualizes words to contain both short and long range word dependency features, and an attention mechanism called Cold-start Aware Attention (CSAA) which considers the existence of the cold-start problem among users and products by using a shared vector and a frequency-guided selective gate, in addition to the original distinct vector. Our experimental results show that our model performs significantly better than previous models. These improvements increase when the level of sparsity in data increases, which confirm that HCSC is able to deal with the cold-start problem.

Acknowledgements

This work was supported by Microsoft Research Asia and the ICT R&D program of MSIT/IITP. [2017-0-01778, Development of Explainable Human-level Deep Machine Learning Inference Framework]

References

Reinald Kim Amplayo and Seung-won Hwang. 2017. Aspect sentiment model for micro reviews. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, pages 727–732.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016a. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1650–1659.

Tao Chen, Ruifeng Xu, Yulan He, Yunqing Xia, and Xuan Wang. 2016b. Learning user and product distributed representations using a sequence model for sentiment analysis. *IEEE Computational Intelligence Magazine* 11(3):34–44.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 193–202.

Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 521–526.

Tomohiro Fukuhara, Hiroshi Nakagawa, and Toyoaki Nishida. 2007. Understanding sentiment of people from news articles: Temporal sentiment analysis of social events. In *ICWSM*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, pages 815–824.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, pages 375–384.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. pages 55–60.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. pages 6297–6308.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Kaisong Song, Wei Gao, Shi Feng, Daling Wang, Kam-Fai Wong, and Chengqi Zhang. 2017. Recommendation vs sentiment analysis: a text-driven latent factor model for rating prediction with cold-start awareness. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, pages 2744–2750.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1014–1023.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 208–212.

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment flow-a general model of web review argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 601–611.

Min Yang, Jincheng Mei, Heng Ji, Zhou Zhao, Xiaojun Chen, et al. 2017. Identifying and tracking sentiments and topics from social media texts during natural disasters. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 527–533.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1480–1489.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR* abs/1212.5701.

Yongfeng Zhang. 2015. Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation. In *Proceedings of the eighth ACM international conference on web search and data mining*. ACM, pages 435–440.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 3485–3495.

Modeling Deliberative Argumentation Strategies on Wikipedia

Khalid Al-Khatib[†] **Henning Wachsmuth**[‡] **Kevin Lang**[†] **Jakob Herpel**[†]
Matthias Hagen[§] **Benno Stein**[†]

[†] Bauhaus-Universität Weimar
Webis Group, Faculty of Media
<firstname>.<lastname>@uni-weimar.de

[‡] Paderborn University
Computational Social Science Group
henningw@upb.de

[§] Halle University
matthias.hagen@informatik.uni-halle.de

Abstract

This paper studies how the argumentation strategies of participants in deliberative discussions can be supported computationally. Our ultimate goal is to predict the best next deliberative move of each participant. In this paper, we present a model for deliberative discussions and we illustrate its operationalization. Previous models have been built manually based on a small set of discussions, resulting in a level of abstraction that is not suitable for move recommendation. In contrast, we derive our model statistically from several types of metadata that can be used for move description. Applied to six million discussions from Wikipedia talk pages, our approach results in a model with 13 categories along three dimensions: discourse acts, argumentative relations, and frames. On this basis, we automatically generate a corpus with about 200,000 turns, labeled for the 13 categories. We then operationalize the model with three supervised classifiers and provide evidence that the proposed categories can be predicted.

1 Introduction

Deliberation is the type of discussions where the aim is to find the best choice from a set of possible actions (Walton, 2010). This type is influential for making decisions in different processes including *collaborative writing*. Studies have shown the positive impact of deliberation on the quality of several document types, such as scientific papers, research proposals, political reports, and Wikipedia articles, among others (Kraut et al., 2012).

However, deliberative discussions may fail, either by agreeing on the wrong action, or by reaching no agreement. While the former is hard to measure, the latter is, for example, clearly reflected in the number of disputed discussions on Wikipedia (Wang and Cardie, 2014).

Although agreement can never be guaranteed, a deliberative argumentation strategy of a discussion's participants makes it more likely (Kittur et al., 2007). With *strategy*, we here mean the sequence of moves that participants take during the discussion. Such a sequence is effective if it leads to a successful discussion. To achieve effectiveness, every participant has to understand the current state of a discussion and to come up with a next deliberative move that *best* serves the discussion. For newcomers, this requires substantial effort and time, especially when a discussion grows due to conflicts and back-and-forth arguments. Here, automated tools can help by annotating ongoing discussions with a label for each move or by providing a textual summary of past moves (Zhang et al., 2017a,b). A way to go beyond that is to let the tool *recommend the best possible moves* according to an effective strategy. This is the ultimate goal of our research.

As a substantial step towards this goal, two fundamental research questions are addressed in the paper at hand: (1) How to model deliberative discussions in light of the aim of agreement, and (2) how to operationalize the model in order to identify different argumentation strategies and to learn about their effectiveness.

Different models of deliberative discussions have been proposed in previous studies. These models were developed based on expert analyses of a *small* set of sampled discussions (see Section 2). However, the small size, in fact, confines the ability to develop a *representative* model, which should ideally cover a wide range of moves while being abstract to fit the majority of discussions.

To overcome this limitation, we propose to derive a model statistically from a large set of discussions. We approach this based on different types of metadata that people use to describe their moves on Wikipedia talk pages, the richest source of deliberative discussions on the web. Particularly, we extract the entire set of about six million discussions from all English Wikipedia talk pages. We parse each discussion to identify its structural components, such as turns, users, and time stamps. Besides, we store four types of *metadata* from the turns: the user tag, a shortcut, an in-line template, and links. To learn from the metadata, we cluster the types' instances based on their semantic similarity. Then, we map each cluster to a specific concept (e.g., 'providing a source'), and the related concepts into a set of categories (e.g., 'providing evidence'). Table 2 shows the categories of our model.

Analyzing the distribution of these categories, we find that each turn ideally should have (1) one of six categories that we call *discourse acts*, (2) one of three categories that we call *argumentative relations*, and (3) one of four categories that we call *frames*. As such, our model is in line with three well-established theories: *speech act theory* (Searle, 1969), *argumentation theory* (Peldszus and Stede, 2013), and *framing theory* (P. Levin et al., 1998). A model instance is sketched in Figure 1.

Based on the model, we generate a new large-scale corpus using the metadata automatically: *Webis-WikiDebate-18* corpus. Basically, if a turn in a discussion has metadata that belongs to a specific category according to the above-mentioned analysis, it is labeled with that category. The corpus includes 2400 turns labeled with a discourse act, 7437 turns labeled with a relation, and 182,321 turns labeled with a frame.

To operationalize our model, we train three supervised classifiers for acts, relations, and frames on the corpus. The classifiers employ a rich set of linguistic features that has been shown to be effective in similar tasks (Ferschke et al., 2012). The results of our experiments suggest that we are able to predict the labels with a comparable performance to the one achieved in similar tasks.

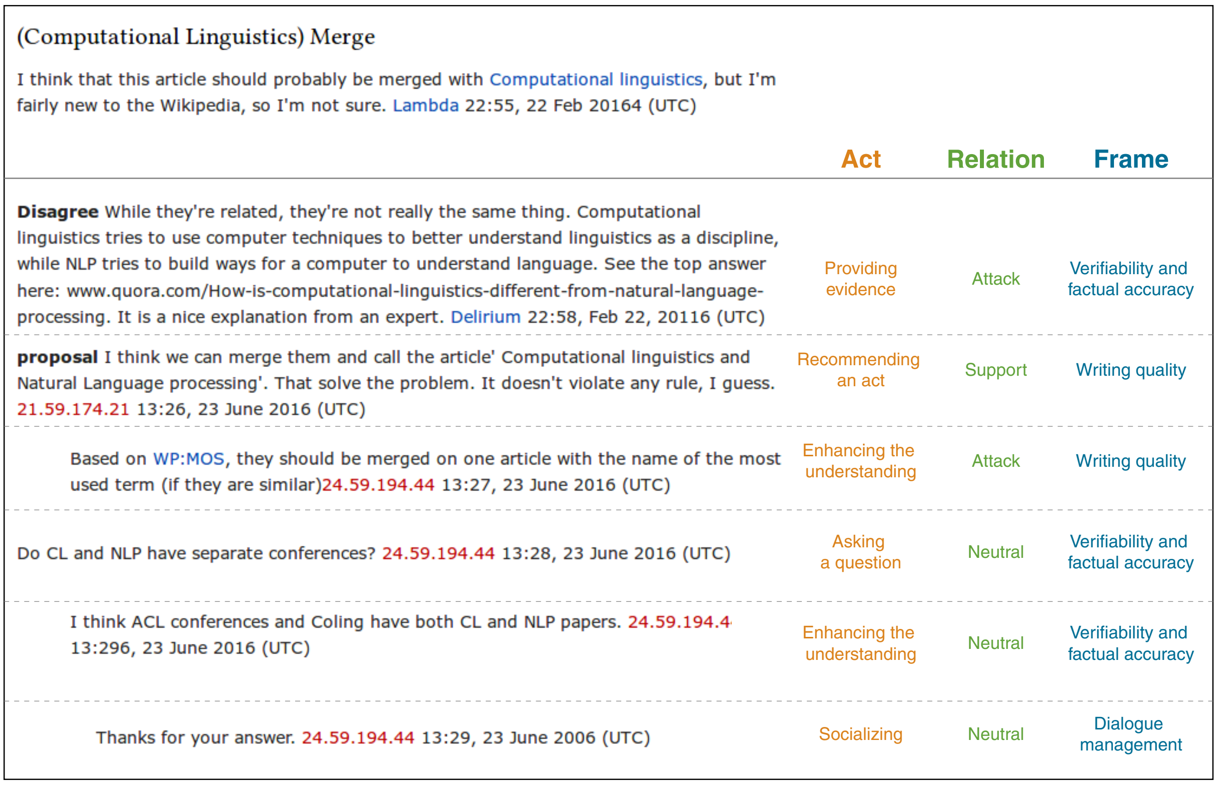
Overall, the contribution of this paper is threefold: (1) A data-driven approach for creating a new model of deliberative discussions that is aligned with well-established theories, (2) a corpus with about 200,000 turns labeled for 13 different categories, and (3) a classification approach that predicts the labels of turns. All developed resources are freely available at `https://www.webis.de/data/data.html`.

2 Related Work

Modeling deliberative discussions in Wikipedia has been already addressed in different studies. The central goal of these studies is to minimize the coordination effort among discussion participants. In particular, Ferschke et al. (2012) have proposed a model of 17 dialogue acts, each belonging to one of four categories: article criticism, explicit performative, information content, and interpersonal. The model was derived by performing a manual analysis of 30 talk pages in the Simple English Wikipedia. Based on the model, a new corpus of 1367 turns has been created and used to train and evaluate a multi-label classifier for predicting the model's acts. Another model is the one proposed by Viegas et al. (2007). The model consists of 11 different dialogue acts. These acts have been used to manually label 25 talk pages from the English Wikipedia. Furthermore, Bender et al. (2011) have developed a model for authority claims and alignment moves in Wikipedia discussions. The model then has been used to label 47 talk pages.

Rooted in the limitation of being derived from a small sample, these models obtain low coverage and/or are over-abstracted. This is indicated by labels such as 'other' (Viegas et al., 2007) or by a very abstract 'information providing' act (Ferschke et al., 2012), which covers 78% of the turns. We argue that recommending such moves for new participants will not be useful. On the other hand, the model of Ferschke et al. (2012) does not include anything similar to 'propose alternative action', for example, although such a concept was shown to be important in deliberative dialogues (Walton, 2010).

Moreover, no existing model distinguishes the three dimensions of turns: act, relation, and frame. They either consider only one dimension or mix an act with a relation, such as in the label: 'criticizing unsuitable or unnecessary content' (Ferschke et al., 2012). This is a problem for predicting the next best deliberative move. For example, consider a discussion about adding new content to an article, where the participants support the action with different acts (e.g., 'providing evidence'), but all of them consider the 'writing quality' frame. A new turn attacks the action by providing evidence that the action would violate the 'neutral point of view'. The best next move should actually consider this frame, since no content that violates 'neutral point of view' policy should be added, regardless of its adherence to the 'writing quality'.


(Computational Linguistics) Merge

I think that this article should probably be merged with Computational linguistics, but I'm fairly new to the Wikipedia, so I'm not sure. Lambda 22:55, 22 Feb 20164 (UTC)

	Act	Relation	Frame
Disagree While they're related, they're not really the same thing. Computational linguistics tries to use computer techniques to better understand linguistics as a discipline, while NLP tries to build ways for a computer to understand language. See the top answer here: www.quora.com/How-is-computational-linguistics-different-from-natural-language-processing. It is a nice explanation from an expert. Delirium 22:58, Feb 22, 20116 (UTC)	Providing evidence	Attack	Verifiability and factual accuracy
proposal I think we can merge them and call the article' Computational linguistics and Natural Language processing'. That solve the problem. It doesn't violate any rule, I guess. 21.59.174.21 13:26, 23 June 2016 (UTC)	Recommending an act	Support	Writing quality
Based on WP:MOS, they should be merged on one article with the name of the most used term (if they are similar)24.59.194.44 13:27, 23 June 2016 (UTC)	Enhancing the understanding	Attack	Writing quality
Do CL and NLP have separate conferences? 24.59.194.44 13:28, 23 June 2016 (UTC)	Asking a question	Neutral	Verifiability and factual accuracy
I think ACL conferences and Coling have both CL and NLP papers. 24.59.194.4 13:296, 23 June 2016 (UTC)	Enhancing the understanding	Neutral	Verifiability and factual accuracy
Thanks for your answer. 24.59.194.44 13:29, 23 June 2006 (UTC)	Socializing	Neutral	Dialogue management

Figure 1: Left: An excerpt of a discussion in a Wikipedia talk page. Right: The labels of each turn in the discussion according to our proposed model.

In contrast, our approach of deriving the model using thousands of different 'descriptions' of moves written by the numerous Wikipedia users is, in our view, more likely to give a representative picture of how people argue in deliberative discussions. This, in turn, leads not only to high coverage, but also to better abstraction. Our model is in line with three well-known theories, which we summarize in the next paragraph.

Speech act is a widely accepted theory in pragmatics (Searle, 1969). Based on this theory, many research papers have been proposed for modeling different domains, such as one-on-one live chat (Kim et al., 2010), persuasiveness in blogs (Anand et al., 2011), twitter conversations (Zarisheva and Scheffler, 2015), and online dialogues (Khanpour et al., 2016). In the context of *argumentation theory* (Peldszus and Stede, 2013), agreement detection is a related direction of work which has been studied in discussions (Rosenthal and McKeown, 2015). Notably, Andreas et al. (2012) annotated 822 turns from 50 talk pages with three labels: 'agreement', 'disagreement', and 'non'. Anyhow, over the last few years, argumentation mining became a hot topic in our community, where several studies have went beyond the agreement detection to investigate the identification of the 'support' and 'attack' relations in argumentation discourses (Peldszus and Stede, 2013). Finally, *framing* is one of the important theories in discourse analysis (Entman, 1993). This theory has been studied widely in different domains, such as news article (Naderi and Hirst, 2017) and political debates (Tsur et al., 2015). These three theories back up the essence of our proposed model. We found that a participant in a discussion writes her text considering a specific act, an argumentative relation, and a frame.

The metadata in Wikipedia have been used for different tasks. The 'infobox' has been exploited in the tasks of question answering (Morales et al., 2016) and summarization (Ye et al., 2009), among others. Moreover, Wang and Cardie (2014) have used specific discussion templates to identify discussions that are disputed. Besides Wikipedia, metadata such as 'point for', 'point against', and 'introduction' have been used successfully for modeling argumentativeness in debate platforms (Al-Khatib et al., 2016a). Also, The metadata for user interactions, such as the 'delta indicator' and users votes in Reddit ChangeMyView discussions have been used to model the persuasiveness of a text (Tan et al., 2016).

We started the investgation of strategies for writing argumentative texts in previous work. In (Al-Khatib et al., 2016b), we have presented a corpus for argumentation strategies in news editorials. We then used this corpus and other data in (Al-Khatib et al., 2017) to identify patterns of strategies across different general topics. In contrast to those two studies targeting monological texts, here we address argumentation strategies in dialogical texts.

3 Modeling Deliberative Discussions

The web is full of platforms where users can share and discuss opinions, beliefs, and ideas. In case of deliberative discussions, in particular, participants try to find the best action from several choices. Apparently, the participants there follow a strategy to achieve an effective discussion, i.e., each participant tries to come with the best deliberative move that leads to achieve the goal of discussion.

The numerous deliberative discussions on these platforms do not only include user-written text, but also different types of metadata that users add to benefit the coordination between them. For example, users vote for specific posts, summarize texts, include references to the sources they use, refer to the discussion policies of a platform, or report bad behavior of others. Overall, the available metadata represents a valuable resource that provides insights into three main aspects of a discussion: The functions of users' moves, the users' roles, and the discussion topics along with their flows. We propose to exploit the metadata for modeling argumentation strategies in deliberative discussions.

To this end, we proceed in four general steps: (1) *metadata inspection*, which includes investigating the used metadata and its functions, (2) *concept origination*, where clusters of similar metadata are created and mapped to corresponding concepts, (3) *concept categorization*, where similar concepts are abstracted into a defined set of categories, and (4) *category composition*, where possible overlaps between categories should be identified.

The idea of this approach is not only to model the strategies, but also to allow for an operationalization of the resulting model by providing a dataset for training classifiers. In particular, the metadata can also be used to label discussions based on distant supervision (Mintz et al., 2009). In the following, we describe how we implement our approach to derive a new model of Wikipedia discussions, using the metadata provided by the participants.

3.1 Discussion Parsing

As part of the management policies of Wikipedia, each article has an associated page called 'Talk'. The main purpose of the talk page is to allow users to discuss how to improve the article through specific actions that they agree on. Most of these discussions can be seen as deliberative, since all participants share the same goal: finding the best action to improve the article.

When a user has a proposal on how to improve an article, she can open a discussion on the article's talk page, specifying a title and the main topic of discussion. Usually, the topic denotes a suggestion to perform a specific action, such as adding, merging, or deleting certain content of the article, among others. Ideally, multiple users then participate in the discussion about whether the action would improve the article or not.

Each single comment written by a user at a specific time is called a 'turn'. A turn may reply directly to the main topic of the discussion or to any other turn. Overall, a discussion consists of the title, the main topic, and a number of turns written by users with attached time stamps (see Figure 1). Based on a manual inspection of the turns' texts of 50 discussions, we found four general types of metadata used by the participants: *user tags*, *shortcuts*, *inline-templates*, and *external links*.

To derive a model from Wikipedia, we need to extract and parse the whole set of discussions on all talk pages, including both ongoing and closed ones. This is all but trivial, particularly due to the fact that the creation of a discussion is solely done by the users; although Wikipedia describes the required format of the different parts of a discussion in detail, not all users follow the format, often forgetting required symbols or mistakenly confusing a symbol with another one. In the implementation of our approach, we built upon the English Wikipedia dump created on March 1st, 2017. Given a Wikipedia dump, we parse it in the following steps:

Extraction of Talk Pages First, we obtain the talk pages. We use the Java Wikipedia Library (JWPL) from Zesch et al. (2008), which converts a Wikipedia dump into a database that provides an easy-to-use access to the dump components.

Extraction of Discussions Next, we extract the discussions from the talk pages. To this end, we develop several regular expressions that capture the format for starting and ending a discussion.

Corpus Component	Instances
Page	5 807 046
Discussion	5 941 534
Discussion template	144 824
Turn	20 816 860
Registered users	739 244
Turns by registered users	10 926 670
Turns by anonymous user	9 890 190
Tag	99 889
Shortcut	425 583
Inline template	3 382 443
Links	4 824 085
Turns with tag and shortcut	2 347
Turns with tag and inline template	61 521
Turns with shortcut and inline template	170 065

Table 1: Instance counts of the different components of the Webis-WikiDiscussions-18 corpus.

Identification of Structure Given the discussion, we identify their structure. We created a specific template to mine the title. The topic of the discussion is simply given by the first turn. To identify and correctly segment all users' turns, we use several indicators, for instance, indentations.

Identification of Turn Metadata Finally, we identify the metadata of each turn. We analyzed how users include the tags in their turns, finding that they usually start a turn with a user tag in triple quotation marks. A shortcut starts with 'WP:', followed by a name for the shortcut, together encapsulated by brackets. Also templates are placed between double parentheses, but they do not start with 'WP:'. Links are simply identified by either of the affixes 'www.' and 'http:'.

3.2 The Webis-WikiDiscussions-18 Corpus

The result of the parsing process is a large-scale corpus of Wikipedia discussions. In particular, the *Webis-WikiDiscussions-18* corpus we created contains about six million discussions, consisting of about 20 million turns. The turns comprise around 74,000 different tags with a total of about 100,000 instances, around 7000 different shortcuts with about 400,000 instances, and around 51,000 different inline templates with about 3.3 million instances. Half of the turns are written by registered users. Table 1 lists the exact instance counts.

3.3 Model Derivation

We now explain how we derive a model of deliberative discussions from the metadata obtained in the previous subsection. The derivation process includes the four steps outlined in the beginning of this section.

Metadata Inspection As mentioned before, a turn on Wikipedia includes up to four types of metadata: user tag, shortcut, inline template, and external link. Each type has a specific definition, a suggested usage, and properties that we discuss in the following paragraphs.

A *user tag* is a short text that a discussion participant uses to describe or summarize her contribution. Most tags indicate the main function of the contribution, such as 'proposal' and 'question'. Users can define any free-text tag they want using a noun, verb, etc. Analyzing the tags in the crawled discussions, we found the most frequent tags to be rather general and meaningful, whereas less frequent tags often capture aspects of the topic of discussion, such as 'Israel-Venezuela relations' in the discussion about 'Foreign relations of Israel'. Sometimes, tags are used to get the attention of specific users, such as 'For who reverted my change'. Unfortunately, many users also misuse tags, for example, by including the whole turn's text there or by encoding meaningless information.

A *shortcut* is an abbreviation text link that redirects the user to some page on Wikipedia. Although shortcuts may link to any Wikipedia page, they are often used to link to rules or policies. The respective pages belong to one of five categories:
(1) Behavioral guidelines: Pages that describe how users should interact with each other (e.g., during a discussion). This includes that users should be "good-faith" (WP:AGF), among others.
(2) Content guidelines: Pages that describe how to identify and include information in the articles, such as those about how an article should have reliable and accepted sources (WP:RELIABLE).
(3) Style guidelines: Pages that contain advice on writing style, formatting, grammar, and similar. This includes how to write the introduction (WP:LEAD) and headings (WP:HEADINGS), and what style to use for the content (WP:MOS).
(4) Notability guidelines: Pages that illustrate the conditions of testing whether a given topic warrants its own article. The most common shortcut in this category is (WP:N).
(5) Editing guidelines: Pages that provide information on the metadata of articles, such as the articles' categories (WP:CAT).

Overall, we found that shortcuts are used particularly frequently for style, content, and behavioral guidelines in Wikipedia discussions. The participants mainly use them to discuss the impact of applying an action that has been proposed to be performed on a Wikipedia article. For example, adding a lot of content to the introduction of an article may violate the style guidelines. A user can indicate this by referring to the style rules using the shortcut (WP:LEAD).

An *inline template* is a Wikipedia page that has been created to be included in other pages. Inline templates usually comprise specific patterns that are used in many articles, such as standard warnings or boilerplate messages. For example, there are templates for including a quotation, citation, or code, among others. Templates are used frequently in Wikipedia discussions, with the objective of writing readable and well structured turns.

An *external link*, finally, points to a web page outside Wikipedia. External links occur both in Wikipedia articles and in Wikipedia discussions. While there are some restrictions for using them in articles, they can be used without restriction in discussions. We found that these links are used in Wikipedia discussions to point to evidence on the linked web pages. In particular, they often link to research, news, search engines, educational institutions, and blogs.

Concept Origination We analyzed the usage of the four types of metadata in Wikipedia discussions and identified a set of concepts. Each concept primarily describes the turn that a participant writes:

User tags: We explored all 376 tags that occurred at least 35 times. As discussed before, the tags could be seen as a keywords that describe the turns. Often, different tags refer to the same concept, for example, 'conclusion', 'summary', and 'overall' all capture the concept of 'summarization', i.e., the main function of the respective turns is to summarize the discussion. As a result, we identified 32 clusters. We examined some turns belonging to each cluster, and mapped each cluster to a specific concept that describes it.

Shortcuts: Analogously, we explored all 99 shortcuts that occurred at least 900 times. Since the shortcuts themselves do not describe the turn, but rather the policy pages they refer to, we analyzed these pages by reading their first paragraphs and by checking their relation to the pages of the five shortcut categories we discussed before (e.g., 'behavioral'). This resulted in the identification of 12 concepts. We found that each shortcut concept describes the main quality aspect that a turn addresses. For example, 'writing content' specifies how a proposed action influences the quality of the writing of the associated article.

Inline-templates: Our investigation of this type led only to concepts that we already found before for the tags and shortcuts, such as 'stating a fact'.

External links: Similar to the templates, we identified concepts in the links that we also observed in the tags, such as 'providing source'.

Concept Categorization The concepts that we identified in the user tags can be grouped into six categories that we see as 'discourse acts':

1. *Socializing:* All concepts related to social interaction, such as thanking, apologizing, or welcoming other users.
2. *Providing evidence:* All concepts concerning the provision of evidence. Evidence may be given in form of a quote, an example, a fact, references, a source, and similar.
3. *Enhancing the understanding:* All concepts related to helping users understand the topic of discussion or a discussion itself. This can be done by giving background information, by clarifying misunderstandings, or by summarizing the discussion, among others.
4. *Recommending an act:* All concepts proposing to add a new aspect to the discussion, to ask more users to participate in the discussion, or to come up with an alternative to the proposed action.
5. *Asking a question:* All concepts related to questions serving different purposes, such as obtaining information on the topic of discussion, requesting reasons of specific decisions, and similar.
6. *Finalizing the discussion:* All concepts related to the decision of a discussion, including reporting the decision, committing it, or closing the discussion to move it to the archive.

In addition, we identified three further categories based on the user tags, which we see as relevant to 'argumentation theory'. Each represents a relation between the turn and the topic of discussion or between the turn and another turn:

1. *Support relation:* The turn agrees with or supports another turn or the topic of discussion,

for instance, by providing an argument in favor of the one in the 'supported' turn.

2. *Attack relation:* The opposite of the 'support relation', i.e., the turn disagrees or attacks another turn or the topic of discussion.
3. *Neutral relation:* The turn has a neutral relation to another turn or the topic of discussion when it neither support nor attack it.

Finally, we identified four categories based on the shortcuts that we see as relevant to 'framing theory'. They target a quality dimension of the article or of the discussion itself:

1. *Writing quality:* Turns that mainly address issues related to the quality of writing of an article, such as whether adding new content complies with the style guidelines for lead sections, the layout, or similar.
2. *Verifiability and factual accuracy:* Turns that address issues related to the quality of references, the reliability of sources, copyright violations, plagiarism, and similar.
3. *Neutral point of view:* Turns that focus on a fair representation of viewpoints and on how to avoid bias.
4. *Dialogue management:* Turns that concentrate on issues related to managing the discussion, such as reporting abusive language, preserving respect between users, encouraging newcomer participants, and similar.

Category Composition Given these categories, we investigated the interaction between them in 20 discussions, for instance, to see whether the categories are orthogonal. We found that each turn may have one discourse act, one relation, and one frame at the same time. For example, a turn may support another turn by providing evidence (say, of the type 'source'), while focusing on the writing quality frame. Table 2 shows the categories of our model and their concepts.

4 Model Operationalization

In this section, we present the operationalization process of our proposed model for deliberative argumentation strategies. First, we explain the construction of *Webis-WikiDebate-18*: a large-scale corpus for our model that we generated automatically based on the metadata in discussions. Then, we discuss the development and evaluation of a classification approach which we use for predicting the model's categories.

4.1 The Webis-WikiDebate-18 Corpus

To create a corpus for our model, we decided to rely again on the metadata. In particular, for each category in our model, we retrieved the metadata instances that had been used to derive the category, and then labeled any turn that included any metadata with this category. For example, the user tag 'overall' was used to originate the concept 'summarization', which was abstracted into the category 'enhancing the understanding'. Accordingly, all the turns that included this tag were labeled with the category 'enhancing the understanding'. This process is in line with the distant supervision paradigm. In case a turn contained metadata belonging to two categories, we excluded it from the corpus. This happened with some shortcuts in particular. Basically, such cases indicate that some turns address more than one frame.

Overall, the corpus comprises 2400 turns labeled with one of the six discourse act categories, 7437 turns with one of the relation categories, and 182,321 turns with one of the frame categories. In order to verify the reliability of the corpus, we randomly sampled about 100 turns from each category, ensuring that all the category's concepts are taken into consideration. The turns in the samples were verified (i.e., whether they belong to the assigned category) by a worker hired from the freelancing platform `upwork.com`. The worker was a native speaker of English with deep expertise in writing. Table 3 shows statistics of the corpus, including the percentage of turns in each sample that belong to the assigned category according to the expert. In general, this verification result is comparable to the inter-annotator agreement achieved in some related studies (Ferschke et al., 2012).

4.2 Classification Approach

Based on the Webis-WikiDebate-18 corpus, we develop three supervised classifiers: one for the discourse acts, one for the relations, and one for the frames. Since this paper does not aim at proposing a novel approach for the classification tasks, but rather at showing the ability to operationalize the model, we follow existing work that has proposed methods for the tasks at hand. Particularly, we implement a rich set of features that have been used by others before. These features capture lexical, semantic, style, and pragmatic properties of turns.

Dimension	Category	Concepts
Discourse act	Socializing	(1) Thank a user, (2) Apologize from a user, (3) Welcome a user, (4) Express anger
	Providing evidence	(1) Provide a quote, (2) Reference, (3) Source, (4) Give an example, (5) State a fact, (6) Explain a rational
	Enhancing the understanding	(1) Provide background info, (2) Info on the history of similar discussions, (3) Introduce the topic of discussion, (4) Clarify a misunderstanding, (5) Correct previous own or other's turn, (6) Write a discussion summary, (7) Conduct a survey on participants, (8) Request info
	Recommending an act	(1) Propose alternative action on the article, (2) Suggest a new process of discussion, (3) Propose asking a third party
	Asking a question	(1) Ask a general question about the topic, (2) Question a proposal or arguments in a turn
	Finalizing the discussion	(1) Report the decision, (2) Commit the decision, (3) Close the discussion
Argumentative relation	Support	(1) Agree, (2) Support
	Neutral	(1) Be neutral.
	Attack	(1) Disagree, (2) Attack, (3) Counter-attack
Frame	Writing quality	(1) Naming articles, (2) Writing content, (3) Formatting, (4) images, (5) Layout and list
	Verifiability and factual accuracy	(1) Reliable sources, (2) Proper citation (3) Good argument
	Neutral point of view	(1) Neutral point of view
	Dialogue management	(1) Be bold. (2) Be civil, (3) Don't game the system

Table 2: The concepts covered by each category of each of the three principle dimensions of our model.

Dimension	Category	Turns	Prec.
Discourse act	Socializing	83	0.71
	Providing evidence	781	0.49
	Enhancing the understanding	671	0.56
	Recommending an act	137	0.82
	Asking a question	106	0.71
	Finalizing the discussion	622	0.71
Argumentative relation	Support	2895	1.00
	Neutral	1937	0.63
	Attack	2605	1.00
Frame	Writing quality	19893	0.51
	Verifiability and factual ac.	72049	0.89
	Neutral point of view	60007	0.89
	Dialogue management	30372	0.74

Table 3: Number of turns in each category of Webis-WikiDebate-18 corpus and the precision of sampled turns for each category according to an expert.

In short, we used the following features: The frequency of word 1–3-grams, character 1–3-grams, chunk 1–3-grams, function word 1–3-grams, and of the first 1–3 tokens in a turn. The number of characters, syllables, tokens, phrases, and sentences in a turn. the frequencies of part-of-speech tag 1–3-grams. The mean SentiWordNet score of the words in a turn (http://sentiwordnet.isti.cnr.it). The frequency of each word class of the General Inquirer (http://www.wjh.harvard.edu/~inquirer). The depth level of turns in the discussion. For the relation classifier, we had additional features that consider the target of the relation (the parent turn), namely, the cosine, euclidean, manhattan, and jaccard similarity between turn and parent turn.

4.3 Experiments and Results

As a preprocessing step, we cleaned the turns in the *Webis-WikiDebate-18* Corpus by removing all the metadata: user tags, shortcuts, user and time stamps, etc. Then, we grouped the turns that belong to the discourse act categories in a single dataset (say, the 'discourse act dataset'). The same was performed for the turns belonging to relations and frames. We then split each of the three datasets randomly into training (60%), development (20%), and test (20%) sets. We ensured that turns from the same discussion should appear only in either of the split sets, in order to avoid biasing the classifiers by topical information.

We trained different machine learning models on the training sets and evaluated them on the development sets. The models included those which had been used before in similar tasks, such as naive bayes, logistic regression, support vector machine, and random forest. We tried both under and over-sampling on the training sets. The best results in the three tasks were achieved by using support vector

Dimension	Category	Prec.	Rec.	F_1
Discourse act	Socializing	0.14	0.11	0.13
	Providing evidence	0.63	0.77	0.69
	Enhancing the understand.	0.62	0.55	0.58
	Recommending an act	0.13	0.09	0.10
	Asking a question	0.80	0.19	0.31
	Finalizing the discussion	0.67	0.74	0.71
Argumentative relation	Support	0.53	0.59	0.56
	Neutral	0.55	0.50	0.52
	Attack	0.50	0.49	0.50
Frame	Writing quality	0.74	0.47	0.57
	Verifiability and factual ac.	0.62	0.74	0.67
	Neutral point of view	0.59	0.56	0.58
	Dialogue management	0.64	0.56	0.60

Table 4: The precision, recall, and F_1-score of our classifiers for all categories of the three dimensions.

machine without sampling the training sets.

We used the support vector machine implementation from the LibLinear library (Fan et al., 2008) on the test sets and report the results in Table 4. Overall, the three classifiers achieved results that are comparable to the results of previous methods on the corresponding tasks (Ferschke et al., 2012; Zhang et al., 2017a). We obtained the best results in the frame task, followed by relations and then discourse acts. Apparently, the results correlate with the size of the datasets. In case of discourse acts, the classifier achieves low F_1-scores for 'socializing', 'recommending an act', and 'asking a question'. These categories have a significantly smaller number of turns compared to other categories, which makes identifying them harder. The effectiveness of classifying the relation and frame categories, on the other hand, appears promising given the difficulty of these tasks.

We point that we considered mainly the turns' texts in our experiments. In principle, this helps to get an idea about the effectiveness of our approach in Wikipedia as well as other registers for discussions. Nevertheless, including the metadata and structural information of the analyzed discussions is definitely worthwhile in general, and will naturally tend to lead to notably higher effectiveness.

5 Discussion and Conclusion

While our approach to modeling argumentation strategies in deliberative discussions may seem Wikipedia-specific, the derivation of concepts and categories from metadata can be transferred to other online discussion platforms. We expect the general derivation steps to be the same, whereas the techniques applied within each step may differ depending on the types, frequency, and quality of metadata. For example, the consistent usage of the most common user tags in Wikipedia discussions helps originating concepts manually. In contrast, other metadata might require the use of computational methods, such as clustering, keyphrase extraction, and textual entailment.

Unlike previous approaches to the modeling of discussions on Wikipedia, our model decouples the three principle dimensions of discussions: *discourse acts*, *argumentative relations*, and *frames*. We argue that the distinction of these dimensions is key to develop tool support for discussion participants, for example, for recommending the best possible move in an ongoing discussion.

Also, our model helps analyzing the influence of user interaction and behavior on the effectiveness of discussion decisions. For example, some Wikipedia users focus on the frame 'well written' while ignoring others, which may negatively affect the accuracy of an article's content. Also, users often attack other turns, instead of considering neutral acts such as clarifications of misunderstandings.

Many categories in our model will apply to deliberative discussions in general, particularly the discourse acts and argumentative relations. While the found frames are more Wikipedia-specific, similar play a role on collaborative writing platforms. For example, when writing a scientific paper, possible frames are the 'writing quality' or the 'verifiability of content and citations'.

Besides the model, we created two large-scale corpora: The *Webis-WikiDiscussions-18* corpus, including the entire set of Wikipedia discussions (at the time of parsing) with annotated discussion structure and metadata, and the *Webis-WikiDebate-18* corpus, where turns are labeled for their discourse acts, argumentative relations, and frames. We believe that these corpora will help foster research on tasks such as argument mining, among others.

Finally, we operationalized our Wikipedia discussion model in three support vector machine classifiers with tailored features. Our experiment results confirm that categories of our model can be predicted successfully. In future work, we plan to study how to distinguish effective from ineffective discussions based on our model as well as how to learn from the strategies used in successful discussions, in order to predict the best next deliberative move in an ongoing discussion.

References

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Kohler, and Benno Stein. 2016a. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL*, pages 1395–1404. Association for Computational Linguistics.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, and Benno Stein. 2017. Patterns of Argumentation Strategies across Topics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1351–1357. Association for Computational Linguistics.

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016b. A News Editorial Corpus for Mining Argumentation Strategies. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING*, pages 3433–3443. Association for Computational Linguistics.

P. Anand, J. King, Jordan Boyd-Graber, E. Wagner, C. Martell, Douglas Oard, and Philip Resnik. 2011. Believe Me—We Can Do This! Annotating Persuasive Acts in Blog Text. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Jacob Andreas, Sara Rosenthal, and Kathleen McKeown. 2012. Annotating Agreement and Disagreement in Threaded Discussion. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC*, pages 818–822.

Emily M. Bender, Jonathan T. Morgan, Meghan Oxley, Mark Zachry, Brian Hutchinson, Alex Marin, Bin Zhang, and Mari Ostendorf. 2011. Annotating Social Acts: Authority Claims and Alignment Moves in Wikipedia Talk Pages. In *Proceedings of the Workshop on Languages in Social Media*, pages 48–57. Association for Computational Linguistics.

Robert M. Entman. 1993. Framing: Toward Clarification of a Fractured Paradigm. *Journal of Communication*, 43(4):51–58.

Rong En Fan, Kai-Wei Chang, Cho-Jui Hsieh, X.-R. Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *JMLR*.

Oliver Ferschke, Iryna Gurevych, and Yevgen Chebotar. 2012. Behind the Article: Recognizing Dialog Acts in Wikipedia Talk Pages. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 777–786. Association for Computational Linguistics.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network. In *Proceedings of 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, COLING*, pages 2012–2021.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying Dialogue Acts in One-on-one Live Chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 862–871. Association for Computational Linguistics.

Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. 2007. He Says, She Says: Conflict and Coordination in Wikipedia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI*, pages 453–462. ACM.

Robert E. Kraut, Paul Resnick, Sara Kiesler, Yuqing Ren, Yan Chen, Moira Burke, Niki Kittur, John Riedl, and Joseph Konstan. 2012. *Building Successful Online Communities: Evidence-Based Social Design*. The MIT Press.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing, ACL*, pages 1003–1011. Association for Computational Linguistics.

Alvaro Morales, Varot Premtoon, Cordelia Avery, Sue Felshin, and Boris Katz. 2016. Learning to Answer Questions from Wikipedia Infoboxes. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1930–1935. Association for Computational Linguistics.

Nona Naderi and Graeme Hirst. 2017. Classifying Frames at the Sentence Level in News Articles. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP*, pages 536–542.

Irwin P. Levin, Sandra L. Schneider, and Gary J. Gaeth. 1998. All Frames Are Not Created Equal: A Typology and Critical Analysis of Framing Effects. *Organizational Behavior and Human Decision Processes*, 76(2):149 – 188.

Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *Int. J. Cogn. Inform. Nat. Intell.*, 7(1):1–31.

Sara Rosenthal and Kathy McKeown. 2015. I Couldn't Agree More: The Role of Conversational Structure in Agreement and Disagreement Detection in Online Discussions. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL*, pages 168–177.

J.R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cam: Verschiedene Aufl. Cambridge University Press.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International World Wide Web Conference*, pages 613–624.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL-IJCNLP*, pages 1629–1638. Association for Computational Linguistics.

Fernanda B. Viegas, Martin Wattenberg, Jesse Kriss, and Frank van Ham. 2007. Talk Before You Type: Coordination in Wikipedia. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, HICSS '07, pages 78–. IEEE Computer Society.

Douglas Walton. 2010. Types of Dialogue and Burdens of Proof. In *Frontiers in Artificial Intelligence and Applications*, volume 216, pages 13–24.

Lu Wang and Claire Cardie. 2014. A Piece of My Mind: A Sentiment Analysis Approach for Online Dispute Detection. In *Proceedings of the52nd Annual Meeting of the Association for Computational Linguistics, ACL*, volume 2, pages 693–699. Association for Computational Linguistics.

Shiren Ye, Tat-Seng Chua, and Jie Lu. 2009. Summarizing Definition from Wikipedia. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing, ACL*, pages 199–207. Association for Computational Linguistics.

Elina Zarisheva and Tatjana Scheffler. 2015. Dialog Act Annotation for Twitter Conversations. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL*, pages 114–123.

Torsten Zesch, Christof Muller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC*. European Language Resources Association (ELRA).

Amy X. Zhang, Bryan Culbertson, and Praveen Paritosh. 2017a. Characterizing Online Discussion Using Coarse Discourse Sequences. In *Proceedings of the 11th International AAAI Conference on Weblogs and Social Media, ICWSM*, pages 357–366.

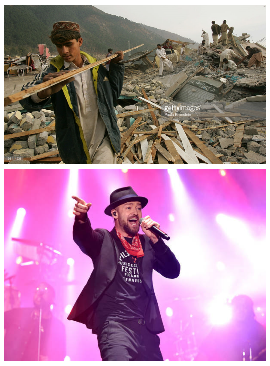
Amy X. Zhang, Lea Verou, and David Karger. 2017b. Wikum: Bridging Discussion Forums and Wikis Using Recursive Summarization. In *Proceedings of the 20th ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW*, pages 2082–2096. ACM.

Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning

Piyush Sharma, Nan Ding, Sebastian Goodman, Radu Soricut
Google AI
Venice, CA 90291
{piyushsharma,dingnan,seabass,rsoricut}@google.com

Abstract

We present a new dataset of image caption annotations, Conceptual Captions, which contains an order of magnitude more images than the MS-COCO dataset (Lin et al., 2014) and represents a wider variety of both images and image caption styles. We achieve this by extracting and filtering image caption annotations from billions of webpages. We also present quantitative evaluations of a number of image captioning models and show that a model architecture based on Inception-ResNet-v2 (Szegedy et al., 2016) for image-feature extraction and Transformer (Vaswani et al., 2017) for sequence modeling achieves the best performance when trained on the Conceptual Captions dataset.



Alt-text: A Pakistani worker helps to clear the debris from the Taj Mahal Hotel November 7, 2005 in Balakot, Pakistan.

Conceptual Captions: a worker helps to clear the debris.

Alt-text: Musician Justin Timberlake performs at the 2017 Pilgrimage Music & Cultural Festival on September 23, 2017 in Franklin, Tennessee.

Conceptual Captions: pop artist performs at the festival in a city.

Figure 1: Examples of images and image descriptions from the Conceptual Captions dataset; we start from existing alt-text descriptions, and automatically process them into Conceptual Captions with a balance of cleanliness, informativeness, fluency, and learnability.

1 Introduction

Automatic image description is the task of producing a natural-language utterance (usually a sentence) which correctly reflects the visual content of an image. This task has seen an explosion in proposed solutions based on deep learning architectures (Bengio, 2009), starting with the winners of the 2015 COCO challenge (Vinyals et al., 2015a; Fang et al., 2015), and continuing with a variety of improvements (see e.g. Bernardi et al. (2016) for a review). Practical applications of automatic image description systems include leveraging descriptions for image indexing or retrieval, and helping those with visual impairments by transforming visual signals into information that can be communicated via text-to-speech technology. The scientific challenge is seen as aligning, exploiting, and pushing further the latest improvements at the intersection of Computer Vision and Natural Language Processing.

There are two main categories of advances responsible for increased interest in this task. The first is the availability of large amounts of annotated data. Relevant datasets include the ImageNet dataset (Deng et al., 2009), with over 14 million images and 1 million bounding-box annotations, and the MS-COCO dataset (Lin et al., 2014), with 120,000 images and 5-way image-caption annotations. The second is the availability of powerful modeling mechanisms such as modern Convolutional Neural Networks (e.g. Krizhevsky et al. (2012)), which are capable of converting image pixels into high-level features with no manual feature-engineering.

In this paper, we make contributions to both the data and modeling categories. First, we present a new dataset of caption annotations[*], Conceptual Captions (Fig. 1), which has an order of magnitude more images than the COCO

[*]https://github.com/google-research-datasets/conceptual-captions

dataset. Conceptual Captions consists of about 3.3M $\langle image, description \rangle$ pairs. In contrast with the curated style of the COCO images, Conceptual Captions images and their raw descriptions are harvested from the web, and therefore represent a wider variety of styles. The raw descriptions are harvested from the Alt-text HTML attribute[†] associated with web images. We developed an automatic pipeline (Fig. 2) that extracts, filters, and transforms candidate image/caption pairs, with the goal of achieving a balance of cleanliness, informativeness, fluency, and learnability of the resulting captions.

As a contribution to the modeling category, we evaluate several image-captioning models. Based on the findings of Huang et al. (2016), we use Inception-ResNet-v2 (Szegedy et al., 2016) for image-feature extraction, which confers optimization benefits via residual connections and computationally efficient Inception units. For caption generation, we use both RNN-based (Hochreiter and Schmidhuber, 1997) and Transformer-based (Vaswani et al., 2017) models. Our results indicate that Transformer-based models achieve higher output accuracy; combined with the reports of Vaswani et al. (2017) regarding the reduced number of parameters and FLOPs required for training & serving (compared with RNNs), models such as T2T8x8 (Section 4) push forward the performance on image-captioning and deserve further attention.

2 Related Work

Automatic image captioning has a long history (Hodosh et al., 2013; Donahue et al., 2014; Karpathy and Fei-Fei, 2015; Kiros et al., 2015). It has accelerated with the success of Deep Neural Networks (Bengio, 2009) and the availability of annotated data as offered by datasets such as Flickr30K (Young et al., 2014) and MS-COCO (Lin et al., 2014).

The COCO dataset is not large (order of 10^6 images), given the training needs of DNNs. In spite of that, it has been very popular, in part because it offers annotations for images with non-iconic views, or non-canonical perspectives of objects, and therefore reflects the composition of everyday scenes (the same is true about Flickr30K (Young et al., 2014)). COCO annotations–category labeling, instance spotting, and instance segmentation–are done for all objects in an image, including those in the background, in a cluttered environment, or partially occluded. Its images are also annotated with captions, i.e. sentences produced by human annotators to reflect the visual content of the images in terms of objects and their actions or relations.

A large number of DNN models for image caption generation have been trained and evaluated using COCO captions (Vinyals et al., 2015a; Fang et al., 2015; Xu et al., 2015; Ranzato et al., 2015; Yang et al., 2016; Liu et al., 2017; Ding and Soricut, 2017). These models are inspired by sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) but use CNN-based encodings instead of RNNs (Hochreiter and Schmidhuber, 1997; Chung et al., 2014). Recently, the Transformer architecture (Vaswani et al., 2017) has been shown to be a viable alternative to RNNs (and CNNs) for sequence modeling. In this work, we evaluate the impact of the Conceptual Captions dataset on the image captioning task using models that combine CNN, RNN, and Transformer layers.

Also related to this work is the Pinterest image and sentence-description dataset (Mao et al., 2016). It is a large dataset (order of 10^8 examples), but its text descriptions do not strictly reflect the visual content of the associated image, and therefore cannot be used directly for training image-captioning models.

3 Conceptual Captions Dataset Creation

The Conceptual Captions dataset is programmatically created using a Flume (Chambers et al., 2010) pipeline. This pipeline processes billions of Internet webpages in parallel. From these webpages, it extracts, filters, and processes candidate $\langle image, caption \rangle$ pairs. The filtering and processing steps are described in detail in the following sections.

Image-based Filtering The first filtering stage, image-based filtering, discards images based on encoding format, size, aspect ratio, and offensive content. It only keeps JPEG images where both dimensions are greater than 400 pixels, and the ratio of larger to smaller dimension is no more than 2. It excludes images that trigger pornography or profanity detectors. These filters discard more than 65% of the candidates.

Text-based Filtering The second filtering stage, text-based filtering, harvests Alt-text from HTML webpages. Alt-text generally accompanies images,

[†]https://en.wikipedia.org/wiki/Alt_attribute

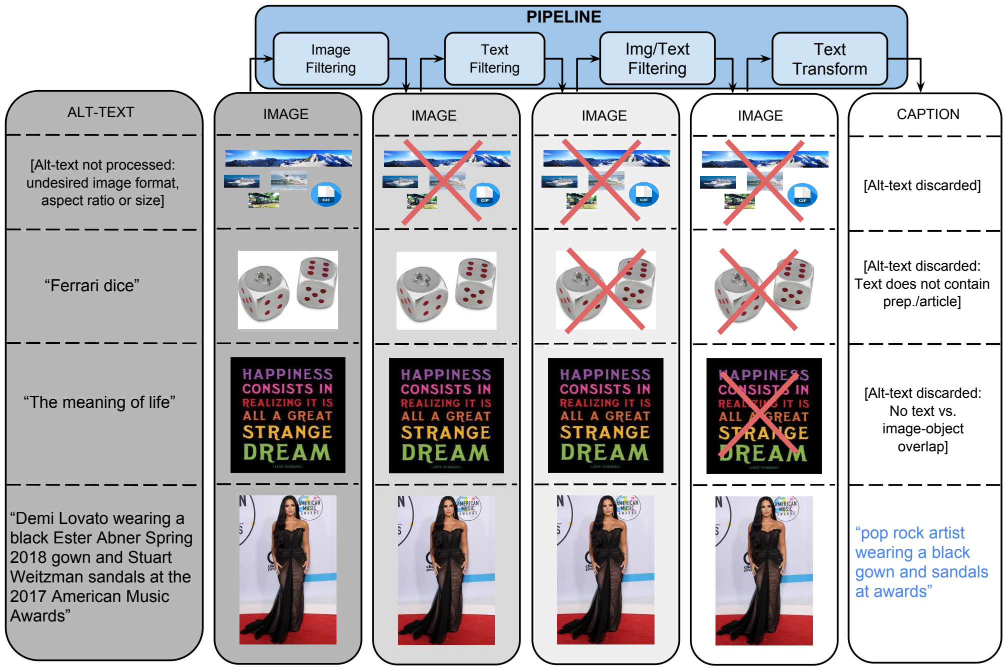

PIPELINE
Image Filtering
Text Filtering
Img/Text Filtering
Text Transform
ALT-TEXT
IMAGE
IMAGE
IMAGE
IMAGE
CAPTION
[Alt-text not processed: undesired image format, aspect ratio or size]
[Alt-text discarded]
"Ferrari dice"
[Alt-text discarded: Text does not contain prep./article]
"The meaning of life"
[Alt-text discarded: No text vs. image-object overlap]
"Demi Lovato wearing a black Ester Abner Spring 2018 gown and Stuart Weitzman sandals at the 2017 American Music Awards"
"pop rock artist wearing a black gown and sandals at awards"

Figure 2: Conceptual Captions pipeline steps with examples and final output.

and intends to describe the nature or the content of the image. Because these Alt-text values are not in any way restricted or enforced to be good image descriptions, many of them have to be discarded, e.g., search engine optimization (SEO) terms, or Twitter hash-tag terms.

We analyze candidate Alt-text using the Google Cloud Natural Language APIs, specifically part-of-speech (POS), sentiment/polarity, and pornography/profanity annotations. On top of these annotations, we have the following heuristics:

- a well-formed caption should have a high unique word ratio covering various POS tags; candidates with no determiner, no noun, or no preposition are discarded; candidates with a high noun ratio are also discarded;

- candidates with a high rate of token repetition are discarded;

- capitalization is a good indicator of well-composed sentences; candidates where the first word is not capitalized, or with too high capitalized-word ratio are discarded;

- highly unlikely tokens are a good indicator of not desirable text; we use a vocabulary V_W of 1B token types, appearing at least 5 times in the English Wikipedia, and discard candidates that contain tokens that are not found in this vocabulary.

- candidates that score too high or too low on the polarity annotations, or trigger the pornography/profanity detectors, are discarded;

- predefined boiler-plate prefix/suffix sequences matching the text are cropped, e.g. "click to enlarge picture", "stock photo"; we also drop text which begins/ends in certain patterns, e.g. "embedded image permalink", "profile photo".

These filters only allow around 3% of the incoming candidates to pass to the later stages.

Image&Text-based Filtering In addition to the separate filtering based on image and text content, we filter out candidates for which none of the text tokens can be mapped to the content of the image. To this end, we use classifiers available via the Google Cloud Vision APIs to assign class labels to images, using an image classifier with a large number of labels (order of magnitude of 10^5). Notably, these labels are also 100% covered by the V_w token types.

Images are generally assigned between 5 to 20 labels, though the exact number depends on the

Original Alt-text	Harrison Ford and Calista Flockhart attend the premiere of 'Hollywood Homicide' at the 29th American Film Festival September 5, 2003 in Deauville, France.
Conceptual Captions	actors attend the premiere at festival.
what-happened	"Harrison Ford and Calista Flockhart" mapped to "actors"; name, location, and date dropped.
Original Alt-text	Side view of a British Airways Airbus A319 aircraft on approach to land with landing gear down - Stock Image
Conceptual Captions	side view of an aircraft on approach to land with landing gear down
what-happened	phrase "British Airways Airbus A319 aircraft" mapped to "aircraft"; boilerplate removed.
Original Alt-text	Two sculptures by artist Duncan McKellar adorn trees outside the derelict Norwich Union offices in Bristol, UK - Stock Image
Conceptual Captions	sculptures by person adorn trees outside the derelict offices
what-happened	object count (e.g. "Two") dropped; proper noun-phrase hypernymized to "person"; proper-noun modifiers dropped; location dropped; boilerplate removed.

Table 1: Examples of Conceptual Captions as derived from their original Alt-text versions.

image. We match these labels against the candidate text, taking into account morphology-based stemming as provided by the text annotation. Candidate $\langle image, caption\rangle$ pairs with no overlap are discarded. This filter discards around 60% of the incoming candidates.

Text Transformation with Hypernymization In the current version of the dataset, we considered over 5 billion images from about 1 billion English webpages. The filtering criteria above are designed to be high-precision (which comes with potentially low recall). From the original input candidates, only 0.2% $\langle image, caption\rangle$ pairs pass the filtering criteria described above.

While the remaining candidate captions tend to be appropriate Alt-text image descriptions (see Alt-text in Fig. 1), a majority of these candidate captions contain proper names (people, venues, locations, etc.), which would be extremely difficult to learn as part of the image captioning task. To give an idea of what would happen in such cases, we train an RNN-based captioning model (see Section 4) on non-hypernymized Alt-text data and present an output example in Fig. 3. If automatic determination of person identity, location, etc. is needed, it should be attempted as a separate task and would need to leverage image meta-information about the image (e.g. location).

Using the Google Cloud Natural Language APIs, we obtain named-entity and syntactic-dependency annotations. We then use the Google Knowledge Graph (KG) Search API to match the named-entities to KG entries and exploit the associated hypernym terms. For instance, both "Harrison Ford" and "Calista Flockhart" identify as named-entities,

Alt-text (groundtruth): Jimmy Barnes performs at the Sydney Entertainment Centre

Model output: Singer Justin Bieber performs onstage during the Billboard Music Awards at the MGM

Figure 3: Example of model output trained on clean, non-hypernymized Alt-text data.

so we match them to their corresponding KG entries. These KG entries have "actor" as their hypernym, so we replace the original surface tokens with that hypernym.

The following steps are applied to achieve text transformations:

- noun modifiers of certain types (proper nouns, numbers, units) are removed;
- dates, durations, and preposition-based locations (e.g., "in Los Angeles") are removed;
- named-entities are identified, matched against the KG entries, and substitute with their hypernym;
- resulting coordination noun-phrases with the same head (e.g., "actor and actor") are resolved into a single-head, pluralized form (e.g., "actors");

Around 20% of samples are discarded during this transformation because it can leave sentences too short or inconsistent.

Finally, we perform another round of text analysis and entity resolution to identify concepts with low-count. We cluster all resolved entities (e.g., "actor", "dog", "neighborhood", etc.) and keep only the candidates for which all detected types have a count of over 100 (around 55% of the candidates). These remaining $\langle image, caption \rangle$ pairs contain around 16,000 entity types, guaranteed to be well represented in terms of number of examples. Table 1 contains several examples of before/after-transformation pairs.

Conceptual Captions Quality To evaluate the precision of our pipeline, we consider a random sample of 4K examples extracted from the test split of the Conceptual Captions dataset. We perform a human evaluation on this sample, using the same methodology described in Section 5.4.

	GOOD (out of 3)		
	1+	2+	3
Conceptual Captions	96.9%	90.3%	78.5%

Table 2: Human evaluation results on a sample from Conceptual Captions.

The results are presented in Table 2 and show that, out of 3 annotations, over 90% of the captions receive a majority (2+) of GOOD judgments. This indicates that the Conceptual Captions pipeline, though involving extensive algorithmic processing, produces high-quality image captions.

	Examples	Unique Tokens	Tokens/Caption		
			Mean	StdDev	Median
Train	3,318,333	51,201	10.3	4.5	9.0
Valid.	28,355	13,063	10.3	4.6	9.0
Test	22,530	11,731	10.1	4.5	9.0

Table 3: Statistics over Train/Validation/Test splits for Conceptual Captions.

We present in Table 3 statistics over the Train/Validation/Test splits for the Conceptual Captions dataset. The training set consists of slightly over 3.3M examples, while there are slightly over 28K examples in the validation set and 22.5K examples in the test set. The size of the training set vocabulary (unique tokens) is 51,201. Note that the test set has been cleaned using human judgements (2+ GOOD), while both the training and validation splits contain all the data, as produced by our automatic pipeline. The mean/stddev/median statistics for tokens-per-caption over the data splits are consistent with each other, at around 10.3/4.5/9.0, respectively.

4 Image Captioning Models

In order to assess the impact of the Conceptual Captions dataset, we consider several image captioning models previously proposed in the literature. These models can be understood using the illustration in Fig. 4, as they mainly differ in the way in which they instantiate some of these components.

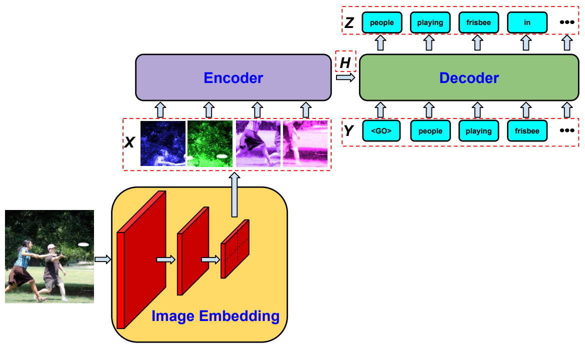


Figure 4: The main model components.

There are three main components to this architecture:

- A deep CNN that takes a (preprocessed) image and outputs a vector of image embeddings $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L)$.
- An Encoder module that takes the image embeddings and encodes them into a tensor $\mathbf{H} = f_{\text{enc}}(\mathbf{X})$.
- A Decoder model that generates outputs $\mathbf{z}_t = f_{\text{dec}}(\mathbf{Y}_{1:t}, \mathbf{H})$ at each step t, conditioned on $\mathbf{H}$ as well as the decoder inputs $\mathbf{Y}_{1:t}$.

We explore two main instantiations of this architecture. One uses RNNs with LSTM cells (Hochreiter and Schmidhuber, 1997) to implement the f_{enc} and f_{dec} functions, corresponding to the Show-And-Tell (Vinyals et al., 2015b) model. The other uses Transformer self-attention networks (Vaswani et al., 2017) to implement f_{enc} and f_{dec}. All models in this paper use Inception-ResNet-v2 as the CNN component (Szegedy et al., 2016).

4.1 RNN-based Models

Our instantiation of the RNN-based model is close to the Show-And-Tell (Vinyals et al., 2015b) model.

$$\mathbf{h}_l \triangleq \text{RNN}_{enc}(\mathbf{x}_l, \mathbf{h}_{l-1}), \text{ and } \mathbf{H} = \mathbf{h}_L,$$
$$\mathbf{z}_t \triangleq \text{RNN}_{dec}(\mathbf{y}_t, \mathbf{z}_{t-1}), \text{ where } \mathbf{z}_0 = \mathbf{H}.$$

In the original Show-And-Tell model, a single image embedding of the entire image is fed to the first cell of an RNN, which is also used for text generation. In our model, a single image embedding is fed to an RNN$_{enc}$ with only one cell, and then a different RNN$_{dec}$ is used for text generation. We tried both single image (1x1) embeddings and 8x8 partitions of the image, where each partition has its own embedding. In the 8x8 case, image embeddings are fed in a sequence to the RNN$_{enc}$. In both cases, we apply plain RNNs without cross attention, same as the Show-And-Tell model. RNNs with cross attention were used in the Show-Attend-Tell model (Xu et al., 2015), but we find its performance to be inferior to the Show-And-Tell model.

4.2 Transformer Model

In the Transformer-based models, both the encoder and the decoder contain a stack of N layers. We denote the n-th layer in the encoder by $\mathbf{X}_n = \{\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,L}\}$, and $\mathbf{X}_0 = \mathbf{X}$, $\mathbf{H} = \mathbf{X}_N$. Each of these layers contains two sub-layers: a multi-head self-attention layer ATTN, and a position-wise feedforward network FFN:

$$\begin{aligned}\mathbf{x}'_{n,j} &= \text{ATTN}(\mathbf{x}_{n,j}, \mathbf{X}_n; \mathbf{W}^e_q, \mathbf{W}^e_k, \mathbf{W}^e_v) \\ &\triangleq \text{softmax}(\langle \mathbf{x}_{n,j}\, \mathbf{W}^e_q, \mathbf{X}_n\, \mathbf{W}^e_k \rangle)\, \mathbf{X}_n\, \mathbf{W}^e_v \\ \mathbf{x}_{(n+1),j} &= \text{FFN}(\mathbf{x}'_{n,j}; \mathbf{W}^e_f)\end{aligned}$$

where $\mathbf{W}^e_q$, $\mathbf{W}^e_k$, and $\mathbf{W}^e_v$ are the encoder weight matrices for query, key, and value transformation in the self-attention sub-layer; and $\mathbf{W}^e_f$ denotes the encoder weight matrix of the feedforward sub-layer. Similar to the RNN-based model, we consider using a single image embedding (1x1) and a vector of 8x8 image embeddings.

In the decoder, we denote the n-th layer by $\mathbf{Z}_n = \{\mathbf{z}_{n,1}, \ldots, \mathbf{z}_{n,T}\}$ and $\mathbf{Z}_0 = \mathbf{Y}$. There are two main differences between the decoder and encoder layers. First, the self-attention sub-layer in the decoder is masked to the right, in order to prevent attending to "future" positions (i.e. $\mathbf{z}_{n,j}$ does not attend to $\mathbf{z}_{n,(j+1)}, \ldots, \mathbf{z}_{n,T}$). Second, in between the self-attention layer and the feedforward layer, the decoder adds a third cross-attention layer that connects $\mathbf{z}_{n,j}$ to the top-layer encoder representation $\mathbf{H} = \mathbf{X}_N$.

$$\begin{aligned}\mathbf{z}'_{n,j} &= \text{ATTN}(\mathbf{z}_{n,j}, \mathbf{Z}_{n,1:j}; \mathbf{W}^d_q, \mathbf{W}^d_k, \mathbf{W}^d_v) \\ \mathbf{z}''_{n,j} &= \text{ATTN}(\mathbf{z}'_{n,j}, \mathbf{H}; \mathbf{W}^c_q, \mathbf{W}^c_k, \mathbf{W}^c_v) \\ \mathbf{z}_{(n+1),j} &= \text{FFN}(\mathbf{z}''_{n,j}; \mathbf{W}^d_f)\end{aligned}$$

where $\mathbf{W}^d_q$, $\mathbf{W}^d_k$, and $\mathbf{W}^d_v$ are the weight matrices for query, key, and value transformation in the decoder self-attention sub-layer; $\mathbf{W}^c_q$, $\mathbf{W}^c_k$, $\mathbf{W}^c_v$ are the corresponding decoder weight matrices in the cross-attention sub-layer; and $\mathbf{W}^d_f$ is the decoder weight matrix of the feedforward sub-layer.

The Transformer-based models utilize position information at the embedding layer. In the 8x8 case, the 64 embedding vectors are serialized to a 1D sequence with positions from $[0, \ldots, 63]$. The position information is modeled by applying sine and cosine functions at each position and with different frequencies for each embedding dimension, as in (Vaswani et al., 2017), and subsequently added to the embedding representations.

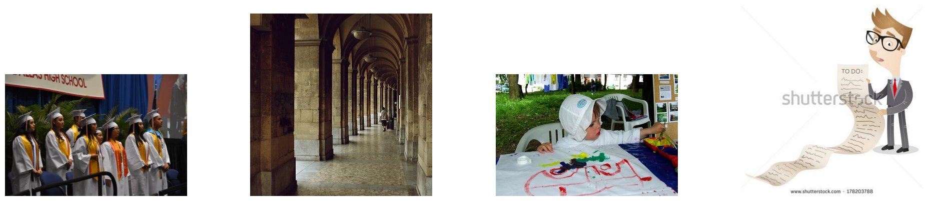
5 Experimental Results

In this section, we evaluate the impact of using the Conceptual Captions dataset (referred to as 'Conceptual' in what follows) for training image captioning models. To this end, we train the models described in Section 4 under two experimental conditions: using the training & development sets provided by the COCO dataset (Lin et al., 2014), versus training & development sets using the Conceptual dataset. We quantitatively evaluate the resulting models using three different test sets: the blind COCO-C40 test set (in-domain for COCO-trained models, out-of-domain for Conceptual-trained models); the Conceptual test set (out-of-domain for COCO-trained models, in-domain for Conceptual-trained models); and the Flickr (Young et al., 2014) 1K test set (out-of-domain for both COCO-trained models and Conceptual-trained models).

5.1 Dataset Details

COCO Image Captions The COCO image captioning dataset is normally divided into 82K images for training, and 40K images for validation. Each of these images comes with at least 5 groundtruth captions. Following standard practice, we combine the training set with most of the validation dataset for training our model, and only hold out a subset of 4K images for validation.

Conceptual Captions The Conceptual Captions dataset contains around 3.3M images for training, 28K for validation and 22.5K for the test set. For more detailed statistics, see Table 3.



COCO-trained				
RNN8x8	a group of men standing in front of a building	a couple of people walking down a walkway	a child sitting at a table with a cake on it	a close up of a stuffed animal on a table
T2T8x8	a group of men in uniform and ties are talking	a narrow hallway with a clock and two doors	a woman cutting a birthday cake at a party	a picture of a fish on the side of a car
Conceptual-trained				
RNN8x8	graduates line up for the commencement ceremony	a view of the nave	a child 's drawing at a birthday party	a cartoon businessman thinking about something
T2T8x8	graduates line up to receive their diplomas	the cloister of the cathedral	learning about the arts and crafts	a cartoon businessman asking for help

Figure 5: Side by side comparison of model outputs under two training conditions. Conceptual-based models (lower half) tend to hallucinate less, are more expressive, and handle well a larger variety of images. The two images in the middle are from Flickr; the other two are from Conceptual Captions.

5.2 Experimental Setup

Image Preprocessing Each input image is first preprocessed by random distortion and cropping (using a random ratio from 50%∼100%). This prevents models from overfitting individual pixels of the training images.

Encoder-Decoder For RNN-based models, we use a 1-layer, 512-dim LSTM as the RNN cell. For the Transformer-based models, we use the default setup from (Vaswani et al., 2017), with $N = 6$ encoder and decoder layers, a hidden-layer size of 512, and 8 attention heads.

Text Handling Training captions are truncated to maximum 15 tokens. We use a token type min-count of 4, which results in around 9,000 token types for the COCO dataset, and around 25,000 token types for the Conceptual Captions dataset. All other tokens are replaced with special token ⟨UNK⟩. The word embedding matrix has size 512 and is tied to the output projection matrix.

Optimization All models are trained using MLE loss and optimized using Adagrad (Duchi et al., 2011) with learning rate 0.01. Mini-batch size is 25. All model parameters are trained for a total number of 5M steps, with batch updates asynchronously distributed across 40 workers. The final model is selected based on the best CIDEr score on the development set for the given training condition.

Inference During inference, the decoder prediction of the previous position is fed to the input of the next position. We use a beam search of beam size 4 to compute the most likely output sequence.

5.3 Qualitative Results

Before we present the numerical results for our experiments, we discuss briefly the patterns that we have observed.

One difference between COCO-trained models and Conceptual-trained models is their ability to use the appropriate natural language terms for the entities in an image. For the left-most image in Fig. 5, COCO-trained models use "group of men" to refer to the people in the image; Conceptual-based models use the more appropriate and informative term "graduates". The second image, from the Flickr test set, makes this even more clear. The Conceptual-trained T2T8x8 model is perfectly rendering the image content as "the cloister of the cathedral". None of the other models come close to producing such an accurate description.

A second difference is that COCO-trained models often seem to hallucinate objects. For instance, they hallucinate "front of building" for the first image, "clock and two doors" for the second, and "birthday cake" for the third image. In contrast, Conceptual-trained models do not seem to have this problem. We hypothesize that the hallucination issue for COCO-based models comes from the high correlations present in the COCO data (e.g., if there is a kid at a table, there is also cake). This high degree of correlation in the data does not allow the captioning model to correctly disentangle and learn representations at the right level of granularity.

Model	Training	1+	2+	3+
RNN8x8	COCO	0.390	0.276	0.173
T2T8x8	COCO	0.478	0.362	0.275
RNN8x8	Conceptual	0.571	0.418	0.277
T2T8x8	Conceptual	0.659	0.506	0.355

Table 4: Human eval results on Flickr 1K Test.

A third difference is the resilience to a large spectrum of image types. COCO only contains natural images, and therefore a cartoon image like the fourth one results in massive hallucination effects for COCO-trained models (“stuffed animal”, “fish”, “side of car”). In contrast, Conceptual-trained models handle such images with ease.

5.4 Quantitative Results

In this section, we present quantitative results on the quality of the outputs produced by several image captioning models. We present both automatic evaluation results and human evaluation results.

5.4.1 Human Evaluation Results

For human evaluations, we use a pool of professional raters (tens of raters), with a double-blind evaluation condition. Raters are asked to assign a GOOD or BAD label to a given $\langle image, caption \rangle$ input, using just common-sense judgment. This approximates the reaction of a typical user, who normally would not accept predefined notions of GOOD vs. BAD. We ask 3 separate raters to rate each input pair and report the percentage of pairs that receive k or more (k+) GOOD annotations.

In Table 4, we report the results on the Flickr 1K test set. This evaluation is out-of-domain for both training conditions, so all models are on relatively equal footing. The results indicate that the Conceptual-based models are superior. In 50.6% (for the T2T8x8 model) of cases, a majority of annotators (2+) assigned a GOOD label. The results also indicate that the Transformer-based models are superior to the RNN-based models by a good margin, by over 8-points (for 2+) under both COCO and Conceptual training conditions.

Model	Training	CIDEr	ROUGE-L	METEOR
RNN1x1	COCO	1.021	0.694	0.348
RNN8x8	COCO	1.044	0.698	0.354
T2T1x1	COCO	1.032	0.700	0.358
T2T8x8	COCO	1.032	0.700	0.356
RNN1x1	Conceptual	0.403	0.445	0.191
RNN8x8	Conceptual	0.410	0.437	0.189
T2T1x1	Conceptual	0.348	0.403	0.171
T2T8x8	Conceptual	0.345	0.400	0.170

Table 5: Auto metrics on the COCO C40 Test.

Model	Training	CIDEr	ROUGE-L	SPICE
RNN1x1	COCO	0.183	0.149	0.062
RNN8x8	COCO	0.191	0.152	0.065
T2T1x1	COCO	0.184	0.148	0.062
T2T8x8	COCO	0.190	0.151	0.064
RNN1x1	Conceptual	1.351	0.326	0.235
RNN8x8	Conceptual	1.401	0.330	0.240
T2T1x1	Conceptual	1.588	0.331	0.254
T2T8x8	Conceptual	1.676	0.336	0.257

Table 6: Auto metrics on the 22.5K Conceptual Captions Test set.

Model	Training	CIDEr	ROUGE-L	SPICE
RNN1x1	COCO	0.340	0.414	0.101
RNN8x8	COCO	0.356	0.413	0.103
T2T1x1	COCO	0.341	0.404	0.101
T2T8x8	COCO	0.359	0.416	0.103
RNN1x1	Conceptual	0.269	0.310	0.076
RNN8x8	Conceptual	0.275	0.309	0.076
T2T1x1	Conceptual	0.226	0.280	0.068
T2T8x8	Conceptual	0.227	0.277	0.066

Table 7: Auto metrics on the Flickr 1K Test.

5.4.2 Automatic Evaluation Results

In this section, we report automatic evaluation results, using established image captioning metrics.

For the COCO C40 test set (Fig. 5), we report the numerical values returned by the COCO online evaluation server[‡], using the CIDEr (Vedantam et al., 2015), ROUGE-L (Lin and Och, 2004), and METEOR (Banerjee and Lavie, 2005) metrics. For Conceptual Captions (Fig. 6) and Flickr (Fig. 7) test sets, we report numerical values for the CIDEr, ROUGE-L, and SPICE (Anderson et al., 2016)[§]. For all metrics, higher number means closer distance between the candidates and the groundtruth captions.

The automatic metrics are good at detecting in- vs out-of-domain situations. For COCO-models tested on COCO, the results in Fig. 5 show CIDEr scores in the 1.02-1.04 range, for both RNN- and Transformer-based models; the scores drop in the 0.35-0.41 range (CIDEr) for the Conceptual-based models tested against COCO groundtruth. For Conceptual-models tested on the Conceptual Captions test set, the results in Fig. 6 show scores as high as 1.468 CIDEr for the T2T8x8 model, which corroborates the human-eval results for the Transformer-based models being superior to the RNN-based models; the scores for the COCO-based models tested against Conceptual Captions groundtruth are all below 0.2 CIDEr.

The automatic metrics fail to corroborate the

[‡]http://mscoco.org/dataset/#captions-eval.

[§]https://github.com/tylin/coco-caption.

human evaluation results. According to the automatic metrics, the COCO-trained models are superior to the Conceptual-trained models (CIDEr scores in the mid-0.3 for the COCO-trained condition, versus mid-0.2 for the Conceptual-trained condition), and the RNN-based models are superior to Transformer-based models. Notably, these are the same metrics which score humans lower than the methods that won the COCO 2015 challenge (Vinyals et al., 2015a; Fang et al., 2015), despite the fact that humans are still much better at this task. The failure of these metrics to align with the human evaluation results casts again grave doubts on their ability to drive progress in this field. A significant weakness of these metrics is that hallucination effects are under-penalized (a small precision penalty for tokens with no correspondent in the reference), compared to human judgments that tend to dive dramatically in the presence of hallucinations.

6 Conclusions

We present a new image captioning dataset, Conceptual Captions, which has several key characteristics: it has around 3.3M examples, an order of magnitude larger than the COCO image-captioning dataset; it consists of a wide variety of images, including natural images, product images, professional photos, cartoons, drawings, etc.; and, its captions are based on descriptions taken from original Alt-text attributes, automatically transformed to achieve a balance between cleanliness, informativeness, and learnability.

We evaluate both the quality of the resulting image/caption pairs, as well as the performance of several image-captioning models when trained on the Conceptual Captions data. The results indicate that such models achieve better performance, and avoid some of the pitfalls seen with COCO-trained models, such as object hallucination. We hope that the availability of the Conceptual Captions dataset will foster considerable progress on the automatic image-captioning task.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. In *ECCV*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.* 2(1):1–127.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *JAIR* 55.

Craig Chambers, Ashish Raniwala, Frances Perry, Stephen Adams, Robert Henry, Robert Bradshaw, and Nathan. 2010. Flumejava: Easy, efficient data-parallel pipelines. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 2 Penn Plaza, Suite 701 New York, NY 10121-0701, pages 363–375. http://dl.acm.org/citation.cfm?id=1806638.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*.

Nan Ding and Radu Soricut. 2017. Cold-start reinforcement learning with softmax policy gradients. In *NIPS*.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2014. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. 2015. From captions to visual concepts and back. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR* .

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. 2016. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR* abs/1611.10012.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2015. Unifying visual-semantic embeddings with multimodal neural language models. *Transactions of the Association for Computational Linguistics* .

A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of ACL*.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR* abs/1405.0312.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Optimization of image description metrics using policy gradient methods. In *International Conference on Computer Vision (ICCV)*.

Junhua Mao, Jiajing Xu, Yushi Jing, and Alan Yuille. 2016. Training and evaluating multimodal word embeddings with large-scale web annotated images. In *NIPS*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR* abs/1602.07261.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015a. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3156–3164.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. of the 32nd International Conference on Machine Learning (ICML)*.

Z. Yang, Y. Yuan, Y. Wu, R. Salakhutdinov, and W. W. Cohen. 2016. Review networks for caption generation. In *NIPS*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL* 2:67–78.

Learning Translations via Images with a Massively Multilingual Image Dataset

John Hewitt* Daphne Ippolito* Brendan Callahan Reno Kriz
Derry Wijaya Chris Callison-Burch
University of Pennsylvania
Computer and Information Science Department
{johnhew,daphnei,rekriz,derry,ccb}@seas.upenn.edu

Abstract

We conduct the most comprehensive study to date into translating words via images. To facilitate research on the task, we introduce a large-scale multilingual corpus of images, each labeled with the word it represents. Past datasets have been limited to only a few high-resource languages and unrealistically easy translation settings. In contrast, we have collected by far the largest available dataset for this task, with images for approximately 10,000 words in each of 100 languages. We run experiments on a dozen high resource languages and 20 low resources languages, demonstrating the effect of word concreteness and part-of-speech on translation quality. To improve image-based translation, we introduce a novel method of predicting word concreteness from images, which improves on a previous state-of-the-art unsupervised technique. This allows us to predict when image-based translation may be effective, enabling consistent improvements to a state-of-the-art text-based word translation system. Our code and the Massively Multilingual Image Dataset (MMID) are available at http://multilingual-images.org/.

1 Introduction

Learning the translations of words is important for machine translation and other tasks in natural language processing. Typically this learning is done using sentence-aligned bilingual parallel texts. However, for many languages, there are not sufficiently large parallel texts to effectively learn translations. In this paper, we explore the question of whether it is possible to learn translations with images. We systematically explore an idea originally proposed by Bergsma and Van Durme (2011): translations can be identified via images associated with words in different languages that have a high degree of visual similarity. This is illustrated in Figure 1.

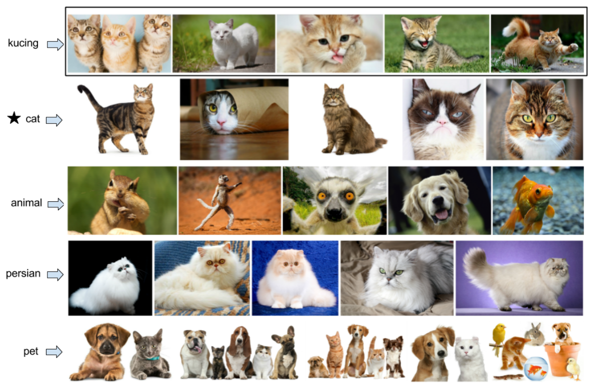


Figure 1: Our dataset and approach allow translations to be discovered by comparing the images associated with foreign and English words. Shown here are five images for the Indonesian word *kucing*, a word with high predicted concreteness, along with its top 4 ranked translations using CNN features.

Most previous image datasets compiled for the task of learning translations were limited to the translation of nouns in a few high-resource languages. In this work, we present a new large-scale dataset that contains images for 100 languages, and is not restricted by part-of-speech. We collected images using Google Image Search for up to 10,000 words in each of 100 foreign languages, and their English translations. For each word, we collected up to 100 images and the text on images' corresponding web pages.

We conduct a broad range of experiments to evaluate the utility of image features across a number of factors:

*These authors contributed equally; listed alphabetically.

- We evaluate on 12 high-resource and 20 low-resource languages.
- We evaluate translation quality stratified by part-of-speech, finding that nouns and adjectives are translated with much higher accuracy than adverbs and verbs.
- We present a novel method for predicting word concreteness from image features that better correlates with human perception than existing methods. We show that choosing concrete subsets of words to translate results in higher accuracy.
- We augment a state-of-the-art text-based word translation system with image feature scores and find consistent improvements to the text-only system, ranging from 3.12% absolute top-1 accuracy improvement at 10% recall to 1.30% absolute improvement at 100% recall.

A further contribution of this paper is our dataset, which is the largest of its kind and should be a standard for future work in learning translations from images. The dataset may facilitate research into multilingual, multimodal models, and translation of low-resource languages.

2 Related Work

The task of learning translations without sentence-aligned bilingual parallel texts is often called bilingual lexicon induction (Rapp, 1999; Fung and Yee, 1998). Most work in bilingual lexicon induction has focused on text-based methods. Some researchers have used similar spellings across related languages to find potential translations (Koehn and Knight, 2002; Haghighi et al., 2008). Others have exploited temporal similarity of word frequencies to induce translation pairs (Schafer and Yarowsky, 2002; Klementiev and Roth, 2006). Irvine and Callison-Burch (2017) provide a systematic study of different text-based features used for bilingual lexicon induction. Recent work has focused on building joint distributional word embedding spaces for multiple languages, leveraging a range of levels of language supervision from bilingual dictionaries to comparable texts (Vulić and Korhonen, 2016; Wijaya et al., 2017).

The most closely related work to ours is research into bilingual lexicon induction using image similarity by Bergsma and Van Durme (2011) and Kiela et al. (2015). Their work differs from ours in that they focused more narrowly on the translation of nouns for a limited number of high resource languages. Bergsma and Van Durme (2011) compiled datasets for Dutch, English, French, German, Italian, and Spanish by downloading 20 images for up to 500 concrete nouns in each of the foreign languages, and 20,000 English words.

Another dataset was generated by Vulic and Moens (2013) who collected images for 1,000 words in Spanish, Italian, and Dutch, along with the English translations for each. Their dataset also consists of only nouns, but includes abstract nouns. Our corpus will allow researchers to explore image similarity for bilingual lexicon induction on a much wider range of languages and parts of speech, which is especially desirable given the potential utility of the method for improving translation between languages with little parallel text.

The ability of images to usefully represent a word is strongly dependent on how concrete or abstract the word is. The terms *abstractness* and *concreteness* are used in the psycholinguistics and cognitive psychology literature. *Concrete* words directly reference a sense experience (Paivio et al., 1968), while abstract words can denote ideas, emotions, feelings, qualities or other abstract or intangible concepts. Concreteness ratings are closely correlated with *imagery* ratings, defined as the ease with which a word arouses a mental image (Gilhooly and Logie, 1980; Friendly et al., 1982). Intuitively, concrete words are easier to represent visually, so a measure of a word's concreteness ought to be able to predict the effectiveness of using images to translate the word.

Kiela et al. (2014) defines an unsupervised method called *image dispersion* that approximates a word's concreteness by taking the average pairwise cosine distance of a set of image representations of the word. Kiela et al. (2015) show that image dispersion helps predict the usefulness of image representations for translation. In this paper, we introduce novel supervised approaches for predicting word concreteness from image and textual features. We make use of a dataset created by Brysbaert et al. (2014) containing human evaluations of concreteness for 39,954 English words.

Concurrently with our work, Hartmann and Søgaard (2017) released an unpublished arXiv draft challenging the efficacy of using images for translation. Their work presents several difficulties of using image features for translation, difficulties which our methods address. They find that image features are only useful in translating simple nouns. While we did indeed find that nouns perform better than other parts of speech, we do not find that images are only effective in translating simple words. Instead, we show a gradual degradation in performance as words become more abstract. Their dataset is restricted to six high-resource languages and a small vocabulary of 557 English words. In contrast, we present results for over 260,000 English words and 32 foreign languages.

Recent research in the NLP and computer vision communities has been enabled by large collections of images associated with words or longer texts. Object recognition has seen dramatic gains in part due to the ImageNet database (Deng et al., 2009), which contains 500-1000 images associated with 80,000 synsets in WordNet. Ferraro et al. (2015) surveys existing corpora that are used in vision and language research. Other NLP+Vision tasks that have been enabled by the availability of large datasets include caption generation for images, action recognition in videos, visual question answering, and others.

Most existing work on multilingual NLP+Vision relies on having a corpus of images manually annotated with captions in several languages, as in the Multi30K dataset (Elliott et al., 2016). Several works have proposed using image features to improve sentence level translations or to translate image captions (Gella et al., 2017; Hitschler and Riezler, 2016; Miyazaki and Shimizu, 2016). Funaki and Nakayama (2015) show that automatically scraped data from websites in English and Japanese can be used to effectively perform zero-shot learning for the task of cross-lingual document retrieval. Since collecting multilingual annotations is difficult at a large-scale or for low-resource languages, our approach relies only on data scraped automatically from the web.

3 Corpus Construction

We present a new dataset for image-based word translation that is more expansive than any previous ones, encompassing all parts-of-speech, the gamut of abstract to concrete, and both low- and high-resource languages.

3.1 Dictionaries

We collect images for words in 100 bilingual dictionaries created by Pavlick et al. (2014). They selected the 10,000 most frequent words on Wikipedia pages in the foreign language, and then collected their translations into English via crowdsourcing. We will denote these dictionaries as CROWDTRANS. The superset of English translations for all foreign words consists of 263,102 translations. The English portion of their data tends to be much noisier than the foreign portion due to its crowdsourced nature (e.g. misspellings, or definition included with translations.)

We computed part-of-speech for entries in each dictionary. We found that while nouns are the most common, other parts-of-speech are reasonably represented (Section 5.1).

3.2 Method

For each English and foreign word, we query Google Image Search to collect 100 images associated with the word. A potential criticism of our use of Google Image Search is that it may be using a bilingual dictionary to translate queries into English (or other high resource languages) and returning images associated with the translated queries (Kilgarriff, 2007). We take steps (Section 3.3) to filter out images that did not appear on pages written in the language that we are gathering images for. After assembling the collection of images associated with words, we construct low-dimensional vector representations of the images using convolutional neural networks (CNNs). We also save the text from each web page that an image appeared on. Further detail on our corpus construction pipeline can be found in Section 2 of the supplemental materials.

3.3 Filtering by Web Page Language

We used the following heuristic to filter images: if text could be extracted from an image's web page, and the expected language was in the top-3 most likely languages output by the CLD2[1] language detection system then we kept the image; otherwise it was discarded. This does not filter all images from webpages with English text; instead it acknowledges the presence of English in the multilingual web and keeps images from pages with some target-language presence. An average of approximately 42% of images for each foreign language remained after the language-filtering step.

[1]`https://github.com/CLD2Owners/cld2`

Language	Concreteness Ratings				Overall
	1-2	2-3	3-4	4-5	
English	.804	.814	.855	.913	.857
French	.622	.653	.706	.828	.721
Indonesian	.505	.569	.665	.785	.661
Uzbek	.568	.530	.594	.683	.601
All	.628	.649	.713	.810	.717
# Words	77	292	292	302	963

Table 1: The proportion of images determined to be good representations of their corresponding word. In columns 2-5, we bucket the results by the word's ground-truth concreteness, while column 6 shows the results over all words. The last row shows the number of words in each bucket of concreteness, and the number of words overall for each language.

3.4 Manual Evaluation of Images

By using a dataset scraped from the web, we expect some fraction of the images for each word to be incorrectly labeled. To confirm the overall quality of our dataset, we asked human evaluators on Amazon Mechanical Turk to label a subset of the images returned by queries in four languages: our target language, English; a representative high-resource language, French; and two low-resource languages, Indonesian and Uzbek. In total, we collected 36,050 judgments of whether the images returned by Google Image Search were a good match for the keyword. Details on the experimental setup can be found in Section 1 of the Supplemental Materials.

Table 1 shows the fraction of images that were judged to be good representations of the search word. It also demonstrates that as the concreteness of a word increases, the proportion of good images associated with that word increases as well. We further discuss the role of concreteness in Section 6.1. Overall, 85% of the English images, 72% of French, 66% of Indonesian, and 60% of Uzbek were judged to be good.

4 Finding Translations Using Images

Can images help us learn translations for low-resource languages? In this section we replicate prior work in high-resource languages, and then evaluate on a wide array of low-resource languages.

Although we scraped images and text for 100 languages, we have selected a representative set of 32 for evaluation. Kiela et al. (2015) established that CNN features are superior to the SIFT plus color histogram features used by Bergsma and Van Durme (2011), and so we restrict all analysis to the former.

4.1 Translation Prediction with AvgMax

To learn the English translation of each foreign word, we rank the English words as candidate translations based on their visual similarity with the foreign words. We take the cosine similarity score for each image i_f associated the foreign word w_f with each of image i_e for the English word w_e, and then compute the average maximum similarity as

$$\textsc{AvgMax}(w_f, w_e) = \frac{1}{|w_f|} \sum_{i_f \in w_f} \max_{i_e \in w_e}(cosine(i_f, i_e))$$

Each image is represented by a 4096-dimensional vector from the fully connected 7th (FC7) layer of a CNN trained on ImageNet (Krizhevsky et al., 2012). AvgMax is the best-performing method described by Bergsma and Van Durme (2011) on images created with SIFT and color histogram features. It was later validated on CNN features by Kiela et al. (2015).

The number of candidate English words is the number of entries in the bilingual dictionary after filtering out dictionary entries where the English word and foreign word are identical. In order to compare with Kiela et al. (2015), we evaluate the models' rankings using Mean Reciprocal Rank (MRR), top-1, top-5 and top-20 accuracy. We prefer the more interpretable top-k accuracy in our subsequent experiments. We choose to follow Wijaya et al. (2017) in standardizing to $k = 10$, and we report top-1 accuracy only when it is particularly informative.

4.2 Replication of Prior Work

We evaluate on the five languages–Dutch, French, German, Italian, and Spanish–which have been the focus of prior work. Table 2 shows the results reported by Kiela et al. (2015) on the BERGSMA500 dataset, along with results using our image crawl method (Section 3.2) on BERGSMA500's vocabulary.

On all five languages, our dataset performs better than that of Kiela et al. (2015). We attribute this to improvements in image search since they collected images. We additionally note that in the BERGSMA500 vocabularies, approximately 11% of the translation pairs are string-identical, like *film* $\leftrightarrow$ *film*. In all subsequent experiments, we remove trivial translation pairs like this.

We also evaluate the identical model on our full data set, which contains 8,500 words, covering all parts of speech and the full range of concreteness ratings. The top-1 accuracy of the model is 23% on our more realistic and challenging data set, versus 68% on the easier concrete nouns set.

4.3 High- and Low-resource Languages

To determine whether image-based translation is effective for low resource languages, we sample 12 high-resource languages (HIGHRES), and 20 low-resource languages (LOWRES). Table 3 reports the top-10 accuracy across all 32 languages.

For each language, we predict a translation for each foreign word in the language's CROWDTRANS dictionary. This comes to approximately 7,000 to 10,000 foreign words per language. We find that high-resource languages' image features are more predictive of translation than those of low-resource languages. Top-10 accuracy is 29% averaged across high-resource languages, but only 16% for low-resource languages. This may be due to the quality of image search in each language, and the number of websites in each language indexed by Google, as suggested by Table 1. The difficulty of the translation task is dependent on the size of the English vocabulary used, as distinguishing between $5,000$ English candidates as in Slovak is not as difficult as distinguishing between $10,000$ words as in Tamil.

4.4 Large Target Vocabulary

How does increasing the number of candidate translations affect accuracy? Prior work used an English vocabulary of 500 or 1,000 words, where the correct English translation is guaranteed to appear. This is unrealistic for many tasks such as machine translation, where the target language vocabulary is likely to be large. To evaluate a more realistic scenario, we take the union of the English vocabulary of every dictionary in CROWDTRANS, and run the same translation experiments as before. We call this large common vocabulary LARGEENG.

Confirming our intuition, experiments with LARGEENG give significantly lower top-10 accuracies across parts of speech, but still provide discriminative power. We find .181 average top-10 accuracy using LARGEENG, whereas on the same languages, average accuracy on the CROWDTRANS vocabularies was .260. The full results for these experiments are reported in Table 4.

5 Evaluation by Part-of-speech

Can images be used to translate words other than nouns? This section presents our methods for de-

dataset	BERGSMA500	BERGSMA500	all
	Kiela et al. (2015)	(ours)	(ours)
# words	500	500	8,500
MRR	0.658	0.704	0.277
Top 1	0.567	0.679	0.229
Top 5	0.692	0.763	0.326
Top 20	0.774	0.811	0.385

Table 2: Our results are consistently better than those reported by Kiela et al. (2015), averaged over Dutch, French, German, Italian, and Spanish on a similar set of 500 concrete nouns. The rightmost column shows the added challenge with our larger, more realistic dataset.

HIGHRES	All	VB	RB	JJ	NN	#
Spanish	.417	.144	.157	.329	.593	9.9k
French	.366	.104	.107	.315	.520	10.5k
Dutch	.365	.085	.064	.262	.511	10.5k
Italian	.323	.086	.085	.233	.487	8.9k
German	.307	.071	.098	.164	.463	10.1k
Swedish	.283	.048	.048	.146	.328	9.6k
Turkish	.263	.035	.143	.233	.346	10.2k
Romanian	.255	.029	.080	.150	.301	9.1k
Hungarian	.240	.030	.082	.193	.352	10.9k
Bulgarian	.236	.024	.106	.116	.372	8.6k
Arabic	.223	.036	.084	.149	.344	10.2k
Serbian	.218	.023	.111	.090	.315	8.3k
Average	*.291*	*.059*	*.097*	*.198*	*.411*	*9.7k*
LOWRES						
Thai	.367	.139	.143	.264	.440	5.6k
Indonesian	.306	.103	.041	.238	.404	10.3k
Vietnamese	.303	.079	.058	.106	.271	6.6k
Bosnian	.212	.035	.084	.103	.277	7.5k
Slovak	.195	.024	.042	.095	.259	6.5k
Ukrainian	.194	.024	.131	.070	.273	5.0k
Latvian	.194	.028	.058	.114	.266	7.1k
Hindi	.163	.024	.068	.057	.231	9.4k
Cebuano	.153	.014	.070	.098	.180	7.7k
Azerbaijani	.150	.016	.031	.113	.174	6.2k
Welsh	.138	.007	.025	.033	.062	7.6k
Albanian	.127	.013	.017	.080	.154	6.0k
Bengali	.120	.026	.050	.063	.173	12.5k
Tamil	.089	.006	.013	.030	.140	9.9k
Uzbek	.082	.093	.066	.114	.077	12.4k
Urdu	.073	.005	.017	.032	.108	11.1k
Telugu	.065	.002	.018	.010	.095	9.6k
Nepali	.059	.002	.039	.018	.089	11.6k
Gujarati	.039	.004	.016	.012	.056	12.0k
Average	*.159*	*.034*	*.052*	*.087*	*.196*	*8.7k*

Table 3: Top-10 accuracy on 12 high-resource languages and 20 low-resource languages. The parts of speech Noun, Adjective, Adverb, and Verb are referred to as NN, JJ, RB, VB, respectively. The "all" column reports accuracy on the entire dictionary. The "#" column reports the size of the English vocabulary used for each experiment.

Language	All	VB	RB	JJ	NN
Arabic	.149	.015	.053	.078	.219
Bengali	.066	.009	.042	.025	.084
Dutch	.265	.042	.039	.164	.350
French	.268	.051	.092	.196	.368
German	.220	.035	.040	.080	.321
Indonesian	.211	.050	.035	.156	.257
Italian	.233	.046	.028	.139	.350
Spanish	.320	.068	.076	.207	.449
Turkish	.171	.011	.086	.139	.201
Uzbek	.057	.121	.075	.104	.045
LARGEENG *Avg*	*.181*	*.041*	*.055*	*.118*	*.244*
SMALL *Avg*	*.260*	*.089*	*.078*	*.210*	*.392*

Table 4: Top-10 accuracy on the expanded English dictionary task. For each experiment, 263,102 English words were used as candidate translations for each foreign word. The SMALL average is given for reference, averaging the results from Table 3 across the same 10 languages.

termining part-of-speech for foreign words even in low-resource languages, and presents our image-based translation results across part-of-speech.

5.1 Assigning POS Labels

To show the performance of our translation method for each particular POS, we first assign a POS tag to each foreign word. Since we evaluate on high- and low-resource languages, many of which do not have POS taggers, we POS tag English words, and transfer the tag to their translations. We scraped the text on the web pages associated with the images of each English word, and collected the sentences that contained each query (English) word. We chose to tag words in sentential context, rather than simply collecting parts of speech from a dictionary, because many words have multiple senses, often with different parts of speech.

We assign universal POS tags (Petrov et al., 2012) using spaCy[2], giving each word its majority tag. We gathered part-of-speech tags for 42% of the English words in our translations. Of the remaining untagged English entries, 40% were multi-word expressions, and 18% were not found in the text of the web pages that we scraped.

When transferring POS tags to foreign words, we only considered foreign words where every English translation had the same POS. Across all 32 languages, on average, we found that, after filtering, 65% of foreign words were nouns, 14% were verbs, 14% were adjectives, 3% were adverbs, and 3% were other (i.e. they were labeled a different POS).

[2]https://spacy.io

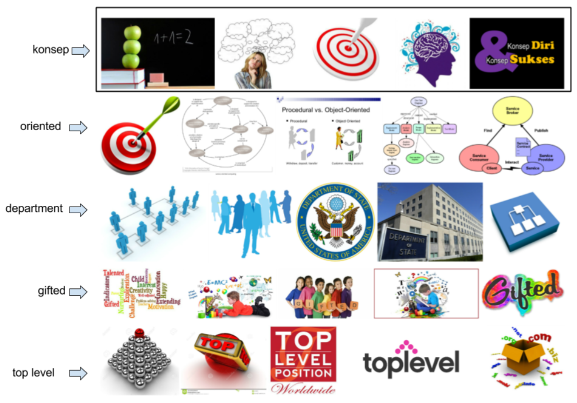


Figure 2: Shown here are five images for the abstract Indonesian word *konsep*, along with its top 4 ranked translations using CNN features. The actual translation, *concept*, was ranked 3,465.

5.2 Accuracy by Part-of-speech

As we see in the results in Table 3, the highest translation performance is obtained for nouns, which confirms the observation by Hartmann and Søgaard (2017). However, we see considerable signal in translating adjectives as well, with top-10 accuracies roughly half that of nouns. This trend extends to low-resource languages. We also see that translation quality is relatively poor for adverbs and verbs. There is higher variation in our performance on adverbs across languages, because there were relatively few adverbs (3% of all words.) From these results, it is clear that one can achieve higher accuracy by choosing to translate only nouns and adjectives.

Analysis by part-of-speech only indirectly addresses the question of when translation with images is useful. For example, Figure 2 shows that nouns like *concept* translate incorrectly because of a lack of consistent visual representation. However, verbs like *walk* may have concrete visual representation. Thus, one might perform better overall at translation on *concrete* words, regardless of part-of-speech.

6 Evaluation by Concreteness

Can we effectively predict the concreteness of words in a variety of languages? If so, can these predictions be used to determine when translation via images is helpful? In this section, we answer both of these questions in the affirmative.

6.1 Predicting Word Concreteness

Previous work has used *image dispersion* as a measure of word concreteness (Kiela et al., 2014). We introduce a novel supervised method for predicting word concreteness that more strongly correlates with human judgements of concreteness.

To train our model, we took Brysbaert et al. (2014)'s dataset, which provides human judgments for about 40k words, each with a 1-5 abstractness-to-concreteness score, and scraped 100 images from English Google Image Search for each word. We then trained a two-layer perceptron with one hidden layer of 32 units, to predict word concreteness. The inputs to the network were the element-wise mean and standard deviation (concatenated into a 8094-dimensional vector)of the CNN features for each of the images corresponding to a word. To better assess this image-only approach, we also experimented with using the distributional word embeddings of Salle et al. (2016) as input. We used these 300-dimensional vectors either seperately or concatentated with the image-based features. Our final network was trained with a cross-entropy loss, although an L2 loss performed nearly as well. We randomly selected 39,000 words as our training set. Results on the remaining held-out validation set are visualized in Figure 3.

Although the concatenated image and word embedding features performed the best, we do not expect to have high-quality word embeddings for words in low-resource languages. Therefore, for the evaluation in Section 6.2, we used the image-embeddings-only model to predict concreteness for every English and foreign word in our dataset.

6.2 Accuracy by Predicted Concreteness

It has already been shown that the images of more abstract words provide a weaker signal for translation (Kiela et al., 2015). Using our method for predicting concreteness, we determine which images sets are most concrete, and thereby estimate the likelihood that we will obtain a high quality translation.

Figure 4 shows the reduction in translation accuracy as increasingly abstract words are included in the set. The concreteness model can be used to establish recall thresholds. For the 25% of foreign words we predict to be most concrete, (25% recall,) AVGMAX achieves top-10 accuracy of 47.0% for high-resource languages and 32.8% for low-resource languages. At a 50% most-concrete recall treshold, top-10 translation accuracies are 25.0% and 37.8% for low- and high-resource languages respectively, compared to 18.6% and 29.3% at 100% recall.

7 Translation with Images and Text

Translation via image features performs worse than state-of-the-art distributional similarity-based methods. For example, Wijaya et al. (2017) demonstrate top-10 accuracies in range of above 85% on the VULIC1000 a 1,000-word dataset, whereas with only image features, Kiela et al. (2015) report top-10 accuracies below 60%. However, there may be utility in combining the two methods, as it is likely that visual and textual distributional representations are contributing different information, and fail in different cases.

We test this intuition by combining image scores with the current state-of-the-art system of Wijaya et al. (2017), which uses Bayesian Personalized Ranking (BPR). In their arXiv draft, Hartmann and Søgaard (2017) presented a negative result when directly combining image representations with distributional representations into a single system. Here, we present a positive result by formalizing the problem as a reranking task. Our intuition is that we hope to guide BPR, clearly the stronger system, with aid from image features and a predicted concreteness value, instead of joining them as equals and potentially washing out the stronger signal.

7.1 Reranking Model

For each foreign word w_f and each English word w_e, we have multiple scores for the pair $p_{f,e} = (w_f, w_e)$, used to rank w_e against all other $w_{e'} \in E$, where E is the English dictionary used in the experiment. Specifically, we have $\textsc{Txt}(p_{f,e})$ and $\textsc{Image}(p_{f,e})$ for all pairs. For each foreign word, we also have the concreteness score, $\textsc{Cnc}(w_f)$, predicted from its image set by the method described in Section 6.1.

We use a small bilingual dictionary, taking all pairs $p_{f,e}$ and labeling them $\{\pm 1\}$, with 1 denoting the words are translations. We construct training data out of the dictionary, treating each labeled pair as an independent observation. We then train a 2-layer perceptron (MLP), with 1 hidden layer of 4 units, to predict translations from the individual scores, minimizing the squared loss.[3]

$$\text{MLP}(p_{f,e}) = \\ \text{MLP}\big([\textsc{Txt}(p_{f,e}); \textsc{Image}(p_{f,e}); \textsc{Cnc}(w_f)]\big) = \{\pm 1\}$$

[3]We use DyNet (Neubig et al., 2017) for constructing and training our network with the Adam optimization method (Kingma and Ba, 2014).

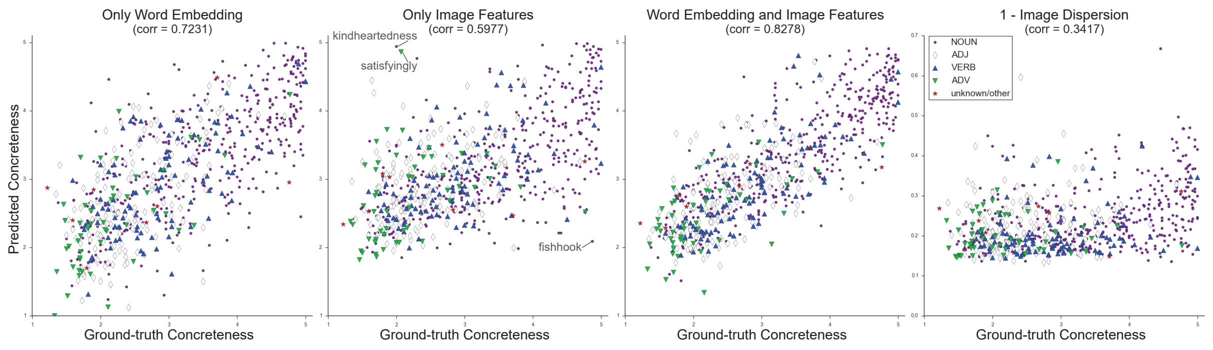


Figure 3: Plots visualizing the distribution of concreteness predictions on the validation set for our three trained models and for image dispersion. Spearman correlation coefficients are shown. For the model trained only on images, the three worst failure cases are annotated. False positives tend to occur when one concrete meaning of an abstract word dominates the search results (i.e. many photos of "satisfyingly" show food). False negatives often stem from related proper nouns or an overabundance of clipart, as is the case for "fishhook."

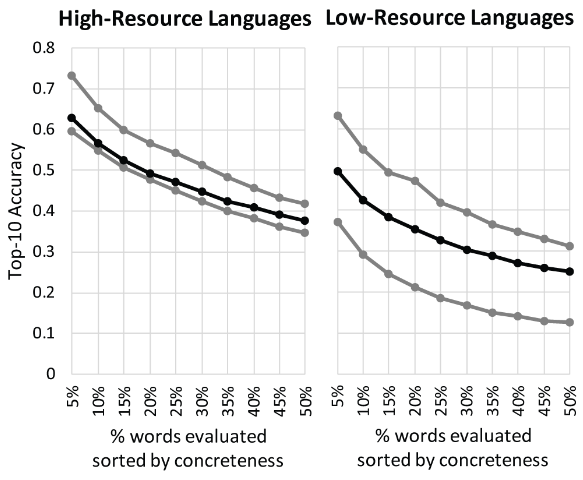


Figure 4: The black curve shows mean top-10 accuracy over the HIGHRES and LOWRES sets sorted by predicted concreteness. The gray curves show the 25th and 75th percentiles.

Once the model is trained, we fix each foreign word w_f, and score all pairs $(w_f, w_{e'})$ for all $e' \in E$, using the learned model MLP$(p_{f,e'})$. Using these scores, we sort E for each w_f.

7.2 Evaluation

We evaluate our text-based and image-based combination method by translating Bosnian, Dutch, French, Indonesian, Italian, and Spanish into English. For each language, we split our bilingual dictionary (of 8,673 entries, on average) into 2,000 entries for a testing set, 20% for training the text-based BPR system, 35% for training the reranking MLP, and the rest for a development set. We filtered out multi-word phrases, and translations where w_f and w_e are string identical.

We compare three models: TXT is Wijaya et al. (2017)'s text-based state-of-the-art model. TXT+IMG is our MLP-learned combination of the two features. TXT+IMG+CNC uses our predicted concreteness of the foreign word as well. We evaluate all models on varying percents of testing data sorted by predicted concreteness, as in Section 6.2. As shown in Figure 5, both image-augmented methods beat TXT across concreteness thresholds on the top-1 accuracy metric.

Results across the 6 languages are reported in Table 5. Confirming our intuition, images are useful at high concreteness, improving the SOA text-based method 3.21% at 10% recall. At 100% recall our method with images still improves the SOA by 1.3%. For example, the text-only system translates the Bosnian word *košarkaški* incorrectly as *football*, whereas the image+text system produces the correct *basketball*.

Further, gains are more pronounced for low-resource languages than for high-resource languages. Concreteness scores are useful for high-resource languages, for example Spanish, where TXT+IMG falls below TXT alone on more abstract words, but TXT+IMG+CNC remains an improvement. Finally, we note that the text-only system also performs better on concrete words than abstract words, indicating a general trend of ease in translating concrete words regardless of method.

8 Summary

We have introduced a large-scale multilingual image resource, and used it to conduct the most comprehensive study to date on using images to learn translations. Our Massively Multilingual Image Dataset will serve as a standard for future work in image-based translation due to its size and generality, covering 100 languages, hundreds of thousands

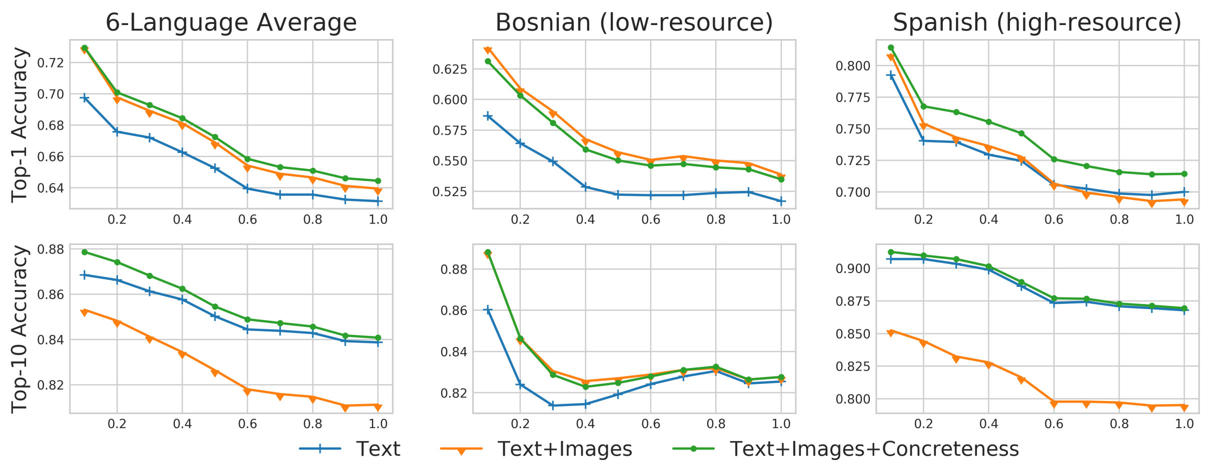


Figure 5: Reranking top-1 and top-10 accuracies of our image+text combination sytems compared to the text-only Bayesian Personalized Ranking system. The X-axis shows percent of foreign words evaluated on, sorted by decreasing predicted concreteness.

	Method	% words evaluated 10%	50%	100%
High-Res	TXT	.746	.696	.673
	TXT+IMG	**.771**	.708	.678
	TXT+IMG+Cnc	**.773**	**.714**	**.685**
Low-Res	TXT	.601	.565	.549
	TXT+IMG	**.646**	**.590**	**.562**
	TXT+IMG+Cnc	**.643**	**.589**	**.563**

Table 5: Top-1 accuracy results across high-resource (Dutch, French, Italian, Spanish) and low-resource (Bosnian, Indonesian) languages. Words evaluated on are again sorted by concreteness for the sake of analysis. The best result on each % of test data is bolded.

of words, and a broad range of parts of speech. Using this corpus, we demonstrated the substantial utility in supervised prediction of word concreteness when using image features, improving over the unsupervised state-of-the-art and finding that image-based translation is much more accurate for concrete words. Because of the text we collected with our corpus, we were also able to collect part-of-speech information and demonstrate that image features are useful in translating adjectives and nouns. Finally, we demonstrate a promising path forward, showing that incorporating images can improve a state-of-the-art text-based word translation system.

9 Dataset and Code

The MMID will be distributed both in raw form and for a subset of languages in memory compact featurized versions from `http://multilingual-images.org` along with code we used in our experiments. Additional details are given in our Supplemental Materials document, which also describes our manual image annotation setup, and gives numerous illustrative examples of our system's predictions.

Acknowledgements

We gratefully acknowledge Amazon for its support of this research through the Amazon Research Awards program and through AWS Research Credits.

This material is based in part on research sponsored by DARPA under grant number HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

References

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods*, 46(3):904–911.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. IEEE Conference on*, pages 248–255. IEEE.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. *CoRR*, abs/1605.00459.

Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. 2015. A survey of current datasets for vision and language research. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 207–213, Lisbon, Portugal. Association for Computational Linguistics.

Michael Friendly, Patricia E. Franklin, David Hoffman, and David C. Rubin. 1982. The Toronto word pool: Norms for imagery, concreteness, orthographic variables, and grammatical usage for 1,080 words. *Behavior Research Methods & Instrumentation*, 14(4):375–399.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 585–590. Association for Computational Linguistics.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 17th international Conference on Computational Linguistics*, volume 1, pages 414–420. Association for Computational Linguistics.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2839–2845, Copenhagen, Denmark. Association for Computational Linguistics.

K. J. Gilhooly and R. H. Logie. 1980. Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, volume 2008, pages 771–779.

Mareike Hartmann and Anders Søgaard. 2017. Limitations of cross-lingual learning from image search. *CoRR*, abs/1709.05914.

Julian Hitschler and Stefan Riezler. 2016. Multimodal pivots for image caption translation. *CoRR*, abs/1601.03916.

Ann Irvine and Chris Callison-Burch. 2017. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics*, 43(2):273–310.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 835–841, Baltimore, Maryland. Association for Computational Linguistics.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred convnet features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 148–158, Lisbon, Portugal. Association for Computational Linguistics.

Adam Kilgarriff. 2007. Googleology is bad science. *Computational linguistics*, 33(1):147–151.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 817–824. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition*, volume 9, pages 9–16. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Takashi Miyazaki and Nobuyuki Shimizu. 2016. Cross-lingual image caption generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1780–1790.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Allan Paivio, John C. Yuille, and Stephen A. Madigan. 1968. Concreteness, imagery, and meaningfulness values for 925 nouns. In *Journal of Experimental Psychology*, volume 76, pages 207–213. American Psychological Association.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The language demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics*, 2:79–92.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1115.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 519–526. Association for Computational Linguistics.

Alexandre Salle, Marco Idiart, and Aline Villavicencio. 2016. Matrix factorization using window sampling and negative sampling for improved word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 419–424. Association for Computational Linguistics.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 247–257, Berlin, Germany. Association for Computational Linguistics.

Ivan Vulic and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 106–116. ACL.

Derry Tanti Wijaya, Brendan Callahan, John Hewitt, Jie Gao, Xiao Ling, Marianna Apidianaki, and Chris Callison-Burch. 2017. Learning translations via matrix completion. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1453–1464, Copenhagen, Denmark. Association for Computational Linguistics.

On the Automatic Generation of Medical Imaging Reports

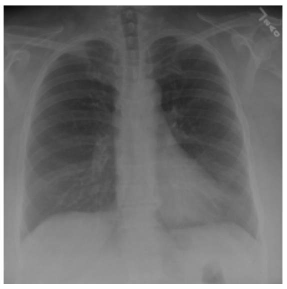
Baoyu Jing[†*] **Pengtao Xie**[†*] **Eric P. Xing**[†]
[†]Petuum Inc, USA
[*]School of Computer Science, Carnegie Mellon University, USA
{baoyu.jing, pengtao.xie, eric.xing}@petuum.com

Abstract

Medical imaging is widely used in clinical practice for diagnosis and treatment. Report-writing can be error-prone for unexperienced physicians, and time-consuming and tedious for experienced physicians. To address these issues, we study the automatic generation of medical imaging reports. This task presents several challenges. First, a complete report contains multiple heterogeneous forms of information, including *findings* and *tags*. Second, abnormal regions in medical images are difficult to identify. Third, the reports are typically long, containing multiple sentences. To cope with these challenges, we (1) build a multi-task learning framework which jointly performs the prediction of tags and the generation of paragraphs, (2) propose a co-attention mechanism to localize regions containing abnormalities and generate narrations for them, (3) develop a hierarchical LSTM model to generate long paragraphs. We demonstrate the effectiveness of the proposed methods on two publicly available datasets.

1 Introduction

Medical images, such as radiology and pathology images, are widely used in hospitals for the diagnosis and treatment of many diseases, such as pneumonia and pneumothorax. The reading and interpretation of medical images are usually conducted by specialized medical professionals. For example, radiology images are read by radiologists. They write textual reports (Figure 1) to narrate the findings regarding each area of the body examined in the imaging study, specifically whether each area was found to be normal, abnormal or potentially abnormal.



Impression: No acute cardiopulmonary abnormality.

Findings: There are no focal areas of consolidation. No suspicious pulmonary opacities. Heart size within normal limits. No pleural effusions. There is no evidence of pneumothorax. Degenerative changes of the thoracic spine.

MTI Tags: degenerative change

Figure 1: An exemplar chest x-ray report. In the *impression* section, the radiologist provides a diagnosis. The *findings* section lists the radiology observations regarding each area of the body examined in the imaging study. The *tags* section lists the keywords which represent the critical information in the findings. These keywords are identified using the Medical Text Indexer (MTI).

For less-experienced radiologists and pathologists, especially those working in the rural area where the quality of healthcare is relatively low, writing medical-imaging reports is demanding. For instance, to correctly read a chest x-ray image, the following skills are needed (Delrue et al., 2011): (1) thorough knowledge of the normal anatomy of the thorax, and the basic physiology of chest diseases; (2) skills of analyzing the radiograph through a fixed pattern; (3) ability of evaluating the evolution over time; (4) knowledge of clinical presentation and history; (5) knowledge of the correlation with other diagnostic results (laboratory results, electrocardiogram, and respiratory function tests).

For experienced radiologists and pathologists, writing imaging reports is tedious and time-consuming. In nations with large population such as China, a radiologist may need to read hundreds of radiology images per day. Typing the findings of each image into computer takes about 5-10 minutes, which occupies most of their working time. In sum, for both unexperienced and experienced medical professionals, writing imaging reports is unpleasant.

This motivates us to investigate whether it is possible to automatically generate medical image reports. Several challenges need to be addressed. First, a complete diagnostic report is comprised of multiple heterogeneous forms of information. As shown in Figure 1, the report for a chest x-ray contains *impression* which is a sentence, *findings* which are a paragraph, and *tags* which are a list of keywords. Generating this heterogeneous information in a unified framework is technically demanding. We address this problem by building a multi-task framework, which treats the prediction of tags as a multi-label classification task, and treats the generation of long descriptions as a text generation task.

Second, how to localize image-regions and attach the right description to them are challenging. We solve these problems by introducing a co-attention mechanism, which simultaneously attends to images and predicted tags and explores the synergistic effects of visual and semantic information.

Third, the descriptions in imaging reports are usually long, containing multiple sentences. Generating such long text is highly nontrivial. Rather than adopting a single-layer LSTM (Hochreiter and Schmidhuber, 1997), which is less capable of modeling long word sequences, we leverage the compositional nature of the report and adopt a hierarchical LSTM to produce long texts. Combined with the co-attention mechanism, the hierarchical LSTM first generates high-level topics, and then produces fine-grained descriptions according to the topics.

Overall, the main contributions of our work are:

- We propose a multi-task learning framework which can simultaneously predict the tags and generate the text descriptions.
- We introduce a co-attention mechanism for localizing sub-regions in the image and generating the corresponding descriptions.
- We build a hierarchical LSTM to generate long paragraphs.
- We perform extensive experiments to show the effectiveness of the proposed methods.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 introduces the method. Section 4 present the experimental results and Section 5 concludes the paper.

2 Related Works

Textual labeling of medical images There have been several works aiming at attaching "texts" to medical images. In their settings, the target "texts" are either fully-structured or semi-structured (e.g. tags, templates), rather than natural texts. Kisilev et al. (2015) build a pipeline to predict the attributes of medical images. Shin et al. (2016) adopt a CNN-RNN based framework to predict tags (e.g. locations, severities) of chest x-ray images. The work closest to ours is recently contributed by (Zhang et al., 2017), which aims at generating semi-structured pathology reports, whose contents are restricted to 5 predefined topics.

However, in the real-world, different physicians usually have different writing habits and different x-ray images will represent different abnormalities. Therefore, collecting semi-structured reports is less practical and thus it is important to build models to learn from natural reports. To the best of our knowledge, our work represents the first one that generates truly natural reports written by physicians, which are usually long and cover diverse topics.

Image captioning with deep learning Image captioning aims at automatically generating text descriptions for given images. Most recent image captioning models are based on a CNN-RNN framework (Vinyals et al., 2015; Fang et al., 2015; Karpathy and Fei-Fei, 2015; Xu et al., 2015; You et al., 2016; Krause et al., 2017).

Recently, attention mechanisms have been shown to be useful for image captioning (Xu et al., 2015; You et al., 2016). Xu et al. (2015) introduce a spatial-visual attention mechanism over image features extracted from intermediate layers of the CNN. You et al. (2016) propose a semantic attention mechanism over tags of given images. To better leverage both the visual features and semantic tags, we propose a co-attention mechanism for report generation.

Instead of only generating one-sentence caption

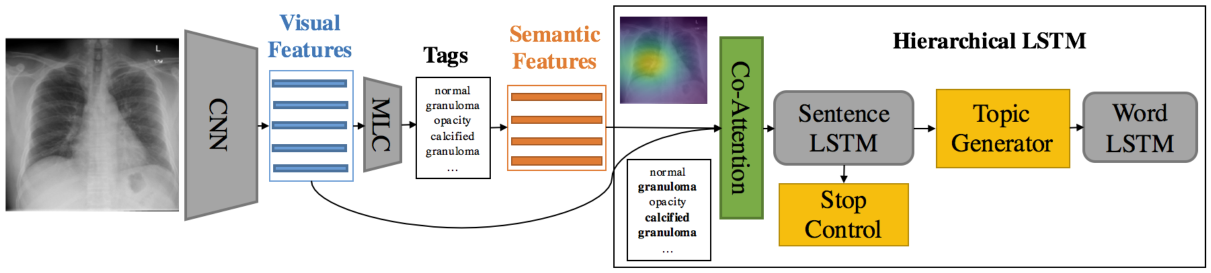


Figure 2: Illustration of the proposed model. MLC denotes a *multi-label classification* network. Semantic features are the word embeddings of the predicted tags. The boldfaced tags "calcified granuloma" and "granuloma" are attended by the co-attention network.

for images, Krause et al. (2017) and Liang et al. (2017) generate paragraph captions using a hierarchical LSTM. Our method also adopts a hierarchical LSTM for paragraph generation, but unlike Krause et al. (2017), we use a co-attention network to generate topics.

3 Methods

3.1 Overview

A complete diagnostic report for a medical image is comprised of both text descriptions (long paragraphs) and lists of tags, as shown in Figure 1. We propose a *multi-task hierarchical model with co-attention* for automatically predicting keywords and generating long paragraphs. Given an image which is divided into regions, we use a CNN to learn visual features for these patches. Then these visual features are fed into a *multi-label classification* (MLC) network to predict the relevant tags. In the tag vocabulary, each tag is represented by a word-embedding vector. Given the predicted tags for a specific image, their word-embedding vectors serve as the semantic features of this image. Then the visual features and semantic features are fed into a *co-attention* model to generate a context vector that simultaneously captures the visual and semantic information of this image. As of now, the encoding process is completed.

Next, starting from the context vector, the decoding process generates the text descriptions. The description of a medical image usually contains multiple sentences, and each sentence focuses on one specific topic. Our model leverages this compositional structure to generate reports in a hierarchical way: it first generates a sequence of high-level topic vectors representing sentences, then generates a sentence from each topic vector. Specifically, the context vector is inputted into a *sentence LSTM*, which unrolls for a few steps and produces a topic vector at each step. A topic vector represents the semantics of a sentence to be generated. Given a topic vector, the *word LSTM* takes it as input and generates a sequence of words to form a sentence. The termination of the unrolling process is controlled by the sentence LSTM.

3.2 Tag Prediction

The first task of our model is predicting the tags of the given image. We treat the tag prediction task as a multi-label classification task. Specifically, given an image I, we first extract its features $\{\mathbf{v}_n\}_{n=1}^{N} \in \mathbb{R}^D$ from an intermediate layer of a CNN, and then feed $\{\mathbf{v}_n\}_{n=1}^{N}$ into a *multi-label classification* (MLC) network to generate a distribution over all of the L tags:

$$\mathbf{p}_{l,pred}(\mathbf{l}_i = 1 | \{\mathbf{v}_n\}_{n=1}^{N}) \propto \exp(\text{MLC}_i(\{\mathbf{v}_n\}_{n=1}^{N})) \quad (1)$$

where $\mathbf{l} \in \mathbb{R}^L$ is a tag vector, $\mathbf{l}_i = 1/0$ denote the presence and absence of the i-th tag respectively, and MLC_i means the i-th output of the MLC network.

For simplicity, we extract visual features from the last convolutional layer of the VGG-19 model (Simonyan and Zisserman, 2014) and use the last two fully connected layers of VGG-19 for MLC.

Finally, the embeddings of the M most likely tags $\{\mathbf{a}_m\}_{m=1}^{M} \in \mathbb{R}^E$ are used as semantic features for topic generation.

3.3 Co-Attention

Previous works have shown that visual attention alone can perform fairly well for localizing objects (Ba et al., 2015) and aiding caption generation (Xu et al., 2015). However, visual attention does not provide sufficient high level semantic information. For example, only looking at the right lower region of the chest x-ray image (Figure 1) without accounting for other areas, we might not be able to recognize what we are looking at, not to even mention detecting the abnormalities. In contrast, the tags can always provide the needed high level information. To this end, we propose a co-attention mechanism which can simultaneously attend to visual and semantic modalities.

In the sentence LSTM at time step s, the joint context vector $\mathbf{ctx}^{(s)} \in \mathbb{R}^C$ is generated by a co-attention network $f_{co_{att}}(\{\mathbf{v}_n\}_{n=1}^N, \{\mathbf{a}_m\}_{m=1}^M, \mathbf{h}_{sent}^{(s-1)})$, where $\mathbf{h}_{sent}^{(s-1)} \in \mathbb{R}^H$ is the sentence LSTM hidden state at time step $s-1$. The co-attention network $f_{co_{att}}$ uses a single layer feed-forward network to compute the soft visual attentions and soft semantic attentions over input image features and tags:

$$\alpha_{\mathbf{v},n} \propto \exp(\mathbf{W}_{\mathbf{v}_{att}} \tanh(\mathbf{W}_{\mathbf{v}}\mathbf{v}_n + \mathbf{W}_{\mathbf{v},\mathbf{h}}\mathbf{h}_{sent}^{(s-1)})) \quad (2)$$

$$\alpha_{\mathbf{a},m} \propto \exp(\mathbf{W}_{\mathbf{a}_{att}} \tanh(\mathbf{W}_{\mathbf{a}}\mathbf{a}_m + \mathbf{W}_{\mathbf{a},\mathbf{h}}\mathbf{h}_{sent}^{(s-1)})) \quad (3)$$

where $\mathbf{W}_{\mathbf{v}}$, $\mathbf{W}_{\mathbf{v},\mathbf{h}}$, and $\mathbf{W}_{\mathbf{v}_{att}}$ are parameter matrices of the visual attention network. $\mathbf{W}_{\mathbf{a}}$, $\mathbf{W}_{\mathbf{a},\mathbf{h}}$, and $\mathbf{W}_{\mathbf{a}_{att}}$ are parameter matrices of the semantic attention network.

The visual and semantic context vectors are computed as:

$$\mathbf{v}_{att}^{(s)} = \sum_{n=1}^{N} \alpha_{\mathbf{v},n}\mathbf{v}_n, \quad \mathbf{a}_{att}^{(s)} = \sum_{m=1}^{M} \alpha_{\mathbf{a},m}\mathbf{a}_m.$$

There are many ways to combine the visual and semantic context vectors such as concatenation and element-wise operations. In this paper, we first concatenate these two vectors as $[\mathbf{v}_{att}^{(s)}; \mathbf{a}_{att}^{(s)}]$, and then use a fully connected layer $\mathbf{W}_{fc}$ to obtain a joint context vector:

$$\mathbf{ctx}^{(s)} = \mathbf{W}_{fc}[\mathbf{v}_{att}^{(s)}; \mathbf{a}_{att}^{(s)}]. \quad (4)$$

3.4 Sentence LSTM

The sentence LSTM is a single-layer LSTM that takes the joint context vector $\mathbf{ctx} \in \mathbb{R}^C$ as its input, and generates topic vector $\mathbf{t} \in \mathbb{R}^K$ for word LSTM through topic generator and determines whether to continue or stop generating captions by a stop control component.

Topic generator We use a deep output layer (Pascanu et al., 2014) to strengthen the context information in topic vector $\mathbf{t}^{(s)}$, by combining the hidden state $\mathbf{h}_{sent}^{(s)}$ and the joint context vector $\mathbf{ctx}^{(s)}$ of the current step:

$$\mathbf{t}^{(s)} = \tanh(\mathbf{W}_{\mathbf{t},\mathbf{h}_{sent}}\mathbf{h}_{sent}^{(s)} + \mathbf{W}_{\mathbf{t},\mathbf{ctx}}\mathbf{ctx}^{(s)}) \quad (5)$$

where $\mathbf{W}_{\mathbf{t},\mathbf{h}_{sent}}$ and $\mathbf{W}_{\mathbf{t},\mathbf{ctx}}$ are weight parameters.

Stop control We also apply a deep output layer to control the continuation of the sentence LSTM. The layer takes the previous and current hidden state $\mathbf{h}_{sent}^{(s-1)}$, $\mathbf{h}_{sent}^{(s)}$ as input and produces a distribution over {*STOP*=1, *CONTINUE*=0}:

$$p(STOP|\mathbf{h}_{sent}^{(s-1)}, \mathbf{h}_{sent}^{(s)}) \propto \exp\{\mathbf{W}_{stop} \tanh(\mathbf{W}_{stop,s-1}\mathbf{h}_{sent}^{(s-1)} + \mathbf{W}_{stop,s}\mathbf{h}_{sent}^{(s)})\} \quad (6)$$

where $\mathbf{W}_{stop}$, $\mathbf{W}_{stop,s-1}$ and $\mathbf{W}_{stop,s}$ are parameter matrices. If $p(STOP|\mathbf{h}_{sent}^{(s-1)}, \mathbf{h}_{sent}^{(s)})$ is greater than a predefined threshold (e.g. 0.5), then the sentence LSTM will stop producing new topic vectors and the word LSTM will also stop producing words.

3.5 Word LSTM

The words of each sentence are generated by a word LSTM. Similar to (Krause et al., 2017), the topic vector $\mathbf{t}$ produced by the sentence LSTM and the special *START* token are used as the first and second input of the word LSTM, and the subsequent inputs are the word sequence.

The hidden state $\mathbf{h}_{word} \in \mathbb{R}^H$ of the word LSTM is directly used to predict the distribution over words:

$$p(word|\mathbf{h}_{word}) \propto \exp(\mathbf{W}_{out}\mathbf{h}_{word}) \quad (7)$$

where $\mathbf{W}_{out}$ is the parameter matrix. After each word-LSTM has generated its word sequences, the final report is simply the concatenation of all the generated sequences.

3.6 Parameter Learning

Each training example is a tuple (I, $\mathbf{l}$, $\mathbf{w}$) where I is an image, $\mathbf{l}$ denotes the ground-truth tag vector, and $\mathbf{w}$ is the diagnostic paragraph, which is comprised of S sentences and each sentence consists of T_s words.

Given a training example (I, $\mathbf{l}$, $\mathbf{w}$), our model first performs multi-label classification on I and produces a distribution $\mathbf{p}_{\mathbf{l},pred}$ over all tags. Note that $\mathbf{l}$ is a binary vector which encodes the presence and absence of tags. We can obtain the ground-truth tag distribution by normalizing $\mathbf{l}$: $\mathbf{p}_{\mathbf{l}} = \mathbf{l}/||\mathbf{l}||_1$. The training loss of this step is a cross-entropy loss ℓ_{tag} between $\mathbf{p}_{\mathbf{l}}$ and $\mathbf{p}_{\mathbf{l},pred}$.

Next, the sentence LSTM is unrolled for S steps to produce topic vectors and also distributions over $\{STOP, CONTINUE\}$: p^s_{stop}. Finally, the S topic vectors are fed into the word LSTM to generate words $\mathbf{w}_{s,t}$. The training loss of caption generation is the combination of two cross-entropy losses: ℓ_{sent} over stop distributions p^s_{stop} and ℓ_{word} over word distributions $p_{s,t}$. Combining the pieces together, we obtain the overall training loss:

$$\begin{aligned} \ell(I, \mathbf{l}, \mathbf{w}) = & \lambda_{tag}\ell_{tag} \\ & + \lambda_{sent} \sum_{s=1}^{S} \ell_{sent}(p^s_{stop}, I\{s = S\}) \\ & + \lambda_{word} \sum_{s=1}^{S} \sum_{t=1}^{T_s} \ell_{word}(p_{s,t}, w_{s,t}) \end{aligned} \quad (8)$$

In addition to the above training loss, there is also a regularization term for visual and semantic attentions. Similar to (Xu et al., 2015), let $\boldsymbol{\alpha} \in \mathbb{R}^{N\times S}$ and $\boldsymbol{\beta} \in \mathbb{R}^{M\times S}$ be the matrices of visual and semantic attentions respectively, then the regularization loss over $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is:

$$\ell_{reg} = \lambda_{reg}[\sum_{n}^{N}(1-\sum_{s}^{S}\alpha_{n,s})^2 + \sum_{m}^{M}(1-\sum_{s}^{S}\beta_{m,s})^2] \quad (9)$$

Such regularization encourages the model to pay equal attention over different image regions and different tags.

4 Experiments

In this section, we evaluate the proposed model with extensive quantitative and qualitative experiments.

4.1 Datasets

We used two publicly available medical image datasets to evaluate our proposed model.

IU X-Ray The Indiana University Chest X-Ray Collection (IU X-Ray) (Demner-Fushman et al., 2015) is a set of chest x-ray images paired with their corresponding diagnostic reports. The dataset contains 7,470 pairs of images and reports. Each report consists of the following sections: *impression*, *findings*, *tags*[1], *comparison*, and *indication*. In this paper, we treat the contents in *impression* and *findings* as the target captions[2] to be generated and the Medical Text Indexer (MTI) annotated tags as the target tags to be predicted (Figure 1 provides an example).

We preprocessed the data by converting all tokens to lowercases, removing all of non-alpha tokens, which resulting in 572 unique tags and 1915 unique words. On average, each image is associated with 2.2 tags, 5.7 sentences, and each sentence contains 6.5 words. Besides, we find that top 1,000 words cover 99.0% word occurrences in the dataset, therefore we only included top 1,000 words in the dictionary. Finally, we randomly selected 500 images for validation and 500 images for testing.

PEIR Gross The Pathology Education Informational Resource (PEIR) digital library[3] is a public medical image library for medical education. We collected the images together with their descriptions in the Gross sub-collection, resulting in the PEIR Gross dataset that contains 7,442 image-caption pairs from 21 different sub-categories. Different from the IU X-Ray dataset, each caption in PEIR Gross contains only one sentence. We used this dataset to evaluate our model's ability of generating single-sentence report.

For PEIR Gross, we applied the same preprocessing as IU X-Ray, which yields 4,452 unique words. On average, each image contains 12.0 words. Besides, for each caption, we selected 5 words with the highest tf-idf scores as tags.

4.2 Implementation Details

We used the full VGG-19 model (Simonyan and Zisserman, 2014) for tag prediction. As for the training loss of the multi-label classification (MLC) task, since the number of tags for semantic attention is fixed as 10, we treat MLC as a multi-label retrieval task and adopt a softmax cross-entropy loss (a multi-label ranking loss), similar to (Gong et al., 2013; Guillaumin et al., 2009).

[1]There are two types of tags: manually generated (MeSH) and Medical Text Indexer (MTI) generated.

[2]The *impression* and *findings* sections are concatenated together as a long paragraph, since *impression* can be viewed as a conclusion or topic sentence of the report.

[3]PEIR is ©University of Alabama at Birmingham, Department of Pathology. (http://peir.path.uab.edu/library/)

Dataset	Methods	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE	CIDER
IU X-Ray	CNN-RNN (Vinyals et al., 2015)	0.316	0.211	0.140	0.095	0.159	0.267	0.111
	LRCN (Donahue et al., 2015)	0.369	0.229	0.149	0.099	0.155	0.278	0.190
	Soft ATT (Xu et al., 2015)	0.399	0.251	0.168	0.118	0.167	0.323	0.302
	ATT-RK (You et al., 2016)	0.369	0.226	0.151	0.108	0.171	0.323	0.155
	Ours-no-Attention	0.505	0.383	0.290	0.224	0.200	0.420	0.259
	Ours-Semantic-only	0.504	0.371	0.291	0.230	0.207	0.418	0.286
	Ours-Visual-only	0.507	0.373	0.297	0.238	0.211	0.426	0.300
	Ours-CoAttention	**0.517**	**0.386**	**0.306**	**0.247**	**0.217**	**0.447**	**0.327**
PEIR Gross	CNN-RNN (Vinyals et al., 2015)	0.247	0.178	0.134	0.092	0.129	0.247	0.205
	LRCN (Donahue et al., 2015)	0.261	0.184	0.136	0.088	0.135	0.254	0.203
	Soft ATT (Xu et al., 2015)	0.283	0.212	0.163	0.113	0.147	0.271	0.276
	ATT-RK (You et al., 2016)	0.274	0.201	0.154	0.104	0.141	0.264	0.279
	Ours-No-Attention	0.248	0.180	0.133	0.093	0.131	0.242	0.206
	Ours-Semantic-only	0.263	0.191	0.145	0.098	0.138	0.261	0.274
	Ours-Visual-only	0.284	0.209	0.156	0.105	**0.149**	0.274	0.280
	Ours-CoAttention	**0.300**	**0.218**	**0.165**	**0.113**	**0.149**	**0.279**	**0.329**

Table 1: Main results for paragraph generation on the IU X-Ray dataset (upper part), and single sentence generation on the PEIR Gross dataset (lower part). BLUE-n denotes the BLEU score that uses up to n-grams.

In paragraph generation, we set the dimensions of all hidden states and word embeddings as 512. For words and tags, different embedding matrices were used since a tag might contain multiple words. We utilized the embeddings of the 10 most likely tags as the semantic feature vectors $\{\mathbf{a}_m\}_{m=1}^{M=10}$. We extracted the visual features from the last convolutional layer of the VGG-19 network, which yields a $14 \times 14 \times 512$ feature map.

We used the Adam (Kingma and Ba, 2014) optimizer for parameter learning. The learning rates for the CNN (VGG-19) and the hierarchical LSTM were 1e-5 and 5e-4 respectively. The weights (λ_{tag}, λ_{sent}, λ_{word} and λ_{reg}) of different losses were set to 1.0. The threshold for stop control was 0.5. Early stopping was used to prevent over-fitting.

4.3 Baselines

We compared our method with several state-of-the-art image captioning models: CNN-RNN (Vinyals et al., 2015), LRCN (Donahue et al., 2015), Soft ATT (Xu et al., 2015), and ATT-RK (You et al., 2016). We re-implemented all of these models and adopt VGG-19 (Simonyan and Zisserman, 2014) as the CNN encoder. Considering these models are built for single sentence captions and to better show the effectiveness of the hierarchical LSTM and the attention mechanism for paragraph generation, we also implemented a hierarchical model without any attention: Ours-no-Attention. The input of Ours-no-Attention is the overall image feature of VGG-19, which has a dimension of 4096. Ours-no-Attention can be viewed as a CNN-RNN (Vinyals et al., 2015) equipped with a hierarchical LSTM decoder. To further show the effectiveness of the proposed co-attention mechanism, we also implemented two ablated versions of our model: Ours-Semantic-only and Ours-Visual-only, which takes solely the semantic attention or visual attention context vector to produce topic vectors.

4.4 Quantitative Results

We report the paragraph generation (upper part of Table 1) and one sentence generation (lower part of Table 1) results using the standard image captioning evaluation tool [4] which provides evaluation on the following metrics: BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004), and CIDER (Vedantam et al., 2015).

For paragraph generation, as shown in the upper part of Table 1, it is clear that models with a single LSTM decoder perform much worse than those with a hierarchical LSTM decoder. Note that the only difference between Ours-no-Attention and CNN-RNN (Vinyals et al., 2015) is that Ours-no-Attention adopts a hierarchical LSTM decoder while CNN-RNN (Vinyals et al., 2015) adopts a single-layer LSTM. The comparison between these two models directly demonstrates the effectiveness of the hierarchical LSTM. This result is not surprising since it is well-known that a single-layer LSTM cannot effectively model long sequences (Liu et al., 2015; Martin and Cundy, 2018). Additionally, employing semantic attention alone (Ours-Semantic-only) or visual attention alone (Ours-Visual-only) to generate topic vectors does not seem to help caption generation a lot. The potential reason might be that visual at-

[4]https://github.com/tylin/coco-caption

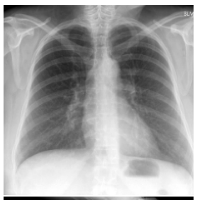
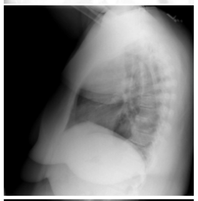
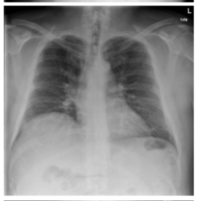
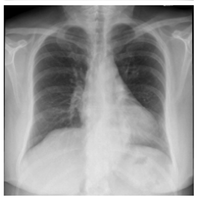
	Ground Truth	Ours-CoAttention	Ours-no-Attention	Soft Attention
	No active disease. The heart and lungs have in the interval. Both lungs are clear and expanded. Heart and mediastinum normal.	No active disease. The heart and lungs have in the interval. Lungs are clear and expanded. Cardiomediastinal silhouette is within normal limits. No pleural effusion or pneumothorax is seen. No pleural effusion. No cavitary or pneumothorax.	The lungs are clear bilaterally. The are grossly normal. No focal lung consolidation. No acute bony abnormality. cm nodule within the right lower lobe on the lateral view. No pneumothorax or pleural effusion. No acute bony abnormality. The heart is not enlarged. The lungs are clear. No acute bony abnormality .	No acute cardiopulmonary abnormality. The lungs are clear bilaterally. Specifically no evidence of focal airspace consolidation pleural effusion or pneumothorax. Cardio mediastinal silhouette is unremarkable. Visualized osseous structures of the thorax are without acute abnormality.
	No evidence of active disease. The lungs are clear. There is no focal airspace consolidation. No pleural effusion or pneumothorax. Heart size and mediastinal contour are within normal limits. There are multilevel degenerative changes of the spine.	No acute cardiopulmonary findings. Heart size is not enlarged. No focal airspace consolidation suspicious pulmonary opacity large pleural effusion or pneumothorax. No focal areas of consolidation. Degenerative changes of the spine. This is moderate exam of the hydropneumothorax. Lungs are clear. There is no focal airspace consolidation pleural effusion or pneumothorax.	The lungs are clear bilaterally. The are grossly normal. No pleural effusion. The heart is normal in size and contour. The lungs are clear. There are no acute bony findings.	No acute cardiopulmonary abnormality. The lungs are clear bilaterally. There is no pleural effusion or pneumothorax. The heart and mediastinum are normal. There is no focal air space opacity to suggest a pneumonia.
	No acute cardiopulmonary abnormality. Normal heart size mediastinal contours. Eventration of the right hemidiaphragm. No focal airspace consolidation. No pleural effusion or pneumothorax.	No acute cardiopulmonary abnormality. Stable appearance of the thoracic aorta. The right lateral lower lobe is noted in the right lower right midlung. No large pleural effusion or focal airspace disease. Mild interstitial opacities. Atherosclerotic calcifications bony structures bilaterally. There is no pleural effusion or pneumothorax developed in the right lower lobe.	The lungs are clear bilaterally. The are grossly normal. No acute bony abnormality. The lungs are otherwise clear. No acute osseous abnormality. No acute osseous abnormality. The heart and mediastinum are normal. There is no focal air space opacity.	No acute cardiopulmonary abnormality. The lungs are clear bilaterally. There is no focal airspace consolidation. No pleural effusion or pneumothorax. Heart size and pulmonary vascularity appear within normal limits.
	No acute cardiopulmonary abnormality. Heart size appears within normal limits . Pulmonary vasculature appears within normal limits. Overlying the middle cardiac silhouette representing a hiatal hernia. No focal consolidation pleural effusion or pneumothorax. No acute bony abnormality.	No active disease. The heart and lungs have in the interval. Nipple and lateral lucency in the lungs suggestive of focal airspace disease. The lungs are hyperexpanded consistent with emphysema in the left lower lobe. This is most at the upper lobes. This may indicate hypoventilated irregularities or effusions. The lungs are otherwise grossly clear. Resolution of by normal pleural effusion.	No acute cardiopulmonary abnormality. The lungs are clear bilaterally. The are grossly normal. No focal airspace consolidation. No pneumothorax or pleural effusion. Heart size and pulmonary vascularity within normal limits. There is no pneumothorax or pleural effusion.	No acute cardiopulmonary abnormality. There is no focal airspace consolidation. No pneumothorax or pleural effusion. No acute bony abnormality. Heart size is normal .

Figure 3: Illustration of paragraph generated by Ours-CoAttention, Ours-no-Attention, and Soft Attention models. The underlined sentences are the descriptions of detected abnormalities. The second image is a lateral x-ray image. Top two images are positive results, the third one is a partial failure case and the bottom one is failure case. These images are from test dataset.

tention can only capture the visual information of sub-regions of the image and is unable to correctly capture the semantics of the entire image. Semantic attention is inadequate of localizing small abnormal image-regions. Finally, our full model (Ours-CoAttention) achieves the best results on all of the evaluation metrics, which demonstrates the effectiveness of the proposed co-attention mechanism.

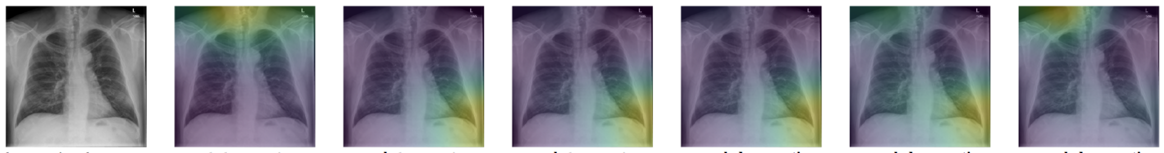
For the single-sentence generation results (shown in the lower part of Table 1), the ablated versions of our model (Ours-Semantic-only and Ours-Visual-only) achieve competitive scores compared with the state-of-the-art methods. Our full model (Ours-CoAttention) outperforms all of the baseline, which indicates the effectiveness of the proposed co-attention mechanism.

4.5 Qualitative Results

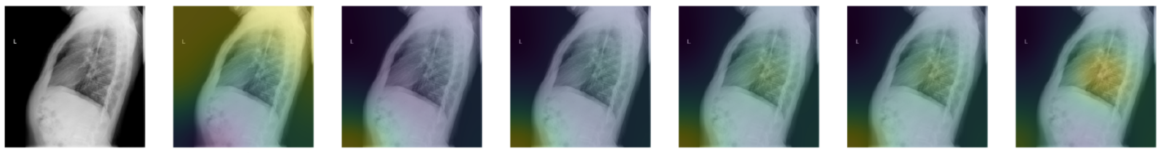
4.5.1 Paragraph Generation

An illustration of paragraph generation by three models (Ours-CoAttention, Ours-no-Attention and Soft Attention models) is shown in Figure 3. We can find that different sentences have different topics. The first sentence is usually a high level description of the image, while each of the following sentences is associated with one area of the image (e.g. "lung", "heart"). Soft Attention and Ours-no-Attention models detect only a few abnormalities of the images and the detected abnormalities are incorrect. In contrast, Ours-CoAttention model is able to correctly describe many true abnormalities (as shown in top three images). This comparison demonstrates that co-attention is better at capturing abnormalities.

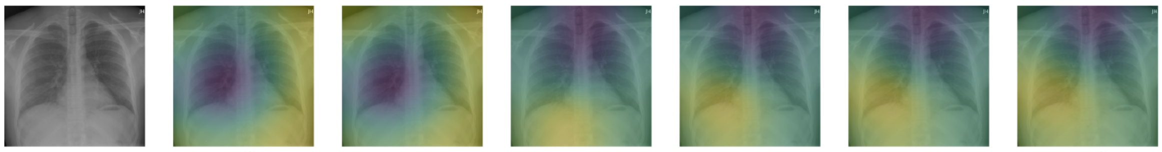
For the third image, Ours-CoAttention model successfully detects the area ("right lower lobe") which is abnormal ("eventration"), however, it fails to precisely describe this abnormality. In addition, the model also finds abnormalities about "interstitial opacities" and "atheroscalerotic calcification", which are not considered as true abnormality by human experts. The potential reason for this mis-description might be that this x-ray image is darker (compared with the above images), and our model might be very sensitive to this change.



degenerative change; obstruction	normal; degenerative change; nodule; calcified granuloma; hyper expansion; granulomatous disease; granuloma; **pneumonia; scarring; sternotomy**	**normal**; degenerative change; **nodule;** calcified granuloma; hyper expansion; granulomatous disease; **granuloma;** pneumonia; scarring; sternotomy	**normal**; degenerative change; **nodule;** calcified granuloma; hyper expansion; granulomatous disease; **granuloma;** pneumonia; scarring; sternotomy	**normal**; **degenerative change**; **nodule**; calcified granuloma; hyper expansion; granulomatous disease; granuloma; pneumonia; scarring; sternotomy	**normal**; **degenerative change**; **nodule**; calcified granuloma; hyper expansion; granulomatous disease; granuloma; pneumonia; scarring; sternotomy	**normal**; **degenerative change**; **nodule**; calcified granuloma; hyper expansion; granulomatous disease; granuloma; pneumonia; scarring; sternotomy
	No acute intrathoracic abnormality.	No bony abnormality.	The cardio mediastinal silhouette is within normal limits for appearance.	No focal areas of pulmonary consolidation.	Breast motion.	There is an age indeterminate deformity of a mid-thoracic vertebral body.
No acute cardiopulmonary finding. The heart size and cardiopulmonary silhouette is normal. There is no focal airspace opacity pleural effusion or pneumothorax. The obstruction are intact with mild degenerative change in the thoracic spine.						



calcified granuloma; granulomas	normal; **cardiomegaly; nodule;** degenerative change; granulomatous disease; opacity; **calcified granuloma;** atelectasis; lymph nodes; hiatal hernia	**normal;** cardiomegaly; nodule; **degenerative change; granulomatous disease;** opacity; calcified granuloma; atelectasis; lymph nodes; hiatal hernia	**normal;** cardiomegaly; nodule; **degenerative change;** granulomatous disease; opacity; calcified granuloma; atelectasis; **lymph nodes;** hiatal hernia	**normal;** cardiomegaly; nodule; **degenerative change; granulomatous disease;** opacity; calcified granuloma; atelectasis; lymph nodes; hiatal hernia	**normal;** cardiomegaly; nodule; **degenerative change;** granulomatous disease; opacity; **calcified granuloma;** atelectasis; lymph nodes; hiatal hernia	normal; **cardiomegaly; nodule;** degenerative change; granulomatous disease; opacity; **calcified granuloma;** atelectasis; lymph nodes; **hiatal hernia**
	Clear lungs.	Lungs are clear .	Left lower lung volumes is clear.	The heart is normal size.	A there is no focal airspace disease.	There is mild blunting of cost phrenic.
No acute cardiopulmonary abnormality. The lungs are clear. There are calcified granulomas. Heart size is normal. No pneumothorax.						



normal	normal; **calcified granuloma; granulomatous disease; granuloma;** scarring; opacity; degenerative change; sternotomy; thoracic aorta; nodule	**normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change;** sternotomy; thoracic aorta; **nodule**	**normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change;** sternotomy; thoracic aorta; **nodule**	**normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change; sternotomy;** thoracic aorta; nodule	**normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change; sternotomy;** thoracic aorta; nodule	normal; **calcified granuloma; granulomatous disease;** granuloma; scarring; opacity; degenerative change; **sternotomy;** thoracic aorta; nodule
	Right upper lobe infiltrate.	Lungs are clear .	Stable heart size and aortic contours.	No acute displaced rib fractures.	No focal airspace opacities or consolidation.	No visualized of pneumothorax.
No acute cardiopulmonary abnormality identified. The examination consists of frontal and lateral radiographs of the chest. The cardio mediastinal contours are within normal limits. Pulmonary vascularity is within normal limits. No focal consolidation pleural effusion or pneumothorax identified. The visualized osseous structures and upper abdomen are unremarkable.						

Figure 4: Visualization of co-attention for three examples. Each example is comprised of four things: (1) image and visual attentions; (2) ground truth tags and semantic attention on predicted tags; (3) generated descriptions; (4) ground truth descriptions. For the semantic attention, three tags with highest attention scores are highlighted. The underlined tags are those appearing in the ground truth.

The image at the bottom is a failure case of Ours-CoAttention. However, even though the model makes the wrong judgment about the major abnormalities in the image, it does find some unusual regions: "lateral lucency" and "left lower lobe".

To further understand models' ability of detecting abnormalities, we present the portion of sentences which describe the normalities and abnormalities in Table 2. We consider sentences which contain "no", "normal", "clear", "stable" as sentences describing normalities. It is clear that Ours-CoAttention best approximates the ground truth distribution over normality and abnormality.

Method	Normality	Abnormality	Total
Soft Attention	0.510	0.490	1.0
Ours-no-Attention	0.753	0.247	1.0
Ours-CoAttention	0.471	0.529	1.0
Ground Truth	0.385	0.615	1.0

Table 2: Portion of sentences which describe the normalities and abnormalities in the image.

4.5.2 Co-Attention Learning

Figure 4 presents visualizations of co-attention. The first property shown by Figure 4 is that the sentence LSTM can generate different topics at different time steps since the model focuses on different image regions and tags for different sentences. The next finding is that visual attention can guide our model to concentrate on relevant regions of the image. For example, the third sentence of the first example is about "cardio", and the visual attention concentrates on regions near the heart. Similar behavior can also be found for semantic attention: for the last sentence in the first example, our model correctly concentrates on "degenerative change" which is the topic of the sentence. Finally, the first sentence of the last example presents a mis-description caused by incorrect semantic attention over tags. Such incorrect attention can be reduced by building a better tag prediction module.

5 Conclusion

In this paper, we study how to automatically generate textual reports for medical images, with the goal to help medical professionals produce reports more accurately and efficiently. Our proposed methods address three major challenges: (1) how to generate multiple heterogeneous forms of information within a unified framework, (2) how to localize abnormal regions and produce accurate descriptions for them, (3) how to generate long texts that contain multiple sentences or even paragraphs. To cope with these challenges, we propose a multi-task learning framework which jointly predicts tags and generates descriptions. We introduce a co-attention mechanism that can simultaneously explore visual and semantic information to accurately localize and describe abnormal regions. We develop a hierarchical LSTM network that can more effectively capture long-range semantics and produce high quality long texts. On two medical datasets containing radiology and pathology images, we demonstrate the effectiveness of the proposed methods through quantitative and qualitative studies.

References

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. *ICLR*.

Louke Delrue, Robert Gosselin, Bart Ilsen, An Van Landeghem, Johan de Mey, and Philippe Duyck. 2011. Difficulties in the interpretation of chest radiography. In *Comparative Interpretation of CT and Standard Radiography of the Chest*, pages 27–49. Springer.

Dina Demner-Fushman, Marc D Kohli, Marc B Rosenman, Sonya E Shooshan, Laritza Rodriguez, Sameer Antani, George R Thoma, and Clement J McDonald. 2015. Preparing a collection of radiology examinations for distribution and retrieval. *Journal of the American Medical Informatics Association*, 23(2):304–310.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482.

Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. 2013. Deep convolutional ranking for multilabel image annotation. *ICLR*.

Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. 2009. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 309–316. IEEE.

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

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pavel Kisilev, Eugene Walach, Ella Barkan, Boaz Ophir, Sharon Alpert, and Sharbell Y Hashoul. 2015. From medical image to automatic medical report generation. *IBM Journal of Research and Development*, 59(2/3):2–1.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P. Xing. 2017. Recurrent topic-transition gan for visual paragraph generation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2326–2335.

Eric Martin and Chris Cundy. 2018. Parallelizing linear recurrent neural nets over sequence length. *ICLR*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. *ICLR*.

Hoo-Chang Shin, Kirk Roberts, Le Lu, Dina Demner-Fushman, Jianhua Yao, and Ronald M Summers. 2016. Learning to read chest x-rays: recurrent neural cascade model for automated image annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2497–2506.

K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659.

Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. 2017. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6428–6436.

Attacking Visual Language Grounding with Adversarial Examples: A Case Study on Neural Image Captioning

Hongge Chen[1*], **Huan Zhang**[23*], **Pin-Yu Chen**[3], **Jinfeng Yi**[4], and **Cho-Jui Hsieh**[2]
[1]MIT, Cambridge, MA 02139, USA
[2]UC Davis, Davis, CA 95616, USA
[3]IBM Research, NY 10598, USA
[4]JD AI Research, Beijing, China
chenhg@mit.edu, ecezhang@ucdavis.edu
pin-yu.chen@ibm.com, yijinfeng@jd.com, chohsieh@ucdavis.edu
[*]Hongge Chen and Huan Zhang contribute equally to this work

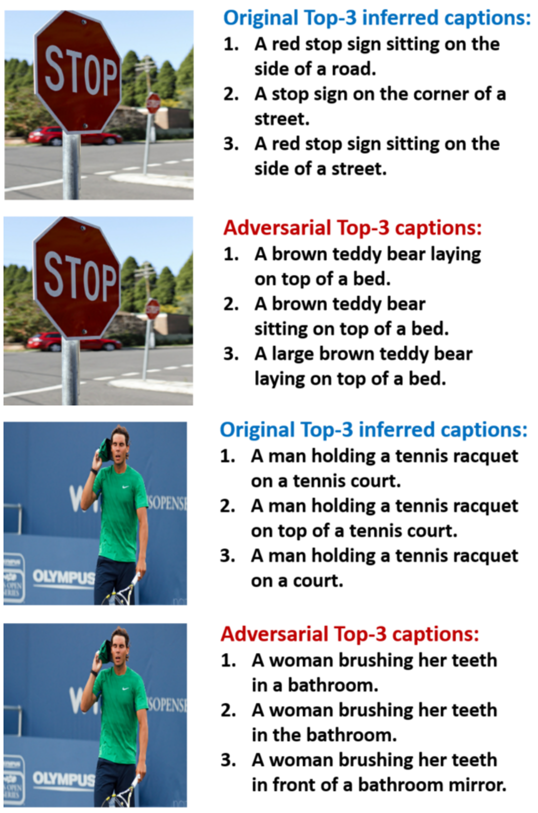
Abstract

Visual language grounding is widely studied in modern neural image captioning systems, which typically adopts an encoder-decoder framework consisting of two principal components: a convolutional neural network (CNN) for image feature extraction and a recurrent neural network (RNN) for language caption generation. To study the robustness of language grounding to adversarial perturbations in machine vision and perception, we propose **Show-and-Fool**, a novel algorithm for crafting adversarial examples in neural image captioning. The proposed algorithm provides two evaluation approaches, which check whether neural image captioning systems can be mislead to output some randomly chosen captions or keywords. Our extensive experiments show that our algorithm can successfully craft visually-similar adversarial examples with randomly targeted captions or keywords, and the adversarial examples can be made highly transferable to other image captioning systems. Consequently, our approach leads to new robustness implications of neural image captioning and novel insights in visual language grounding.

1 Introduction

In recent years, language understanding grounded in machine vision and perception has made remarkable progress in natural language processing (NLP) and artificial intelligence (AI), such as image captioning and visual question answering. Image captioning is a multimodal learning task and has been used to study the interaction between language and vision models (Shekhar et al., 2017). It takes an image as an input and generates a language caption that best describes its visual contents, and has many important applications such as developing image search engines with complex natural language queries, building AI agents that can see and talk, and promoting equal web access for people who are blind or visually impaired. Modern image captioning systems typically adopt an encoder-decoder framework composed of two principal modules: a convolutional neural network (CNN) as an encoder for image feature extraction and a recurrent neural network (RNN) as a decoder for caption generation. This CNN+RNN architecture includes popular image captioning models such as Show-and-Tell (Vinyals et al., 2015), Show-Attend-and-Tell (Xu et al., 2015) and NeuralTalk (Karpathy and Li, 2015).

Recent studies have highlighted the vulnerability of CNN-based image classifiers to adversarial examples: adversarial perturbations to benign images can be easily crafted to mislead a well-trained classifier, leading to visually indistinguishable adversarial examples to human (Szegedy et al., 2014; Goodfellow et al., 2015). In this study, we investigate a more challenging problem in visual language grounding domain that evaluates the robustness of multimodal RNN in the form of a CNN+RNN architecture, and use neural image captioning as a case study. Note that crafting adversarial examples in image captioning tasks is strictly harder than in well-studied image classification tasks, due to the following reasons: (i) class attack v.s. caption attack: unlike classification tasks where the class labels are well defined, the output of image captioning is a set of top-ranked captions. Simply treating different captions as distinct classes will result in an enormous number of classes that can even precede the number of training images. In addition, semantically similar



Original Top-3 inferred captions:

1. A red stop sign sitting on the side of a road.
2. A stop sign on the corner of a street.
3. A red stop sign sitting on the side of a street.

Adversarial Top-3 captions:

1. A brown teddy bear laying on top of a bed.
2. A brown teddy bear sitting on top of a bed.
3. A large brown teddy bear laying on top of a bed.

Original Top-3 inferred captions:

1. A man holding a tennis racquet on a tennis court.
2. A man holding a tennis racquet on top of a tennis court.
3. A man holding a tennis racquet on a court.

Adversarial Top-3 captions:

1. A woman brushing her teeth in a bathroom.
2. A woman brushing her teeth in the bathroom.
3. A woman brushing her teeth in front of a bathroom mirror.

Figure 1: Adversarial examples crafted by Show-and-Fool using the targeted caption method. The target captioning model is Show-and-Tell (Vinyals et al., 2015), the original images are selected from the MSCOCO validation set, and the targeted captions are randomly selected from the top-1 inferred caption of other validation images.

captions can be expressed in different ways and hence should not be viewed as different classes; and (ii) CNN v.s. CNN+RNN: attacking RNN models is significantly less well-studied than attacking CNN models. The CNN+RNN architecture is unique and beyond the scope of adversarial examples in CNN-based image classifiers.

In this paper, we tackle the aforementioned challenges by proposing a novel algorithm called *Show-and-Fool*. We formulate the process of crafting adversarial examples in neural image captioning systems as optimization problems with novel objective functions designed to adopt the CNN+RNN architecture. Specifically, our objective function is a linear combination of the distortion between benign and adversarial examples as well as some carefully designed loss functions. The proposed Show-and-Fool algorithm provides two approaches to craft adversarial examples in neural image captioning under different scenarios:

1. **Targeted caption method:** Given a targeted caption, craft adversarial perturbations to any image such that its generated caption matches the targeted caption.
2. **Targeted keyword method:** Given a set of keywords, craft adversarial perturbations to any image such that its generated caption contains the specified keywords. The captioning model has the freedom to make sentences with target keywords *in any order*.

As an illustration, Figure 1 shows an adversarial example crafted by Show-and-Fool using the targeted caption method. The adversarial perturbations are visually imperceptible while can successfully mislead Show-and-Tell to generate the targeted captions. Interestingly and perhaps surprisingly, our results pinpoint the Achilles heel of the language and vision models used in the tested image captioning systems. Moreover, the adversarial examples in neural image captioning highlight the inconsistency in visual language grounding between humans and machines, suggesting a possible weakness of current machine vision and perception machinery. Below we highlight our major contributions:

- We propose *Show-and-Fool*, a novel optimization based approach to crafting adversarial examples in image captioning. We provide two types of adversarial examples, targeted caption and targeted keyword, to analyze the robustness of neural image captioners. To the best of our knowledge, this is the very first work on crafting adversarial examples for image captioning.
- We propose powerful and generic loss functions that can craft adversarial examples and evaluate the robustness of the encoder-decoder pipelines in the form of a CNN+RNN architecture. In particular, our loss designed for targeted keyword attack only requires the adversarial caption to contain a few specified keywords; and we allow the neural network to *make meaningful sentences with these keywords on its own*.
- We conduct extensive experiments on the MSCOCO dataset. Experimental results show that our targeted caption method attains a 95.8% attack success rate when crafting adversarial examples with randomly assigned captions. In addition, our targeted keyword attack yields an even higher success rate. We also show that attacking CNN+RNN models is inherently different and more challenging than only attacking

CNN models.

- We also show that Show-and-Fool can produce highly transferable adversarial examples: an adversarial image generated for fooling Show-and-Tell can also fool other image captioning models, leading to new robustness implications of neural image captioning systems.

2 Related Work

In this section, we review the existing work on visual language grounding, with a focus on neural image captioning. We also review related work on adversarial attacks on CNN-based image classifiers. Due to space limitations, we defer the second part to the supplementary material.

Visual language grounding represents a family of multimodal tasks that bridge visual and natural language understanding. Typical examples include image and video captioning (Karpathy and Li, 2015; Vinyals et al., 2015; Donahue et al., 2015b; Pasunuru and Bansal, 2017; Venugopalan et al., 2015), visual dialog (Das et al., 2017; De Vries et al., 2017), visual question answering (Antol et al., 2015; Fukui et al., 2016; Lu et al., 2016; Zhu et al., 2017), visual storytelling (Huang et al., 2016), natural question generation (Mostafazadeh et al., 2017, 2016), and image generation from captions (Mansimov et al., 2016; Reed et al., 2016). In this paper, we focus on studying the robustness of neural image captioning models, and believe that the proposed method also sheds lights on robustness evaluation for other visual language grounding tasks using a similar multimodal RNN architecture.

Many image captioning methods based on deep neural networks (DNNs) adopt a multimodal RNN framework that first uses a CNN model as the encoder to extract a visual feature vector, followed by a RNN model as the decoder for caption generation. Representative works under this framework include (Chen and Zitnick, 2015; Devlin et al., 2015; Donahue et al., 2015a; Karpathy and Li, 2015; Mao et al., 2015; Vinyals et al., 2015; Xu et al., 2015; Yang et al., 2016; Liu et al., 2017a,b), which are mainly differed by the underlying CNN and RNN architectures, and whether or not the attention mechanisms are considered. Other lines of research generate image captions using semantic information or via a compositional approach (Fang et al., 2015; Gan et al., 2017; Tran et al., 2016; Jia et al., 2015; Wu et al., 2016; You et al., 2016).

The recent work in (Shekhar et al., 2017) touched upon the robustness of neural image captioning for language grounding by showing its insensitivity to one-word (foil word) changes in the language caption, which corresponds to the *untargeted attack* category in adversarial examples. In this paper, we focus on the more challenging *targeted attack* setting that requires to fool the captioning models and enforce them to generate pre-specified captions or keywords.

3 Methodology of Show-and-Fool

3.1 Overview of the Objective Functions

We now formally introduce our approaches to crafting adversarial examples for neural image captioning. The problem of finding an adversarial example for a given image I can be cast as the following optimization problem:

$$\begin{aligned} \min_{\delta} \quad & c \cdot \text{loss}(I+\delta) + \|\delta\|_2^2 \\ \text{s.t.} \quad & I+\delta \in [-1,1]^n. \end{aligned} \tag{1}$$

Here δ denotes the adversarial perturbation to I. $\|\delta\|_2^2 = \|(I+\delta) - I\|_2^2$ is an ℓ_2 distance metric between the original image and the adversarial image. $\text{loss}(\cdot)$ is an attack loss function which takes different forms in different attacking settings. We will provide the explicit expressions in Sections 3.2 and 3.3. The term $c > 0$ is a pre-specified regularization constant. Intuitively, with larger c, the attack is more likely to succeed but at the price of higher distortion on δ. In our algorithm, we use a binary search strategy to select c. The box constraint on the image $I \in [-1,1]^n$ ensures that the adversarial example $I+\delta \in [-1,1]^n$ lies within a valid image space.

For the purpose of efficient optimization, we convert the constrained minimization problem in (1) into an unconstrained minimization problem by introducing two new variables $y \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$ such that

$$y = \text{arctanh}(I) \quad \text{and} \quad w = \text{arctanh}(I+\delta) - y,$$

where arctanh denotes the inverse hyperbolic tangent function and is applied element-wisely. Since $\tanh(y_i + w_i) \in [-1,1]$, the transformation will automatically satisfy the box constraint. Consequently, the constrained optimization problem in

(1) is equivalent to

$$\min_{w\in\mathbb{R}^n} \quad c \cdot \text{loss}(\tanh(w+y)) \\ +\|\tanh(w+y)-\tanh(y)\|_2^2. \tag{2}$$

In the following sections, we present our designed loss functions for different attack settings.

3.2 Targeted Caption Method

Note that a targeted caption is denoted by

$$S = (S_1,\ S_2,\ ...,\ S_t,\ ...,\ S_N),$$

where S_t indicates the index of the t-th word in the vocabulary list $\mathcal{V}$, S_1 is a start symbol and S_N indicates the end symbol. N is the length of caption S, which is not fixed but does not exceed a predefined maximum caption length. To encourage the neural image captioning system to output the targeted caption S, one needs to ensure the log probability of the caption S conditioned on the image $I+\delta$ attains the maximum value among all possible captions, that is,

$$\log P(S|I+\delta) = \max_{S'\in\Omega} \log P(S'|I+\delta), \tag{3}$$

where Ω is the set of all possible captions. It is also common to apply the chain rule to the joint probability and we have

$$\log P(S'|I+\delta) = \sum_{t=2}^{N} \log P(S'_t|I+\delta, S'_1, ..., S'_{t-1}).$$

In neural image captioning networks, $p(S'_t|I+\delta, S'_1, ..., S'_{t-1})$ is usually computed by a RNN/LSTM cell f, with its hidden state h_{t-1} and input S'_{t-1}:

$$z_t = f(h_{t-1}, S'_{t-1}) \text{ and } p_t = \text{softmax}(z_t), \tag{4}$$

where $z_t := [z_t^{(1)}, z_t^{(2)}, ..., z_t^{(|\mathcal{V}|)}] \in \mathbb{R}^{|\mathcal{V}|}$ is a vector of the *logits* (unnormalized probabilities) for each possible word in the vocabulary. The vector p_t represents a probability distribution on $\mathcal{V}$ with each coordinate $p_t^{(i)}$ defined as:

$$p_t^{(i)} := P(S'_t = i|I+\delta, S'_1, ..., S'_{t-1}).$$

Following the definition of softmax function:

$$P(S'_t|I+\delta, S'_1, ..., S'_{t-1}) = \exp(z_t^{(S'_t)}) / \sum_{i\in\mathcal{V}} \exp(z_t^{(i)}).$$

Intuitively, to maximize the targeted caption's probability, we can directly use its negative log probability (5) as a loss function. The inputs of the RNN are the first $N-1$ words of the targeted caption $(S_1, S_2, ..., S_{N-1})$.

$$\begin{aligned}\text{loss}_{S,\text{log-prob}}(I+\delta) &= -\log P(S|I+\delta) \\ &= -\sum_{t=2}^{N} \log P(S_t|I+\delta, S_1, ..., S_{t-1}).\end{aligned} \tag{5}$$

Applying (5) to (2), the formulation of targeted caption method given a targeted caption S is:

$$\min_{w\in\mathbb{R}^n} c \cdot \text{loss}_{S,\text{log prob}}(\tanh(w+y)) \\ + \ \|\tanh(w+y)-\tanh(y)\|_2^2.$$

Alternatively, using the definition of the softmax function,

$$\begin{aligned}\log P(S'|I+\delta) &= \sum_{t=2}^{N} [z_t^{(S'_t)} - \log(\sum_{i\in\mathcal{V}} \exp(z_t^{(i)}))] \\ &= \sum_{t=2}^{N} z_t^{(S'_t)} - \text{constant},\end{aligned} \tag{6}$$

(3) can be simplified as

$$\log P(S|I+\delta) \propto \sum_{t=2}^{N} z_t^{(S_t)} = \max_{S'\in\Omega} \sum_{t=2}^{N} z_t^{(S'_t)}.$$

Instead of making each $z_t^{(S_t)}$ as large as possible, it is sufficient to require the target word S_t to attain the largest (top-1) logit (or probability) among all the words in the vocabulary at position t. In other words, we aim to minimize the difference between the maximum logit except S_t, denoted by $\max_{k\in\mathcal{V},k\neq S_t}\{z_t^{(k)}\}$, and the logit of S_t, denoted by $z_t^{(S_t)}$. We also propose a ramp function on top of this difference as the final loss function:

$$\text{loss}_{S,\text{logits}}(I+\delta) = \sum_{t=2}^{N-1} \max\{-\epsilon, \max_{k\neq S_t}\{z_t^{(k)}\} - z_t^{(S_t)}\}, \tag{7}$$

where $\epsilon > 0$ is a confidence level accounting for the gap between $\max_{k\neq S_t}\{z_t^{(k)}\}$ and $z_t^{(S_t)}$. When $z_t^{(S_t)} > \max_{k\neq S_t}\{z_t^{(k)}\} + \epsilon$, the corresponding term in the summation will be kept at $-\epsilon$ and does not contribute to the gradient of the loss function, encouraging the optimizer to focus on minimizing other terms where $z_t^{(S_t)}$ is not large enough.

Applying the loss (7) to (1), the final formulation of targeted caption method given a targeted caption S is

$$\min_{w \in \mathbb{R}^n} c \cdot \sum_{t=2}^{N-1} \max\{-\epsilon, \max_{k \neq S_t}\{z_t^{(k)}\} - z_t^{(S_t)}\} + \| \tanh(w+y) - \tanh(y) \|_2^2.$$

We note that (Carlini and Wagner, 2017) has reported that in CNN-based image classification, using logits in the attack loss function can produce better adversarial examples than using probabilities, especially when the target network deploys some gradient masking schemes such as defensive distillation (Papernot et al., 2016b). Therefore, we provide both logit-based and probability-based attack loss functions for neural image captioning.

3.3 Targeted Keyword Method

In addition to generating an exact targeted caption by perturbing the input image, we offer an intermediate option that aims at generating captions with specific keywords, denoted by $\mathcal{K} := \{K_1, \cdots, K_M\} \subset \mathcal{V}$. Intuitively, finding an adversarial image generating a caption with specific keywords might be easier than generating an exact caption, as we allow more degree of freedom in caption generation. However, as we need to ensure a valid and meaningful inferred caption, finding an adversarial example with specific keywords in its caption is difficult in an optimization perspective. Our target keyword method can be used to investigate the generalization capability of a neural captioning system given only a few keywords.

In our method, we do not require a target keyword K_j, $j \in [M]$ to appear at a particular position. Instead, we want a loss function that allows K_j to become the top-1 prediction (plus a confidence margin ϵ) at any position. Therefore, we propose to use the minimum of the hinge-like loss terms over all $t \in [N]$ as an indication of K_j appearing at any position as the top-1 prediction, leading to the following loss function:

$$\text{loss}_{K,\text{logits}} = \sum_{j=1}^{M} \min_{t \in [N]} \{\max\{-\epsilon, \max_{k \neq K_j}\{z_t^{(k)}\} - z_t^{(K_j)}\}\}. \tag{8}$$

We note that the loss functions in (4) and (5) require an input S'_{t-1} to predict z_t for each $t \in \{2, \ldots, N\}$. For the targeted caption method, we use the targeted caption S as the input of RNN. In contrast, for the targeted keyword method we no longer know the exact targeted sentence, but only require the presence of specified keywords in the final caption. To bridge the gap, we use the originally inferred caption $S^0 = (S_1^0, \cdots, S_N^0)$ from the benign image as the initial input to RNN. Specifically, after minimizing (8) for T iterations, we run inference on $I + \delta$ and set the RNN's input S^1 as its current top-1 prediction, and continue this process. With this iterative optimization process, the desired keywords are expected to gradually appear in top-1 prediction.

Another challenge arises in targeted keyword method is the problem of "keyword collision". When the number of keywords $M \geq 2$, more than one keywords may have large values of $\max_{k \neq K_j}\{z_t^{(k)}\} - z_t^{(K_j)}$ at a same position t. For example, if `dog` and `cat` are top-2 predictions for the second word in a caption, the caption can either start with "A dog ..." or "A cat ...". In this case, despite the loss (8) being very small, a caption with both `dog` and `cat` can hardly be generated, since only one word is allowed to appear at the same position. To alleviate this problem, we define a gate function $g_{t,j}(x)$ which masks off all the other keywords when a keyword becomes top-1 at position t:

$$g_{t,j}(x) = \begin{cases} A, \text{ if } \arg\max_{i \in \mathcal{V}} z_t^{(i)} \in \mathcal{K} \setminus \{K_j\} \\ x, \text{ otherwise,} \end{cases}$$

where A is a predefined value that is significantly larger than common logits values. Then (8) becomes:

$$\sum_{j=1}^{M} \min_{t \in [N]} \{g_{t,j}(\max\{-\epsilon, \max_{k \neq K_j}\{z_t^{(k)}\} - z_t^{(K_j)}\})\}. \tag{9}$$

The log-prob loss for targeted keyword method is discussed in the Supplementary Material.

4 Experiments

4.1 Experimental Setup and Algorithms

We performed extensive experiments to test the effectiveness of our Show-and-Fool algorithm and study the robustness of image captioning systems under different problem settings. In our experiments[1], we use the pre-trained TensorFlow implementation[2] of Show-and-Tell (Vinyals et al., 2015)

[1]Our source code is available at: https://github.com/huanzhang12/ImageCaptioningAttack

[2]https://github.com/tensorflow/models/tree/master/research/im2txt

with Inception-v3 as the CNN for visual feature extraction. Our testbed is Microsoft COCO (Lin et al., 2014) (MSCOCO) data set. Although some more recent neural image captioning systems can achieve better performance than Show-and-Tell, they share a similar framework that uses CNN for feature extraction and RNN for caption generation, and Show-and-Tell is the vanilla version of this CNN+RNN architecture. Indeed, we find that the adversarial examples on Show-and-Tell are transferable to other image captioning models such as Show-Attend-and-Tell (Xu et al., 2015) and NeuralTalk2[3], suggesting that the attention mechanism and the choice of CNN and RNN architectures do not significantly affect the robustness. We also note that since Show-and-Fool is the first work on crafting adversarial examples for neural image captioning, to the best of our knowledge, there is no other method for comparison.

We use ADAM to minimize our loss functions and set the learning rate to 0.005. The number of iterations is set to $1,000$. All the experiments are performed on a single Nvidia GTX 1080 Ti GPU. For targeted caption and targeted keyword methods, we perform a binary search for 5 times to find the best c: initially $c = 1$, and c will be increased by 10 times until a successful adversarial example is found. Then, we choose a new c to be the average of the largest c where an adversarial example can be found and the smallest c where an adversarial example cannot be found. We fix $\epsilon = 1$ except for transferability experiments. For each experiment, we randomly select 1,000 images from the MSCOCO validation set. We use BLEU-1 (Papineni et al., 2002), BLEU-2, BLEU-3, BLEU-4, ROUGE (Lin, 2004) and METEOR (Lavie and Agarwal, 2005) scores to evaluate the correlations between the inferred captions and the targeted captions. These scores are widely used in NLP community and are adopted by image captioning systems for quality assessment. Throughout this section, we use the *logits loss* (7)(9). The results of using the *log-prob loss* (5) are similar and are reported in the supplementary material.

4.2 Targeted Caption Results

Unlike the image classification task where all possible labels are predefined, the space of possible captions in a captioning system is almost infinite. However, the captioning system is only able to output relevant captions learned from the training set. For instance, the captioning model cannot generate a passive-voice sentence if the model was never trained on such sentences. Therefore, we need to ensure that the targeted caption lies in the space where the captioning system can possibly generate. To address this issue, we use the generated caption of a randomly selected image (other than the image under investigation) from MSCOCO validation set as the targeted caption S. The use of a generated caption as the targeted caption excludes the effect of out-of-domain captioning, and ensures that the target caption is within the output space of the captioning network.

Here we use the logits loss (7) plus a ℓ_2 distortion term (as in (2)) as our objective function. A successful adversarial example is found if the inferred caption after adding the adversarial perturbation δ is *exactly the same* as the targeted caption. In our setting, 1,000 ADAM iterations take about 38 seconds for one image. The overall success rate and average distortion of adversarial perturbation δ are shown in Table 1. Among all the tested images, our method attains 95.8% attack success

[3]https://github.com/karpathy/neuraltalk2

Table 1: Summary of targeted caption method (Section 3.2) and targeted keyword method (Section 3.3) using logits loss. The ℓ_2 distortion of adversarial noise $\|\delta\|_2$ is averaged over successful adversarial examples. For comparison, we also include CNN based attack methods (Section 4.5).

Experiments	Success Rate	Avg. $\|\delta\|_2$
targeted caption	95.8%	2.213
1-keyword	97.1%	1.589
2-keyword	97.5%	2.363
3-keyword	96.0%	2.626
C&W on CNN	22.4%	2.870
I-FGSM on CNN	34.5%	15.596

Table 2: Statistics of the 4.2% failed adversarial examples using the targeted caption method and logits loss (7). All correlation scores are computed using the top-5 inferred captions of an adversarial image and the targeted caption (higher score means better targeted attack performance).

c	1	10	10^2	10^3	10^4
ℓ_2 Distortion	1.726	3.400	7.690	16.03	23.31
BLEU-1	.567	.725	.679	.701	.723
BLEU-2	.420	.614	.559	.585	.616
BLEU-3	.320	.509	.445	.484	.514
BLEU-4	.252	.415	.361	.402	.417
ROUGE	.502	.664	.629	.638	.672
METEOR	.258	.407	.375	.403	.399

rate. Moreover, our adversarial examples have small ℓ_2 distortions and are visually identical to the original images, as displayed in Figure 1. We also examine the failed adversarial examples and summarize their statistics in Table 2. We find that their generated captions, albeit not entirely identical to the targeted caption, are in fact highly correlated to the desired one. Overall, the high success rate and low ℓ_2 distortion of adversarial examples clearly show that Show-and-Tell is not robust to targeted adversarial perturbations.

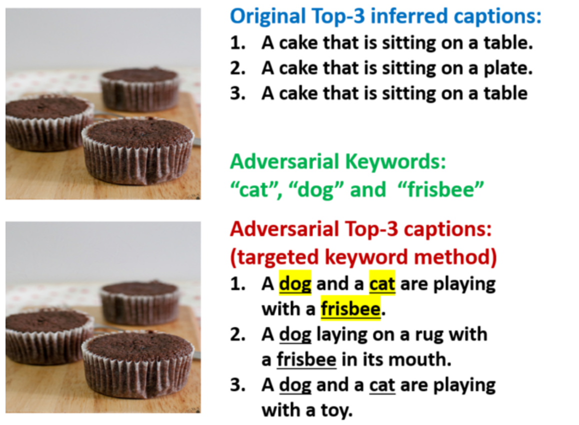
4.3 Targeted Keyword Results

In this task, we use (9) as our loss function, and choose the number of keywords $M = \{1, 2, 3\}$. We run an inference step on $I + \delta$ every $T = 5$ iterations, and use the top-1 caption as the input of RNN/LSTMs. Similar to Section 4.2, for each image the targeted keywords are selected from the caption generated by a randomly selected validation set image. To exclude common words like "a", "the", "and", we look up each word in the targeted sentence and only select nouns, verbs, adjectives or adverbs. We say an adversarial image is successful when its caption contains *all* specified keywords. The overall success rate and average distortion are shown in Table 1. When compared to the targeted caption method, targeted keyword method achieves an even higher success rate (at least 96% for 3-keyword case and at least 97% for 1-keyword and 2-keyword cases). Figure 2 shows an adversarial example crafted from our targeted keyword method with three keywords - "dog", "cat" and "frisbee". Using Show-and-Fool, the top-1 caption of a cake image becomes "A dog and a cat are playing with a frisbee" while the adversarial image remains visually indistinguishable to the original one. When $M = 2$ and 3, even if we cannot find an adversarial image yielding all specified keywords, we might end up with a caption that contains some of the keywords (partial success). For example, when $M = 3$, Table 3 shows the number of keywords appeared in the captions (M') for those *failed* examples (not all 3 targeted keywords are found). These results clearly show that the 4% failed examples are still partially successful: the generated captions contain about 1.5 targeted keywords on average.

4.4 Transferability of Adversarial Examples

It has been shown that in image classification tasks, adversarial examples found for one machine learning model may also be effective against another model, even if the two models have different architectures (Papernot et al., 2016a; Liu et al., 2017c). However, unlike image classification where correct labels are made explicit, two different image captioning systems may generate quite different, yet semantically similar, captions for the same benign image. In image captioning, we say an adversarial example is *transferable* when the adversarial image found on model A with a target sentence S_A can generate a similar (rather than exact) sentence S_B on model B.

In our setting, model A is Show-and-Tell, and we choose Show-Attend-and-Tell (Xu et al., 2015) as model B. The major differences between Show-and-Tell and Show-Attend-and-Tell are the addition of attention units in LSTM network for caption generation, and the use of last convolutional layer (rather than the last fully-connected layer) feature maps for feature extraction. We use Inception-v3 as the CNN architecture for both models and train them on the MSCOCO 2014 data set. However, their CNN parameters are different due to the fine-tuning process.


Original Top-3 inferred captions:
1. A cake that is sitting on a table.
2. A cake that is sitting on a plate.
3. A cake that is sitting on a table

Adversarial Keywords:
"cat", "dog" and "frisbee"

Adversarial Top-3 captions:
(targeted keyword method)
1. A dog and a cat are playing with a frisbee.
2. A dog laying on a rug with a frisbee in its mouth.
3. A dog and a cat are playing with a toy.

Figure 2: An adversarial example ($\|\delta\|_2 = 1.284$) of an cake image crafted by the Show-and-Fool targeted keyword method with three keywords - "dog", "cat" and "frisbee".

Table 3: Percentage of partial success with different c in the 4.0% failed images that do not contain all the 3 targeted keywords.

c	Avg. $\|\delta\|_2$	$M' \geq 1$	$M' = 2$	Avg. M'
1	2.49	72.4%	34.5%	1.07
10	5.40	82.7%	37.9%	1.21
10^2	12.95	93.1%	58.6%	1.52
10^3	24.77	96.5%	51.7%	1.48
10^4	29.37	100.0%	58.6%	1.59

Table 4: Transferability of adversarial examples from Show-and-Tell to Show-Attend-and-Tell, using different ϵ and c. **ori** indicates the scores between the generated captions of the *original* images and the transferred adversarial images on Show-Attend-and-Tell. **tgt** indicates the scores between the *targeted* captions on Show-and-Tell and the generated captions of transferred adversarial images on Show-Attend-and-Tell. A smaller **ori** or a larger **tgt** value indicates better transferability. **mis** measures the differences between captions generated by the two models given the same benign image (*model mismatch*). When $C = 1000$, $\epsilon = 10$, **tgt** is close to **mis**, indicating the discrepancy between adversarial captions on the two models is mostly bounded by model mismatch, and the adversarial perturbation is highly transferable.

	$\epsilon=1$						$\epsilon=5$						$\epsilon=10$						
	C=10		C=100		C=1000		C=10		C=100		C=1000		C=10		C=100		C=1000		
	ori	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**ori**	**tgt**	**mis**
BLEU-1	.474	.395	.384	.462	.347	.484	.441	.429	.368	.488	**.337**	.527	.431	.421	.360	.485	.339	**.534**	.649
BLEU-2	.337	.236	.230	.331	.186	.342	.300	.271	.212	.343	.175	.389	.287	.266	.204	.342	**.174**	**.398**	.521
BLEU-3	.256	.154	.151	.224	.114	.254	.220	.184	.135	.254	.103	.299	.210	.185	.131	.254	**.102**	**.307**	.424
BLEU-4	.203	.109	.107	.172	.077	.198	.170	.134	.093	.197	.068	.240	.162	.138	.094	.197	**.066**	**.245**	.352
ROUGE	.463	.371	.374	.438	.336	.465	.429	.402	.359	.464	.329	.502	.421	.398	.351	.463	**.328**	**.507**	.604
METEOR	.201	.138	.139	.180	.118	.201	.177	.157	.131	.199	.110	.228	.172	.157	.127	.202	**.110**	**.232**	.300
$\|\delta\|_2$	3.268		4.299		4.474		7.756		10.487		10.952		15.757		21.696		21.778		

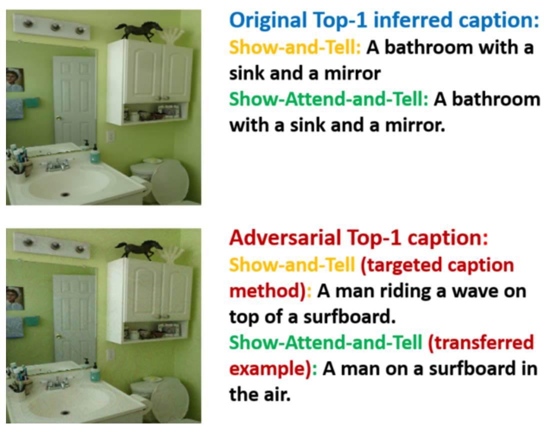

Original Top-1 inferred caption:
Show-and-Tell: A bathroom with a sink and a mirror
Show-Attend-and-Tell: A bathroom with a sink and a mirror.
Adversarial Top-1 caption:
Show-and-Tell (targeted caption method): A man riding a wave on top of a surfboard.
Show-Attend-and-Tell (transferred example): A man on a surfboard in the air.

Figure 3: A highly transferable adversarial example ($\|\delta\|_2 = 15.226$) crafted by Show-and-Tell targeted caption method, transfers to Show-Attend-and-Tell, yielding similar adversarial captions.

To investigate the transferability of adversarial examples in image captioning, we first use the targeted caption method to find adversarial examples for 1,000 images in model A with different c and ϵ, and then transfer successful adversarial examples (which generate the exact target captions on model A) to model B. The generated captions by model B are recorded for transferability analysis. The transferability of adversarial examples depends on two factors: the intrinsic difference between two models even when the same benign image is used as the input, i.e., *model mismatch*, and the transferability of adversarial perturbations.

To measure the mismatch between Show-and-Tell and Show-Attend-and-Tell, we generate captions of the same set of 1,000 original images from both models, and report their mutual BLEU, ROUGE and METEOR scores in Table 4 under the **mis** column. To evaluate the effectiveness of transferred adversarial examples, we measure the scores for two set of captions: (i) the captions of original images and the captions of transferred adversarial images, both generated by Show-Attend-and-Tell (shown under column **ori** in Table 4); and (ii) the targeted captions for generating adversarial examples on Show-and-Tell, and the captions of the transferred adversarial image on Show-Attend-and-Tell (shown under column **tgt** in Table 4). Small values of **ori** suggest that the adversarial images on Show-Attend-and-Tell generate significantly different captions from original images' captions. Large values of **tgt** suggest that the adversarial images on Show-Attend-and-Tell generate similar adversarial captions as on the Show-and-Tell model. We find that increasing c or ϵ helps to enhance transferability at the cost of larger (but still acceptable) distortion. When $C = 1{,}000$ and $\epsilon = 10$, Show-and-Fool achieves the best transferability results: **tgt** is close to **mis**, indicating that the discrepancy between adversarial captions on the two models is mostly bounded by the intrinsic model mismatch rather than the transferability of adversarial perturbations, and implying that the adversarial perturbations are easily transferable. In addition, the adversarial examples generated by our method can also fool NeuralTalk2. When $c = 10^4, \epsilon = 10$, the average ℓ_2 distortion, BLEU-4 and METEOR scores between the original and transferred adversarial captions are 38.01, 0.440 and 0.473, respectively. The high transferability of adversarial examples crafted by Show-

and-Fool also indicates the problem of common robustness leakage between different neural image captioning models.

4.5 Attacking Image Captioning v.s. Attacking Image Classification

In this section we show that attacking image captioning models is inherently more challenging than attacking image classification models. In the classification task, a targeted attack usually becomes harder when the number of labels increases, since an attack method needs to change the classification prediction to a specific label over all the possible labels. In the targeted attack on image captioning, if we treat each caption as a label, we need to change the original label to a specific one over an almost infinite number of possible labels, corresponding to a nearly zero volume in the search space. This constraint forces us to develop non-trivial methods that are significantly different from the ones designed for attacking image classification models.

To verify that the two tasks are inherently different, we conducted additional experiments on attacking *only* the CNN module using two state-of-the-art image classification attacks on ImageNet dataset. Our experiment setup is as follows. Each selected ImageNet image has a label corresponding to a WordNet synset ID. We randomly selected 800 images from ImageNet dataset such that their synsets have at least one word in common with Show-and-Tell's vocabulary, while ensuring the Inception-v3 CNN (Show-and-Tell's CNN) classify them correctly. Then, we perform Iterative Fast Gradient Sign Method (I-FGSM) (Kurakin et al., 2017) and Carlini and Wagner's (C&W) attack (Carlini and Wagner, 2017) on these images. The attack target labels are randomly chosen and their synsets also have at least one word in common with Show-and-Tell's vocabulary. Both I-FGSM and C&W achieve 100% targeted attack success rate on the Inception-v3 CNN. These adversarial examples were further employed to attack Show-and-Tell model. An attack is considered successful if *any* word in the targeted label's synset or its hypernyms up to 5 levels is presented in the resulting caption. For example, for the chain of hypernyms 'broccoli'⇒'cruciferous vegetable'⇒'vegetable, veggie, veg'⇒'produce, green goods, green groceries, garden truck'⇒'food, solid food', we include 'broccoli','cruciferous','vegetable','veggie' and all other following words. Note that this criterion of success is much weaker than the criterion we use in the targeted caption method, since a caption with the targeted image's hypernyms does not necessarily leads to similar meaning of the targeted image's captions. To achieve higher attack success rates, we allow relatively larger distortions and set $\epsilon_\infty = 0.3$ (maximum ℓ_∞ distortion) in I-FGSM and $\kappa = 10$, $C = 100$ in C&W. However, as shown in Table 1, the attack success rates are only 34.5% for I-FGSM and 22.4% for C&W, respectively, which are much lower than the success rates of our methods despite larger distortions. This result further confirms that performing targeted attacks on neural image captioning requires a careful design (as proposed in this paper), and attacking image captioning systems is not a trivial extension to attacking image classifiers.

5 Conclusion

In this paper, we proposed a novel algorithm, **Show-and-Fool**, for crafting adversarial examples and providing robustness evaluation of neural image captioning. Our extensive experiments show that the proposed targeted caption and keyword methods yield high attack success rates while the adversarial perturbations are still imperceptible to human eyes. We further demonstrate that Show-and-Fool can generate highly transferable adversarial examples. The high-quality and transferable adversarial examples in neural image captioning crafted by Show-and-Fool highlight the inconsistency in visual language grounding between humans and machines, suggesting a possible weakness of current machine vision and perception machinery. We also show that attacking neural image captioning systems are inherently different from attacking CNN-based image classifiers.

Our method stands out from the well-studied adversarial learning on image classifiers and CNN models. To the best of our knowledge, this is the very first work on crafting adversarial examples for neural image captioning systems. Indeed, our Show-and-Fool algorithm[1] can be easily extended to other applications with RNN or CNN+RNN architectures. We believe this paper provides potential means to evaluate and possibly improve the robustness (for example, by adversarial training or data augmentation) of a wide range of visual language grounding and other NLP models.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. EAD: elastic-net attacks to deep neural networks via adversarial examples. *AAAI*.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS*, pages 15–26.

Xinlei Chen and C. Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*, pages 2422–2431.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *CVPR*.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 100–105.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015a. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015b. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *CVPR*, pages 1473–1482.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 457–468.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5630–5639.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *ICLR; arXiv preprint arXiv:1412.6572*.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1233–1239.

Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. 2015. Guiding the long-short term memory model for image caption generation. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2407–2415. IEEE.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. *ICLR; arXiv preprint arXiv:1611.01236*.

Alon Lavie and Abhaya Agarwal. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, pages 65–72.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan L Yuille. 2017a. Attention correctness in neural image captioning. In *AAAI*, pages 4176–4182.

Feng Liu, Tao Xiang, Timothy M Hospedales, Wankou Yang, and Changyin Sun. 2017b. Semantic regularisation for recurrent image annotation. *CVPR*.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017c. Delving into transferable adversarial examples and black-box attacks. *ICLR; arXiv preprint arXiv:1611.02770*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems (NIPS)*, pages 289–297.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2016. Generating images from captions with attention. *ICLR; arXiv preprint arXiv:1511.02793*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR; arXiv preprint arXiv:1412.6632*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *CVPR*.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 462–472.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1802–1813.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016a. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *annual meeting on association for computational linguistics (ACL)*, pages 311–318.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1273–1283.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurelie Herbelot, Moin Nabi, Enver Sangineto, Raffaella Bernardi, et al. 2017. Foil it! Find one mismatch between image and language caption. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *ICLR;arXiv preprint arXiv:1312.6199*.

Kenneth Tran, Xiaodong He, Lei Zhang, and Jian Sun. 2016. Rich image captioning in the wild. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2016 IEEE Conference on*, pages 434–441. IEEE.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *NAACL-HLT*, pages 1494–1504.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton van den Hengel. 2016. What value do explicit high level concepts have in vision to language problems? In *CVPR*, pages 203–212.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057.

Zhilin Yang, Ye Yuan, Yuexin Wu, William W. Cohen, and Ruslan Salakhutdinov. 2016. Review networks for caption generation. In *NIPS*, pages 2361–2369.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *CVPR*, pages 4651–4659.

Linchao Zhu, Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. 2017. Uncovering the temporal context for video question answering. *International Journal of Computer Vision*, 124(3):409–421.

Think Visually: Question Answering through Virtual Imagery

Ankit Goyal **Jian Wang** **Jia Deng**
Computer Science and Engineering
University of Michigan, Ann Arbor
{ankgoyal, jianwolf, jiadeng}@umich.edu

Abstract

In this paper, we study the problem of geometric reasoning in the context of question-answering. We introduce Dynamic Spatial Memory Network (DSMN), a new deep network architecture designed for answering questions that admit latent visual representations. DSMN learns to generate and reason over such representations. Further, we propose two synthetic benchmarks, FloorPlanQA and ShapeIntersection, to evaluate the geometric reasoning capability of QA systems. Experimental results validate the effectiveness of our proposed DSMN for visual thinking tasks[1].

1 Introduction

The ability to reason is a hallmark of intelligence and a requirement for building question-answering (QA) systems. In AI research, reasoning has been strongly associated with logic and symbol manipulation, as epitomized by work in automated theorem proving (Fitting, 2012). But for humans, reasoning involves not only symbols and logic, but also images and shapes. Einstein famously wrote: "The psychical entities which seem to serve as elements in thought are certain signs and more or less clear images which can be 'voluntarily' reproduced and combined... Conventional words or other signs have to be sought for laboriously only in a secondary state..." And the history of science abounds with discoveries from visual thinking, from the Benzene ring to the structure of DNA (Pinker, 2003).

There are also plenty of ordinary examples of human visual thinking. Consider a square room with a door in the middle of its southern wall. Suppose you are standing in the room such that the eastern wall of the room is behind you. Where is the door with respect to you? The answer is 'to your left.' Note that in this case both the question and answer are just text. But in order to answer the question, it is natural to construct a mental picture of the room and use it in the process of reasoning. Similar to humans, the ability to 'think visually' is desirable for AI agents like household robots. An example could be to construct a rough map and navigation plan for an unknown environment from verbal descriptions and instructions.

In this paper, we investigate how to model geometric reasoning (a form of visual reasoning) using deep neural networks (DNN). Specifically, we address the task of answering questions through geometric reasoning—both the question and answer are expressed in symbols or words, but a geometric representation is created and used as part of the reasoning process.

In order to focus on geometric reasoning, we do away with natural language by designing two synthetic QA datasets, FloorPlanQA and ShapeIntersection. In FloorPlanQA, we provide the blueprint of a house in words and ask questions about location and orientation of objects in it. For ShapeIntersection, we give a symbolic representation of various shapes and ask how many places they intersect. In both datasets, a reference visual representation is provided for each sample.

Further, we propose Dynamic Spatial Memory Network (DSMN), a novel DNN that uses virtual imagery for QA. DSMN is similar to existing memory networks (Kumar et al., 2016; Sukhbaatar et al., 2015; Henaff et al., 2016) in that it uses vector embeddings of questions and memory modules to perform reasoning. The main novelty of DSMN is that it creates virtual images for the input question and uses a spatial memory to aid the reasoning

[1] Code and datasets: `https://github.com/umich-vl/think_visually`

process.

We show through experiments that with the aid of an internal visual representation and a spatial memory, DSMN outperforms strong baselines on both FloorPlanQA and ShapeIntersection. We also demonstrate that explicitly learning to create visual representations further improves performance. Finally, we show that DSMN is substantially better than the baselines even when visual supervision is provided for only a small proportion of the samples.

It's important to note that our proposed datasets consist of synthetic questions as opposed to natural texts. Such a setup allows us to sidestep difficulties in parsing natural language and instead focus on geometric reasoning. However, synthetic data lacks the complexity and diversity of natural text. For example, spatial terms used in natural language have various ambiguities that need to resolved by context (e.g. how far is "far" and whether "to the left" is relative to the speaker or the listener) (Shariff, 1998; Landau and Jackendoff, 1993), but our synthetic data lacks such complexities. Therefore, our method and results do not automatically generalize to real-life tasks involving natural language. Additional research is needed to extend and validate our approach on natural data.

Our contributions are three-fold: First, we present Dynamic Spatial Memory Network (DSMN), a novel DNN that performs geometric reasoning for QA. Second, we introduce two synthetic datasets that evaluate a system's visual thinking ability. Third, we demonstrate that on synthetic data, DSMN achieves superior performance for answering questions that require visual thinking.

2 Related Work

Natural language datasets for QA: Several natural language QA datasets have been proposed to test AI systems on various reasoning abilities (Levesque et al., 2011; Richardson et al., 2013). Our work differs from them in two key aspects: first, we use synthetic data instead of natural data; and second, we specialize in geometrical reasoning instead of general language understanding. Using synthetic data helps us simplify language parsing and thereby focus on geometric reasoning. However, additional research is necessary to generalize our work to natural data.

Synthetic datasets for QA: Recently, synthetic datasets for QA are also becoming crucial in AI. In particular, bAbI (Weston et al., 2015) has driven the development of several recent DNN-based QA systems (Kumar et al., 2016; Sukhbaatar et al., 2015; Henaff et al., 2016). bAbI consists of 20 tasks to evaluate different reasoning abilities. Two tasks, Positional Reasoning (PR) and Path Finding (PF), are related to geometric reasoning. However, each Positional Reasoning question contains only two sentences, and can be solved through simple logical deduction such as 'A is left of B implies B is right of A'. Similarly, Path Finding involves a search problem that requires simple spatial deductions such as 'A is east of B implies B is west of A'. In contrast, the questions in our datasets involve longer descriptions, more entities, and more relations; they are thus harder to answer with simple deductions. We also provide reference visual representation for each sample, which is not available in bAbI.

Mental Imagery and Visual Reasoning: The importance of visual reasoning has been long recognized in AI (Forbus et al., 1991; Lathrop and Laird, 2007). Prior works in NLP (Seo et al., 2015; Lin and Parikh, 2015) have also studied visual reasoning. Our work is different from them as we use synthetic language instead of natural language. Our synthetic language is easier to parse, allowing our evaluation to mainly reflect the performance of geometric reasoning. On the other hand, while our method and conclusions can potentially apply to natural text, this remains to be validated and involves nontrivial future work. There are other differences to prior works as well. Specifically, (Seo et al., 2015) combined information from textual questions and diagrams to build a model for solving SAT geometry questions. However, our task is different as diagrams are not provided as part of the input, but are generated from the words/symbols themselves. Also, (Lin and Parikh, 2015) take advantage of synthetic images to gather semantic common sense knowledge (visual common sense) and use it to perform fill-in-the-blank (FITB) and visual paraphrasing tasks. Similar to us, they also form 'mental images'. However, there are two differences (apart from natural vs synthetic language): first, their benchmark tests higher level semantic knowledge (like "Mike is having lunch when he sees a bear." $\Longrightarrow$ "Mike tries to hide."), while ours is more focused on geometric reasoning. Second, their model is based on hand-crafted features while we use a DNN.

Spatial language for Human-Robot Interaction: Our work is also related to prior work on making robots understand spatial commands (e.g. “put that box here”, “move closer to the box”) and complete tasks such as navigation and assembly. Earlier work (Müller et al., 2000; Gribble et al., 1998; Zelek, 1997) in this domain used template-based commands, whereas more recent work (Skubic et al., 2004) tried to make the commands more natural. This line of work differs from ours in that the robot has visual perception of its environment that allows grounding of the textual commands, whereas in our case the agent has no visual perception, and an environment needs to be imagined.

Image Generation: Our work is related to image generation using DNNs which has a large body of literature, with diverse approaches (Reed et al., 2016; Gregor et al., 2015). We also generate an image from the input. But in our task, image generation is in the service of reasoning rather than an end goal in itself—as a result, photorealism or artistic style of generated images is irrelevant and not considered.

Visual Question Answering: Our work is also related to visual QA (VQA) (Johnson et al., 2016; Antol et al., 2015; Lu et al., 2016). Our task is different from VQA because our questions are in terms of words/symbols whereas in VQA the questions are visual, consisting of both text descriptions and images. The images involved in our task are internal and virtual, and are not part of the input or output.

Memory and Attention: Memory and attention have been increasingly incorporated into DNNs, especially for tasks involving algorithmic inference and/or natural language (Graves et al., 2014; Vaswani et al., 2017). For QA tasks, memory and attention play an important role in state-of-the-art (SOTA) approaches. (Sukhbaatar et al., 2015) introduced End-To-End Memory Network (MemN2N), a DNN with memory and recurrent attention mechanism, which can be trained end-to-end for diverse tasks like textual QA and language modeling. Concurrently, (Kumar et al., 2016) introduced Dynamic Memory Network (DMN), which also uses attention and memory. (Xiong et al., 2016) proposed DMN+, with several improvements over the previous version of DMN and achieved SOTA results on VQA (Antol et al., 2015) and bAbI (Weston et al., 2015). Our proposed DSMN is a strict generalization of DMN+ (see Sec. 4.1). On removing the images and spatial memory from DSMN, it reduces to DMN+. Recently (Gupta et al., 2017) also used spatial memory in their deep learning system, but for visual navigation. We are using spatial memory for QA.

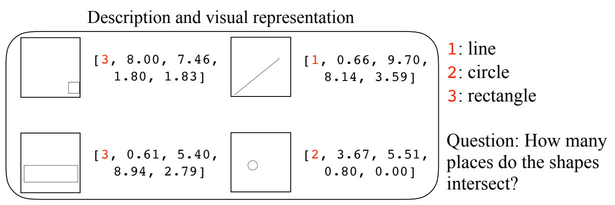


Figure 1: An example in the ShapeIntersection dataset.

3 Datasets

We introduce two synthetically-generated QA datasets to evaluate a system’s goemetrical reasoning ability: FloorPlanQA and ShapeIntersection. These datasets are not meant to test natural language understanding, but instead focus on geometrical reasoning. Owing to their synthetic nature, they are easy to parse, but nevertheless they are still challenging for DNNs like DMN+ (Xiong et al., 2016) and MemN2N (Sukhbaatar et al., 2015) that achieved SOTA results on existing QA datasets (see Table 2a).

The proposed datasets are similar in spirit to bAbI (Weston et al., 2015), which is also synthetic. In spite of its synthetic nature, bAbI has proved to be a crucial benchmark for the development of new models like MemN2N, DMN+, variants of which have proved successful in various natural domains (Kumar et al., 2016; Perez and Liu, 2016). Our proposed dataset is first to explicitly test ‘visual thinking’, and its synthetic nature helps us avoid the expensive and tedious task of collecting human annotations. Meanwhile, it is important to note that conclusions drawn from synthetic data do not automatically translate to natural data, and methods developed on synthetic benchmarks need additional validation on natural domains.

The proposed datasets also contain visual representations of the questions. Each of them has 38,400 questions, evenly split into a training set, a validation set and a test set (12,800 each).

Component	Template
House door	The house door is in the middle of the {*nr, sr, er, wr*} wall of the house. The house door is located in the {*n-er, s-er, n-wr, s-wr, n-er, s-er, n-wr, s-wr*} side of the house, such that it opens towards {*n, s, e, w*}.
Room door	The door for this room is in the middle of its {*nr, sr, er, wr*} wall. This room's door is in the middle of its {*nr, sr, er, wr*} wall. The door for this room is located in its {*n-er, s-er, n-wr, s-wr, n-er, s-er, n-wr, s-wr*} side, such that it opens towards {*n, s, e, w*}. This room's door is located in its {*n-er, s-er, n-wr, s-wr, n-er, s-er, n-wr, s-wr*} side, such that it opens towards {*n, s, e, w*}.
Small room	Room {1, 2, 3} is small in size and it is located in the {*n, s, e, w, c, n-e, s-e, n-w, s-w*} of the house. Room {1, 2, 3} is located in the {*n, s, e, w, c, n-e, s-e, n-w, s-w*} of the house and is small in size.
Medium room	Room {1, 2, 3} is medium in size and it extends from the {*n, s, e, w, c, n-e, s-e, n-w, s-w*} to the {*n, s, e, w, c, n-e, s-e, n-w, s-w*} of the house. Room {1, 2, 3} extends from the {*n, s, e, w, c, n-e, s-e, n-w, s-w*} to the {*n, s, e, w, c, n-e, s-e, n-w, s-w*} of the house and is medium in size.
Large room	Room {1, 2, 3} is large in size and it stretches along the {*n-s, e-w*}direction in the {*n, s, e, w, c*} of the house. Room {1, 2, 3} stretches along the {*n-s, e-w*} direction in the {*n, s, e, w, c*} of the house and is large in size.
Object	A {*cu, cd, sp, co*} is located in the middle of the {*nr, sr, er, wr*} part of the house. A {*cu, cd, sp, co*} is located in the {*n-er, s-er, n-wr, s-wr, n-er, s-er, n-wr, s-wr, cr*} part of the house. A {*cu, cd, sp, co*} is located in the middle of the {*nr, sr, er, wr*} part of this room. A {*cu, cd, sp, co*} is located in the {*n-er, s-er, n-wr, s-wr, n-er, s-er, n-wr, s-wr, cr*} part of this room.

Table 1: Templates used by the description generator for FloorPlanQA. For compactness we used the following notations, *n* - north, *s* - south, *e* - east, *w* - west, *c* - center, *nr* - northern, *sr* - southern, *er* - eastern, *wr* - western, *cr* - central, *cu* - cube, *cd* - cuboid, *sp* - sphere and *co* - cone.

FloorPlanQA: Each sample in FloorPlanQA involves the layout of a house that has multiple rooms (max 3). The rooms are either small, medium or large. All the rooms and the house have a door. Additionally, each room and empty-space in the house (i.e. the space in the house that is not part of any room) might also contain an object (either a cube, cuboid, sphere, or cone).

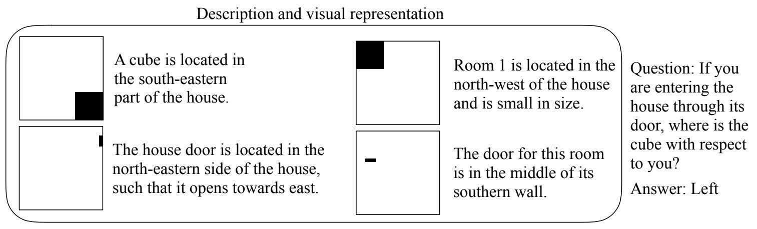
Each sample has four components, *a description, a question, an answer*, and *a visual representation*. Each sentence in the description describes either a room, a door or an object. A question is of the following template: *Suppose you are entering the* {*house, room 1, room 2, room 3*}, *where is the* {*house door, room 1 door, room 2 door, room 3 door, cube, cuboid, sphere, cone*} *with respect to you?*. The answer is either of *left, right, front,* or *back*. Other characteristics of FloorPlanQA are summarized in Fig. 2.

The visual representation of a sample consists of an ordered set of image channels, one per sentence in the description. An image channel pictorially represents the location and/or orientation of the described item (room, door, object) w.r.t. the house. An example is shown in Fig. 2.

To generate samples for FloorPlanQA, we define a probabilistic generative process which produces tree structures representing layouts of houses, similar to scene graphs used in computer graphics. The root node of a tree represents an entire house, and the leaf nodes represent rooms. We use a description and visual generator to produce respectively the description and visual representation from the tree structure. The templates used by the description generator are described in Table 1. Furthermore, the order of sentences in a description is randomized while making sure that the description still makes sense. For example, in some sample, the description of room 1 can appear before that of the house-door, while in another sample, it could be reversed. Similarly, for a room, the sentence describing the room's door could appear before or after the sentence describing the object in the room (if the room contains one). We perform rejection sampling to ensure that all the answers are equally likely, and thus removing bias.

ShapeIntersection: As the name suggests, ShapeIntersection is concerned with counting the number of intersection points between shapes. In this dataset, the description consists of symbols representing various shapes, and the question is always "how many points of intersection are there among these shapes?"

There are three types of shapes in ShapeIntersection: rectangles, circles, and lines. The description of shapes is provided in the form of a sequence of 1D vectors, each vector representing one shape. A vector in ShapeIntersection is analogous to a sentence in FloorPlanQA. Hence,


Description and visual representation
A cube is located in the south-eastern part of the house.
The house door is located in the north-eastern side of the house, such that it opens towards east.
Room 1 is located in the north-west of the house and is small in size.
The door for this room is in the middle of its southern wall.
Question: If you are entering the house through its door, where is the cube with respect to you?
Answer: Left

vocabulary size	66
# unique sentences	264
# unique descriptions	38093
# unique questions	32
# unique question-description pairs	38228
Avg. # words per sentence	15
Avg. # sentences per description	6.61

Figure 2: An example and characteristics of FloorPlanQA (when considering all the 38,400 samples i.e. training, validation and test sets combined).

for ShapeIntersection, the term 'sentence' actually refers to a vector. Each sentence describing a shape consists of 5 real numbers. The first number stands for the type of shape: 1 - line, 2 - circle, and 3 - rectangle. The subsequent four numbers specify the size and location of the shape. For example, in case of a rectangle, they represent its height, its width, and coordinates of its bottom-left corner. Note that one can also describe the shapes using a sentence, e.g. "there is a rectangle at (5, 5), with a height of 2 cm and width of 8 cm." However, as our focus is to evaluate 'visual thinking', we work directly with the symbolic encoding.

In a given description, there are 6.5 shapes on average, and at most 6 lines, 3 rectangles and 3 circles. All the shapes in the dataset are unique and lie on a 10×10 canvas. While generating the dataset, we do rejection sampling to ensure that the number of intersections is uniformly distributed from 0 to the maximum possible number of intersections, regardless of the number of lines, rectangles, and circles. This ensures that the number of intersections cannot be estimated from the number of lines, circles or rectangles.

Similar to FloorPlanQA, the visual representation for a sample in this dataset is an ordered set of image channels. Each channel is associated with a sentence, and it plots the described shape. An example is shown in Figure 1.

4 Dynamic Spatial Memory Network

We propose Dynamic Spatial Memory Network (DSMN), a novel DNN designed for QA with geometric reasoning. What differentiates DSMN from other QA DNNs is that it forms an internal visual representation of the input. It then uses a spatial memory to reason over this visual representation.

A DSMN can be divided into five modules: *the input module, visual representation module, question module, spatial memory module*, and *answer module*. The input module generates an embedding for each sentence in the description. The visual representation module uses these embeddings to produce an intermediate visual representation for each sentence. In parallel, the question module produces an embedding for the question. The spatial memory module then goes over the question embedding, the sentence embeddings, and the visual representation multiple times to update the spatial memory. Finally, the answer module uses the spatial memory to output the answer. Fig. 3 illustrates the overall architecture of DSMN.

Input Module: This module produces an embedding for each sentence in the description. It is therefore customized based on how the descriptions are provided in a dataset. Since the descriptions are in words for FloorPlanQA, a position encoding (PE) layer is used to produce the initial sentence embeddings. This is done to ensure a fair comparison with DMN+ (Xiong et al., 2016) and MemN2N (Sukhbaatar et al., 2015), which also use a PE layer. A PE layer combines the word-embeddings to encode the position of words in a sentence (Please see (Sukhbaatar et al., 2015) for more information). For ShapeIntersection, the description is given as a sequence of vectors. Therefore, two FC layers (with ReLU in between) are used to obtain the initial sentence embeddings.

These initial sentence embeddings are then fed into a bidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014) to propagate the information across sentences. Let $\overrightarrow{s_i}$ and $\overleftarrow{s_i}$ be the respective output of the forward and backward GRU at i^{th} step. Then, the final sentence embedding for the i^{th} sentence is given by $s_i = \overrightarrow{s_i} + \overleftarrow{s_i}$.

Question Module: This module produces an embedding for the question. It is also customized to the dataset. For FloorPlanQA, the embeddings of the words in the question are fed to a GRU, and the final hidden state of the GRU is used as the question embedding. For ShapeIntersection, the question is always fixed, so we use an all-zero vector as the question embedding.

Visual Representation Module: This module generates a visual representation for each sentence in the description. It consists of two subcomponents: an attention network and an encoder-decoder network. The attention network gathers information from previous sentences that is important to produce the visual representation for the current sentence. For example, suppose the current sentence describes the location of an object with respect to a room. Then in order to infer the location of the object with respect to the house, one needs the location of the room with respect to the house, which is described in some previous sentence.

The encoder-decoder network encodes the visual information gathered by the attention network, combines it with the current sentence embedding, and decodes the visual representation of the current sentence. An encoder ($En(.)$) takes an image as input and produces an embedding, while a decoder ($De(.)$) takes an embedding as input and produces an image. An encoder is composed of series of convolution layers and a decoder is composed of series of deconvolution layers.

Suppose we are currently processing the sentence $\boldsymbol{s_t}$. This means we have already processed the sentences $\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_{t-1}}$ and produced the corresponding visual representations $\boldsymbol{S_1}, \boldsymbol{S_2}, \ldots, \boldsymbol{S_{t-1}}$. We also add $\boldsymbol{s_0}$ and $\boldsymbol{S_0}$, which are all-zero vectors to represent the null sentence. The attention network produces a scalar attention weight a_i for the i^{th} sentence which is given by $a_i = \text{Softmax}(\boldsymbol{w_s}^t \boldsymbol{z_i} + b_s)$ where $\boldsymbol{z_i} = [|\boldsymbol{s_i} - \boldsymbol{s_t}|; \boldsymbol{s_i} \circ \boldsymbol{s_t}]$. Here, $\boldsymbol{w_s}$ is a vector, b_s is a scalar, $\circ$ represents element-wise multiplication, $|.|$ represents element-wise absolute value, and $[\boldsymbol{v1}; \boldsymbol{v2}]$ represents the concatenation of vectors $\boldsymbol{v1}$ and $\boldsymbol{v2}$.

The gathered visual information is $\bar{\boldsymbol{S_t}} = \sum_{i=0}^{t-1} a_i \boldsymbol{S_i}$. It is fed into the encoder-decoder network. The visual representation for $\boldsymbol{s_t}$ is given by $\boldsymbol{S_t} = De_s\Big([\boldsymbol{s_t}; En_s(\bar{\boldsymbol{S_t}})]\Big)$. The parameters of $En_s(.)$, $De_s()$, $\boldsymbol{w_s}$, and b_s are shared across multiple iterations.

In the proposed model, we make the simplifying assumption that the visual representation of the current sentence does not depend on future sentences. In other words, it can be completely determined from the previous sentences in the description. Both FloorPlanQA and ShapeIntersection satisfy this assumption.

Spatial Memory Module: This module gathers relevant information from the description and updates memory accordingly. Similar to DMN+ and MemN2N, it collects information and updates memory multiple times to perform transitive reasoning. One iteration of information collection and memory update is referred as a 'hop'.

The memory consists of two components: a 2D spatial memory and a tag vector. The 2D spatial memory can be thought of as a visual scratch pad on which the network 'sketches' out the visual information. The tag vector is meant to represent what is 'sketched' on the 2D spatial memory. For example, the network can sketch the location of room 1 on its 2D spatial memory, and store the fact that it has sketched room 1 in the tag vector.

As mentioned earlier, each step of the spatial memory module involves gathering of relevant information and updating of memory. Suppose we are in step t. Let $\boldsymbol{M^{(t-1)}}$ represent the 2D spatial memory and $\boldsymbol{m^{(t-1)}}$ represent the tag vector after step $t-1$. The network gathers the relevant information by calculating the attention value for each sentence based on the question and the current memory. For sentence $\boldsymbol{s_i}$, the scalar attention value $g_i^{(t)}$ equal to $\text{Softmax}(\boldsymbol{w_y^t p_i^{(t)}} + b_y)$, where $\boldsymbol{p_i^{(t)}}$ is given as

$$\begin{aligned} \boldsymbol{p_i^{(t)}} = \big[&|\boldsymbol{m^{(t-1)}} - \boldsymbol{s_i}|;\ \boldsymbol{m^{(t-1)}} \circ \boldsymbol{s_i};\ |\boldsymbol{q} - \boldsymbol{s_i}|; \\ &\boldsymbol{q} \circ \boldsymbol{s_i};\ En_{p_1}^{(t)}(|\boldsymbol{M^{(t-1)}} - \boldsymbol{S_i}|); \\ &En_{p_2}^{(t)}(\boldsymbol{M^{(t-1)}} \circ \boldsymbol{S_i})\big] \end{aligned} \quad (1)$$

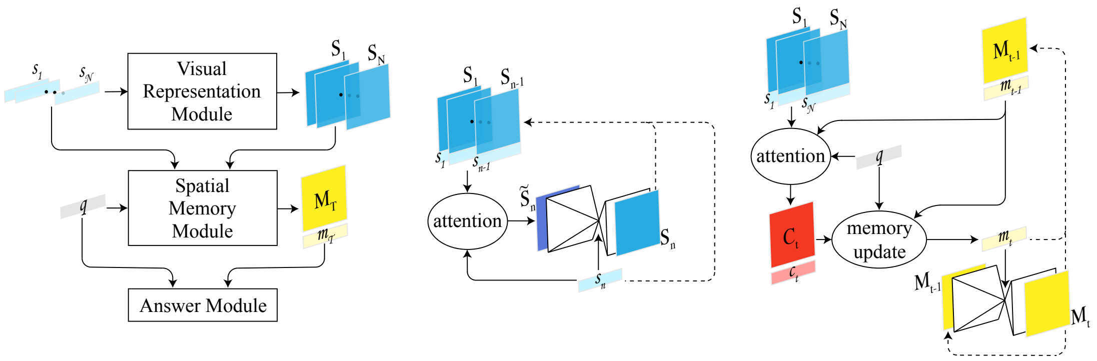
$\boldsymbol{M^{(0)}}$ and $\boldsymbol{m^{(0)}}$ represent initial blank memory, and their elements are all zero. Then, gathered information is represented as a context tag vector, $\boldsymbol{c^{(t)}} = \text{AttGRU}(g_i{}^{(t)}\boldsymbol{s_i})$ and 2D context, $\boldsymbol{C^{(t)}} = \sum_{i=0}^{n} g_i{}^{(t)} \boldsymbol{S_i}$. Please refer to (Xiong et al., 2016) for information about AttGRU(.). Finally, we use the 2D context and context tag vector to update the memory as follows:

$$\begin{aligned} \boldsymbol{m^{(t)}} = \text{ReLU}\Big(&\boldsymbol{W}_m{}^{(t)}\big[\boldsymbol{m^{(t-1)}};\ \boldsymbol{q};\ \boldsymbol{c^{(t)}}; \\ &En_c(\boldsymbol{C^{(t)}})\big] + \boldsymbol{b_m}{}^{(t)}\Big) \end{aligned} \quad (2)$$

$$\boldsymbol{M^{(t)}} = De_m^{(t)}\Big(\big[\boldsymbol{m^{(t)}};\ En_m^{(t)}(\boldsymbol{M^{(t-1)}})\big]\Big) \quad (3)$$

Answer Module: This module uses the final memory and question embedding to generate the output. The feature vector used for predicting the answer is given by $\boldsymbol{f}$, where $\boldsymbol{M^{(T)}}$ and $\boldsymbol{m^{(T)}}$ represent the final memory.

$$\boldsymbol{f} = [En_f(\boldsymbol{M^{(T)}});\ \boldsymbol{m^{(T)}};\ \boldsymbol{q}] \quad (4)$$



(a) Overall architecture (b) Visual represenation module (c) Spatial memory module

Figure 3: The architecture of the proposed Dynamic Spatial Memory Network (DSMN).

To obtain the output, an FC layer is applied to $\boldsymbol{f}$ in case of regression, while the FC layer is followed by softmax in case of classification. To keep DSMN similar to DMN+, we apply a dropout layer on sentence encodings ($\boldsymbol{s_i}$) and $\boldsymbol{f}$.

4.1 DSMN as a strict generalization of DMN

DSMN is a strict generalization of a DMN+. If we remove the visual representation of the input along with the 2D spatial memory, and just use vector representations with memory tags, then a DSMN reduces to DMN+. This ensures that comparison with DMN+ is fair.

4.2 DSMN with or without intermediate visual supervision

As described in previous sections, a DSMN forms an intermediate visual representation of the input. Therefore, if we have a 'ground-truth' visual representation for the training data, we could use it to train our network better. This leads to two different ways for training a DSMN, one with intermediate visual supervision and one without it. Without intermediate visual supervision, we train the network in an end-to-end fashion by using a loss (L_{w/o_vi}) that compares the predicted answer with the ground truth. With intermediate visual supervision, we train our network using an additional visual representation loss (L_{vi}) that measures how close the generated visual representation is to the ground-truth representation. Thus, the loss used for training with intermediate supervision is given by $L_{w\_vi} = \lambda_{vi} L_{vi} + (1 - \lambda_{vi}) L_{w/o\_vi}$, where λ_{vi} is a hyperparameter which can be tuned for each dataset. Note that in neither case do we need any visual input once the network is trained. During testing, the only input to the network is the description and question.

Also note that we can provide intermediate visual supervision to DSMN even when the visual representations for only a portion of samples in the training data are available. This can be useful when obtaining visual representation is expensive and time-consuming.

5 Experiments

Baselines: LSTM (Hochreiter and Schmidhuber, 1997) is a popular neural network for sequence processing tasks. We use two versions of LSTM-based baselines. LSTM-1 is a common version that is used as a baseline for textual QA (Sukhbaatar et al., 2015; Graves et al., 2016). In LSTM-1, we concatenate all the sentences and the question to a single string. For FloorPlanQA, we do word embedding look-up, while for ShapeIntersection, we project each real number into higher dimension via a series of FC layers. The sequence of vectors is fed into an LSTM. The final output vector of the LSTM is then used for prediction.

We develop another version of LSTM that we call LSTM-2, in which the question is concatenated to the description. We use a two-level hierarchy to embed the description. We first extract an embedding for each sentence. For FloorPlanQA, we use an LSTM to get the sentence embeddings, and for ShapeIntersection, we use a series of FC layers. We then feed the sentence embeddings into an LSTM, whose output is used for prediction.

Further, we compare our model to DMN+ (Xiong et al., 2016) and MemN2N (Sukhbaatar et al., 2015), which achieved state-of-the-art results on bAbI (Weston et al., 2015). In particular, we compare the 3-hop versions of DSMN, DMN+, and MemN2N.

Training Details: We used ADAM (Kingma and Ba, 2014) to train all models, and the learning rate

MODEL	FloorPlanQA (accuracy in %)	ShapeIntersection (rmse)
LSTM-1	41.36	3.28
LSTM-2	50.69	2.99
MemN2N	45.92	3.51
DMN+	60.29	2.98
DSMN	68.01	2.84
DSMN*	**97.73**	**2.14**

(a) The test set performance of different models on FloorPlanQA and ShapeIntersection. DSMN* refers to the model with intermediate supervision.

MODEL	$\boldsymbol{f}$ in Eqn. 4	FloorPlanQA (accuracy in %)
DSMN	$[\boldsymbol{m}^{(\boldsymbol{T})}; \boldsymbol{q}]$	67.65
DSMN	$[En_f(\boldsymbol{M}^{(\boldsymbol{T})}); \boldsymbol{q}]$	43.90
DSMN	$[En_f(\boldsymbol{M}^{(\boldsymbol{T})}); \boldsymbol{m}^{(\boldsymbol{T})}; \boldsymbol{q}]$	68.12
DSMN*	$[\boldsymbol{m}^{(\boldsymbol{T})}; \boldsymbol{q}]$	97.24
DSMN*	$[En_f(\boldsymbol{M}^{(\boldsymbol{T})}); \boldsymbol{q}]$	95.17
DSMN*	$[En_f(\boldsymbol{M}^{(\boldsymbol{T})}); \boldsymbol{m}^{(\boldsymbol{T})}; \boldsymbol{q}]$	98.08

(b) The validation set performances for the ablation study on the usefulness of tag ($\boldsymbol{m}^{(\boldsymbol{T})}$) and 2D spatial memory ($\boldsymbol{M}^{(\boldsymbol{T})}$) in the answer feature vector for $\boldsymbol{f}$.

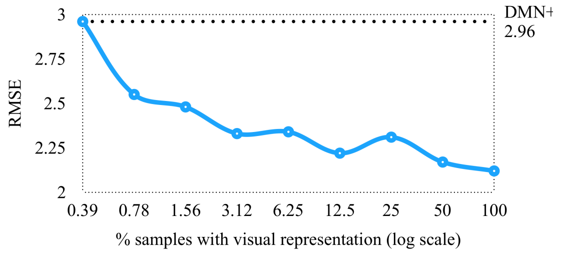
MODEL	FloorPlanQA (accuracy in %)
1-Hop DSMN	63.32
2-Hop DSMN	65.59
3-Hop DSMN	68.12
1-Hop DSMN*	90.09
2-Hop DSMN*	97.45
3-Hop DSMN*	98.08

(c) The validation set performance for the ablation study on variation in performance with hops.

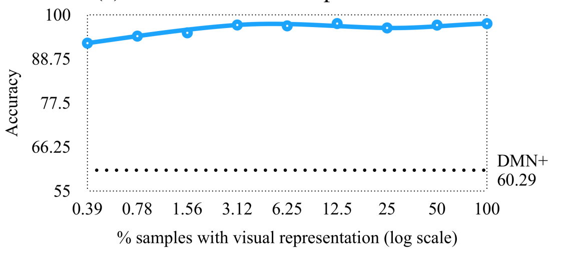
Table 2: Experimental results showing comparison with baselines, and ablation study of DSMN

for each model is tuned for each dataset. We tune the embedding size and l_2 regularization weight for each model and dataset pair separately. For reproducibility, the value of the best-tuned hyperparameters is mentioned in the supplementary material. As reported by (Sukhbaatar et al., 2015; Kumar et al., 2016; Henaff et al., 2016), we also observe that the results of memory networks are unstable across multiple runs. Therefore for each hyperparameter choice, we run all the models 10 times and select the run with the best performance on the validation set. For FloorPlanQA, all models are trained up to a maximum of 1600 epochs, with early stopping after 80 epochs if the validation accuracy did not increase. The maximum number of epochs for ShapeIntersection is 800 epochs, with early stopping after 80 epochs. Additionally, we modify the input module and question module of DMN+ and MemN2N to be same as ours for the ShapeIntersection dataset.

For MemN2N, we use the publicly available im-



(a) Test set rmse on ShapeIntersection.



(b) Test set accuracy on FloorPlanQA.

Figure 4: Performance of DSMN* with varying percentage of intermediate visual supervision.

plementation[2] and train it exactly as all other models (same optimizer, total epochs, and early stopping criteria) for fairness. While the reported best result for MemN2N is on the version with position encoding, linear start training, and random-injection of time index noise (Sukhbaatar et al., 2015), the version we use has only position encoding. Note that the comparison is still meaningful because linear start training and time index noise are not used in DMN+ (and as a result, neither in our proposed DSMN).

Results: The results for FloorPlanQA and ShapeIntersection are summarized in Table 2a. For brevity, we will refer to the DSMN model trained without intermediate visual supervision as DSMN, and the one with intermediate visual supervision as DSMN*. We see that DSMN (i.e the one without intermediate supervision) outperforms DMN+, MemN2N and the LSTM baselines on both datasets. However, we consider DSMN to be only slightly better than DMN+ because both are observed to be unstable across multiple runs and so the gap between the two has a large variance. Finally, DSMN* outperforms all other approaches by a large margin on both datasets, which demonstrates the utility of visual supervision in proposed tasks. While the variation can be significant across runs, if we run each model 10 times and choose the best run, we observe consistent results. We visualized the intermediate visual representations, but when no visual supervision is pro-

[2]`https://github.com/domluna/memn2n`

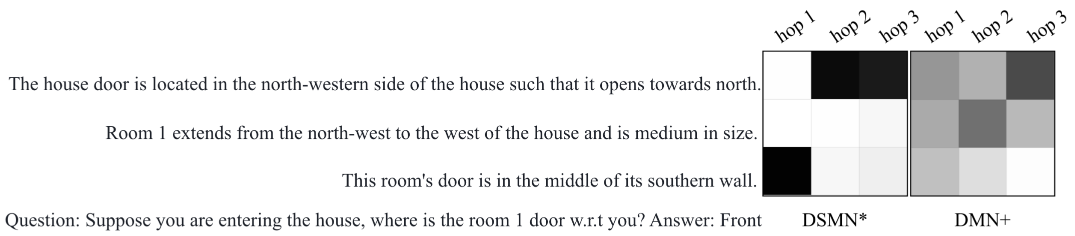


Figure 5: Attention values on each sentence during different memory 'hops' for a sample from FloorPlanQA. Darker color indicates more attention. To answer, one needs the location of room 1's door and the house door. To infer the location of room 1's door, DSMN* directly jumps to sent. 3. Since DMN+ does not form a visual representation, it tries to infer the location of room 1's door w.r.t the house by finding the location of the room's door w.r.t the room (sent. 3) and the location of the room w.r.t the house (sent. 2). Both DSMN* and DMN+ use one hop to infer the location of the house door (sent. 1).

vided, they were not interpretable (sometimes they looked like random noise, sometimes blank). In the case when visual supervision is provided, the intermediate visual representation is well-formed and similar to the ground-truth.

We further investigate how DSMN* performs when intermediate visual supervision is available for only a portion of training samples. As shown in Fig. 4, DSMN* outperforms DMN+ by a large margin, even when intermediate visual supervision is provided for only 1% of the training samples. This can be useful when obtaining visual representations is expensive and time-consuming. One possible justification for why visual supervision (even in a small amount) helps a lot is that it constrains the high-dimensional space of possible intermediate visual representations. With limited data and no explicit supervision, automatically learning these high-dimensional representations can be difficult.

Additonally, we performed ablation study (see Table 2b) on the usefulness of final memory tag vector ($\boldsymbol{m^{(T)}}$) and 2D spatial memory ($\boldsymbol{M^{(T)}}$) in the answer feature vector $\boldsymbol{f}$ (see Eqn. 4). We removed each of them one at a time, and retrained (with hyperparameter tuning) the DSMN and DSMN* models. Note that they are removed only from the final feature vector $\boldsymbol{f}$, and both of them are still coupled. The model with both tag and 2D spatial memory ($\boldsymbol{f} = \big[En_f(\boldsymbol{M^{(T)}}); \boldsymbol{m^{(T)}}; \boldsymbol{q}\big]$) performs slightly better than the only tag vector model ($\boldsymbol{f} = \big[\boldsymbol{m^{(T)}}; \boldsymbol{q}\big]$). Also, as expected the only 2D spatial memory model ($\boldsymbol{f} = \big[En_f(\boldsymbol{M^{(T)}}); \boldsymbol{q}\big]$) performs much better for DSMN* than DSMN becuase of the intermdiate supervision.

Further, Table 2c shows the effect of varying the number of memory 'hops' for DSMN and DSMN* on FloorPlanQA. The performance of both DSMN and DSMN* increases with the number of 'hops'. Note that even the 1-hop DSMN* performs well (better than baselines). Also, note that the difference in performance between 2-hop DSMN* and 3-hop DSMN* is not much. A possible justification for why DSMN* performs well even with fewer memory 'hops' is that DSMN* completes some 'hops of reasoning' in the visual representation module itself. Suppose one needs to find the location of an object placed in a room, w.r.t. the house. To do so, one first needs to find the location of the room w.r.t. the house, and then the location of the object w.r.t. the room. However, if one has already 'sketched' out the location of the object in the house, one can directly fetch it. It is during sketching the object's location that one has completed a 'hop of reasoning'. For a sample from FloorPlanQA, we visualize the attention maps in the memory module of 3-hop DMN+ and 3-hop DSMN* in Fig. 5. To infer the location of room 1's door, DSMN* directly fetches sentence 3, while DMN+ tries to do so by fetching two sentences (one for the room's door location w.r.t the room and one for the room's location w.r.t the house).

Conclusion: We have investigated how to use DNNs for modeling visual thinking. We have introduced two synthetic QA datasets, FloorPlanQA and ShapeIntersection, that test a system's ability to think visually. We have developed DSMN, a novel DNN that reasons in the visual space for answering questions. Experimental results have demonstrated the effectiveness of DSMN for geometric reasoning on synthetic data.

Acknowledgements: This work is partially supported by the National Science Foundation under Grant No. 1633157.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *ICCV*, pages 2425–2433.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Melvin Fitting. 2012. *First-order logic and automated theorem proving*. Springer Science & Business Media.

Kenneth D Forbus, Paul Nielsen, and Boi Faltings. 1991. Qualitative spatial reasoning: The clock project. *Artificial Intelligence*, 51(1-3):417–471.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, pages 471–476.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

William S Gribble, Robert L Browning, Micheal Hewett, Emilio Remolina, and Benjamin J Kuipers. 1998. Integrating vision and spatial reasoning for assistive navigation. In *Assistive Technology and artificial intelligence*, pages 179–193.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. 2017. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, pages 1735–1780.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2016. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *arXiv preprint arXiv:1612.06890*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, pages 1378–1387.

Barbara Landau and Ray Jackendoff. 1993. Whence and whither in spatial language and spatial cognition? *Behavioral and brain sciences*, 16:255–265.

Scott D Lathrop and John E Laird. 2007. Towards incorporating visual imagery into a cognitive architecture. In *International conference on cognitive modeling*, page 25.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI Spring Symposium*, volume 46, page 47.

Xiao Lin and Devi Parikh. 2015. Don't just listen, use your imagination: Leveraging visual common sense for non-visual tasks. In *ICCV*, pages 2984–2993.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*, pages 289–297.

Rolf Müller, Thomas Röfer, Axel Lankenau, Alexandra Musto, Klaus Stein, and Andreas Eisenkolb. 2000. Coarse qualitative descriptions in robot navigation. In *Spatial Cognition II*, pages 265–276.

Julien Perez and Fei Liu. 2016. Dialog state tracking, a machine reading approach using memory network. *arXiv preprint arXiv:1606.04052*.

Steven Pinker. 2003. *The language instinct: How the mind creates language*. Penguin UK.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 3, page 4.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*, pages 1466–1476.

A Rashid BM Shariff. 1998. Natural-language spatial relations between linear and areal objects: the topology and metric of english-language terms. *International journal of geographical information science*, 12:215–245.

Marjorie Skubic, Dennis Perzanowski, Samuel Blisard, Alan Schultz, William Adams, Magda Bugajska, and Derek Brock. 2004. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, pages 154–167.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*, pages 2440–2448.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*, pages 2397–2406.

John S Zelek. 1997. Human-robot interaction with minimal spanning natural language template for autonomous and tele-operated control. In *IROS*, pages 299–305.

Interactive Language Acquisition with One-shot Visual Concept Learning through a Conversational Game

Haichao Zhang[†], **Haonan Yu**[†], **and Wei Xu** [†§]
[†] Baidu Research - Institue of Deep Learning, Sunnyvale USA
[§] National Engineering Laboratory for Deep Learning Technology and Applications, Beijing China
{zhanghaichao,haonanyu,wei.xu}@baidu.com

Abstract

Building intelligent agents that can communicate with and learn from humans in natural language is of great value. Supervised language learning is limited by the ability of capturing mainly the statistics of training data, and is hardly adaptive to new scenarios or flexible for acquiring new knowledge without inefficient retraining or catastrophic forgetting. We highlight the perspective that conversational interaction serves as a natural interface both for language learning and for novel knowledge acquisition and propose a joint imitation and reinforcement approach for grounded language learning through an interactive conversational game. The agent trained with this approach is able to actively acquire information by asking questions about novel objects and use the just-learned knowledge in subsequent conversations in a one-shot fashion. Results compared with other methods verified the effectiveness of the proposed approach.

1 Introduction

Language is one of the most natural forms of communication for human and is typically viewed as fundamental to human intelligence; therefore it is crucial for an intelligent agent to be able to use language to communicate with human as well. While supervised training with deep neural networks has led to encouraging progress in language learning, it suffers from the problem of capturing mainly the statistics of training data, and from a lack of adaptiveness to new scenarios and being flexible for acquiring new knowledge without inefficient retraining or catastrophic forgetting. Moreover, supervised training of deep neural network models needs a large number of training samples while many interesting applications require rapid learning from a small amount of data, which poses an even greater challenge to the supervised setting.

In contrast, humans learn in a way very different from the supervised setting (Skinner, 1957; Kuhl, 2004). First, humans act upon the world and learn from the consequences of their actions (Skinner, 1957; Kuhl, 2004; Petursdottir and Mellor, 2016). While for mechanical actions such as movement, the consequences mainly follow geometrical and mechanical principles, for language, humans act by speaking, and the consequence is typically a response in the form of verbal and other behavioral feedback (*e.g.*, nodding) from the conversation partner (*i.e.*, teacher). These types of feedback typically contain informative signals on how to improve language skills in subsequent conversations and play an important role in humans' language acquisition process (Kuhl, 2004; Petursdottir and Mellor, 2016). Second, humans have shown a celebrated ability to learn new concepts from small amount of data (Borovsky et al., 2003). From even just one example, children seem to be able to make inferences and draw plausible boundaries between concepts, demonstrating the ability of one-shot learning (Lake et al., 2011).

The language acquisition process and the one-shot learning ability of human beings are both impressive as a manifestation of human intelligence, and are inspiring for designing novel settings and algorithms for computational language learning. In this paper, we leverage conversation as both an interactive environment for language learning (Skinner, 1957) and a natural interface for acquiring new knowledge (Baker et al., 2002). We propose an approach for interactive language acquisition with one-shot concept learning ability. The proposed approach allows an agent to learn grounded language from scratch, acquire the transferable skill of actively seeking and memorizing information about novel objects, and develop the one-shot learning ability, purely through conversational interaction with a teacher.

2 Related Work

Supervised Language Learning. Deep neural network-based language learning has seen great success on many applications, including machine translation (Cho et al., 2014b), dialogue generation (Wen et al., 2015; Serban et al., 2016), image captioning and visual question answering (**?**Antol et al., 2015). For training, a large amount of labeled data is needed, requiring significant efforts to collect. Moreover, this setting essentially captures the statistics of training data and does not respect the interactive nature of language learning, rendering it less flexible for acquiring new knowledge without retraining or forgetting (Stent and Bangalore, 2014).

Reinforcement Learning for Sequences. Some recent studies used reinforcement learning (RL) to tune the performance of a pre-trained language model according to certain metrics (Ranzato et al., 2016; Bahdanau et al., 2017; Li et al., 2016; Yu et al., 2017). Our work is also related to RL in natural language action space (He et al., 2016) and shares a similar motivation with Weston (2016) and Li et al. (2017), which explored language learning through pure textual dialogues. However, in these works (He et al., 2016; Weston, 2016; Li et al., 2017), a set of candidate sequences is provided and the action is to select one from the set. Our main focus is rather on learning language from scratch: the agent has to learn to *generate* a sequence action rather than to simply *select* one from a provided candidate set.

Communication and Emergence of Language. Recent studies have examined learning to communicate (Foerster et al., 2016; Sukhbaatar et al., 2016) and invent language (Lazaridou et al., 2017; Mordatch and Abbeel, 2018). The emerged language needs to be interpreted by humans via post-processing (Mordatch and Abbeel, 2018). We, however, aim to achieve language learning from the dual perspectives of understanding and generation, and the speaking action of the agent is readily understandable without any post-processing. Some studies on language learning have used a guesser-responder setting in which the guesser tries to achieve the final goal (*e.g.*, classification) by collecting additional information through asking the responder questions (Strub et al., 2017; Das et al., 2017). These works try to optimize the question being asked to help the guesser achieve the final goal, while we focus on transferable speaking and one-shot ability.

One-shot Learning and Active Learning. One-shot learning has been investigated in some recent works (Lake et al., 2011; Santoro et al., 2016; Woodward and Finn, 2016). The memory-augmented network (Santoro et al., 2016) stores visual representations mixed with ground truth class labels in an external memory for one-shot learning. A class label is always provided following the presentation of an image; thus the agent receives information from the teacher in a passive way. Woodward and Finn (2016) present efforts toward active learning, using a vanilla recurrent neural network (RNN) without an external memory. Both lines of study focus on image classification only, meaning the class label is directly provided for memorization. In contrast, we target language and one-shot learning via conversational interaction, and the learner has to learn to extract important information from the teacher's sentences for memorization.

3 The Conversational Game

We construct a conversational game inspired by experiments on language development in infants from cognitive science (Waxman, 2004). The game is implemented with the XWORLD simulator (Yu et al., 2018; Zhang et al., 2017) and is publicly available online.[1] It provides an environment for the agent[2] to learn language and develop the one-shot learning ability. One-shot learning here means that during test sessions, no further training happens to the agent and it has to answer teacher's questions correctly about novel images of never-before-seen classes after being taught only once by the teacher, as illustrated in Figure 1. To succeed in this game, the agent has to learn to 1) speak by generating sentences, 2) extract and memorize useful information with only one exposure and use it in subsequent conversations, and 3) behave adaptively according to context and its own knowledge (*e.g.*, asking questions about unknown objects and answering questions about something known), all achieved through interacting with the

[1] `https://github.com/PaddlePaddle/XWorld`

[2] We use the term *agent* interchangeably with *learner*.

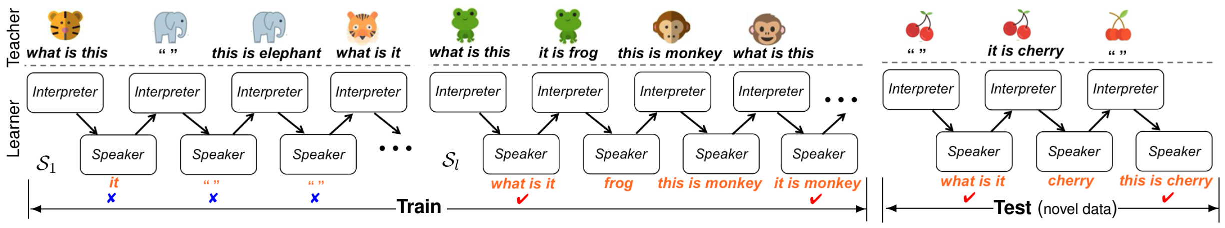


Figure 1: **Interactive language and one-shot concept learning.** Within a session $\mathcal{S}_l$, the teacher may ask questions, answer learner's questions, make statements, or say nothing. The teacher also provides reward feedback based on learner's responses as (dis-)encouragement. The learner alternates between interpreting teacher's sentences and generating a response through *interpreter* and *speaker*. **Left:** Initially, the learner can barely say anything meaningful. **Middle:** Later it can produce meaningful responses for interaction. **Right:** After training, when confronted with an image of *cherry*, which is a novel class that the learner never saw before during training, the learner can ask a question about it ("*what is it*") and generate a correct statement ("*this is cherry*") for another instance of cherry after only being taught once.

teacher. This makes our game distinct from other seemingly relevant games, in which the agent cannot speak (Wang et al., 2016) or "speaks" by *selecting* a candidate from a provided set (He et al., 2016; Weston, 2016; Li et al., 2017) rather than *generating* sentences by itself, or games mainly focus on slow learning (Das et al., 2017; Strub et al., 2017) and falls short on one-shot learning.

In this game, sessions ($\mathcal{S}_l$) are randomly instantiated during interaction. Testing sessions are constructed with a separate dataset with concepts that never appear before during training to evaluate the language and one-shot learning ability. Within a session, the teacher randomly selects an object and interacts with the learner about the object by randomly 1) posing a question (*e.g.*, "*what is this*"), 2) saying nothing (*i.e.*, "") or 3) making a statement (*e.g.*, "*this is monkey*"). When the teacher asks a question or says nothing, i) if the learner raises a question, the teacher will provide a statement about the object asked (*e.g.*, "*it is frog*") with a *question-asking reward* (+0.1); ii) if the learner says nothing, the teacher will still provide an answer (*e.g.*, "*this is elephant*") but with an *incorrect-reply reward* (−1) to discourage the learner from remaining silent; iii) for all other incorrect responses from the learner, the teacher will provide an *incorrect-reply reward* and move on to the next random object for interaction. When the teacher generates a statement, the learner will receive no reward if a correct statement is generated otherwise an *incorrect-reply reward* will be given. The session ends if the learner answers the teacher's question correctly, generates a correct statement when the teacher says nothing (receiving a *correct-answer reward* +1), or when the maximum number of steps is reached. The sentence from teacher at each time step is generated using a context-free grammar as shown in Table 1.

A success is reached if the learner behaves correctly during the whole session: asking questions about novel objects, generating answers when asked, and making statements when the teacher says nothing about objects that have been taught within the session. Otherwise it is a failure.

Table 1: Grammar for the teacher's sentences.

start	→	question | silence | statement
question	→	Q1 | Q2 | Q3
silence	→	" "
statement	→	A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8
Q1	→	"*what*"
Q2	→	"*what*" M
Q3	→	"*tell what*" N
M	→	"*is it*" | "*is this*" | "*is there*" | "*do you see*" | "*can you see*" | "*do you observe*" | "*can you observe*"
N	→	"*it is*" | "*this is*" | "*there is*" | "*you see*" | "*you can see*" | "*you observe*" | "*you can observe*"
A1	→	G
A2	→	"*it is*" G
A3	→	"*this is*" G
A4	→	"*there is*" G
A5	→	"*i see*" G
A6	→	"*i observe*" G
A7	→	"*i can see*" G
A8	→	"*i can observe*" G
G	→	object name

4 Interactive Language Acquisition via Joint Imitation and Reinforcement

Motivation. The goal is to *learn to converse and develop the one-shot learning ability by conversing with a teacher and improving from teacher's feedback*. We propose to use a *joint imitation and reinforce* approach to achieve this goal. *Imitation* helps the agent to develop the basic ability to generate sensible sentences. As learning is done by observing the teacher's behaviors during conversion, the agent essentially imitates the teacher from a *third-person perspective* (Stadie et al., 2017) rather than imitating an expert agent who is conversing with the teacher (Das et al., 2017; Strub et al., 2017). During conversations, the agent perceives sentences and images without any explicit labeling of ground truth answers, and it has to learn to make sense of raw perceptions, extract useful information, and save it for later use when generating an answer to teacher's question. While it is tempting to purely imitate the teacher, the agent trained this way only develops *echoic behavior* (Skinner, 1957), *i.e.*, mimicry. *Reinforce* leverages confirmative feedback from the teacher for learning to converse adaptively beyond mimicry by adjusting the action policy. It enables the learner to use the acquired speaking ability and adapt it according to reward feedback. This is analogous to some views on the babies' language-learning process that babies use the acquired speaking skills by trial and error with parents and improve according to the consequences of speaking actions (Skinner, 1957; Petursdottir and Mellor, 2016). The fact that babies don't fully develop the speaking capabilities without the ability to hear (Houston and Miyamoto, 2011), and that it is hard to make a meaningful conversation with a trained parrot signifies the importance of both imitation and reinforcement in language learning.

Formulation. The agent's response can be modeled as a sample from a probability distribution over the possible sequences. Specifically, for one session, given the visual input $\mathbf{v}^t$ and conversation history $\mathcal{H}^t = \{\mathbf{w}^1, \mathbf{a}^1, \cdots, \mathbf{w}^t\}$, the agent's response $\mathbf{a}^t$ can be generated by sampling from a distribution of the speaking action $\mathbf{a}^t \sim p_\theta^{\mathrm{S}}(\mathbf{a}|\mathcal{H}^t, \mathbf{v}^t)$. The agent interacts with the teacher by outputting the utterance $\mathbf{a}^t$ and receives feedback from the teacher in the next step, with $\mathbf{w}^{t+1}$ a sentence as verbal feedback and r^{t+1} reward feedback (with positive values as encouragement while negative values as discouragement, according to $\mathbf{a}^t$, as described in Section 3). Central to the goal is learning $p_\theta^{\mathrm{S}}(\cdot)$. We formulate the problem as the minimization of a cost function as:

$$\mathcal{L}_\theta = \underbrace{\mathbb{E}_{\mathcal{W}}\left[-\textstyle\sum_t \log p_\theta^{\mathrm{I}}(\mathbf{w}^t|\cdot)\right]}_{\text{Imitation } \mathcal{L}_\theta^{\mathrm{I}}} + \underbrace{\mathbb{E}_{p_\theta^{\mathrm{S}}}\left[-\textstyle\sum_t [\gamma]^{t-1} \cdot r^t\right]}_{\text{Reinforce } \mathcal{L}_\theta^{\mathrm{R}}}$$

where $\mathbb{E}_{\mathcal{W}}(\cdot)$ is the expectation over all the sentences $\mathcal{W}$ from teacher, γ is a reward discount factor, and $[\gamma]^t$ denotes the exponentiation over γ. While the imitation term learns directly the predictive distribution $p_\theta^{\mathrm{I}}(\mathbf{w}^t|\mathcal{H}^{t-1}, \mathbf{a}^t)$, it contributes to $p_\theta^{\mathrm{S}}(\cdot)$ through *parameter sharing* between them.

Architecture. The learner comprises four major components: *external memory*, *interpreter*, *speaker*, and *controller*, as shown in Figure 2. *External memory* is flexible for storing and retrieving information (Graves et al., 2014; Santoro et al., 2016), making it a natural component of our network for one-shot learning. The *interpreter* is responsible for interpreting the teacher's sentences, extracting information from the perceived signals, and saving it to the external memory. The *speaker* is in charge of generating sentence responses with reading access to the external memory. The response could be a question asking for information or a statement answering a teacher's question, leveraging the information stored in the external memory. The *controller* modulates the behavior of the speaker to generate responses according to context (*e.g.*, the learner's knowledge status).

At time step t, the *interpreter* uses an interpreter-RNN to encode the input sentence $\mathbf{w}^t$ from the teacher as well as historical conversational information into a state vector $\mathbf{h}_{\mathrm{I}}^t$. $\mathbf{h}_{\mathrm{I}}^t$ is then passed through a residue-structured network, which is an identity mapping augmented with a learnable controller $f(\cdot)$ implemented with fully connected layers for producing $\mathbf{c}^t$. Finally, $\mathbf{c}^t$ is used as the initial state of the speaker-RNN for generating the response $\mathbf{a}^t$. The final state $\mathbf{h}_{\mathrm{last}}^t$ of the speaker-RNN will be used as the initial state of the interpreter-RNN at the next time step.

4.1 Imitation with Memory Augmented Neural Network for Echoic Behavior

The teacher's way of speaking provides a source for the agent to imitate. For example, the syntax for composing a sentence is a useful skill the agent can learn from the teacher's sentences, which could benefit both *interpreter* and *speaker*. Imitation is achieved by predicting teacher's future sentences with *interpreter* and parameter sharing between *interpreter* and *speaker*. For prediction, we can represent the probability of the next sentence $\mathbf{w}^t$ conditioned on the image $\mathbf{v}^t$ as well as previous sentences from both the teacher and the

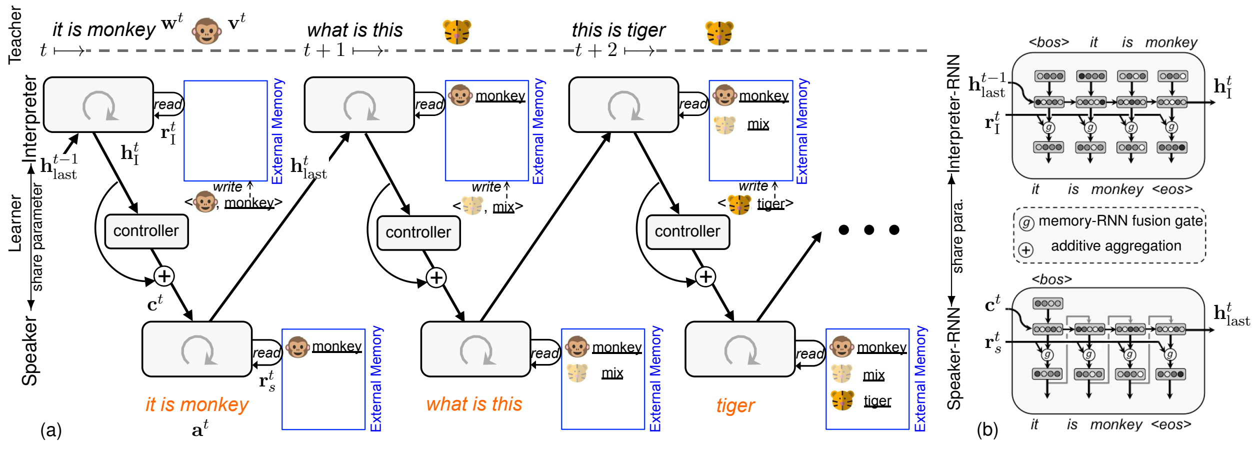


Figure 2: **Network structure.** (a) Illustration of the overall architecture. At each time step, the learner uses the interpreter module to encode the teacher's sentence. The visual perception is also encoded and used as a key to retrieve information from the external memory. The last state of the interpreter-RNN will be passed through a controller. The controller's output will be added to the input and used as the initial state of the speaker-RNN. The interpreter-RNN will update the external memory with an importance (illustrated with transparency) weighted information extracted from the perception input. 'Mix' denotes a mixture of word embedding vectors. (b) The structures of the interpreter-RNN (top) and the speaker-RNN (bottom). The interpreter-RNN and speaker-RNN share parameters.

learner $\{\mathbf{w}^1, \mathbf{a}^1, \cdots, \mathbf{w}^{t-1}, \mathbf{a}^{t-1}\}$ as

$$\begin{aligned} & p^{\mathrm{I}}_{\theta}(\mathbf{w}^t|\mathcal{H}^{t-1}, \mathbf{a}^{t-1}, \mathbf{v}^t) \\ & = \textstyle\prod_i p^{\mathrm{I}}_{\theta}(w^t_i|w^t_{1:i-1}, \mathbf{h}^{t-1}_{\text{last}}, \mathbf{v}^t), \end{aligned} \tag{1}$$

where $\mathbf{h}^{t-1}_{\text{last}}$ is the last state of the RNN at time step t–1 as the summarization of $\{\mathcal{H}^{t-1}, \mathbf{a}^{t-1}\}$ (*c.f.*, Figure 2), and i indexes words within a sentence.

It is natural to model the probability of the i-th word in the t-th sentence with an RNN, where the sentences up to t and words up to i within the t-th sentence are captured by a fixed-length state vector $\mathbf{h}^t_i = \text{RNN}(\mathbf{h}^t_{i-1}, w^t_i)$. To incorporate knowledge learned and stored in the external memory, the generation of the next word is *adaptively* based on i) the predictive distribution of the next word from the state of the RNN to capture the *syntactic structure of sentences*, and ii) the information from the external memory to represent the previously learned knowledge, via a fusion gate g:

$$p^{\mathrm{I}}_{\theta}(w^t_i|\mathbf{h}^t_i, \mathbf{v}^t) = (1-g)\cdot p_{\mathbf{h}} + g\cdot p_{\mathbf{r}}, \tag{2}$$

where $p_{\mathbf{h}} = \text{softmax}\big(\mathbf{E}^{\mathsf{T}} f_{\text{MLP}}(\mathbf{h}^t_i)\big)$ and $p_{\mathbf{r}} = \text{softmax}\big(\mathbf{E}^{\mathsf{T}}\mathbf{r}\big)$. $\mathbf{E}\in\mathbb{R}^{d\times k}$ is the word embedding table, with d the embedding dimension and k the vocabulary size. $\mathbf{r}$ is a vector read out from the external memory using a visual key as detailed in the next section. $f_{\text{MLP}}(\cdot)$ is a multi-layer Multi-Layer Perceptron (MLP) for bridging the semantic gap between the RNN state space and the word embedding space. The fusion gate g is computed as $g = f(\mathbf{h}^t_i, c)$, where c is the confidence score $c = \max(\mathbf{E}^{\mathsf{T}}\mathbf{r})$, and a well-learned concept should have a large score by design (Appendix A.2).

Multimodal Associative Memory. We use a multimodal memory for storing visual (v) and sentence (s) features with each modality while preserving the correspondence between them (Baddeley, 1992). Information organization is more structured than the single modality memory as used in Santoro et al. (2016) and cross modality retrieval is straightforward under this design. A visual encoder implemented as a convolutional neural network followed by fully connected layers is used to encode the visual image $\mathbf{v}$ into a visual key $\mathbf{k}_v$, and then the corresponding sentence feature can be retrieved from the memory as:

$$\mathbf{r} \leftarrow \textsc{Read}(\mathbf{k}_v, \mathbf{M}_v, \mathbf{M}_s). \tag{3}$$

$\mathbf{M}_v$ and $\mathbf{M}_s$ are memories for visual and sentence modalities with the same number of slots (columns). Memory read is implemented as $\mathbf{r} = \mathbf{M}_s\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}$ a soft reading weight obtained through the visual modality by calculating the cosine similarities between $\mathbf{k}_v$ and slots of $\mathbf{M}_v$.

Memory write is similar to Neural Turing Machine (Graves et al., 2014), but with a content importance gate g_{mem} to adaptively control whether the content $\mathbf{c}$ should be written into memory:

$$\mathbf{M}_m \leftarrow \textsc{Write}(\mathbf{M}_m, \mathbf{c}_m, g_{\text{mem}}),\ m\in\{v, s\}.$$

For the visual modality $\mathbf{c}_v \triangleq \mathbf{k}_v$. For the sentence modality, $\mathbf{c}_s$ has to be selectively extracted from the sentence generated by the teacher. We use an attention mechanism to achieve this by $\mathbf{c}_s = \mathbf{W}\boldsymbol{\eta}$, where $\mathbf{W}$ denotes the matrix with columns being the embedding vectors of all the words in the sentence. $\boldsymbol{\eta}$ is a normalized attention vector representing the relative importance of each word in the sentence as measured by the cosine similarity between the sentence representation vector and each word's context vector, computed using a bidirectional-RNN. The scalar-valued content importance gate g_{mem} is computed as a function of the sentence from the teacher, meaning that the importance of the content to be written into memory depends on the content itself (*c.f.*, Appendix A.3 for more details). The memory write is achieved with an erase and an add operation:

$$\tilde{\mathbf{M}}_m = \mathbf{M}_m - \mathbf{M}_m \odot (g_{\text{mem}} \cdot \mathbf{1} \cdot \boldsymbol{\beta}^{\mathsf{T}}),$$
$$\mathbf{M}_m = \tilde{\mathbf{M}}_m + g_{\text{mem}} \cdot \mathbf{c}_m \cdot \boldsymbol{\beta}^{\mathsf{T}}, \ m \in \{v, s\}.$$

$\odot$ denotes Hadamard product and the write location $\boldsymbol{\beta}$ is determined with a Least Recently Used Access mechanism (Santoro et al., 2016).

4.2 Context-adaptive Behavior Shaping through Reinforcement Learning

Imitation fosters the basic language ability for generating echoic behavior (Skinner, 1957), but it is not enough for conversing adaptively with the teacher according to context and the knowledge state of the learner. Thus we leverage reward feedback to shape the behavior of the agent by optimizing the policy using RL. The agent's response $\mathbf{a}^t$ is generated by the *speaker*, which can be modeled as a sample from a distribution over all possible sequences, given the conversation history $\mathcal{H}^t = \{\mathbf{w}^1, \mathbf{a}^1, \cdots, \mathbf{w}^t\}$ and visual input $\mathbf{v}^t$:

$$\mathbf{a}^t \sim p_\theta^{\mathrm{S}}(\mathbf{a}|\mathcal{H}^t, \mathbf{v}^t). \qquad (4)$$

As $\mathcal{H}^t$ can be encoded by the interpreter-RNN as $\mathbf{h}_{\mathrm{I}}^t$, the action policy can be represented as $p_\theta^{\mathrm{S}}(\mathbf{a}|\mathbf{h}_{\mathrm{I}}^t, \mathbf{v}^t)$. To leverage the language skill that is learned via imitation through the *interpreter*, we can generate the sentence by implementing the *speaker* with an RNN, sharing parameters with the interpreter-RNN, but with a conditional signal modulated by a controller network (Figure 2):

$$p_\theta^{\mathrm{S}}(\mathbf{a}^t|\mathbf{h}_{\mathrm{I}}^t, \mathbf{v}^t) = p_\theta^{\mathrm{I}}(\mathbf{a}^t|\mathbf{h}_{\mathrm{I}}^t + f(\mathbf{h}_{\mathrm{I}}^t, c), \mathbf{v}^t). \qquad (5)$$

The reason for using a controller $f(\cdot)$ for modulation is that the basic language model only offers the learner the echoic ability to generate a sentence, but not necessarily the adaptive behavior according to context (*e.g.* asking questions when facing novel objects and providing an answer for a previously learned object according to its own knowledge state). Without any additional module or learning signals, the agent's behaviors would be the same as those of the teacher because of parameter sharing; thus, it is difficult for the agent to learn to speak in an adaptive manner.

To learn from consequences of speaking actions, the policy $p_\theta^{\mathrm{S}}(\cdot)$ is adjusted by maximizing expected future reward as represented by $\mathcal{L}_\theta^{\mathrm{R}}$. As a non-differentiable sampling operation is involved in Eqn.(4), policy gradient theorem (Sutton and Barto, 1998) is used to derive the gradient for updating $p_\theta^{\mathrm{S}}(\cdot)$ in the reinforce module:

$$\nabla_\theta \mathcal{L}_\theta^{\mathrm{R}} = \mathbb{E}_{p_\theta^{\mathrm{S}}}\left[\textstyle\sum_t A^t \cdot \nabla_\theta \log p_\theta^{\mathrm{S}}(\mathbf{a}^t|\mathbf{c}^t)\right], \qquad (6)$$

where $A^t = V(\mathbf{h}_{\mathrm{I}}^t, c^t) - r^{t+1} - \gamma V(\mathbf{h}_{\mathrm{I}}^{t+1}, c^{t+1})$ is the advantage (Sutton and Barto, 1998) estimated using a value network $V(\cdot)$. The imitation module contributes by implementing $\mathcal{L}_\theta^{\mathrm{I}}$ with a cross-entropy loss (Ranzato et al., 2016) and minimizing it with respect to the parameters in $p_\theta^{\mathrm{I}}(\cdot)$, which are shared with $p_\theta^{\mathrm{S}}(\cdot)$. The training signal from imitation takes the shortcut connection without going through the controller. More details on $f(\cdot)$, $V(\cdot)$ are provided in Appendix A.2.

5 Experiments

We conduct experiments with comparison to baseline approaches. We first experiment with a word-level task in which the teacher and the learner communicate a single word each time. We then investigate the impact of image variations on concept learning. We further perform evaluation on the more challenging sentence-level task in which the teacher and the agent communicate in the form of sentences with varying lengths.

Setup. To evaluate the performance in learning a transferable ability, rather than the ability of fitting a particular dataset, we use an Animal dataset for training and test the trained models on a Fruit dataset (Figure 1). More details on the datasets are provided in Appendix A.1. Each session consists of two randomly sampled classes, and the maximum number of interaction steps is six.

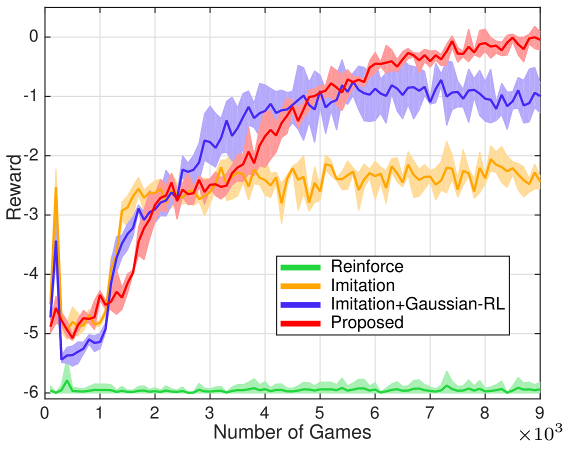


Figure 3: **Evolution of reward** during training for the word-level task without image variations.

Baselines. The following methods are compared:

- **Reinforce**: a baseline model with the same network structure as the proposed model and trained using RL only, *i.e.* minimizing $\mathcal{L}_{\theta}^{\mathrm{R}}$;
- **Imitation**: a recurrent encoder decoder (Serban et al., 2016) model with the same structure as ours and trained via imitation (minimizing $\mathcal{L}_{\theta}^{\mathrm{I}}$);
- **Imitation+Gaussian-RL**: a joint imitation and reinforcement method using a Gaussian policy (Duan et al., 2016) in the latent space of the control vector $\mathbf{c}^t$ (Zhang et al., 2017). The policy is changed by modifying the control vector $\mathbf{c}^t$ the action policy depends upon.

Training Details. The training algorithm is implemented with the deep learning platform PaddlePaddle.[3] The whole network is trained from scratch in an end-to-end fashion. The network is randomly initialized without any pre-training and is trained with decayed Adagrad (Duchi et al., 2011). We use a batch size of 16, a learning rate of 1×10^{-5} and a weight decay rate of 1.6×10^{-3}. We also exploit experience replay (Wang et al., 2017; Yu et al., 2018). The reward discount factor γ is 0.99, the word embedding dimension d is 1024 and the dictionary size k is 80. The visual image size is 32×32, the maximum length of generated sentence is 6 and the memory size is 10. Word embedding vectors are initialized as random vectors and remain fixed during training. A sampling operation is used for sentence generation during training for exploration while a max operation is used during testing both for **Proposed** and for **Reinforce** baseline. The max operation is used in both training and testing for **Imitation** and **Imitation+Gaussian-RL** baselines.

[3]https://github.com/PaddlePaddle/Paddle

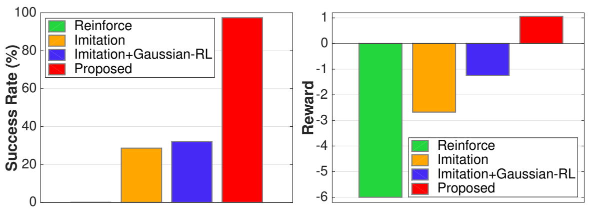


Figure 4: **Test performance** for the word-level task without image variations. Models are trained on the Animal dataset and tested on the Fruit dataset.

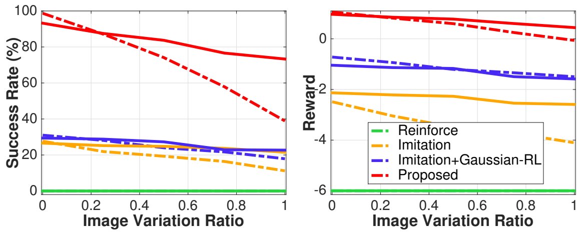


Figure 5: **Test success rate and reward** for the word-level task on the Fruit dataset under different test image variation ratios for models trained on the Animal dataset with a variation ratio of 0.5 (solid lines) and without variation (dashed lines).

5.1 Word-Level Task

In this experiment, we focus on a word-level task, which offers an opportunity to analyze and understand the underlying behavior of different algorithms while being free from distracting factors. Note that although the teacher speaks a word each time, the learner still has to learn to generate a full-sentence ended with an end-of-sentence symbol.

Figure 3 shows the evolution curves of the rewards during training for different approaches. It is observed that **Reinforce** makes very little progress, mainly due to the difficulty of exploration in the large space of sequence actions. **Imitation** obtains higher rewards than **Reinforce** during training, as it can avoid some penalty by generating sensible sentences such as questions. **Imitation+Gaussian-RL** gets higher rewards than both **Imitation** and **Reinforce**, indicating that the RL component reshapes the action policy toward higher rewards. However, as the Gaussian policy optimizes the action policy indirectly in a latent feature space, it is less efficient for exploration and learning. **Proposed** achieves the highest final reward during training.

We train the models using the Animal dataset and evaluate them on the Fruit dataset; Figure 4 sum-

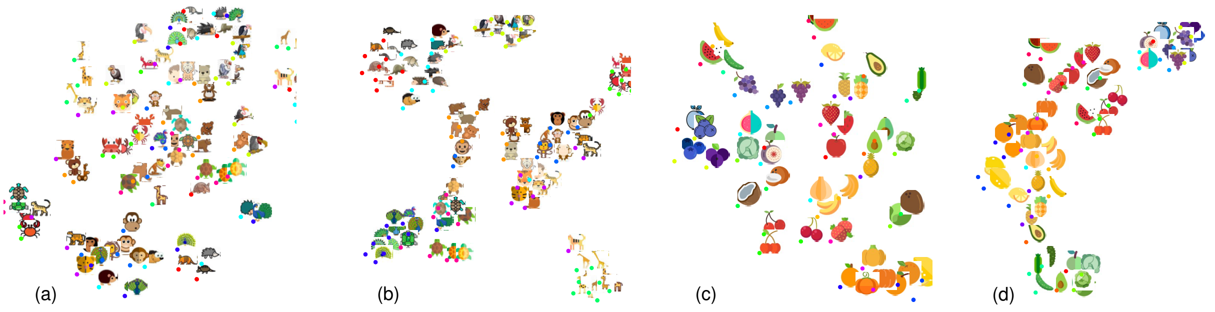


Figure 6: **Visualization of the CNN features** with t-SNE. Ten classes randomly sampled from **(a-b)** the Animal dataset and **(c-d)** the Fruit dataset, with features extracted using the visual encoder trained without (a, c) and with (b, d) image variations on the the Animal dataset.

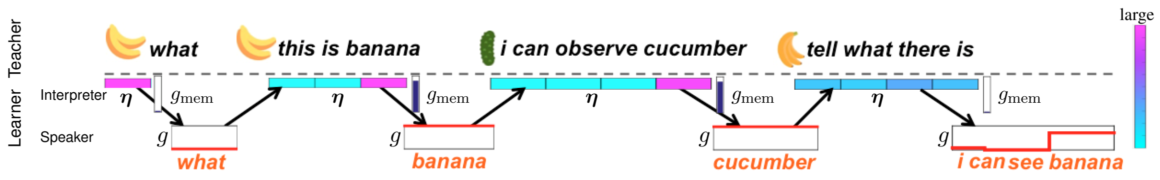


Figure 7: **Example results** of the proposed approach on novel classes. The learner can ask about the new class and use the interpreter to extract useful information from the teacher's sentence via word-level attention $\boldsymbol{\eta}$ and content importance g_{mem} jointly. The speaker uses the fusion gate g to adaptively switch between signals from RNN (small g) and external memory (large g) to generate sentence responses.

marizes the success rate and average reward over 1K testing sessions. As can be observed, **Reinforce** achieves the lowest success rate (0.0%) and reward (-6.0) due to its inherent inefficiency in learning. **Imitation** performs better than **Reinforce** in terms of both its success rate (28.6%) and reward value (-2.7). **Imitation+Gaussian-RL** achieves a higher reward (-1.2) during testing, but its success rate (32.1%) is similar to that of **Imitation**, mainly due to the rigorous criteria for success. **Proposed** reaches the highest success rate (97.4%) and average reward ($+1.1$)[4], outperforming all baseline methods by a large margin. From this experiment, it is clear that imitation with a proper usage of reinforcement is crucial for achieving adaptive behaviors (*e.g.*, asking questions about novel objects and generating answers or statements about learned objects proactively).

5.2 Learning with Image Variations

To evaluate the impact of within-class image variations on one-shot concept learning, we train models with and without image variations, and during testing compare their performance under different image variation ratios (the chance of a novel image instance being present within a session) as shown in Figure 5. It is observed that the performance of the model trained without image variations drops significantly as the variation ratio increases. We also evaluate the performance of models trained under a variation ratio of 0.5. Figure 5 clearly shows that although there is also a performance drop, which is expected, the performance degrades more gradually, indicating the importance of image variation for learning one-shot concepts. Figure 6 visualizes sampled training and testing images represented by their corresponding features extracted using the visual encoder trained without and with image variations. Clusters of visually similar concepts emerge in the feature space when trained with image variations, indicating that a more discriminative visual encoder was obtained for learning generalizable concepts.

5.3 Sentence-Level Task

We further evaluate the model on sentence-level tasks. Teacher's sentences are generated using the grammar as shown in Table 1 and have a number of variations with sentence lengths ranging from one to five. Example sentences from the teacher are presented in Appendix A.1. This task is more challenging than the word-level task in two ways: i) information processing is more difficult as the learner has to learn to extract useful information which could appear at different locations of the sentence; ii) the sentence generation is also more

[4]The testing reward is higher than the training reward mainly due to the action sampling in training for exploration.

difficult than the word-level task and the learner has to adaptively fuse information from RNN and external memory to generate a complete sentence. Comparison of different approaches in terms of their success rates and average rewards on the novel test set are shown in Figure 8. As can be observed from the figure, **Proposed** again outperforms all other compared methods in terms of both success rate (82.8%) and average reward (+0.8), demonstrating its effectiveness even for the more complex sentence-level task.

We also visualize the information extraction and the adaptive sentence composing process of the proposed approach when applied to a test set. As shown in Figure 7, the agent learns to extract useful information from the teacher's sentence and use the content importance gate to control what content is written into the external memory. Concretely, sentences containing object names have a larger g_{mem} value, and the word corresponding to object name has a larger value in the attention vector $\boldsymbol{\eta}$ compared to other words in the sentence. The combined effect of $\boldsymbol{\eta}$ and g_{mem} suggests that words corresponding to object names have higher likelihoods of being written into the external memory. The agent also successfully learns to use the external memory for storing the information extracted from the teacher's sentence, to fuse it adaptively with the signal from the RNN (capturing the syntactic structure) and to generate a complete sentence with the new concept included. The value of the fusion gate g is small when generating words like "*what*,", "*i*," "*can*," and "*see*," meaning it mainly relies on the signal from the RNN for generation (*c.f.*, Eqn.(2) and Figure 7). In contrast, when generating object names (*e.g.*, "*banana*," and "*cucumber*"), the fusion gate g has a large value, meaning that there is more emphasis on the signal from the external memory. This experiment showed that the proposed approach is applicable to the more complex sentence-level task for language learning and one-shot learning. More interestingly, it learns an interpretable operational process, which can be easily understood. More results including example dialogues from different approaches are presented in Appendix A.4.

6 Discussion

We have presented an approach for grounded language acquisition with one-shot visual concept learning in this work. This is achieved by purely

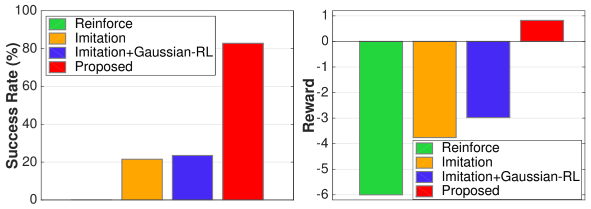


Figure 8: **Test performance** for sentence-level task with image variations (variation ratio=0.5).

interacting with a teacher and learning from feedback arising naturally during interaction through joint imitation and reinforcement learning, with a memory augmented neural network. Experimental results show that the proposed approach is effective for language acquisition with one-shot visual concept learning across several different settings compared with several baseline approaches.

In the current work, we have designed and used a computer game (synthetic task with synthetic language) for training the agent. This is mainly due to the fact that there is no existing dataset to the best of our knowledge that is adequate for developing our addressed interactive language learning and one-shot learning problem. For our current design, although it is an artificial game, there is a reasonable amount of variations both within and across sessions, *e.g.*, the object classes to be learned within a session, the presentation order of the selected classes, the sentence patterns and image instances to be used etc. All these factors contribute to the increased complexity of the learning task, making it non-trivial and already very challenging to existing approaches as shown by the experimental results. While offering flexibility in training, one downside of using a synthetic task is its limited amount of variation compared with real-world scenarios with natural languages. Although it might be non-trivial to extend the proposed approach to real natural language directly, we regard this work as an initial step towards this ultimate ambitious goal and our game might shed some light on designing more advanced games or performing real-world data collection. We plan to investigate the generalization and application of the proposed approach to more realistic environments with more diverse tasks in future work.

Acknowledgments

We thank the reviewers and PC members for theirs efforts in helping improving the paper. We thank Xiaochen Lian and Xiao Chu for their discussions.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Alan Baddeley. 1992. Working memory. *Science*, 255(5044):556–559.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Ann C. Baker, Patricia J. Jensen, and David A. Kolb. 2002. *Conversational Learning: An Experiential Approach to Knowledge Creation*. Copley Publishing Group.

Arielle Borovsky, Marta Kutas, and Jeff Elman. 2003. Learning to use words: Event related potentials index single-shot contextual word learning. *Cognzition*, 116(2):289–296.

K. Cho, B. Merrienboer, C. Glehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014a. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Abhishek Das, Satwik Kottur, , José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *International Conference on Computer Vision (ICCV)*.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *International Conference on International Conference on Machine Learning (ICML)*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Association for Computational Linguistics (ACL)*.

Derek M. Houston and Richard T. Miyamoto. 2011. Effects of early auditory experience on word learning and speech perception in deaf children with cochlear implants: Implications for sensitive periods of language development. *Otol Neurotol*, 31(8):1248–1253.

Patricia K. Kuhl. 2004. Early language acquisition: cracking the speech code. *Nat Rev Neurosci*, 5(2):831–843.

Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. 2011. One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society*.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations (ICLR)*.

Jiwei Li, Alexander H. Miller, Sumit Chopra, MarcAurelio Ranzato, and Jason Weston. 2017. Learning through dialogue interactions by asking questions. In *International Conference on Learning Representations (ICLR)*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. Playing Atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Igor Mordatch and Pieter Abbeel. 2018. Emergence of grounded compositional language in multi-agent populations. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Anna Ingeborg Petursdottir and James R. Mellor. 2016. Reinforcement contingencies in language acquisition. *Policy Insights from the Behavioral and Brain Sciences*, 4(1):25–32.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning (ICML)*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

B. F. Skinner. 1957. *Verbal Behavior*. Copley Publishing Group.

Bradly C. Stadie, Pieter Abbeel, and Ilya Sutskever. 2017. Third-person imitation learning. In *International Conference on Learning Representations (ICLR)*.

Amanda Stent and Srinivas Bangalore. 2014. *Natural Language Generation in Interactive Systems*. Cambridge University Press.

Florian Strub, Harm de Vries, Jérémie Mary, Bilal Piot, Aaron C. Courville, and Olivier Pietquin. 2017. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NIPS)*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

S. I. Wang, P. Liang, and C. Manning. 2016. Learning language games through interaction. In *Association for Computational Linguistics (ACL)*.

Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. Freitas. 2017. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations (ICLR)*.

Sandra R. Waxman. 2004. *Everything had a name, and each name gave birth to a new thought: links between early word learning and conceptual organization*. Cambridge, MA: The MIT Press.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jason Weston. 2016. Dialog-based language learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Mark Woodward and Chelsea Finn. 2016. Active one-shot learning. In *NIPS Deep Reinforcement Learning Workshop*.

Haonan Yu, Haichao Zhang, and Wei Xu. 2018. Interactive grounded language acquisition and generalization in a 2D world. In *International Conference on Learning Representations (ICLR)*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Haichao Zhang, Haonan Yu, and Wei Xu. 2017. Listen, interact and talk: Learning to speak via interaction. In *NIPS Workshop on Visually-Grounded Interaction and Language*.

A Purely End-to-end System for Multi-speaker Speech Recognition

Hiroshi Seki[1,2]*, Takaaki Hori[1], Shinji Watanabe[3], Jonathan Le Roux[1], John R. Hershey[1]
[1]Mitsubishi Electric Research Laboratories (MERL)
[2]Toyohashi University of Technology
[3]Johns Hopkins University

Abstract

Recently, there has been growing interest in multi-speaker speech recognition, where the utterances of multiple speakers are recognized from their mixture. Promising techniques have been proposed for this task, but earlier works have required additional training data such as isolated source signals or senone alignments for effective learning. In this paper, we propose a new sequence-to-sequence framework to directly decode multiple label sequences from a single speech sequence by unifying source separation and speech recognition functions in an end-to-end manner. We further propose a new objective function to improve the contrast between the hidden vectors to avoid generating similar hypotheses. Experimental results show that the model is directly able to learn a mapping from a speech mixture to multiple label sequences, achieving 83.1% relative improvement compared to a model trained without the proposed objective. Interestingly, the results are comparable to those produced by previous end-to-end works featuring explicit separation and recognition modules.

1 Introduction

Conventional automatic speech recognition (ASR) systems recognize a single utterance given a speech signal, in a one-to-one transformation. However, restricting the use of ASR systems to situations with only a single speaker limits their applicability. Recently, there has been growing interest in single-channel multi-speaker speech recognition, which aims at generating multiple transcriptions from a single-channel mixture of multiple speakers' speech (Cooke et al., 2009).

To achieve this goal, several previous works have considered a two-step procedure in which the mixed speech is first separated, and recognition is then performed on each separated speech signal (Hershey et al., 2016; Isik et al., 2016; Yu et al., 2017; Chen et al., 2017). Dramatic advances have recently been made in speech separation, via the *deep clustering* framework (Hershey et al., 2016; Isik et al., 2016), hereafter referred to as DPCL. DPCL trains a deep neural network to map each time-frequency (T-F) unit to a high-dimensional embedding vector such that the embeddings for the T-F unit pairs dominated by the same speaker are close to each other, while those for pairs dominated by different speakers are farther away. The speaker assignment of each T-F unit can thus be inferred from the embeddings by simple clustering algorithms, to produce masks that isolate each speaker. The original method using k-means clustering (Hershey et al., 2016) was extended to allow end-to-end training by unfolding the clustering steps using a permutation-free mask inference objective (Isik et al., 2016). An alternative approach is to perform *direct mask inference* using the permutation-free objective function with networks that directly estimate the labels for a fixed number of sources. Direct mask inference was first used in Hershey et al. (2016) as a baseline method, but without showing good performance. This approach was revisited in Yu et al. (2017) and Kolbaek et al. (2017) under the name permutation-invariant training (PIT). Combination of such single-channel speaker-independent multi-speaker speech separation systems with ASR was first considered in Isik et al. (2016) using a conventional Gaussian Mixture Model/Hidden Markov Model

*This work was done while H. Seki, Ph.D. candidate at Toyohashi University of Technology, Japan, was an intern at MERL.

(GMM/HMM) system. Combination with an end-to-end ASR system was recently proposed in (Settle et al., 2018). Both these approaches either trained or pre-trained the source separation and ASR networks separately, making use of mixtures and their corresponding isolated clean source references. While the latter approach could in principle be trained without references for the isolated speech signals, the authors found it difficult to train from scratch in that case. This ability can nonetheless be used when adapting a pre-trained network to new data without such references.

In contrast with this two-stage approach, Qian et al. (2017) considered direct optimization of a deep-learning-based ASR recognizer without an explicit separation module. The network is optimized based on a permutation-free objective defined using the cross-entropy between the system's hypotheses and reference labels. The best permutation between hypotheses and reference labels in terms of cross-entropy is selected and used for backpropagation. However, this method still requires reference labels in the form of senone alignments, which have to be obtained on the clean isolated sources using a single-speaker ASR system. As a result, this approach still requires the original separated sources. As a general caveat, generation of multiple hypotheses in such a system requires the number of speakers handled by the neural network architecture to be determined before training. However, Qian et al. (2017) reported that the recognition of two-speaker mixtures using a model trained for three-speaker mixtures showed almost identical performance with that of a model trained on two-speaker mixtures. Therefore, it may be possible in practice to determine an upper bound on the number of speakers.

Chen et al. (2018) proposed a progressive training procedure for a hybrid system with explicit separation motivated by curriculum learning. They also proposed self-transfer learning and multi-output sequence discriminative training methods for fully exploiting pairwise speech and preventing competing hypotheses, respectively.

In this paper, we propose to circumvent the need for the corresponding isolated speech sources when training on a set of mixtures, by using an end-to-end multi-speaker speech recognition without an explicit speech separation stage. In separation based systems, the spectrogram is segmented into complementary regions according to sources, which generally ensures that different utterances are recognized for each speaker. Without this complementarity constraint, our direct multi-speaker recognition system could be susceptible to redundant recognition of the same utterance. In order to prevent degenerate solutions in which the generated hypotheses are similar to each other, we introduce a new objective function that enhances contrast between the network's representations of each source. We also propose a training procedure to provide permutation invariance with low computational cost, by taking advantage of the joint CTC/attention-based encoder-decoder network architecture proposed in (Hori et al., 2017a). Experimental results show that the proposed model is able to directly convert an input speech mixture into multiple label sequences without requiring any explicit intermediate representations. In particular no frame-level training labels, such as phonetic alignments or corresponding unmixed speech, are required. We evaluate our model on spontaneous English and Japanese tasks and obtain comparable results to the DPCL based method with explicit separation (Settle et al., 2018).

2 Single-speaker end-to-end ASR

2.1 Attention-based encoder-decoder network

An attention-based encoder-decoder network (Bahdanau et al., 2016) predicts a target label sequence $Y = (y_1, \dots, y_N)$ without requiring intermediate representation from a T-frame sequence of D-dimensional input feature vectors, $O = (o_t \in \mathbb{R}^D | t = 1, \dots, T)$, and the past label history. The probability of the n-th label y_n is computed by conditioning on the past history $y_{1:n-1}$:

$$p_{\text{att}}(Y|O) = \prod_{n=1}^{N} p_{\text{att}}(y_n|O, y_{1:n-1}). \qquad (1)$$

The model is composed of two main sub-modules, an encoder network and a decoder network. The encoder network transforms the input feature vector sequence into a high-level representation $H = (h_l \in \mathbb{R}^C | l = 1, \dots, L)$. The decoder network emits labels based on the label history y and a context vector c calculated using an attention mechanism which weights and sums the C-dimensional sequence of representation H with attention weight a. A hidden state e of the decoder is updated based on the previous state, the previous context vector, and the emitted label. This mechanism is summarized as follows:

$$H = \mathrm{Encoder}(O), \quad (2)$$

$$y_n \sim \mathrm{Decoder}(c_n, y_{n-1}), \quad (3)$$

$$c_n, a_n = \mathrm{Attention}(a_{n-1}, e_n, H), \quad (4)$$

$$e_n = \mathrm{Update}(e_{n-1}, c_{n-1}, y_{n-1}). \quad (5)$$

At inference time, the previously emitted labels are used. At training time, they are replaced by the reference label sequence $R = (r_1, \ldots, r_N)$ in a *teacher-forcing* fashion, leading to conditional probability $p_{\text{att}}(Y_R|O)$, where Y_R denotes the output label sequence variable in this condition. The detailed definitions of Attention and Update are described in Section A of the supplementary material. The encoder and decoder networks are trained to maximize the conditional probability of the reference label sequence R using backpropagation:

$$\mathcal{L}_{\text{att}} = \mathrm{Loss}_{\text{att}}(Y_R, R) \triangleq -\log p_{\text{att}}(Y_R = R|O), \quad (6)$$

where $\mathrm{Loss}_{\text{att}}$ is the cross-entropy loss function.

2.2 Joint CTC/attention-based encoder-decoder network

The joint CTC/attention approach (Kim et al., 2017; Hori et al., 2017a), uses the connectionist temporal classification (CTC) objective function (Graves et al., 2006) as an auxiliary task to train the network. CTC formulates the conditional probability by introducing a framewise label sequence Z consisting of a label set $\mathcal{U}$ and an additional blank symbol defined as $Z = \{z_l \in \mathcal{U} \cup \{\texttt{'blank'}\} | l = 1, \cdots, L\}$:

$$p_{\text{ctc}}(Y|O) = \sum_{Z} \prod_{l=1}^{L} p(z_l|z_{l-1}, Y) p(z_l|O), \quad (7)$$

where $p(z_l|z_{l-1}, Y)$ represents monotonic alignment constraints in CTC and $p(z_l|O)$ is the frame-level label probability computed by

$$p(z_l|O) = \mathrm{Softmax}(\mathrm{Linear}(h_l)), \quad (8)$$

where h_l is the hidden representation generated by an encoder network, here taken to be the encoder of the attention-based encoder-decoder network defined in Eq. (2), and $\mathrm{Linear}(\cdot)$ is the final linear layer of the CTC to match the number of labels. Unlike the attention model, the forward-backward algorithm of CTC enforces monotonic alignment between the input speech and the output label sequences during training and decoding. We adopt the joint CTC/attention-based encoder-decoder network as the monotonic alignment helps the separation and extraction of high-level representation. The CTC loss is calculated as:

$$\mathcal{L}_{\text{ctc}} = \mathrm{Loss}_{\text{ctc}}(Y, R) \triangleq -\log p_{\text{ctc}}(Y = R|O). \quad (9)$$

The CTC loss and the attention-based encoder-decoder loss are combined with an interpolation weight $\lambda \in [0, 1]$:

$$\mathcal{L}_{\text{mtl}} = \lambda \mathcal{L}_{\text{ctc}} + (1 - \lambda) \mathcal{L}_{\text{att}}. \quad (10)$$

Both CTC and encoder-decoder networks are also used in the inference step. The final hypothesis is a sequence that maximizes a weighted conditional probability of CTC in Eq. (7) and attention-based encoder decoder network in Eq. (1):

$$\hat{Y} = \arg\max_{Y} \{\gamma \log p_{\text{ctc}}(Y|O) + (1 - \gamma) \log p_{\text{att}}(Y|O)\}, \quad (11)$$

where $\gamma \in [0, 1]$ is an interpolation weight.

3 Multi-speaker end-to-end ASR

3.1 Permutation-free training

In situations where the correspondence between the outputs of an algorithm and the references is an arbitrary permutation, neural network training faces a *permutation problem*. This problem was first addressed by deep clustering (Hershey et al., 2016), which circumvented it in the case of source separation by comparing the relationships between pairs of network outputs to those between pairs of labels. As a baseline for deep clustering, Hershey et al. (2016) also proposed another approach to address the permutation problem, based on an objective which considers all permutations of references when computing the error with the network estimates. This objective was later used in Isik et al. (2016) and Yu et al. (2017). In the latter, it was referred to as permutation-invariant training.

This permutation-free training scheme extends the usual one-to-one mapping of outputs and labels for backpropagation to one-to-many by selecting the proper permutation of hypotheses and references, thus allowing the network to generate multiple independent hypotheses from a single-channel speech mixture. When a speech mixture contains speech uttered by S speakers simultaneously, the network generates S label sequence variables $Y^s = (y_1^s, \dots, y_{N_s}^s)$ with N_s labels from the T-frame sequence of D-dimensional input feature vectors, $O = (o_t \in \mathbb{R}^D | t = 1, \dots, T)$:

$$Y^s \sim g^s(O),\ s = 1, \dots, S, \tag{12}$$

where the transformations g^s are implemented as neural networks which typically share some components with each other. In the training stage, all possible permutations of the S sequences $R^s = (r_1^s, \dots, r_{N_s'}^s)$ of N_s' reference labels are considered (considering permutations on the hypotheses would be equivalent), and the one leading to minimum loss is adopted for backpropagation. Let $\mathcal{P}$ denote the set of permutations on $\{1, \dots, S\}$. The final loss $\mathcal{L}$ is defined as

$$\mathcal{L} = \min_{\pi \in \mathcal{P}} \sum_{s=1}^{S} \mathrm{Loss}(Y^s, R^{\pi(s)}), \tag{13}$$

where $\pi(s)$ is the s-th element of a permutation π. For example, for two speakers, $\mathcal{P}$ includes two permutations $(1, 2)$ and $(2, 1)$, and the loss is defined as:

$$\begin{aligned}\mathcal{L} = \min(&\mathrm{Loss}(Y^1, R^1) + \mathrm{Loss}(Y^2, R^2),\\ &\mathrm{Loss}(Y^1, R^2) + \mathrm{Loss}(Y^2, R^1)).\end{aligned} \tag{14}$$

Figure 1 shows an overview of the proposed end-to-end multi-speaker ASR system. In the following Section 3.2, we describe an extension of encoder network for the generation of multiple hidden representations. We further introduce a permutation assignment mechanism for reducing the computation cost in Section 3.3, and an additional loss function $\mathcal{L}_{KL}$ for promoting the difference between hidden representations in Section 3.4.

3.2 End-to-end permutation-free training

To make the network output multiple hypotheses, we consider a stacked architecture that combines both shared and unshared (or specific) neural network modules. The particular architecture we consider in this paper splits the encoder network into three stages: the first stage, also referred to as mixture encoder, processes the input mixture and

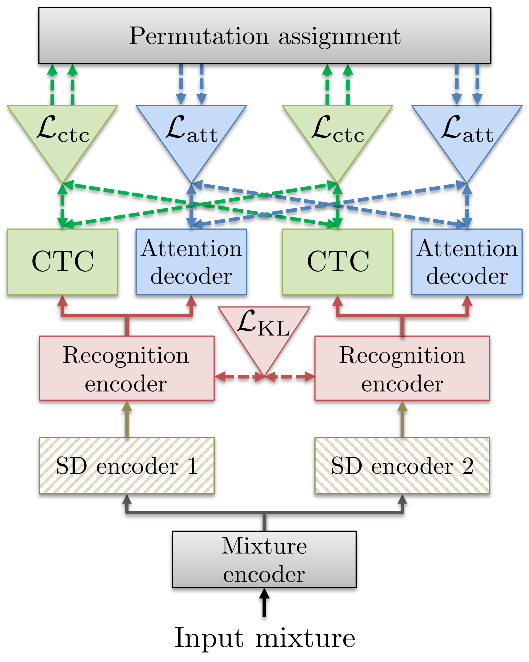


Figure 1: End-to-end multi-speaker speech recognition. We propose to use the permutation-free training for CTC and attention loss functions $\mathrm{Loss}_{\mathrm{ctc}}$ and $\mathrm{Loss}_{\mathrm{att}}$, respectively.

outputs an intermediate feature sequence H; that sequence is then processed by S independent encoder sub-networks which do not share parameters, also referred to as speaker-differentiating (SD) encoders, leading to S feature sequences H^s; at the last stage, each feature sequence H^s is independently processed by the same network, also referred to as recognition encoder, leading to S final high-level representations G^s.

Let $u \in \{1 \dots, S\}$ denote an output index (corresponding to the transcription of the speech by one of the speakers), and $v \in \{1 \dots, S\}$ denote a reference index. Denoting by $\mathrm{Encoder}_{\mathrm{Mix}}$ the mixture encoder, $\mathrm{Encoder}_{\mathrm{SD}}^{\mathrm{u}}$ the u-th speaker-differentiating encoder, and $\mathrm{Encoder}_{\mathrm{Rec}}$ the recognition encoder, an input sequence O corresponding to an input mixture can be processed by the encoder network as follows:

$$H = \mathrm{Encoder}_{\mathrm{Mix}}(O), \tag{15}$$
$$H^u = \mathrm{Encoder}_{\mathrm{SD}}^{\mathrm{u}}(\mathrm{H}), \tag{16}$$
$$G^u = \mathrm{Encoder}_{\mathrm{Rec}}(H^u). \tag{17}$$

The motivation for designing such an architecture can be explained as follows, following analogies with the architectures in (Isik et al., 2016) and (Settle et al., 2018) where separation and recognition are performed explicitly in separate steps: the first stage in Eq. (15) corresponds to a speech separation module which creates embedding vectors that can be used to distinguish between the multiple sources; the speaker-differentiating second stage in Eq. (16) uses the first stage's output to disentangle each speaker's speech content from the mixture, and prepare it for recognition; the final stage in Eq. (17) corresponds to an acoustic model that encodes the single-speaker speech for final decoding.

The decoder network computes the conditional probabilities for each speaker from the S outputs of the encoder network. In general, the decoder network uses the reference label R as a history to generate the attention weights during training, in a teacher-forcing fashion. However, in the above permutation-free training scheme, the reference label to be attributed to a particular output is not determined until the loss function is computed, so we here need to run the attention decoder for all reference labels. We thus need to consider the conditional probability of the decoder output variable $Y^{u,v}$ for each output G^u of the encoder network under the assumption that the reference label for that output is R^v:

$$p_{\text{att}}(Y^{u,v}|O) = \prod_n p_{\text{att}}(y_n^{u,v}|O, y_{1:n-1}^{u,v}), \quad (18)$$

$$c_n^{u,v}, a_n^{u,v} = \text{Attention}(a_{n-1}^{u,v}, e_n^{u,v}, G^u), \quad (19)$$

$$e_n^{u,v} = \text{Update}(e_{n-1}^{u,v}, c_{n-1}^{u,v}, r_{n-1}^{v}), \quad (20)$$

$$y_n^{u,v} \sim \text{Decoder}(c_n^{u,v}, r_{n-1}^{v}). \quad (21)$$

The final loss is then calculated by considering all permutations of the reference labels as follows:

$$\mathcal{L}_{\text{att}} = \min_{\pi \in \mathcal{P}} \sum_s \text{Loss}_{\text{att}}(Y^{s,\pi(s)}, R^{\pi(s)}). \quad (22)$$

3.3 Reduction of permutation cost

In order to reduce the computational cost, we fixed the permutation of the reference labels based on the minimization of the CTC loss alone, and used the same permutation for the attention mechanism as well. This is an advantage of using a joint CTC/attention based end-to-end speech recognition. Permutation is performed only for the CTC loss by assuming synchronous output where the permutation is decided by the output of CTC:

$$\hat{\pi} = \arg\min_{\pi \in \mathcal{P}} \sum_s \text{Loss}_{\text{ctc}}(Y^s, R^{\pi(s)}), \quad (23)$$

where Y^u is the output sequence variable corresponding to encoder output G^u. Attention-based decoding is then performed on the same hidden representations G^u, using teacher forcing with the labels determined by the permutation $\hat{\pi}$ that minimizes the CTC loss:

$$p_{\text{att}}(Y^{u,\hat{\pi}(u)}|O) = \prod_n p_{\text{att}}(y_n^{u,\hat{\pi}(u)}|O, y_{1:n-1}^{u,\hat{\pi}(u)}),$$

$$c_n^{u,\hat{\pi}(u)}, a_n^{u,\hat{\pi}(u)} = \text{Attention}(a_{n-1}^{u,\hat{\pi}(u)}, e_n^{u,\hat{\pi}(u)}, G^u),$$

$$e_n^{u,\hat{\pi}(u)} = \text{Update}(e_{n-1}^{u,\hat{\pi}(u)}, c_{n-1}^{u,\hat{\pi}(u)}, r_{n-1}^{\hat{\pi}(u)}),$$

$$y_n^{u,\hat{\pi}(u)} \sim \text{Decoder}(c_n^{u,\hat{\pi}(u)}, r_{n-1}^{\hat{\pi}(u)}).$$

This corresponds to the "permutation assignment" in Fig. 1. In contrast with Eq. (18), we only need to run the attention-based decoding once for each output G^u of the encoder network. The final loss is defined as the sum of two objective functions with interpolation λ:

$$\mathcal{L}_{\text{mtl}} = \lambda \mathcal{L}_{\text{ctc}} + (1-\lambda)\mathcal{L}_{\text{att}}, \quad (24)$$

$$\mathcal{L}_{\text{ctc}} = \sum_s \text{Loss}_{\text{ctc}}(Y^s, R^{\hat{\pi}(s)}), \quad (25)$$

$$\mathcal{L}_{\text{att}} = \sum_s \text{Loss}_{\text{att}}(Y^{s,\hat{\pi}(s)}, R^{\hat{\pi}(s)}). \quad (26)$$

At inference time, because both CTC and attention-based decoding are performed on the same encoder output G^u and should thus pertain to the same speaker, their scores can be incorporated as follows:

$$\hat{Y}^u = \arg\max_{Y^u} \{\gamma \log p_{\text{ctc}}(Y^u|G^u) + (1-\gamma)\log p_{\text{att}}(Y^u|G^u)\}, \quad (27)$$

where $p_{\text{ctc}}(Y^u|G^u)$ and $p_{\text{att}}(Y^u|G^u)$ are obtained with the same encoder output G^u.

3.4 Promoting separation of hidden vectors

A single decoder network is used to output multiple label sequences by independently decoding the multiple hidden vectors generated by the encoder network. In order for the decoder to generate multiple different label sequences the encoder needs to generate sufficiently differentiated hidden vector sequences for each speaker. We propose to encourage this contrast among hidden vectors by introducing in the objective function a new term based on the negative symmetric Kullback-Leibler

(KL) divergence. In the particular case of two-speaker mixtures, we consider the following additional loss function:

$$\mathcal{L}_{\mathrm{KL}} = -\eta \sum_{l} \{\mathrm{KL}(\bar{G}^1(l) \,||\, \bar{G}^2(l)) + \mathrm{KL}(\bar{G}^2(l) \,||\, \bar{G}^1(l))\}, \quad (28)$$

where η is a small constant value, and $\bar{G}^u = (\mathrm{softmax}(G^u(l)) \mid l = 1, \dots, L)$ is obtained from the hidden vector sequence G^u at the output of the recognition encoder $\mathrm{Encoder}_{\mathrm{Rec}}$ as in Fig. 1 by applying an additional frame-wise softmax operation in order to obtain a quantity amenable to a probability distribution.

3.5 Split of hidden vector for multiple hypotheses

Since the network maps acoustic features to label sequences directly, we consider various architectures to perform implicit separation and recognition effectively. As a baseline system, we use the concatenation of a VGG-motivated CNN network (Simonyan and Zisserman, 2014) (referred to as VGG) and a bi-directional long short-term memory (BLSTM) network as the encoder network. For the splitting point in the hidden vector computation, we consider two architectural variations as follows:

- Split by BLSTM: The hidden vector is split at the level of the BLSTM network. 1) the VGG network generates a single hidden vector H; 2) H is fed into S independent BLSTMs whose parameters are not shared with each other; 3) the output of each independent BLSTM $H^u, u = 1, \dots, S$, is further separately fed into a unique BLSTM, the same for all outputs. Each step corresponds to Eqs. (15), (16), and (17).

- Split by VGG: The hidden vector is split at the level of the VGG network. The number of filters at the last convolution layer is multiplied by the number of mixtures S in order to split the output into S hidden vectors (as in Eq. (16)). The layers prior to the last VGG layer correspond to the network in Eq. (15), while the subsequent BLSTM layers implement the network in (17).

4 Experiments

4.1 Experimental setup

We used English and Japanese speech corpora, WSJ (Wall street journal) (Consortium, 1994; Garofalo et al., 2007) and CSJ (Corpus of spontaneous Japanese) (Maekawa, 2003). To show the effectiveness of the proposed models, we generated mixed speech signals from these corpora to simulate single-channel overlapped multi-speaker recording, and evaluated the recognition performance using the mixed speech data. For WSJ, we used WSJ1 SI284 for training, Dev93 for development, and Eval92 for evaluation. For CSJ, we followed the Kaldi recipe (Moriya et al., 2015) and used the full set of academic and simulated presentations for training, and the standard test sets 1, 2, and 3 for evaluation.

Table 1: Duration (hours) of unmixed and mixed corpora. The mixed corpora are generated by Algorithm 1 in Section B of the supplementary material, using the training, development, and evaluation set respectively.

	TRAIN	DEV.	EVAL
WSJ (UNMIXED)	81.5	1.1	0.7
WSJ (MIXED)	98.5	1.3	0.8
CSJ (UNMIXED)	583.8	6.6	5.2
CSJ (MIXED)	826.9	9.1	7.5

We created new corpora by mixing two utterances with different speakers sampled from existing corpora. The detailed algorithm is presented in Section B of the supplementary material. The sampled pairs of two utterances are mixed at various signal-to-noise ratios (SNR) between 0 dB and 5 dB with a random starting point for the overlap. Duration of original unmixed and generated mixed corpora are summarized in Table 1.

4.1.1 Network architecture

As input feature, we used 80-dimensional log Mel filterbank coefficients with pitch features and their delta and delta delta features ($83 \times 3 = 249$-dimension) extracted using Kaldi tools (Povey et al., 2011). The input feature is normalized to zero mean and unit variance. As a baseline system, we used a stack of a 6-layer VGG network and a 7-layer BLSTM as the encoder network. Each BLSTM layer has 320 cells in each direction, and is followed by a linear projection layer with 320 units to combine the forward and backward LSTM outputs. The decoder network has an 1-layer LSTM with 320 cells. As described in Section 3.5, we adopted two types of encoder architectures for multi-speaker speech recognition. The network architectures are summarized in Table 2. The split-by-VGG network had speaker differentiating encoders with a convolution layer

Table 2: Network architectures for the encoder network. The number of layers is indicated in parentheses. $\mathrm{Encoder}_{\mathrm{Mix}}$, $\mathrm{Encoder}^{\mathrm{u}}_{\mathrm{SD}}$, and $\mathrm{Encoder}_{\mathrm{Rec}}$ correspond to Eqs. (15), (16), and (17).

SPLIT BY	$\mathrm{Encoder}_{\mathrm{Mix}}$	$\mathrm{Encoder}^{\mathrm{u}}_{\mathrm{SD}}$	$\mathrm{Encoder}_{\mathrm{Rec}}$
NO	VGG (6)	—	BLSTM (7)
VGG	VGG (4)	VGG (2)	BLSTM (7)
BLSTM	VGG (6)	BLSTM (2)	BLSTM (5)

(and the following maxpooling layer). The split-by-BLSTM network had speaker differentiating encoders with two BLSTM layers. The architectures were adjusted to have the same number of layers. We used characters as output labels. The number of characters for WSJ was set to 49 including alphabets and special tokens (e.g., characters for space and unknown). The number of characters for CSJ was set to 3,315 including Japanese Kanji/Hiragana/Katakana characters and special tokens.

4.1.2 Optimization

The network was initialized randomly from uniform distribution in the range -0.1 to 0.1. We used the AdaDelta algorithm (Zeiler, 2012) with gradient clipping (Pascanu et al., 2013) for optimization. We initialized the AdaDelta hyperparameters as $\rho = 0.95$ and $\epsilon = 1^{-8}$. ϵ is decayed by half when the loss on the development set degrades. The networks were implemented with Chainer (Tokui et al., 2015) and ChainerMN (Akiba et al., 2017). The optimization of the networks was done by synchronous data parallelism with 4 GPUs for WSJ and 8 GPUs for CSJ.

The networks were first trained on single-speaker speech, and then retrained with mixed speech. When training on unmixed speech, only one side of the network only (with a single speaker differentiating encoder) is optimized to output the label sequence of the single speaker. Note that only character labels are used, and there is no need for clean source reference corresponding to the mixed speech. When moving to mixed speech, the other speaker-differentiating encoders are initialized using the already trained one by copying the parameters with random perturbation, $w' = w \times (1 + \mathrm{Uniform}(-0.1, 0.1))$ for each parameter w. The interpolation value λ for the multiple objectives in Eqs. (10) and (24) was set to 0.1 for WSJ and to 0.5 for CSJ. Lastly, the model is retrained with the additional negative KL divergence loss in Eq. (28) with $\eta = 0.1$.

Table 3: Evaluation of unmixed speech without multi-speaker training.

TASK	AVG.
WSJ	2.6
CSJ	7.8

4.1.3 Decoding

In the inference stage, we combined a pre-trained RNNLM (recurrent neural network language model) in parallel with the CTC and decoder network. Their label probabilities were linearly combined in the log domain during beam search to find the most likely hypothesis. For the WSJ task, we used both character and word level RNNLMs (Hori et al., 2017b), where the character model had a 1-layer LSTM with 800 cells and an output layer for 49 characters. The word model had a 1-layer LSTM with 1000 cells and an output layer for 20,000 words, i.e., the vocabulary size was 20,000. Both models were trained with the WSJ text corpus. For the CSJ task, we used a character level RNNLM (Hori et al., 2017c), which had a 1-layer LSTM with 1000 cells and an output layer for 3,315 characters. The model parameters were trained with the transcript of the training set in CSJ. We added language model probabilities with an interpolation factor of 0.6 for character-level RNNLM and 1.2 for word-level RNNLM.

The beam width for decoding was set to 20 in all the experiments. Interpolation γ in Eqs. (11) and (27) was set to 0.4 for WSJ and 0.5 for CSJ.

4.2 Results

4.2.1 Evaluation of unmixed speech

First, we examined the performance of the baseline joint CTC/attention-based encoder-decoder network with the original unmixed speech data. Table 3 shows the character error rates (CERs), where the baseline model showed 2.6% on WSJ and 7.8% on CSJ. Since the model was trained and evaluated with unmixed speech data, these CERs are considered lower bounds for the CERs in the succeeding experiments with mixed speech data.

4.2.2 Evaluation of mixed speech

Table 4 shows the CERs of the generated mixed speech from the WSJ corpus. The first column indicates the position of split as mentioned in Section 3.5. The second, third and forth columns indicate CERs of the high energy speaker (HIGH E. SPK.), the low energy speaker (LOW E. SPK.), and the average (AVG.), respectively. The baseline model has very high CERs because

Table 4: CER (%) of mixed speech for WSJ.

Split	High E. spk.	Low E. spk.	Avg.
no (baseline)	86.4	79.5	83.0
VGG	17.4	15.6	16.5
BLSTM	14.6	**13.3**	14.0
+ KL loss	**14.0**	**13.3**	**13.7**

Table 5: CER (%) of mixed speech for CSJ.

Split	High E. spk.	Low E. spk.	Avg.
no (baseline)	93.3	92.1	92.7
BLSTM	**11.0**	**18.8**	**14.9**

it was trained as a single-speaker speech recognizer without permutation-free training, and it can only output one hypothesis for each mixed speech. In this case, the CERs were calculated by duplicating the generated hypothesis and comparing the duplicated hypotheses with the corresponding references. The proposed models, i.e., split-by-VGG and split-by-BLSTM networks, obtained significantly lower CERs than the baseline CERs, the split-by-BLSTM model in particular achieving 14.0% CER. This is an 83.1% relative reduction from the baseline model. The CER was further reduced to 13.7% by retraining the split-by-BLSTM model with the negative KL loss, a 2.1% relative reduction from the network without retraining. This result implies that the proposed negative KL loss provides better separation by actively improving the contrast between the hidden vectors of each speaker. Examples of recognition results are shown in Section C of the supplementary material. Finally, we profiled the computation time for the permutations based on the decoder network and on CTC. Permutation based on CTC was 16.3 times faster than that based on the decoder network, in terms of the time required to determine the best match permutation given the encoder network's output in Eq. (17).

Table 5 shows the CERs for the mixed speech from the CSJ corpus. Similarly to the WSJ experiments, our proposed model significantly reduced the CER from the baseline, where the average CER was 14.9% and the reduction ratio from the baseline was 83.9%.

4.2.3 Visualization of hidden vectors

We show a visualization of the encoder networks outputs in Fig. 2 to illustrate the effect of the negative KL loss function. Principal component analysis (PCA) was applied to the hidden vectors on the vertical axis. Figures 2(a) and 2(b) show the hidden vectors generated by the split-by-BLSTM model without the negative KL divergence loss for an example mixture of two speakers. We can observe different activation patterns showing that the hidden vectors were successfully separated to the individual utterances in the mixed speech, although some activity from one speaker can be seen as leaking into the other. Figures 2(c) and 2(d) show the hidden vectors generated after retraining with the negative KL divergence loss. We can more clearly observe the different patterns and boundaries of activation and deactivation of hidden vectors. The negative KL loss appears to regularize the separation process, and even seems to help in finding the end-points of the speech.

4.2.4 Comparison with earlier work

We first compared the recognition performance with a hybrid (non end-to-end) system including DPCL-based speech separation and a Kaldi-based ASR system. It was evaluated under the same evaluation data and metric as in (Isik et al., 2016) based on the WSJ corpus. However, there are differences in the size of training data and the options in decoding step. Therefore, it is not a fully matched condition. Results are shown in Table 6. The word error rate (WER) reported in (Isik et al., 2016) is 30.8%, which was obtained with jointly trained DPCL and second-stage speech enhancement networks. The proposed end-to-end ASR gives an 8.4% relative reduction in WER even though our model does not require any explicit frame-level labels such as phonetic alignment, or clean signal reference, and does not use a phonetic lexicon for training. Although this is an unfair comparison, our purely end-to-end system outperformed a hybrid system for multi-speaker speech recognition.

Next, we compared our method with an end-to-end explicit separation and recognition network (Settle et al., 2018). We retrained our model previously trained on our WSJ-based corpus using the training data generated by Settle et al. (2018), because the direct optimization from scratch on their data caused poor recognition performance due to data size. Other experimental conditions are shared with the earlier work. Interestingly, our method showed comparable performance to the end-to-end explicit separation and recognition network, without having to pre-train using clean signal training references. It remains to be seen if this parity of performance holds in other tasks and conditions.

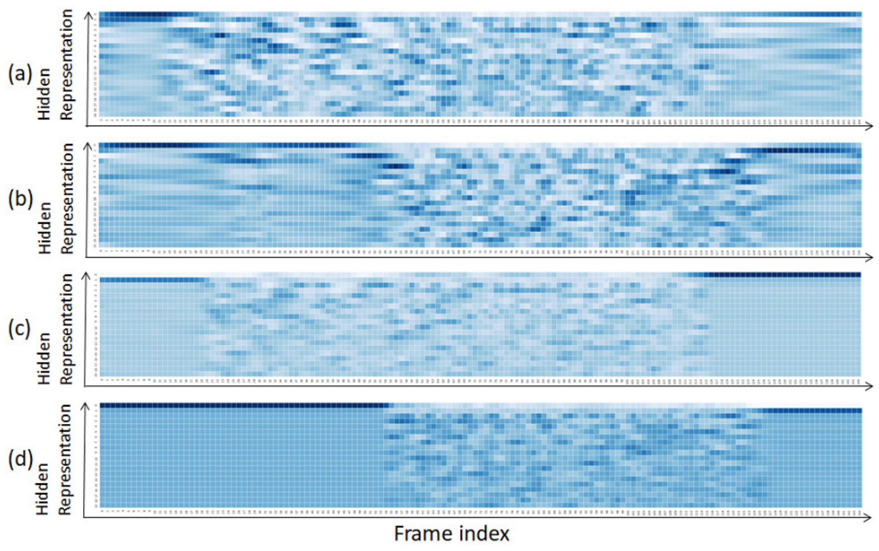


Figure 2: Visualization of the two hidden vector sequences at the output of the split-by-BLSTM encoder on a two-speaker mixture. (a,b): Generated by the model without the negative KL loss. (c,d): Generated by the model with the negative KL loss.

Table 6: Comparison with conventional approaches

METHOD	WER (%)
DPCL + ASR (ISIK ET AL., 2016)	30.8
Proposed end-to-end ASR	**28.2**
METHOD	CER (%)
END-TO-END DPCL + ASR (CHAR LM) (SETTLE ET AL., 2018)	**13.2**
Proposed end-to-end ASR (char LM)	14.0

5 Related work

Several previous works have considered an explicit two-step procedure (Hershey et al., 2016; Isik et al., 2016; Yu et al., 2017; Chen et al., 2017, 2018). In contrast with our work which uses a single objective function for ASR, they introduced an objective function to guide the separation of mixed speech.

Qian et al. (2017) trained a multi-speaker speech recognizer using permutation-free training without explicit objective function for separation. In contrast with our work which uses an end-to-end architecture, their objective function relies on a senone posterior probability obtained by aligning unmixed speech and text using a model trained as a recognizer for single-speaker speech. Compared with (Qian et al., 2017), our method directly maps a speech mixture to multiple character sequences and eliminates the need for the corresponding isolated speech sources for training.

6 Conclusions

In this paper, we proposed an end-to-end multi-speaker speech recognizer based on permutation-free training and a new objective function promoting the separation of hidden vectors in order to generate multiple hypotheses. In an encoder-decoder network framework, teacher forcing at the decoder network under multiple references increases computational cost if implemented naively. We avoided this problem by employing a joint CTC/attention-based encoder-decoder network.

Experimental results showed that the model is able to directly convert an input speech mixture into multiple label sequences under the end-to-end framework without the need for any explicit intermediate representation including phonetic alignment information or pairwise unmixed speech. We also compared our model with a method based on explicit separation using deep clustering, and showed comparable result. Future work includes data collection and evaluation in a real world scenario since the data used in our experiments are simulated mixed speech, which is already extremely challenging but still leaves some acoustic aspects, such as Lombard effects and real room impulse responses, that need to be alleviated for further performance improvement. In addition, further study is required in terms of increasing the number of speakers that can be simultaneously recognized, and further comparison with the separation-based approach.

References

Takuya Akiba, Keisuke Fukuda, and Shuji Suzuki. 2017. ChainerMN: Scalable Distributed Deep Learning Framework. In *Proceedings of Workshop on ML Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949.

Zhehuai Chen, Jasha Droppo, Jinyu Li, and Wayne Xiong. 2018. Progressive joint modeling in unsupervised single-channel overlapped speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(1):184–196.

Zhuo Chen, Yi Luo, and Nima Mesgarani. 2017. Deep attractor network for single-microphone speaker separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 246–250.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–585.

Linguistic Data Consortium. 1994. CSR-II (wsj1) complete. *Linguistic Data Consortium, Philadelphia*, LDC94S13A.

Martin Cooke, John R Hershey, and Steven J Rennie. 2009. Monaural speech separation and recognition challenge. *Computer Speech and Language*, 24(1):1–15.

John Garofalo, David Graff, Doug Paul, and David Pallett. 2007. CSR-I (wsj0) complete. *Linguistic Data Consortium, Philadelphia*, LDC93S6A.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine learning (ICML)*, pages 369–376.

John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. 2016. Deep clustering: Discriminative embeddings for segmentation and separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35.

Takaaki Hori, Shinji Watanabe, and John R Hershey. 2017a. Joint CTC/attention decoding for end-to-end speech recognition. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technologies: long papers*.

Takaaki Hori, Shinji Watanabe, and John R Hershey. 2017b. Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Takaaki Hori, Shinji Watanabe, Yu Zhang, and Chan William. 2017c. Advances in joint CTC-Attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *Interspeech*, pages 949–953.

Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R. Hershey. 2016. Single-channel multi-speaker separation using deep clustering. In *Proc. Interspeech*, pages 545–549.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839.

Morten Kolbæk, Dong Yu, Zheng-Hua Tan, and Jesper Jensen. 2017. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(10):1901–1913.

Kikuo Maekawa. 2003. Corpus of Spontaneous Japanese: Its design and evaluation. In *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*.

Takafumi Moriya, Takahiro Shinozaki, and Shinji Watanabe. 2015. Kaldi recipe for Japanese spontaneous speech recognition and its evaluation. In *Autumn Meeting of ASJ*, 3-Q-7.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning (ICML)*, pages 1310–1318.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Yanmin Qian, Xuankai Chang, and Dong Yu. 2017. Single-channel multi-talker speech recognition with permutation invariant training. *arXiv preprint arXiv:1707.06527*.

Shane Settle, Jonathan Le Roux, Takaaki Hori, Shinji Watanabe, and John R. Hershey. 2018. End-to-end multi-speaker speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4819–4823.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in NIPS*.

Dong Yu, Morten Kolbk, Zheng-Hua Tan, and Jesper Jensen. 2017. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

A Structured Variational Autoencoder for Contextual Morphological Inflection

Lawrence Wolf-Sonkin[*] **Jason Naradowsky**[*] **Sebastian J. Mielke**[*] **Ryan Cotterell**[*]
Department of Computer Science, Johns Hopkins University
{lawrencews,narad,sjmielke,ryan.cotterell}@jhu.edu

Abstract

Statistical morphological inflectors are typically trained on fully supervised, type-level data. One remaining open research question is the following: How can we effectively exploit raw, token-level data to improve their performance? To this end, we introduce a novel generative latent-variable model for the semi-supervised learning of inflection generation. To enable posterior inference over the latent variables, we derive an efficient variational inference procedure based on the wake-sleep algorithm. We experiment on 23 languages, using the Universal Dependencies corpora in a simulated low-resource setting, and find improvements of over 10% absolute accuracy in some cases.

1 Introduction

The majority of the world's languages overtly encodes syntactic information on the word form itself, a phenomenon termed inflectional morphology (Dryer et al., 2005). In English, for example, the verbal lexeme with lemma talk has the four forms: talk, talks, talked and talking. Other languages, such as Archi (Kibrik, 1998), distinguish more than a thousand verbal forms. Despite the cornucopia of unique variants a single lexeme may mutate into, native speakers can flawlessly predict the correct variant that the lexeme's syntactic context dictates. Thus, in computational linguistics, a natural question is the following: Can we estimate a probability model that can do the same?

The topic of inflection generation has been the focus of a flurry of individual attention of late and, moreover, has been the subject of two shared tasks

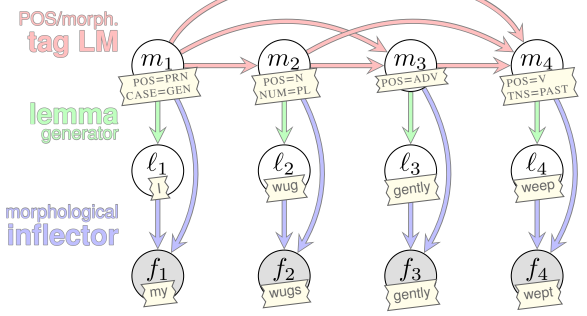


Figure 1: A length-4 example of our generative model factorized as in Eq. (1) and overlayed with example values of the random variables in the sequence. We highlight that all the conditionals in the Bayesian network are recurrent neural networks, e.g., we note that m_i depends on $\boldsymbol{m}_{<i}$ because we employ a recurrent neural network to model the morphological tag sequence.

(Cotterell et al., 2016, 2017). Most work, however, has focused on the fully supervised case—a source lemma and the morpho-syntactic properties are fed into a model, which is asked to produce the desired inflection. In contrast, our work focuses on the semi-supervised case, where we wish to make use of unannotated raw text, i.e., a sequence of inflected tokens.

Concretely, we develop a generative directed graphical model of inflected forms *in context*. A contextual inflection model works as follows: Rather than just generating the proper inflection for a single given word form *out of context* (for example *walking* as the gerund of *walk*), our generative model is actually a fully-fledged language model. In other words, it generates *sequences* of inflected words. The graphical model is displayed in Fig. 1 and examples of words it may generate are pasted on top of the graphical model notation. That our model is a language model enables it to exploit both inflected lexicons and unlabeled raw text in a principled semi-supervised way. In order to train using

[*] All authors contributed equally.

	SG	PL	SG	PL
NOM	Wort	Wörter	Herr	Herren
GEN	Wortes	Wörter	Herrn	Herren
ACC	Wort	Wörter	Herrn	Herren
DAT	Worte	Wörtern	Herrn	Herren

Table 1: As an exhibit of morphological inflection, full paradigms (two numbers and four cases, 8 slots total) for the German nouns Wort ("word") and Herr ("gentleman"), with abbreviated and tabularized UniMorph annotation.

raw-text corpora (which is useful when we have less annotated data), we marginalize out the unobserved lemmata and morpho-syntactic annotation from unlabeled data. In terms of Fig. 1, this refers to marginalizing out $m_1, \ldots, m_4$ and $\ell_1, \ldots, \ell_4$. As this marginalization is intractable, we derive a variational inference procedure that allows for efficient approximate inference. Specifically, we modify the wake-sleep procedure of Hinton et al. (1995). It is the inclusion of raw text in this fashion that makes our model *token level*, a novelty in the camp of inflection generation, as much recent work in inflection generation (Dreyer et al., 2008; Durrett and DeNero, 2013; Nicolai et al., 2015; Ahlberg et al., 2015; Faruqui et al., 2016), trains a model on *type-level* lexicons.

We offer empirical validation of our model's utility with experiments on 23 languages from the Universal Dependencies corpus in a simulated low-resource setting.[1] Our semi-supervised scheme improves inflection generation by over 10% absolute accuracy in some cases.

2 Background: Morphological Inflection

2.1 Inflectional Morphology

To properly discuss models of inflectional morphology, we require a formalization. We adopt the framework of word-based morphology (Aronoff, 1976; Spencer, 1991). Note in the present paper, we omit derivational morphology.

We define an **inflected lexicon** as a set of 4-tuples consisting of a part-of-speech tag, a lexeme, an inflectional slot, and a surface form. A **lexeme** is a discrete object that indexes the word's core meaning and part of speech. In place of such an abstract lexeme, lexicographers will often use a **lemma**, denoted by ℓ, which is a designated[2] surface form of the lexeme (such as the infinitive). For the remainder of this paper, we will use the lemma as a proxy for the lexeme, wherever convenient, although we note that lemmata may be ambiguous: bank is the lemma for at least two distinct nouns and two distinct verbs. For inflection, this ambiguity will rarely[3] play a role—for instance, all senses of bank inflect in the same fashion.

A **part-of-speech (POS) tag**, denoted t, is a coarse syntactic category such as VERB. Each POS tag allows some set of lexemes, and also allows some set of inflectional **slots**, denoted as σ, such as $\left[\text{TNS=PAST}, \text{PERSON=3}\right]$. Each allowed $\langle$tag, lexeme, slot$\rangle$ triple is realized—in only one way—as an inflected **surface form**, a string over a fixed phonological or orthographic alphabet Σ. (In this work, we take Σ to be an orthographic alphabet.) Additionally, we will define the term **morphological tag**, denoted by m, which we take to be the POS-slot pair $m = \langle t, \sigma \rangle$. We will further define $\mathcal{T}$ as the set of all POS tags and $\mathcal{M}$ as the set of all morphological tags.

A **paradigm** $\pi(t, \ell)$ is the mapping from tag t's slots to the surface forms that "fill" those slots for lexeme/lemma ℓ. For example, in the English paradigm $\pi(\text{VERB}, \text{talk})$, the past-tense slot is said to be filled by talked, meaning that the lexicon contains the tuple $\langle \text{VERB}, \text{talk}, \text{PAST}, \text{talked} \rangle$.

A cheat sheet for the notation is provided in Tab. 2.

We will specifically work with the UniMorph annotation scheme (Sylak-Glassman, 2016). Here, each slot specifies a morpho-syntactic bundle of inflectional features such as tense, mood, person, number, and gender. For example, the German surface form Wörtern is listed in the lexicon with tag NOUN, lemma Wort, and a slot specifying the feature bundle $\left[\text{NUM=PL}, \text{CASE=DAT}\right]$. The full paradigms $\pi(\text{NOUN}, \text{Wort})$ and $\pi(\text{NOUN}, \text{Herr})$ are found in Tab. 1.

2.2 Morphological Inflection

Now, we formulate the task of context-free **morphological inflection** using the notation developed in §2. Given a set of N form-tag-lemma triples $\{\langle f_i, m_i, \ell_i \rangle\}_{i=1}^{N}$, the goal of morphological inflection is to map the pair $\langle m_i, \ell_i \rangle$ to the form f_i. As

[1]We make our code and data available at: https://github.com/lwolfsonkin/morph-svae.

[2]A specific slot of the paradigm is chosen, depending on the part-of-speech tag – all these terms are defined next.

[3]One example of a paradigm where the lexeme, rather than the lemma, may influence inflection is hang. If one chooses the lexeme that licenses animate objects, the proper past tense is hanged, whereas it is hung for the lexeme that licenses inanimate objects.

object	symbol	example
form	f	talking
lemma	ℓ	talk
POS	t	VERB
slot	σ	[TNS=GERUND]
morph. tag	m	[POS=V, TNS=GERUND]

Table 2: Notational cheat sheet for the paper.

the definition above indicates, the task is traditionally performed at the *type level*. In this work, however, we focus on a generalization of the task to the *token level*—we seek to map a bisequence of lemma-tag pairs to the sequence of inflected forms in context. Formally, we will denote the lemma-morphological tag bisequence as $\langle \boldsymbol{\ell}, \boldsymbol{m} \rangle$ and the form sequence as $\boldsymbol{f}$. Foreshadowing, the primary motivation for this generalization is to enable the use of raw-text in a semi-supervised setting.

3 Generating Sequences of Inflections

The primary contribution of this paper is a novel generative model over sequences of inflected words in their sentential context. Following the notation laid out in §2.2, we seek to jointly learn a distribution over sequences of forms $\boldsymbol{f}$, lemmata $\boldsymbol{\ell}$, and morphological tags $\boldsymbol{m}$. The generative procedure is as follows: First, we sample a sequence of tags $\boldsymbol{m}$, each morphological tag coming from a language model over morphological tags: $m_i \sim p_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{m}_{<i})$. Next, we sample the sequence of lemmata $\boldsymbol{\ell}$ given the previously sampled sequence of tags $\boldsymbol{m}$— these are sampled conditioned only on the corresponding morphological tag: $\ell_i \sim p_{\boldsymbol{\theta}}(\cdot \mid m_i)$. Finally, we sample the sequence of inflected words $\boldsymbol{f}$, where, again, each word is chosen conditionally independent of other elements of the sequence: $f_i \sim p_{\boldsymbol{\theta}}(\cdot \mid \ell_i, m_i)$.[4] This yields the factorized joint distribution:

$$p_{\boldsymbol{\theta}}(\boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m}) = \tag{1}$$
$$\left(\prod_{i=1}^{|\boldsymbol{f}|} \underbrace{p_{\boldsymbol{\theta}}(f_i \mid \ell_i, m_i)}_{\substack{\textit{morphological inflector} \\ \textcircled{3}}} \cdot \underbrace{p_{\boldsymbol{\theta}}(\ell_i \mid m_i)}_{\substack{\textit{lemma generator} \\ \textcircled{2}}} \right) \cdot \underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{m})}_{\substack{\textit{m-tag LM} \\ \textcircled{1}}}$$

We depict the corresponding directed graphical model in Fig. 1.

[4]Note that we denote all three distributions as $p_{\boldsymbol{\theta}}$ to simplify notation and emphasize the joint modeling aspect; context will always resolve the ambiguity in this paper. We will discuss their parameterization in §4.

Relation to Other Models in NLP. As the graphical model drawn in Fig. 1 shows, our model is quite similar to a Hidden Markov Model (HMM) (Rabiner, 1989). There are two primary differences. First, we remark that an HMM directly emits a form f_i conditioned on the tag m_i. Our model, in contrast, emits a lemma ℓ_i conditioned on the morphological tag m_i and, then, conditioned on both the lemma ℓ_i and the tag m_i, we emit the inflected form f_i. In this sense, our model resembles the hierarchical HMM of Fine et al. (1998) with the difference that we do not have interdependence between the lemmata ℓ_i. The second difference is that our model is non-Markovian: we sample the i^{th} morphological tag m_i from a distribution that depends on *all* previous tags, using an LSTM language model (§4.1). This yields richer interactions among the tags, which may be necessary for modeling long-distance agreement phenomena.

Why a Generative Model? What is our interest in a generative model of inflected forms? Eq. (1) is a syntax-only language model in that it only allows for interdependencies between the morpho-syntactic tags in $p_{\boldsymbol{\theta}}(\boldsymbol{m})$. However, given a tag sequence $\boldsymbol{m}$, the individual lemmata and forms are *conditionally independent*. This prevents the model from learning notions such as semantic frames and topicality. So what is this model good for? Our chief interest is the ability to *train a morphological inflector on unlabeled data*, which is a boon in a low-resource setting. As the model is generative, we may consider the latent-variable model:

$$p_{\boldsymbol{\theta}}(\boldsymbol{f}) = \sum_{\langle \boldsymbol{\ell}, \boldsymbol{m} \rangle} p_{\boldsymbol{\theta}}(\boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m}), \tag{2}$$

where we marginalize out the latent lemmata and morphological tags from raw text. The sum in Eq. (2) is unabashedly intractable—given a sequence $\boldsymbol{f}$, it involves consideration of an exponential (in $|\boldsymbol{f}|$) number of tag sequences and an *infinite* number of lemmata sequences. Thus, we will fall back on an approximation scheme (see §5).

4 Recurrent Neural Parameterization

The graphical model from §3 specifies a family of models that obey the conditional independence assumptions dictated by the graph in Fig. 1. In this section we define a specific parameterization using long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997) language models (Sundermeyer et al., 2012).

4.1 LSTM Language Models

Before proceeding, we review the modeling of sequences with LSTM language models. Given some alphabet Δ, the distribution over sequences $\mathbf{x} \in \Delta^*$ can be defined as follows:

$$p(\mathbf{x}) = \prod_{j=1}^{|\mathbf{x}|} p(x_j \mid \mathbf{x}_{<j}), \tag{3}$$

where $\mathbf{x}_{<j} = x_1, \ldots, x_{j-1}$. The prediction at time step j of a single element x_j is then parametrized by a neural network:

$$p(x_j \mid \mathbf{x}_{<j}) = \text{softmax}\left(\mathbf{W} \cdot \mathbf{h}_j + \mathbf{b}\right), \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{|\Delta| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|\Delta|}$ are learned parameters (for some number of hidden units d) and the hidden state $\mathbf{h}_j \in \mathbb{R}^d$ is defined through the recurrence given by Hochreiter and Schmidhuber (1997) from the previous hidden state and an embedding of the previous character (assuming some learned embedding function $\mathbf{e}\colon \Delta \to \mathbb{R}^c$ for some number of dimensions c):

$$\mathbf{h}_j = \text{LSTM}\big(\mathbf{h}_{j-1},\, \mathbf{e}(x_{j-1})\big) \tag{5}$$

4.2 Our Conditional Distributions

We discuss each of the factors in Eq. (1) in turn.

① **Morphological Tag Language Model: $p_{\boldsymbol{\theta}}(\boldsymbol{m})$.** We define $p_{\boldsymbol{\theta}}(\boldsymbol{m})$ as an LSTM language model, as defined in §4.1, where we take $\Delta = \mathcal{M}$, i.e., the elements of the sequence that are to be predicted are tags like $\left[\text{POS=V, TNS=GERUND}\right]$. Note that the embedding function $\mathbf{e}$ does not treat them as atomic units, but breaks them up into individual attribute-value pairs that are embedded individually and then summed to yield the final vector representation. To be precise, each tag is first encoded by a multi-hot vector, where each component corresponds to a attribute-value pair in the slot, and then this multi-hot vector is multiplied with an embedding matrix.

② **Lemma Generator: $p_{\boldsymbol{\theta}}(\ell_i \mid m_i)$.** The next distribution in our model is a lemma generator which we define to be a *conditional* LSTM language model over characters (we take $\Delta = \Sigma$), i.e., each x_i is a single (orthographic) character. The language model is conditioned on t_i (the part-of-speech information contained in the morphological tag $m_i = \langle t_i, \sigma_i \rangle$), which we embed into a low-dimensional space and feed to the LSTM by concatenating its embedding with that of the current character. Thusly, we obtain the new recurrence relation for the hidden state:

$$\mathbf{h}_j = \text{LSTM}\Big(\mathbf{h}_{j-1},\, \Big[\,\mathbf{e}\big([\ell_i]_{j-1}\big)\,;\,\mathbf{e}'\big(t_i\big)\,\Big]\Big), \tag{6}$$

where $[\ell_i]_j$ denotes the j^{th} character of the generated lemma ℓ_i and $\mathbf{e}' : \mathcal{T} \to \mathbb{R}^{c'}$ for some c' is a learned embedding function for POS tags. Note that we embed only the POS tag, rather than the entire morphological tag, as we assume the lemma depends on the part of speech exclusively.

③ **Morphological Inflector: $p_{\boldsymbol{\theta}}(f_i \mid \ell_i, m_i)$.** The final conditional in our model is a morphological inflector, which we parameterize as a neural recurrent sequence-to-sequence model (Sutskever et al., 2014) with Luong dot-style attention (Luong et al., 2015). Our particular model uses a single encoder-decoder architecture (Kann and Schütze, 2016) for all tag pairs within a language and we refer to reader to that paper for further details. Concretely, the encoder runs over a string consisting of the desired slot and all characters of the lemma that is to be inflected (e.g. `<w> V PST t a l k </w>`), one LSTM running left-to-right, the other right-to-left. Concatenating the hidden states of both RNNs at each time step results in hidden states $\mathbf{h}_j^{(enc)}$. The decoder, again, takes the form of an LSTM language model (we take $\Delta = \Sigma$), producing the inflected form character by character, but at each time step not only the previous hidden state and the previously generated token are considered, but attention (a convex combination) over all encoder hidden states $\mathbf{h}_j^{(\text{enc})}$, with the distribution given by another neural network; see Luong et al. (2015).

5 Semi-Supervised Wake-Sleep

We train the model with the wake-sleep procedure, which requires us to perform posterior inference over the latent variables. However, the exact computation in the model is intractable—it involves a sum over all possible lemmatizations and taggings of the sentence, as shown in Eq. (2). Thus, we fall back on a variational approximation (Jordan et al., 1999). We train an **inference network** $q_{\boldsymbol{\phi}}(\boldsymbol{\ell}, \boldsymbol{m} \mid \boldsymbol{f})$ that approximates the true posterior over the latent variables $p_{\boldsymbol{\theta}}(\boldsymbol{\ell}, \boldsymbol{m} \mid \boldsymbol{f})$.[5] The variational family we choose in this work will be

[5]Inference networks are also known as **stochastic inverses** (Stuhlmüller et al., 2013) or **recognition models** (Dayan et al., 1995).

detailed in §5.5. We fit the distribution q_{ϕ} using a semi-supervised extension of the wake-sleep algorithm (Hinton et al., 1995; Dayan et al., 1995; Bornschein and Bengio, 2014). We derive the algorithm in the following subsections and provide pseudo-code in Alg. 1.

Note that the wake-sleep algorithm shows structural similarities to the expectation-maximization (EM) algorithm (Dempster et al., 1977), and, presaging the exposition, we note that the wake-sleep procedure is a type of variational EM (Beal, 2003). The key difference is that the E-step minimizes an inclusive KL divergence, rather than the exclusive one typically found in variational EM.

5.1 Data Requirements of Wake-Sleep

We emphasize again that we will train our model in a semi-supervised fashion. Thus, we will assume a set of labeled sentences, $\mathcal{D}_{labeled}$, represented as a set of triples $\langle \boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m} \rangle$, and a set of unlabeled sentences, $\mathcal{D}_{unlabeled}$, represented as a set of surface form sequences $\boldsymbol{f}$.

5.2 The Sleep Phase

Wake-sleep first dictates that we find an approximate posterior distribution q_{ϕ} that minimizes the KL divergences for all form sequences:

$$D_{KL}\Big(\underbrace{p_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)}_{\textit{full joint: Eq. (1)}} \;||\; \underbrace{q_{\phi}(\cdot,\cdot\mid\cdot)}_{\textit{variational approximation}} \Big) \quad (7)$$

with respect to the parameters ϕ, which control the variational approximation q_{ϕ}. Because q_{ϕ} is trained to be a variational approximation for *any* input $\boldsymbol{f}$, it is called an inference network. In other words, it will return an approximate posterior over the latent variables for any observed sequence. Importantly, note that computation of Eq. (7) is still hard—it requires us to normalize the distribution $p_{\boldsymbol{\theta}}$, which, in turn, involves a sum over all lemmatizations and taggings. However, it does lend itself to an efficient Monte Carlo approximation. As our model is fully generative and directed, we may easily take samples from the complete joint. Specifically, we will take K samples $\langle \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle \sim p_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)$ by forward sampling and define them as $\widetilde{\mathcal{D}}_{sleep}$. We remark that we use a tilde to indicate that a form, lemmata or tag is sampled, rather than human annotated. Using K samples, we obtain the objective

$$\mathcal{S}_{unsup} = {}^{1}\!/_{K} \cdot \sum_{\langle \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle \in \widetilde{\mathcal{D}}_{sleep}} \log q_{\phi}(\tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \mid \tilde{\boldsymbol{f}}), \quad (8)$$

which we could maximize by fitting the model q_{ϕ} through backpropagation (Rumelhart et al., 1986), as one would during maximum likelihood estimation.

5.3 The Wake Phase

Now, given our approximate posterior $q_{\phi}(\boldsymbol{\ell}, \boldsymbol{m} \mid \boldsymbol{f})$, we are in a position to re-estimate the parameters of the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m})$. Given a set of unannotated sentences $\mathcal{D}_{unlabeled}$, we again first consider the objective

$$\mathcal{W}_{unsup} = {}^{1}\!/_{M} \cdot \sum_{\langle \boldsymbol{f}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle \in \widetilde{\mathcal{D}}_{wake}} \log p_{\boldsymbol{\theta}}(\boldsymbol{f}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}}) \quad (9)$$

where $\widetilde{\mathcal{D}}_{wake}$ is a set of triples $\langle \boldsymbol{f}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle$ with $\boldsymbol{f} \in \mathcal{D}_{unlabeled}$ and $\langle \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle \sim q_{\phi}(\cdot,\cdot \mid \boldsymbol{f})$, maximizing with respect to the parameters $\boldsymbol{\theta}$ (we may stochastically backprop through the expectation simply by backpropagating through this sum). Note that Eq. (9) is a Monte Carlo approximation of the inclusive divergence of the data distribution of $\mathcal{D}_{unlabeled}$ times q_{ϕ} with $p_{\boldsymbol{\theta}}$.

5.4 Adding Supervision to Wake-Sleep

So far we presented a purely unsupervised training method that makes no assumptions about the latent lemmata and morphological tags. In our case, however, we have a very clear idea what the latent variables should look like. For instance, we are quite certain that the lemma of talking is talk and that it is in fact a GERUND. And, indeed, we have access to annotated examples $\mathcal{D}_{labeled}$ in the form of an annotated corpus. In the presence of these data, we optimize the supervised sleep phase objective,

$$\mathcal{S}_{sup} = {}^{1}\!/_{N} \cdot \sum_{\langle \boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m} \rangle \in \mathcal{D}_{labeled}} \log q_{\phi}(\boldsymbol{\ell}, \boldsymbol{m} \mid \boldsymbol{f}). \quad (10)$$

which is a Monte Carlo approximation of $D_{KL}(\mathcal{D}_{labeled} \mid\mid q_{\phi})$. Thus, when fitting our variational approximation q_{ϕ}, we will optimize a joint objective $\mathcal{S} = \mathcal{S}_{sup} + \gamma_{sleep} \cdot \mathcal{S}_{unsup}$, where $\mathcal{S}_{sup}$, to repeat, uses actual annotated lemmata and morphological tags; we balance the two parts of the objective with a scaling parameter γ_{sleep}. Note that on the first sleep phase iteration, we set $\gamma_{sleep} = 0$ since taking samples from an untrained $p_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)$ when we have available labeled data is of little utility. We will discuss the provenance of our data in §7.2.

Likewise, in the wake phase we can neglect the approximation q_{ϕ} in favor of the annotated latent

Algorithm 1 semi-supervised wake-sleep

```
1: input $\mathcal{D}_{labeled}$                    ▷ labeled training data
2: input $\mathcal{D}_{unlabeled}$                  ▷ unlabeled training data
3: for $i = 1$ to $I$ do
4:     $\widetilde{\mathcal{D}}_{sleep} \leftarrow \emptyset$
5:     if $i > 1$ then
6:         for $k = 1$ to $K$ do
7:             $\langle \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle \sim p_{\boldsymbol{\theta}}(\cdot, \cdot, \cdot)$
8:             $\widetilde{\mathcal{D}}_{sleep} \leftarrow \widetilde{\mathcal{D}}_{sleep} \cup \{\langle \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle\}$
9:     maximize $\log q_{\boldsymbol{\phi}}$ on $\mathcal{D}_{labeled} \cup \widetilde{\mathcal{D}}_{sleep}$
                                  ▷ this corresponds to Eq. (10) + Eq. (8)
10:    $\widetilde{\mathcal{D}}_{wake} \leftarrow \emptyset$
11:    for $\boldsymbol{f} \in \mathcal{D}_{unlabeled}$ do
12:        $\langle \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle \sim q_{\boldsymbol{\phi}}(\cdot, \cdot \mid \boldsymbol{f})$
13:        $\widetilde{\mathcal{D}}_{wake} \leftarrow \widetilde{\mathcal{D}}_{wake} \cup \{\langle \boldsymbol{f}, \tilde{\boldsymbol{\ell}}, \tilde{\boldsymbol{m}} \rangle\}$
14:    maximize $\log p_{\boldsymbol{\theta}}$ on $\mathcal{D}_{labeled} \cup \widetilde{\mathcal{D}}_{wake}$
                                  ▷ this corresponds to Eq. (11) + Eq. (9)
```

variables found in $\mathcal{D}_{labeled}$; this leads to the following supervised objective

$$\mathcal{W}_{sup} = {}^{1}\!/\!{}_{N} \cdot \sum_{\langle \boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m} \rangle \in \mathcal{D}_{labeled}} \log p_{\boldsymbol{\theta}}(\boldsymbol{f}, \boldsymbol{\ell}, \boldsymbol{m}), \qquad (11)$$

which is a Monte Carlo approximation of $D_{KL}(\mathcal{D}_{labeled} \mid\mid p_{\boldsymbol{\theta}})$. As in the sleep phase, we will maximize $\mathcal{W} = \mathcal{W}_{sup} + \gamma_{wake} \cdot \mathcal{W}_{unsup}$, where γ_{wake} is, again, a scaling parameter.

5.5 Our Variational Family

How do we choose the variational family $q_{\boldsymbol{\phi}}$? In terms of NLP nomenclature, $q_{\boldsymbol{\phi}}$ represents a joint morphological tagger and lemmatizer. The open-source tool LEMMING (Müller et al., 2015) represents such an object. LEMMING is a higher-order linear-chain conditional random field (CRF; Lafferty et al., 2001), that is an extension of the morphological tagger of Müller et al. (2013). Interestingly, LEMMING is a linear model that makes use of simple character n-gram feature templates. On both the tasks of morphological tagging and lemmatization, neural models have supplanted linear models in terms of performance in the high-resource case (Heigold et al., 2017). However, we are interested in producing an accurate approximation to the posterior in the presence of minimal annotated examples and potentially noisy samples produced during the sleep phase, where linear models still outperform non-linear approaches (Cotterell and Heigold, 2017). We note that our variational approximation is compatible with any family.

5.6 Interpretation as an Autoencoder

We may also view our model as an autoencoder, following Kingma and Welling (2013), who saw that a variational approximation to any generative model naturally has this interpretation. The crucial difference between Kingma and Welling (2013) and this work is that our model is a **structured** variational autoencoder in the sense that the space of our latent code is structured: the inference network encodes a sentence into a pair of lemmata and morphological tags $\langle \boldsymbol{\ell}, \boldsymbol{m} \rangle$. This bisequence is then decoded back into the sequence of forms $\boldsymbol{f}$ through a morphological inflector. The reason the model is called an autoencoder is that we arrive at an auto-encoding-like objective if we combine the $p_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{\phi}}$ as so:

$$p(\boldsymbol{f} \mid \hat{\boldsymbol{f}}) = \sum_{\langle \boldsymbol{\ell}, \boldsymbol{m} \rangle} p_{\boldsymbol{\theta}}(\boldsymbol{f} \mid \boldsymbol{\ell}, \boldsymbol{m}) \cdot q_{\boldsymbol{\phi}}(\boldsymbol{\ell}, \boldsymbol{m} \mid \hat{\boldsymbol{f}}) \qquad (12)$$

where $\hat{\boldsymbol{f}}$ is a copy of the original sentence $\boldsymbol{f}$.

Note that this choice of latent space sadly precludes us from making use of the *reparametrization trick* that makes inference in VAEs particularly efficient. In fact, our whole inference procedure is quite different as we do not perform gradient descent on both $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$ jointly but alternatingly optimize both (using wake-sleep). We nevertheless call our model a VAE to uphold the distinction between the VAE as a model (essentially a specific Helmholtz machine (Dayan et al., 1995), justified by variational inference) and the end-to-end inference procedure that is commonly used.

Another way of viewing this model is that it tries to force the words in the corpus through a syntactic bottleneck. Spiritually, our work is close to the conditional random field autoencoder of Ammar et al. (2014).

We remark that many other structured NLP tasks can be “autoencoded” in this way and, thus, trained by a similar wake-sleep procedure. For instance, any two tasks that effectively function as inverses, e.g., translation and backtranslation, or language generation and parsing, can be treated with a similar variational autoencoder. While this work only focuses on the creation of an improved morphological inflector $p_{\boldsymbol{\theta}}(\boldsymbol{f} \mid \boldsymbol{\ell}, \boldsymbol{m})$, one could imagine a situation where the encoder was also a task of interest. That is, the goal would be to improve both the decoder (the generation model) *and* the encoder (the variational approximation).

6 Related Work

Closest to our work is Zhou and Neubig (2017), who describe an unstructured variational autoencoder. However, the exact use case of our respective models is distinct. Our method models the syntactic dynamics with an LSTM language model over morphological tags. Thus, in the semi-supervised setting, we require token-level annotation. Additionally, our latent variables are interpretable as they correspond to well-understood linguistic quantities. In contrast, Zhou and Neubig (2017) infer latent lemmata as real vectors. To the best of our knowledge, we are only the second attempt, after Zhou and Neubig (2017), to attempt to perform semi-supervised learning for a neural inflection generator. Other non-neural attempts at semi-supervised learning of morphological inflectors include Hulden et al. (2014). Models in this vein are non-neural and often focus on exploiting corpus statistics, e.g., token frequency, rather than explicitly modeling the forms in context. All of these approaches are designed to learn from a type-level lexicon, rendering direct comparison difficult.

7 Experiments

While we estimate all the parameters in the generative model, the purpose of this work is to improve the performance of morphological inflectors through semi-supervised learning with the incorporation of unlabeled data.

7.1 Low-Resource Inflection Generation

The development of our method was primarily aimed at the low-resource scenario, where we observe a limited number of annotated data points. Why low-resource? When we have access to a preponderance of data, morphological inflection is close to being a solved problem, as evinced in SIGMORPHON's 2016 shared task. However, the CoNLL-SIGMORPHON 2017 shared task showed there is much progress to be made in the low-resource case. Semi-supervision is a clear avenue.

7.2 Data

As our model requires *token-level* morphological annotation, we perform our experiments on the Universal Dependencies (UD) dataset (Nivre et al., 2017). As this stands in contrast to most work on morphological inflection (which has used the UniMorph (Sylak-Glassman et al., 2015)[6] datasets), we use a converted version of UD data, in which the UD morphological tags have been deterministically converted into UniMorph tags.

For each of the treebanks in the UD dataset, we divide the training portion into three chunks consisting of the first 500, 1000 and 5000 tokens, respectively. These *labeled* chunks will constitute three unique sets $\mathcal{D}_{labeled}$. The remaining sentences in the training portion will be used as *unlabeled* data $\mathcal{D}_{unlabeled}$ for each language, i.e., we will discard those labels. The development and test portions will be left untouched.

Languages. We explore a typologically diverse set of languages of various stocks: Indo-European, Afro-Asiatic, Turkic and Finno-Ugric, as well as the language isolate Basque. We have organized our experimental languages in Tab. 3 by genetic grouping, highlighting sub-families where possible. The Indo-European languages mostly exhibit fusional morphologies of varying degrees of complexity. The Basque, Turkic, and Finno-Ugric languages are agglutinative. Both of the Afro-Asiatic languages, Arabic and Hebrew, are Semitic and have templatic morphology with fusional affixes.

7.3 Evaluation

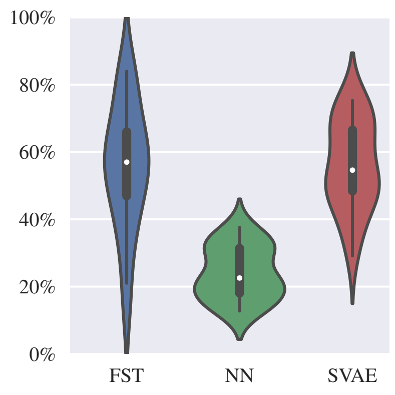
The end product of our procedure is a morphological inflector, whose performance is to be improved through the incorporation of unlabeled data. Thus, we evaluate using the standard metric accuracy. We will evaluate at the *type level*, as is traditional in the morphological inflection literature, even though the UD treebanks on which we evaluate are *token-level* resources. Concretely, we compile an incomplete type-level morphological lexicon from the token-level resource. To create this resource, we gather all unique form-lemma-tag triples $\langle f, \ell, m\rangle$ present in the UD test data.[7]

7.4 Baselines

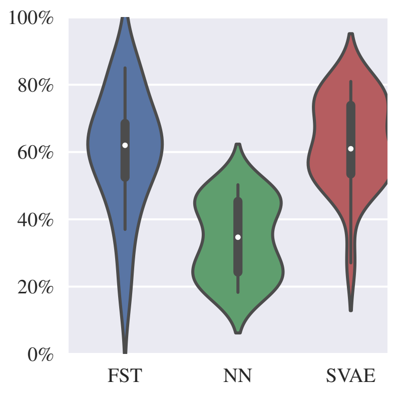
As mentioned before, most work on morphological inflection has considered the task of estimating statistical inflectors from type-level lexicons. Here, in

[6]The two annotation schemes are similar. For a discussion, we refer the reader to `http://universaldependencies.org/v2/features.html`; sadly there are differences that render all numbers reported in this work incomparable with previous work, see §7.4.

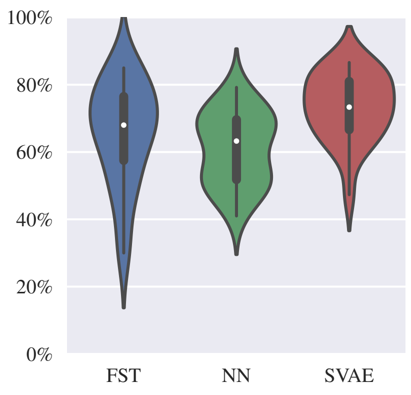
[7]Some of these form-lemma-tag triples will overlap with those seen in the training data.



(a) 500 training tokens



(b) 1000 training tokens



(c) 5000 training tokens

Figure 2: Violin plots showing the distribution over accuracies. The structured variational autoencoder (SVAE) always outperforms the neural network (NN), but only outperformed the FST-based approach when trained on 5000 annotated tokens. Thus, while semi-supervised training helps neural models reduce their sample complexity, roughly 5000 annotated tokens are still required to boost their performance above more symbolic baselines.

contrast, we require token-level annotation to estimate our model. For this reason, there is neither a competing approach whose numbers we can make a fair comparison to nor is there an open-source system we could easily run in the token-level setting. This is why we treat our token-level data as a list of "types"[8] and then use two simple type-based baselines.

First, we consider the probabilistic finite-state transducer used as the baseline for the CoNLL-SIGMORPHON 2017 shared task.[9] We consider this a relatively strong baseline, as we seek to generalize from a minimal amount of data. As described by Cotterell et al. (2017), the baseline performed quite competitively in the task's low-resource setting. Note that the finite-state machine is created by heuristically extracting prefixes and suffixes from the word forms, based on an unsupervised alignment step. The second baseline is our neural inflector $p(f \mid \ell, m)$ given in §4 *without* the semi-supervision; this model is state-of-the-art on the high-resource version of the task.

We will refer to our baselines as follows: **FST** is the probabilistic transducer, **NN** is the neural sequence-to-sequence model *without* semi-supervision, and **SVAE** is the structured variational autoencoder, which is equivalent to **NN** but also trained using wake-sleep and unlabeled data.

[8]Typical type-based inflection lexicons are likely not i.i.d. samples from natural utterances, but we have no other choice if we want to make use of only our token-level data and not additional resources like frequency and regularity of forms.

[9]`https://sites.google.com/view/conll-sigmorphon2017/`

7.5 Results

We ran the three models on 23 languages with the hyperparameters and experimental details described in App. A. We present our results in Fig. 2 and in Tab. 3. We also provide sample output of the generative model created using the dream step in App. B. The high-level take-away is that on almost all languages we are able to exploit the unlabeled data to improve the sequence-to-sequence model using unlabeled data, i.e., SVAE outperforms the NN model on *all* languages across *all* training scenarios. However, in many cases, the FST model is a better choice—the FST can sometimes generalize better from a handful of supervised examples than the neural network, even with semi-supervision (SVAE). We highlight three finer-grained observations below.

Observation 1: FST Good in Low-Resource. As clearly evinced in Fig. 2, the baseline FST is still competitive with the NN, or even our SVAE when data is extremely scarce. Our neural architecture is quite general, and lacks the prior knowledge and inductive biases of the rule-based system, which become more pertinent in low-resource scenarios. Even though our semi-supervised strategy clearly improves the performance of NN, we cannot always recommend SVAE for the case when we only have 500 annotated tokens, but on average it does slightly better. The SVAE surpasses the FST when moving up to 1000 annotated tokens, becoming even more pronounced at 5000 annotated tokens.

		500 tokens					1000 tokens					5000 tokens				
	lang	FST	NN	SVAE	Δ_{FST}	Δ_{NN}	FST	NN	SVAE	Δ_{FST}	Δ_{NN}	FST	NN	SVAE	Δ_{FST}	Δ_{NN}
Romance	ca	81.0	28.11	71.76	-9.24	43.65	85.0	42.58	78.46	-6.54	35.88	84.0	74.22	85.77	1.77	11.55
	fr	84.0	36.25	74.75	-9.25	38.5	85.0	47.04	79.97	-5.03	32.93	85.0	79.21	83.96	-1.04	4.75
	it	81.0	31.30	67.48	-13.52	36.18	81.0	43.58	77.37	-3.63	33.79	82.0	71.09	73.11	-8.89	2.02
	la	21.0	14.02	29.12	8.12	15.10	26.0	19.62	27.06	1.06	7.44	30.0	41.00	47.32	17.32	6.32
	pt	81.0	31.58	72.54	-8.46	40.96	83.0	47.27	73.24	-9.76	25.97	82.0	74.17	86.13	4.13	11.96
	ro	56.0	22.56	52.48	-3.52	29.92	62.0	34.68	58.30	-3.70	23.62	68.0	51.77	75.49	7.49	23.72
	es	57.0	34.34	75.32	18.32	40.98	60.0	46.14	80.97	20.97	34.83	72.0	71.99	84.44	12.44	12.45
Germanic	nl	63.0	19.22	49.14	-13.86	29.92	65.0	26.05	53.12	-11.88	27.07	70.0	53.70	65.97	-4.03	12.27
	da	68.0	31.25	65.58	-2.42	34.33	73.0	44.51	72.82	-0.18	28.31	79.0	67.92	80.12	1.12	12.20
	no	69.0	32.51	65.46	-3.54	32.95	71.0	46.26	74.49	3.49	28.23	79.0	71.31	81.25	2.25	9.94
	nn	64.0	20.29	54.62	-9.38	34.33	65.0	24.32	60.97	-4.03	36.65	72.0	50.40	73.35	1.35	22.95
	sv	63.0	19.02	58.15	-4.85	39.13	66.0	36.35	67.18	1.18	30.83	74.0	59.82	78.23	4.23	18.41
Slavic	bg	44.0	15.51	47.22	3.22	31.71	51.0	21.00	57.18	6.18	36.18	59.0	49.06	71.15	12.15	22.09
	pl	50.0	12.75	48.62	-1.38	35.87	57.0	19.88	55.90	-1.10	36.02	64.0	54.44	67.15	3.15	12.71
	si	52.0	15.60	55.69	3.69	40.09	61.0	26.39	61.22	0.22	34.83	68.0	66.65	75.40	7.40	8.75
Semit.	ar	14.0	31.47	63.53	49.53	32.06	17.0	48.53	71.52	54.52	22.99	34.0	68.16	80.72	46.72	12.56
	he	60.0	37.61	71.11	11.11	33.50	66.0	50.28	76.32	10.32	26.04	72.0	64.37	86.60	14.6	22.23
Finn.-Urg.	hu	53.0	22.56	48.64	-4.36	26.08	56.0	28.62	60.74	4.74	32.12	61.0	66.45	72.84	11.84	6.39
	et	39.0	21.81	42.16	3.16	20.35	45.0	29.66	51.75	6.75	22.09	49.0	46.82	58.91	9.91	12.09
	fi	37.0	12.97	35.78	-1.22	22.81	42.0	19.03	47.65	5.65	28.62	49.0	46.75	62.76	13.76	16.01
other	lv	57.0	17.16	48.29	-8.71	31.13	63.0	18.30	53.58	-9.42	35.28	66.0	51.84	66.12	0.12	14.28
	eu	50.0	24.46	48.72	-1.28	24.26	54.0	35.14	53.39	-0.61	18.25	56.0	56.29	62.33	6.33	6.04
	tr	34.0	20.67	37.92	3.92	17.25	37.0	24.33	49.67	12.67	25.34	48.0	63.26	69.35	21.35	6.09
	avg	55.57	24.04	**55.83**	0.26	31.79	59.61	33.89	**62.73**	3.12	6.90	65.35	60.90	**73.41**	8.06	12.51

Table 3: Type-level morphological inflection accuracy across different models, training scenarios, and languages

Observation 2: Agglutinative Languages. The next trend we remark upon is that languages of an agglutinating nature tend to benefit more from the semi-supervised learning. Why should this be? Since in our experimental set-up, every language sees the *same number of tokens*, it is naturally harder to generalize on languages that have more distinct morphological variants. Also, by the nature of agglutinative languages, relevant morphemes could be arbitrarily far from the edges of the string, making the (NN and) SVAE's ability to learn more generic rules even more valuable.

Observation 3: Non-concatenative Morphology. One interesting advantage that the neural models have over the FSTs is the ability to learn non-concatenative phenomena. The FST model is based on prefix and suffix rewrite rules and, naturally, struggles when the correctly reinflected form is more than the concatenation of these parts. Thus we see that for the two semitic language, the SVAE is the best method across all resource settings.

8 Conclusion

We have presented a novel generative model for morphological inflection generation in context. The model allows us to exploit unlabeled data in the training of morphological inflectors. As the model's rich parameterization prevents tractable inference, we craft a variational inference procedure, based on the wake-sleep algorithm, to marginalize out the latent variables. Experimentally, we provide empirical validation on 23 languages. We find that, especially in the lower-resource conditions, our model improves by large margins over the baselines.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1024–1029, Denver, CO. Association for Computational Linguistics.

Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Advances in Neural Information Processing Systems*, pages 3311–3319.

Mark Aronoff. 1976. *Word Formation in Generative Grammar*. Number 1 in Linguistic Inquiry Monographs. MIT Press, Cambridge, MA.

Matthew James Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. University College London.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jörg Bornschein and Yoshua Bengio. 2014. Reweighted wake-sleep. *CoRR*, abs/1406.2751.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759, Copenhagen, Denmark. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, Vancouver, Canada. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Peter Dayan, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. 1995. The Helmholtz machine. *Neural Computation*, 7(5):889–904.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1080–1089, Honolulu, Hawaii. Association for Computational Linguistics.

Matthew S. Dryer, David Gil, Bernard Comrie, Hagen Jung, Claudia Schmidt, et al. 2005. The world atlas of language structures.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 634–643, San Diego, California. Association for Computational Linguistics.

Shai Fine, Yoram Singer, and Naftali Tishby. 1998. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 505–513, Valencia, Spain. Association for Computational Linguistics.

Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. 1995. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

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

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578, Gothenburg, Sweden. Association for Computational Linguistics.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Aleksandr E. Kibrik. 1998. Archi. In Andrew Spencer and Arnold M. Zwicky, editors, *The Handbook of Morphology*, pages 455–476.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 922–931, Denver, Colorado. Association for Computational Linguistics.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, and Dobrovoljc. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics ('UFAL), Faculty of Mathematics and Physics, Charles University.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Andrew Spencer. 1991. *Morphological Theory: An Introduction to Word Structure in Generative Grammar*. Wiley-Blackwell.

Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. 2013. Learning stochastic inverses. In *Advances in Neural Information Processing Systems*, pages 3048–3056.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (Unimorph schema). Technical report, Johns Hopkins University.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 674–680, Beijing, China. Association for Computational Linguistics.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint:1212.5701*.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320, Vancouver, Canada. Association for Computational Linguistics.

Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, Joshua Maynez
Google Inc.
{bohnetbd,ryanmcd,gsimoes,andor,epitler,joshuahm}@google.com

Abstract

The rise of neural networks, and particularly recurrent neural networks, has produced significant advances in part-of-speech tagging accuracy (Zeman et al., 2017). One characteristic common among these models is the presence of rich initial word encodings. These encodings typically are composed of a recurrent character-based representation with learned and pre-trained word embeddings. However, these encodings do not consider a context wider than a single word and it is only through subsequent recurrent layers that word or sub-word information interacts. In this paper, we investigate models that use recurrent neural networks with sentence-level context for initial character and word-based representations. In particular we show that optimal results are obtained by integrating these context sensitive representations through synchronized training with a meta-model that learns to combine their states. We present results on part-of-speech and morphological tagging with state-of-the-art performance on a number of languages.

1 Introduction

Morphosyntactic tagging accuracy has seen dramatic improvements through the adoption of recurrent neural networks—specifically BiLSTMs (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005) to create sentence-level context sensitive encodings of words. A successful recipe is to first create an initial context insensitive word representation, which usually has three main parts: 1) A dynamically trained word embedding; 2) a fixed pre-trained word-embedding, induced from a large corpus; and 3) a sub-word character model, which itself is usually the final state of a recurrent model that ingests one character at a time. Such word/sub-word models originated with Plank et al. (2016). Recently, Dozat et al. (2017) used precisely such a context insensitive word representation as input to a BiLSTM in order to obtain context sensitive word encodings used to predict part-of-speech tags. The Dozat et al. model had the highest accuracy of all participating systems in the CoNLL 2017 shared task (Zeman et al., 2017).

In such a model, sub-word character-based representations only interact indirectly via subsequent recurrent layers. For example, consider the sentence *I had shingles, which is a painful disease*. Context insensitive character and word representations may have learned that for unknown or infrequent words like ‘shingles’, ‘s’ and more so ‘es’ is a common way to end a plural noun. It is up to the subsequent BiLSTM layer to override this once it sees the singular verb is to the right. Note that this differs from traditional linear models where word and sub-word representations are directly concatenated with similar features in the surrounding context (Giménez and Marquez, 2004).

In this paper we aim to investigate to what extent having initial sub-word and word context insensitive representations affects performance. We propose a novel model where we learn context sensitive initial character and word representations through two separate sentence-level recurrent models. These are then combined via a meta-BiLSTM model that builds a unified representation of each word that is then used for syntactic tagging. Critically, while each of these three models—character, word and meta—are trained synchronously, they are ultimately separate models using different network configurations, training hyperparameters and loss functions. Empirically, we found this optimal as it allowed control over the fact that each representation has a different learning capacity.

We tested the system on the 2017 CoNLL shared task data sets and gain improvements compared to the top performing systems for the majority of languages for part-of-speech and morphological tagging. As we will see, a pattern emerged where gains were largest for morphologically rich languages, especially those in the Slavic family group. We also applied the approach to the benchmark English PTB data, where our model achieved 97.9 using the standard train/dev/test split, which constitutes a relative reduction in error of 12% over the previous best system.

2 Related Work

While sub-word representations are often attributed to the advent of deep learning in NLP, it was, in fact, commonplace for linear featurized machine learning methods to incorporate such representations. While the literature is too large to enumerate, Giménez and Marquez (2004) is a good example of an accurate linear model that uses both word and sub-word features. Specifically, like most systems of the time, they use n-gram affix features, which were made context sensitive via manually constructed conjunctions with features from other words in a fixed window.

Collobert and Weston (2008) was perhaps the first modern neural network for tagging. While this first study used only word embeddings, a subsequent model extended the representation to include suffix embeddings (Collobert et al., 2011).

The seminal dependency parsing paper of Chen and Manning (2014) led to a number of tagging papers that used their basic architecture of highly featurized (and embedded) feed-forward neural networks. Botha et al. (2017), for example, studied this architecture in a low resource setting using word, sub-word (prefix/suffix) and induced cluster features to obtain competitive accuracy with the state-of-the-art. Zhou et al. (2015), Alberti et al. (2015) and Andor et al. (2016) extended the work of Chen et al. to a structured prediction setting, the later two use again a mix of word and sub-word features.

The idea of using a recurrent layer over characters to induce a complementary view of a word has occurred in numerous papers. Perhaps the earliest is Santos and Zadrozny (2014) who compare character-based LSTM encodings to traditional word-based embeddings. Ling et al. (2015) take this a step further and combine the word embeddings with a recurrent character encoding of the word—instead of just relying on one or the other. Alberti et al. (2017) use characters encodings for parsing. Peters et al. (2018) show that contextual embeddings using character convolutions improve accuracy for number of NLP tasks. Plank et al. (2016) is probably the jumping-off point for most current architectures for tagging models with recurrent neural networks. Specifically, they used a combined word embedding and recurrent character encoding as the initial input to a BiLSTM that generated context sensitive word encodings. Though, like most previous studies, these initial encodings were context insensitive and relied on subsequent layers to encode sentence-level interactions.

Finally, Dozat et al. (2017) showed that subword/word combination representations lead to state-of-the-art morphosyntactic tagging accuracy across a number of languages in the CoNLL 2017 shared task (Zeman et al., 2017). Their word representation consisted of three parts: 1) A dynamically trained word embedding; 2) a fixed pretrained word embedding; 3) a character LSTM encoding that summed the final state of the recurrent model with vector constructed using an attention mechanism over all character states. Again, the initial representations are all context insensitive. As this model is currently the state-of-the-art in morphosyntactic tagging, it will serve as a baseline during our discussion and experiments.

3 Models

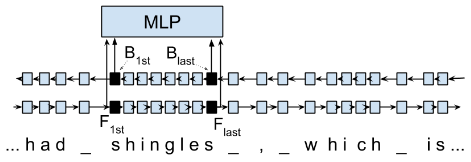
In this section, we introduce models that we investigate and experiment with in §4.

3.1 Sentence-based Character Model

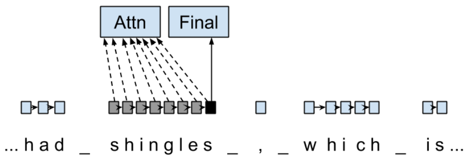
The feature that distinguishes our model most from previous work is that we apply a bidirectional recurrent layer (LSTM) on all characters of a sentence to induce fully context sensitive initial word encodings. That is, we do not restrict the context of this layer to the words themselves (as in Figure 1b). Figure 1a shows the sentence-based character model applied to an example token in context.

The character model uses, as input, sentences split into UTF8 characters. We include the spaces between the tokens[1] in the input and map each

[1]As input, we assume the sentence has been tok-



(a) Sentence-based Character Model. The representation for the token *shingles* is the concatenation of the four shaded boxes. Note the surrounding sentence context affects the representation.



(b) Token-based Character Model[a]. The token is represented by the concatenation of attention over the lightly shaded boxes with the final cell (dark shaded box). The rest of the sentence has no impact on the representation.

[a]This is specifically the model of Dozat et al. (2017).

Figure 1: Token representations are sensitive to the context in the sentence-based character model (§3.1) and are context-independent in the token-based character model (§3.2).

character to a dynamically learned embedding.

Next, a forward LSTM reads the characters from left to right and a backward LSTM reads sentences from right to left, in standard BiLSTM fashion.

More formally, for an n-character sentence, we apply for each character embedding $(e_1^{char}, ..., e_n^{char})$ a BiLSTM:

$$f_{c,i}^0, b_{c,i}^0 \quad = \quad \text{BiLSTM}(r_0, (e_1^{char}, ..., e_n^{char}))_i$$

As is also typical, we can have multiple such layers (l) that feed into each other through the concatenation of previous layer encodings. The last layer l has both forward $(f_{c,1}^l, ..., f_{c,n}^l)$ and backward $(b_{c,1}^l, ..., b_{c,n}^l)$ output vectors for each character.

To create word encodings, we need to combine a relevant subset of these context sensitive character encodings. These word encodings can then be used in a model that assigns morphosyntactic tags to each word directly or via subsequent layers. To accomplish this, the model concatenates up to four character output vectors: the {*forward*, *backward*} output of the {*first*, *last*} character in the token $(F_{1st}(w), F_{last}(w), B_{1st}(w), B_{last}(w))$. In Figure 1a, the four shaded boxes indicate these four outputs for the example token. Thus, the proposed model concatenates all four of these and passes it as input to an multilayer perceptron (MLP):

$$\begin{aligned} g_i \quad &= \quad \text{concat}(F_{1st}(w), F_{last}(w), \\ &\qquad B_{1st}(w), B_{last}(w)) \qquad (1) \\ m_i^{chars} \quad &= \quad \text{MLP}(g_i) \end{aligned}$$

A tag can then be predicted with a linear classifier that takes as input the output of the MLP m_i^{chars}, applies a softmax function and chooses for each word the tag with highest probability. Table 8 investigates the empirical impact of alternative definitions of g_i that concatenate only subsets of $\{F_{1st}(w), F_{last}(w), B_{1st}(w), B_{last}(w)\}$.

enized/segmented.

3.2 Word-based Character Model

To investigate whether a sentence sensitive character model is better than a character model where the context is restricted to the characters of a word, we reimplemented the word-based character model of Dozat et al. (2017) as shown in Figure 1a. This model uses the final state of a unidirectional LSTM over the characters of the word, combined with the attention mechanism of Cao and Rei (2016) over all characters. We refer the reader to those works for more details. Critically, however, all the information fed to this representation comes from the word itself, and not a wider sentence-level context.

3.3 Sentence-based Word Model

We used a similar setup for our context sensitive word encodings as the character encodings. There are a few differences. Obviously, the inputs are the words of the sentence. For each of the words, we use pretrained word embeddings $(p_1^{word}, ..., p_n^{word})$ summed with a dynamically learned word embedding for each word in the corpus $(e_1^{word}, ..., e_n^{word})$:

$$in_i^{word} = e_i^{word} + p_i^{word}$$

The summed embeddings in_i are passed as input to one or more BiLSTM layers whose output $f_{w,i}^l, b_{w,i}^l$ is concatenated and used as the final encoding, which is then passed to an MLP

$$o_i^{word} = \text{concat}(f_{w,i}^l, b_{w,i}^l)$$
$$m_i^{word} = \text{MLP}(o_i^{word})$$

It should be noted, that the output of this BiLSTM is essentially the Dozat et al. model before tag prediction, with the exception that the word-based character encodings are excluded.

3.4 Meta-BiLSTM: Model Combination

Given initial word encodings, both character and word-based, a common strategy is to pass these through a sentence-level BiLSTM to create context sensitive encodings, e.g., this is precisely what Plank et al. (2016) and Dozat et al. (2017) do. However, we found that if we trained each of the character-based and word-based encodings with their own loss, and combined them using an additional meta-BiLSTM model, we obtained optimal performance. In the meta-BiLSTM model, we concatenate the output, for each word, of its context sensitive character and word-based encodings, and put this through another BiLSTM to create an additional combined context sensitive encoding. This is followed by a final MLP whose output is passed to a linear layer for tag prediction.

$$\begin{aligned} cw_i &= \text{concat}(m_i^{char}, m_i^{word}) \\ f_{m,i}^l, b_{m,i}^l &= \text{BiLSTM}(r_0, (cw_0, ..., cw_n))_i \\ m_i^{comb} &= \text{MLP}(\text{concat}(f_{m,i}^l, b_{m,i}^l)) \end{aligned}$$

With this setup, each of the models can be optimized independently which we describe in more detail in §3.5. Figure 2b depicts the architecture of the combined system and contrasts it with that of the Dozat et al. model (Figure 2a).

3.5 Training Schema

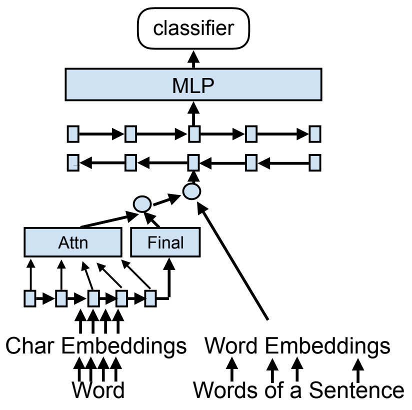
As mentioned in the previous section, the character and word-based encoding models have their own tagging loss functions, which are trained independently and joined via the meta-BiLSTM. I.e., the loss of each model is minimized independently by separate optimizers with their own hyperparameters. Thus, it is in some sense a multi-task learning model and we must define a schedule in which individual models are updated. We opted for a simple synchronous schedule outline in Algorithm 1. Here, during each epoch, we update each of the models in sequence—character, word and meta—using the entire training data.

In terms of model selection, after each epoch, the algorithm evaluates the tagging accuracy of the development set and keeps the parameters of the best model. Accuracy is measured using the meta-BiLSTM tagging layer, which requires a forward pass through all three models. Though we use all three losses to update the models, only the meta-BiLSTM layer is used for model selection and test-time prediction.

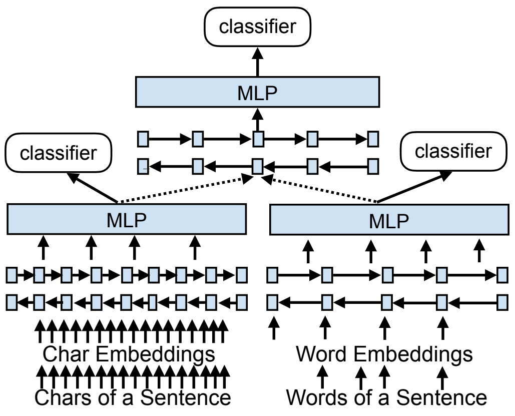
```
Data: train-corpus, dev-corpus
/* The following models are defined
   in §3.  */
Input: char-model, word-model, meta-model
/* Model optimizers */
Input: char-opt, word-opt, meta-opt
/* Results are parameter sets for
   each model.  */
Result: best-char, best-word, best-meta
/* Initialize parameter sets (cf.
   Table 1) */
Initialize($pa_c, pa_w, pa_m$)
/* Iteration on over training
   corpus.  */
for epoch = 1 to MAX do
    /* Update character model.  */
    char-logits, char-preds =
      char-model(train-corpus, $pa_c$)
    $pa_c$ = char-opt.update(char-preds, train-data)
    /* Update word model.  */
    word-logits, word-preds =
      word-model(train-corpus, $pa_w$)
    $pa_w$ = word-opt.update(char-preds, train-data)
    /* Update Meta-BiLSTM model.  */
    meta-preds = meta-model(train-corpus,
      $pa_c, pa_w, pa_m$)
    $pa_m$ = meta-opt.update(train-corpus,
      meta-preds)
    /* Evaluate model due to dev set
       accuracy.  */
    F1 = DevEval($par_c, par_w, par_m$)
    /* Keep the best model.  */
    if F1 > best-F1 then
        best-char = $pa_c$; best-word = $pa_w$
        best-meta = $pa_m$; best-F1 = F1
    end
end
```

Algorithm 1: Training procedure for learning initial character and word-based context sensitive encodings synchronously with meta-BiLSTM.

While each of the three models—character, word and meta—are trained with their own loss functions, it should be emphasized that training is synchronous in the sense that the meta-BiLSTM model is trained in tandem with the two encoding models, and not after those models have converged. Since accuracy from the meta-BiLSTM model on the development set determines the best parameters, training is not completely independent. We found this to improve accuracy overall. Crucially, when we allowed the meta-BiLSTM to back-propagate through the whole network, performance degraded regardless of whether one or multiple loss functions were used.



(a) The overall architecture of Dozat et al. (2017)



(b) The overall architecture of the system. The data flows along the arrows. The optimizers minimizes the loss of the classifiers independently and backpropagates along the bold arrows.

Figure 2: Tagging architectures. (a) Dozat et al. (2017); (b) Meta-BiLSTM architecture of this work.

Each language could in theory use separate hyperparameters, optimized for highest accuracy. However, for our main experiments we use identical settings for each language which worked well for large corpora and simplified things. We provide an overview of the selected hyperparameters in §4.1. We explored more settings for selected individual languages with a grid search and ablation experiments and present the results in §5.

4 Experiments and Results

In this section, we present the experimental setup and the selected hyperparameter for the main experiments where we use the CoNLL Shared Task 2017 treebanks and compare with the best systems of the shared task.

4.1 Experimental Setup

For our main results, we selected one network configuration and set of the hyperparameters. These settings are not optimal for all languages. However, since hyperparameter exploration is computationally demanding due to the number of languages we optimized these hyperparameters on initial development data experiments over a few languages. Table 1 shows an overview of the architecture, hyperparameters and the initialization settings of the network. The word embeddings are initialized with zero values and the pre-trained embeddings are not updated during training. The dropout used on the embeddings is achieved by a single dropout mask and we use dropout on the input and the states of the LSTM.

Architecture		
Model	**Parameter**	**Value**
Chr, Wrd	BiLSTM layers	3
Mt	BiLSTM layers	1
Chr, Wrd, Mt	BiLSTM size	400
Chr, Wrd, Mt	Dropout LSTMs	0.33
Chr, Wrd, Mt	Dropout MLP	0.33
Wrd	Dropout embeddings	0.33
Chr	Dropout embeddings	0.05
Chr, Wrd, Mt	Nonlinear act. (MLP)	ELU
Initialization		
Model	**Parameter**	**Value**
Wrd	embeddings	Zero
Chr	embeddings	Gaussian
Chr, Wrd, Mt	MLP	Gaussian
Training		
Model	**Parameter**	**Value**
Chr, Wrd, Mt	Optimizer	Adam
Chr, Wrd, Mt	Loss	Cross entropy
Chr, Wrd, Mt	Learning rate	0.002
Chr, Wrd, Mt	Decay	0.999994
Chr, Wrd, Mt	Adam epsilon	1e-08
Chr, Wrd, Mt	beta1	0.9
Chr, Wrd, Mt	beta2	0.999

Table 1: Selected hyperparameters and initialization of parameters for our models. *Chr*, *Wrd*, and *Mt* are used to indicate the character, word, and meta models respectively. The Gaussian distribution is used with a mean of 0 and variance of 1 to generate the random values.

As is standard, model selection was done measuring development accuracy/F1 score after each epoch and taking the model with maximum value on the development set.

4.2 Data Sets

For the experiments, we use the data sets as provided by the CoNLL Shared Task 2017 (Zeman et al., 2017). For training, we use the training sets which were denoted as big treebanks [2].

We followed the same methodology used in the CoNLL Shared Task. We use the training treebank for training only and the development sets for hyperparameter tuning and early stopping. To keep our results comparable with the Shared Task, we use the provided precomputed word embeddings. We excluded Gothic from our experiments as the available downloadable content did not include embeddings for this language.

As input to our system—for both part-of-speech tagging and morphological tagging—we use the output of the UDPipe-base baseline system (Straka and Straková, 2017) which provides segmentation. The segmentation differs from the gold segmentation and impacts accuracy negatively for a number of languages. Most of the top performing systems for part-of-speech tagging used as input UDPipe to obtain the segmentation for the input data. For morphology, the top system for most languages (IMS) used its own segmentation (Björkelund et al., 2017). For the evaluation, we used the official evaluation script (Zeman et al., 2017).

4.3 Part-of-Speech Tagging Results

In this section, we present the results of the application of our model to part-of-speech tagging. In our first experiment, we used our model in the setting of the CoNLL 2017 Shared Task to annotate words with XPOS[3] tags (Zeman et al., 2017). We compare our results against the top systems of the CoNLL 2017 Shared Task. Table 2 contains the results of this task for the large treebanks.

Because Dozat et al. (2017) won the challenge for the majority of the languages, we first compare our results with the performance of their system. Our model outperforms Dozat et al. (2017) in 32 of the 54 treebanks with 13 ties. These ties correspond mostly to languages where XPOS tagging anyhow obtains accuracies above 99%. Our model tends to produce better results, especially for morphologically rich languages (e.g. Slavic

[2]In the CONLL 2017 Shared Task, a big treebank is one that contains a development set. In total, there are 55 out of the 64 UD treebanks which are considered big treebanks.

[3]These are the language specific fine-grained part-of-speech tags from the Universal Dependency Treebanks.

lang.	CONLL Winner	**DQM**	**ours**	**RRIE**
cs_cac	95.16	95.16	**96.91**	36.2
cs	95.86	95.86	**97.28**	35.5
fi	97.37	97.37	**97.81**	16.7
sl	94.74	94.74	**95.54**	15.2
la_ittb	94.79	94.79	**95.56**	14.8
grc	84.47	84.47	**86.51**	13.1
bg	96.71	96.71	**97.05**	10.3
ca	98.58	98.58	**98.72**	9.9
grc_proiel	97.51	97.51	**97.72**	8.4
pt	83.04	83.04	**84.39**	8.0
cu	96.20	96.20	**96.49**	7.6
it	97.93	97.93	**98.08**	7.2
fa	97.12	97.12	**97.32**	6.9
ru	96.73	96.73	**96.95**	6.7
sv	96.40	96.40	**96.64**	6.7
ko	93.02	93.02	**93.45**	6.2
sk	85.00	85.00	**85.88**	5.9
nl	90.61	90.61	**91.10**	5.4
fi_ftb	95.31	95.31	**95.56**	5.3
de	97.29	97.29	**97.39**	4.7
tr	93.11	93.11	**93.43**	4.6
hi	97.01	97.01	**97.13**	4.0
es_ancora	98.73	98.73	**98.78**	3.9
ro	96.98	96.98	**97.08**	3.6
la_proiel	96.93	96.93	**97.00**	2.3
pl	91.97	91.97	**92.12**	1.9
ar	87.66	87.66	**87.82**	1.3
gl	97.50	97.50	**97.53**	1.2
sv_lines	94.84	94.84	**94.90**	1.2
cs_clt	89.98	89.98	**90.09**	1.1
lv	80.05	80.05	**80.20**	0.8
zh	**88.40**	85.07	85.10	0.2
da	**100.00**	99.96	99.96	0.0
es	**99.81**	99.69	99.69	0.0
eu	**99.98**	99.96	99.96	0.0
fr_sequoia	**99.49**	99.06	99.06	0.0
fr	**99.50**	98.87	98.87	0.0
hr	**99.93**	**99.93**	**99.93**	0.0
hu	**99.85**	99.82	99.82	0.0
id	**100.00**	99.99	99.99	0.0
ja	**98.59**	89.68	89.68	0.0
nl_lassy	**99.99**	99.93	99.93	0.0
no_bok.	**99.88**	99.75	99.75	0.0
no_nyn.	**99.93**	99.85	99.85	0.0
ru_syn.	**99.58**	99.57	99.57	0.0
en_lines	**95.41**	**95.41**	95.39	-0.4
ur	**92.30**	**92.30**	92.21	-1.2
he	**83.24**	82.45	82.16	-1.7
vi	**75.42**	73.56	73.12	-1.7
gl_treegal	**91.65**	**91.65**	91.40	-3.0
en	**94.82**	**94.82**	94.66	-3.1
en_partut	**95.08**	**95.08**	94.81	-5.5
pt_br	**98.22**	**98.22**	98.11	-6.2
et	**95.05**	**95.05**	94.72	-6.7
el	**97.76**	**97.76**	97.53	-10.3
macro-avg	93.18	93.11	**93.40**	-

Table 2: Results for XPOS tags. The first column shows the language acronym, the column named **DQM** shows the results of Dozat et al. (2017). Our system outperforms Dozat et al. (2017) on 32 out of 54 treebanks and Dozat et al. outperforms our model on 10 of 54 treebanks, with 13 ties. **RRIE** is the relative reduction in error. We excluded ties in the calculation of macro-avg since these treebanks do not contain meaningful xpos tags.

System	**Accuracy**
Søgaard (2011)	97.50
Huang et al. (2015)	97.55
Choi (2016)	97.64
Andor et al. (2016).	97.44
Dozat et al. (2017)	97.41
ours	**97.96**

Table 3: Results on WSJ test set.

languages), whereas Dozat et al. (2017) showed higher performance in 10 languages in particular English, Greek, Brazilian Portuguese and Estonian.

4.4 Part-of-Speech Tagging on WSJ

We also performed experiments on the Penn Treebank with the usual split in train, development and test set. Table 3 shows the results of our model in comparison to the results reported in state-of-the-art literature. Our model significantly outperforms these systems, with an absolute difference of 0.32% in accuracy, which corresponds to a RRIE of 12%.

4.5 Morphological Tagging Results

In addition to the XPOS tagging experiments, we performed experiments with morphological tagging. This annotation was part of the CONLL 2017 Shared Task and the objective was to predict a bundle of morphological features for each token in the text. Our model treats the morphological bundle as one tag making the problem equivalent to a sequential tagging problem. Table 4 shows the results.

Our models tend to produce significantly better results than the winners of the CoNLL 2017 Shared Task (i.e., 1.8% absolute improvement on average, corresponding to a RRIE of 21.20%). The only cases for which this is not true are again languages that require significant segmentation efforts (i.e., Hebrew, Chinese, Vietnamese and Japanese) or when the task was trivial.

Given the fact that Dozat et al. (2017) obtained the best results in part-of-speech tagging by a significant margin in the CoNLL 2017 Shared Task, it would be expected that their model would also perform significantly well in morphological tagging since the tasks are very similar. Since they did not participate in this particular challenge, we decided to reimplement their system to serve

lang.	CONLL Winner	**DQM** Reimpl.	**ours**	**RRIE**
cs_cac	90.72	94.66	**96.41**	27.9
ru_syn.	94.55	96.70	**97.53**	23.1
cs	93.14	96.32	**97.14**	22.3
la_ittb	94.28	96.45	**97.12**	18.9
sl	90.08	95.26	**96.03**	16.2
ca	97.23	97.85	**98.13**	13.0
fi_ftb	93.43	95.96	**96.42**	11.4
no_bok.	95.56	96.95	**97.26**	10.2
grc_proiel	90.24	91.35	**92.22**	10.1
fr_sequoia	96.10	96.62	**97.62**	10.1
la_proiel	89.22	91.52	**92.35**	9.8
es_ancora	97.72	98.15	**98.32**	9.7
da	94.83	96.62	**96.94**	9.5
fi	92.43	94.29	**94.83**	9.5
sv	95.15	96.52	**96.84**	9.2
pt	94.62	95.89	**96.27**	9.2
grc	88.00	90.39	**91.13**	9.0
no_nyn.	95.25	96.79	**97.08**	9.0
de	83.11	89.78	**90.70**	9.0
ru	87.27	91.99	**92.69**	8.7
hi	91.03	90.72	**91.78**	8.1
cu	88.90	88.93	**89.82**	8.0
fa	96.34	97.23	**97.45**	7.9
tr	87.03	89.39	**90.21**	7.7
en_partut	92.69	93.93	**94.40**	7.7
sk	81.23	87.54	**88.48**	7.5
eu	89.57	92.48	**93.04**	7.4
pt_br	99.73	99.73	**99.75**	7.4
es	96.34	96.42	**96.68**	7.3
ko	99.41	99.44	**99.48**	7.1
ar	87.15	85.45	**88.29**	6.7
it	97.37	97.72	**97.86**	6.1
nl_lassy	97.55	98.04	**98.15**	5.2
nl	90.04	92.06	**92.47**	5.2
pl	86.53	91.71	**92.14**	5.2
ur	81.03	83.16	**84.02**	5.1
bg	96.47	97.71	**97.82**	4.8
hr	85.82	90.64	**91.50**	3.8
he	**85.06**	79.34	79.76	2.0
et	84.62	88.18	**88.25**	0.6
zh	**92.90**	87.67	87.74	0.6
vi	**86.92**	82.23	82.30	0.4
ja	**96.84**	89.65	89.66	0.1
en_lines	99.96	99.99	99.99	0.0
fr	96.12	95.98	95.98	0.0
gl	**99.78**	99.72	99.72	0.0
id	**99.55**	99.50	99.50	0.0
ro	96.24	97.26	97.26	0.0
sv_lines	99.98	99.98	99.98	0.0
cs_cltt	87.88	**90.41**	90.36	-0.5
lv	84.14	**87.00**	86.92	-0.6
el	91.37	**94.00**	93.92	-1.3
hu	72.61	**82.67**	82.44	-1.3
en	94.49	**95.93**	95.71	-5.4
macro-avg	91.51	92.89	**93.31**	-

Table 4: Results for morphological features. The column **CoNLL Winner** shows the top system of the ST 17, the **DQM Reimpl.** shows our reimplementation of Dozat et al. (2017), the column **ours** shows our system with a sentence-based character model; **RRIE** gives the relative reduction in error between the Reimpl. DQM and sentence-based character system. Our system outperforms the CoNLL Winner by 48 out of 54 treebanks and the reimplementation of DQM, by 43 of 54 treebanks, with 6 ties.

as a strong baseline. As expected, our reimplementation of Dozat et al. (2017) tends to significantly outperform the winners of the CONLL 2017 Shared Task. However, in general, our models still obtain better results, outperforming Dozat et al. on 43 of the 54 treebanks, with an absolute difference of 0.42% on average.

5 Ablation Study

The model proposed in this paper of a Meta-BiLSTM with a sentence-based character model differs from prior work in multiple aspects. In this section, we perform ablations to determine the relative impact of each modeling decision.

For the experimental setup of the ablation experiments, we report accuracy scores for the development sets. We split off 5% of the sentences from each training corpus and we use this part for early stopping. Ablation experiments are either performed on a few selected treebanks to show individual language results or averaged across all treebanks for which tagging is non-trivial.

Impact of the Training Schema We first compare jointly training the three model components (Meta-BiLSTM, character model, word model) to training each separately. Table 5 shows that separately optimized models are significantly more accurate on average than jointly optimized models. Separate optimization leads to better accuracy for 34 out of 40 treebanks for the morphological features task and for 30 out of 39 treebanks for xpos tagging. Separate optimization outperformed joint optimization by up to 2.1 percent absolute, while joint never out-performed separate by more than 0.5% absolute. We hypothesize that separately training the models forces each submodel (word and character) to be strong enough to make high accuracy predictions and in some sense serves as a regularizer in the same way that dropout does for individual neurons.

Impact of the Sentence-based Character Model We compared the setup with sentence-based character context (Figure 1a) to word-based character context (Figure 1b). We selected for these experiments a number of morphological rich languages. The results are shown in Table 6. The accuracy of the word-based character model joint with a word-based model were significantly lower than a sentence-based character model. We conclude also from these results and comparing with results of the reimplementation of DQM that early integration of the word-based character model performs much better as late integration via Meta-BiLSTM for a word-based character model.

Optimization	Avg. F1 Score morphology	Avg. F1 Score xpos
separate	**94.57**	**94.85**
jointly	94.15	94.48

Table 5: Comparison of optimization methods: Separate optimization of the word, character and meta model is more accurate on average than full back-propagation using a single loss function.The results are statistically significant with two-tailed paired t-test for xpos with p<0.001 and for morphology with p <0.0001.

dev. set	word char model	sentence char model
el	89.05	93.41
la_ittb	93.22	95.69
ru	88.94	92.31
tr	87.78	90.77

Table 6: F1 score for selected languages on sentence vs. word level character models for the prediction of morphology using late integration.

dev. set	num.	mean	mean	mean	stdev	stdev	stdev
lang.	exp.	char	word	joint	char	word	joint
el	10	96.43	95.36	**97.01**	0.13	0.11	0.09
grc	10	88.28	73.52	**88.85**	0.21	0.29	0.22
la_ittb	10	91.45	87.98	**91.94**	0.14	0.30	0.05
ru	10	95.98	93.50	**96.61**	0.06	0.17	0.07
tr	10	93.77	90.48	**94.67**	0.11	0.33	0.14

Table 7: F1 score for the character, word and joint models. The standard deviation of 10 random restarts of each model is show in the last three columns. The differences in means are all statistically significant at $p < 0.001$ (paired t-test).

Impact of the Meta-BiLSTM Model Combination The proposed model trains word and character models independently while training a joint model on top. Here we investigate the part-of-speech tagging performance of the joint model compared with the word and character models on their own (using hyperparameters from in 4.1).

Table 5 shows, for selected languages, the results averaged over 10 runs in order to measure standard deviation. The examples show that the combined model has significantly higher accuracy compared with either the character and word models individually.

dev. set. lang.	F_{last} B_{1st}	F_{1st} B_{last}	F_{last} B_{last}	F_{1st} B_{1st}	DQM	|xpos|
el	**96.6**	**96.6**	96.2	96.1	95.9	16
grc	**87.3**	87.1	87.1	86.8	86.7	3130
la_ittb	91.1	91.5	**91.9**	91.3	91.0	811
ru	95.6	95.4	95.6	95.3	**95.8**	49
tr	93.5	93.3	93.2	92.5	**93.9**	37

Table 8: F1 score of char models and their performance on the dev. set for selected languages with different gather strategies, concatenate to g_i (Equation 1). DQM shows results for our reimplementation of Dozat et al. (2017) (cf. §3.2), where we feed in only the characters. The final column shows the number of xpos tags in the training set.

Concatenation Strategies for the Context-Sensitive Character Encodings The proposed model bases a token encoding on both the forward and the backward character representations of both the first and last character in the token (see Equation 1). Table 8 reports, for a few morphological rich languages, the part-of-speech tagging performance of different strategies to gather the characters when creating initial word encodings. The strategies were defined in §3.1. The Table also contains a column with results for our reimplementation of Dozat et al. (2017). We removed, for all systems, the word model in order to assess each strategy in isolation. The performance is quite different per language. E.g., for Latin, the outputs of the forward and backward LSTMs of the last character scored highest.

Sensitivity to Hyperparameter Search We picked Vietnamese for a more in-depth analysis since it did not perform well and investigated the influence of the sizes of LSTMs for the word and character model on the accuracy of development set. With larger network sizes, the capacity of the network increases, however, on the other hand it is prune to overfitting. We fixed all the hyperparameters except those for the network size of the character model and the word model, and ran a grid search over dimension sizes from 200 to 500 in steps of 50. The surface plot in 3 shows that the grid peaks with more moderate settings around 350 LSTM cells which might lead to a higher accuracy. For all of the network sizes in the grid search, we still observed during training that the accuracy reach a high value and degrades with more iterations for the character and word model. This suggests that future variants of this model might benefit from higher regularization.

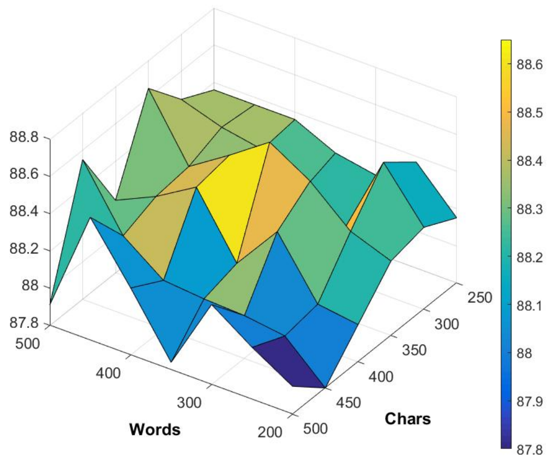


Figure 3: 3D surface plot for development set accuracy for XPOS (y-axis) depending on LSTM size of the character and word model for the Vietnamese treebank. The snapshot is take after 195 training epochs and we average the values of neighboring epochs.

Discussion Generally, the fact that different techniques for creating word encodings from character encodings and different network sizes can lead to different accuracies per language suggests that it should be possible to increase the accuracy of our model on a per language basis via a grid search over all possibilities. In fact, there are many variations on the models we presented in this work (e.g., how the character and word models are combined with the meta-BiLSTM). Since we are using separate losses, we could also change our training schema. For example, one could use methods like stack-propagation (Zhang and Weiss, 2016) where we burn-in the character and word models and then train the meta-BiLSTM backpropagating throughout the entire network.

6 Conclusions

We presented an approach to morphosyntactic tagging that combines context-sensitive initial character and word encodings with a meta-BiLSTM layer to obtain state-of-the art accuracies for a wide variety of languages.

Acknowledgments

We would like to thank the anonymous reviewers as well as Terry Koo, Slav Petrov, Vera Axelrod, Kellie Websterk, Jan Botha, Kuzman Ganchev, Zhuoran Yu, Yuan Zhang, Eva Schlinger, Ji Ma, and John Alex for their helpful suggestions, comments and discussions.

References

Chris Alberti, Daniel Andor, Ivan Bogatyy, Michael Collins, Dan Gillick, Lingpeng Kong, Terry Koo, Ji Ma, Mark Omernick, Slav Petrov, Chayut Thanapirom, Zora Tung, and David Weiss. 2017. Syntaxnet models for the conll 2017 shared task http://arxiv.org/abs/1703.04929.

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2442–2452.

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. Ims at the conll 2017 ud shared task: Crfs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 40–51.

Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald, and Slav Petrov. 2017. Natural language processing with small feed-forward networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2879–2885.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. pages 18–26.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 740–750.

Jinho D. Choi. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*. San Diego, CA, NAACL'16, pages 271–281.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 160–167.

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

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017*. pages 20–30.

Jesús Giménez and Lluis Marquez. 2004. Fast and accurate part-of-speech tagging: The svm approach revisited. *Recent Advances in Natural Language Processing III* pages 153–162.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602–610.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging http://arxiv.org/abs/1508.01991.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations http://arxiv.org/abs/1802.05365.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 412–418.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 48–52.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 88–99.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 1–19.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1557–1566.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 1213–1222.

Neural Factor Graph Models for Cross-lingual Morphological Tagging

Chaitanya Malaviya and **Matthew R. Gormley** and **Graham Neubig**
Language Technologies Institute, Machine Learning Department
Carnegie Mellon University
{cmalaviy,mgormley,gneubig}@cs.cmu.edu

Abstract

Morphological analysis involves predicting the syntactic traits of a word (e.g. {*POS: Noun, Case: Acc, Gender: Fem*}). Previous work in morphological tagging improves performance for low-resource languages (LRLs) through cross-lingual training with a high-resource language (HRL) from the same family, but is limited by the strict—often false—assumption that tag sets exactly overlap between the HRL and LRL. In this paper we propose a method for cross-lingual morphological tagging that aims to improve information sharing between languages by relaxing this assumption. The proposed model uses factorial conditional random fields with neural network potentials, making it possible to (1) utilize the expressive power of neural network representations to smooth over superficial differences in the surface forms, (2) model pairwise and transitive relationships between tags, and (3) accurately generate tag sets that are unseen or rare in the training data. Experiments on four languages from the Universal Dependencies Treebank (Nivre et al., 2017) demonstrate superior tagging accuracies over existing cross-lingual approaches.[1]

1 Introduction

Morphological analysis (Hajič and Hladká (1998), Oflazer and Kuruöz (1994), *inter alia*) is the task of predicting fine-grained annotations about the syntactic properties of tokens in a language such as part-of-speech, case, or tense. For instance, in Figure 1, the given Portuguese sentence is labeled with the respective morphological tags such as Gender and its label value Masculine.

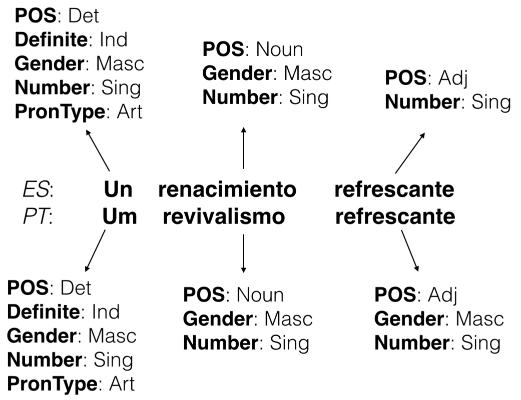


Figure 1: Morphological tags for a UD sentence in Portuguese and a translation in Spanish

The accuracy of morphological analyzers is paramount, because their results are often a first step in the NLP pipeline for tasks such as translation (Vylomova et al., 2017; Tsarfaty et al., 2010) and parsing (Tsarfaty et al., 2013), and errors in the upstream analysis may cascade to the downstream tasks. One difficulty, however, in creating these taggers is that only a limited amount of annotated data is available for a majority of the world's languages to learn these morphological taggers. Fortunately, recent efforts in morphological annotation follow a standard annotation schema for these morphological tags across languages, and now the Universal Dependencies Treebank (Nivre et al., 2017) has tags according to this schema in 60 languages.

Cotterell and Heigold (2017) have recently shown that combining this shared schema with cross-lingual training on a related high-resource language (HRL) gives improved performance

[1]Our code and data is publicly available at www.github.com/chaitanyamalaviya/NeuralFactorGraph.

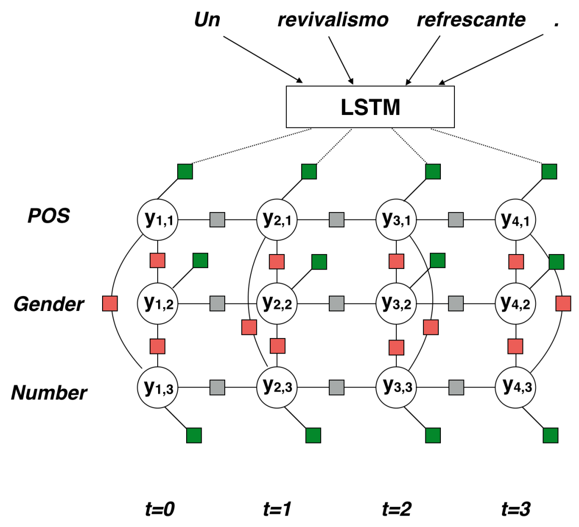


Figure 2: FCRF-LSTM Model for morphological tagging

on tagging accuracy for low-resource languages (LRLs). The output space of this model consists of tag sets such as {*POS: Adj, Gender: Masc, Number: Sing*}, which are predicted for a token at each time step. However, this model relies heavily on the fact that *the entire space of tag sets* for the LRL must match those of the HRL, which is often not the case, either due to linguistic divergence or small differences in the annotation schemes between the two languages.[2] For instance, in Figure 1 "refrescante" is assigned a gender in the Portuguese UD treebank, but not in the Spanish UD treebank.

In this paper, we propose a method that instead of predicting full tag sets, makes predictions over single tags separately but ties together each decision by modeling variable dependencies between tags over time steps (e.g. capturing the fact that nouns frequently occur after determiners) and pairwise dependencies between all tags at a single time step (e.g. capturing the fact that infinitive verb forms don't have tense). The specific model is shown in Figure 2, consisting of a factorial conditional random field (FCRF; Sutton et al. (2007)) with neural network potentials calculated by long short-term memory (LSTM; (Hochreiter and Schmidhuber, 1997)) at every variable node (§3). Learning and inference in the model is made tractable through belief propagation over the possible tag combinations, allowing the model to consider an exponential label space in polynomial time (§3.5).

This model has several advantages:

- The model is able to generate tag sets unseen in training data, and share information between similar tag sets, alleviating the main disadvantage of previous work cited above.
- Our model is empirically strong, as validated in our main experimental results: it consistently outperforms previous work in cross-lingual low-resource scenarios in experiments.
- Our model is more interpretable, as we can probe the model parameters to understand which variable dependencies are more likely to occur in a language, as we demonstrate in our analysis.

In the following sections, we describe the model and these results in more detail.

2 Problem Formulation and Baselines

2.1 Problem Formulation

Formally, we define the problem of morphological analysis as the task of mapping a length-T string of tokens $\mathbf{x} = x_1, \ldots, x_T$ into the target morphological tag sets for each token $\mathbf{y} = \mathbf{y}_1, \ldots, \mathbf{y}_T$. For the tth token, the target label $\mathbf{y}_t = y_{t,1}, \ldots, y_{t,m}$ defines a set of tags (e.g. {*Gender: Masc, Number: Sing, POS: Verb*}). An annotation schema defines a set $\mathcal{S}$ of M possible tag types and with the mth type (e.g. *Gender*) defining its set of possible labels $\mathcal{Y}_m$ (e.g. {*Masc, Fem, Neu*}) such that $y_{t,m} \in \mathcal{Y}_m$. We must note that not all tags or attributes need to be specified for a token; usually, a subset of $\mathcal{S}$ is specified for a token and the remaining tags can be treated as mapping to a `NULL` $\in \mathcal{Y}_m$ value. Let $\mathcal{Y} = \{(y_1, \ldots, y_M) : y_1 \in \mathcal{Y}_1, \ldots, y_M \in \mathcal{Y}_M\}$ denote the set of all possible tag sets.

2.2 Baseline: Tag Set Prediction

Data-driven models for morphological analysis are constructed using training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ consisting of N training examples. The baseline model (Cotterell and Heigold, 2017) we compare with regards the output space of the model as a subset $\tilde{\mathcal{Y}} \subset \mathcal{Y}$ where $\tilde{\mathcal{Y}}$ is the

[2]In particular, the latter is common because many UD resources were created by full or semi-automatic conversion from treebanks with less comprehensive annotation schemes than UD. Our model can generate label values for these tags too, which could possibly aid the enhancement of UD annotations, although we do not examine this directly in our work.

set of all tag sets seen in this training data. Specifically, they solve the task as a multi-class classification problem where the classes are individual tag sets. In low-resource scenarios, this indicates that $|\tilde{\mathcal{Y}}| << |\mathcal{Y}|$ and even for those tag sets existing in $\tilde{\mathcal{Y}}$ we may have seen very few training examples. The conditional probability of a sequence of tag sets given the sentence is formulated as a 0th order CRF.

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{x}) \tag{1}$$

Instead, we would like to be able to generate any combination of tags from the set $\mathcal{Y}$, and share statistical strength among similar tag sets.

2.3 A Relaxation: Tag-wise Prediction

As an alternative, we could consider a model that performs prediction for each tag's label $y_{t,m}$ independently.

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} \prod_{m=1}^{M} p(y_{t,m}|\mathbf{x}) \tag{2}$$

This formulation has an advantage: the tag-predictions within a single time step are now independent, it is now easy to generate any combination of tags from $\mathcal{Y}$. On the other hand, now it is difficult to model the interdependencies between tags in the same tag set $\mathbf{y}_i$, a major disadvantage over the previous model. In the next section, we describe our proposed neural factor graph model, which can model not only dependencies within tags for a single token, but also dependencies across time steps while still maintaining the flexibility to generate any combination of tags from $\mathcal{Y}$.

3 Neural Factor Graph Model

Due to the correlations between the syntactic properties that are represented by morphological tags, we can imagine that capturing the relationships between these tags through pairwise dependencies can inform the predictions of our model. These dependencies exist both among tags for the same token (intra-token pairwise dependencies), and across tokens in the sentence (inter-token transition dependencies). For instance, knowing that a token's POS tag is a Noun, would strongly suggest that this token would have a `NULL` label for the tag Tense, with very few exceptions (Nordlinger and Sadler, 2004). In a language where nouns follow adjectives, a tag set prediction {*POS: Adj, Gender: Fem*} might inform the model that the next token is likely to be a noun and have the same gender. The baseline model can not explicitly model such interactions given their factorization in equation 1.

To incorporate the dependencies discussed above, we define a factorial CRF (Sutton et al., 2007), with pairwise links between cotemporal variables and transition links between the same types of tags. This model defines a distribution over the tag-set sequence $\mathbf{y}$ given the input sentence $\mathbf{x}$ as,

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \prod_{\alpha \in \mathcal{C}} \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x}, t) \tag{3}$$

where $\mathcal{C}$ is the set of factors in the factor graph (as shown in Figure 2), α is one such factor, and $\mathbf{y}_\alpha$ is the assignment to the subset of variables neighboring factor α. We define three types of potential functions: neural ψ_{NN}, pairwise ψ_P, and transition ψ_T, described in detail below.

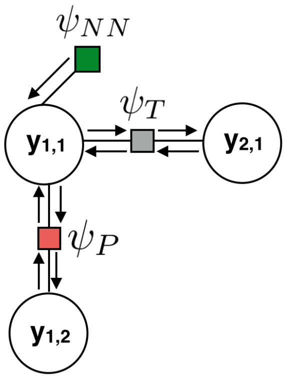


Figure 3: Factors in the Neural Factor Graph model (red: Pairwise, grey: Transition, green: Neural Network)

3.1 Neural Factors

The flexibility of our formulation allows us to include any form of custom-designed potentials in our model. Those for the neural factors have a fairly standard log-linear form,

$$\psi_{NN,i}(y_{t,m}) = \exp\left\{\sum_k \lambda_{\text{nn},k} f_{\text{nn},k}(\mathbf{x}, t)\right\} \tag{4}$$

except that the features $f_{\text{nn},k}$ are themselves given by a neural network. There is one such factor per variable. We obtain our neural factors using a biLSTM over the input sequence $\mathbf{x}$, where the input word embedding for each token is obtained from a character-level biLSTM embedder. This component of our model is similar to the model proposed by Cotterell and Heigold (2017). Given an input token $\mathbf{x}_t = c_1...c_n$, we compute an input embedding v_t as,

$$v_t = [\text{cLSTM}(c_1...c_n); \text{cLSTM}(c_n...c_1)] \quad (5)$$

Here, cLSTM is a character-level LSTM function that returns the last hidden state. This input embedding v_t is then used in the biLSTM tagger to compute an output representation e_t. Finally, the scores $f_{\text{nn}}(\mathbf{x}, t)$ are obtained as,

$$f_{\text{nn}}(\mathbf{x}, t) = W_l e_t + b_l \quad (6)$$

We use a language-specific linear layer with weights W_l and bias b_l.

3.2 Pairwise Factors

As discussed previously, the pairwise factors are crucial for modeling correlations between tags. The pairwise factor potential for a tag i and tag j at timestep t is given in equation 7. Here, the dimension of f_p is $(|\mathcal{Y}_i|, |\mathcal{Y}_j|)$. These scores are used to define the neural factors as,

$$\psi_{P_{i,j}}(y_{t,i}, y_{t,j}) = \exp\left\{\sum_k \lambda_{\text{p},k} f_{\text{p},k}(y_{t,i}, y_{t,j})\right\} \quad (7)$$

3.3 Transition Factors

Previous work has experimented with the use of a linear chain CRF with factors from a neural network (Huang et al., 2015) for sequence tagging tasks. We hypothesize that modeling transition factors in a similar manner can allow the model to utilize information about neighboring tags and capture word order features of the language. The transition factor for tag i and timestep t is given below for variables $y_{t,i}$ and $y_{t+1,i}$. The dimension of f_T is $(|\mathcal{Y}_i|, |\mathcal{Y}_i|)$.

$$\psi_{T_{i,t}}(y_{t,i}, y_{t+1,i}) = \exp\left\{\sum_k \lambda_{\text{T},k} f_{\text{T},k}(y_{t,i}, y_{t+1,i})\right\} \quad (8)$$

In our experiments, $f_{\text{p},k}$ and $f_{\text{T},k}$ are simple indicator features for the values of tag variables with no dependence on $\mathbf{x}$.

3.4 Language-Specific Weights

As an enhancement to the information encoded in the transition and pairwise factors, we experiment with training general and language-specific parameters for the transition and the pairwise weights. We define the weight matrix λ_{gen} to learn the general trends that hold across both languages, and the weights λ_{lang} to learn the exceptions to these trends. In our model, we sum both these parameter matrices before calculating the transition and pairwise factors. For instance, the transition weights λ_T are calculated as $\lambda_T = \lambda_{\text{T, gen}} + \lambda_{\text{T, lang}}$.

3.5 Loopy Belief Propagation

Since the graph from Figure 2 is a loopy graph, performing exact inference can be expensive. Hence, we use loopy belief propagation (Murphy et al., 1999; Ihler et al., 2005) for computation of approximate variable and factor marginals. Loopy BP is an iterative message passing algorithm that sends messages between variables and factors in a factor graph. The message updates from variable v_i, with neighboring factors $N(i)$, to factor α is

$$\mu_{i\to\alpha}(v_i) = \prod_{\alpha \in N(i)\setminus\alpha} \mu_{\alpha\to i}(v_i) \quad (9)$$

The message from factor α to variable v_i is

$$\mu_{\alpha\to i}(v_i) = \sum_{\mathbf{v}_\alpha : \mathbf{v}_\alpha[i]=v_i} \psi_\alpha(\mathbf{v}_\alpha) \prod_{j\in N(\alpha)\setminus i} \mu_{j\to\alpha}(\mathbf{v}_\alpha[i]) \quad (10)$$

where $\mathbf{v}_\alpha$ denote an assignment to the subset of variables adjacent to factor α, and $\mathbf{v}_\alpha[i]$ is the assignment for variable v_i. Message updates are performed asynchronously in our model. Our message passing schedule was similar to that of foward-backward: the forward pass sends all messages from the first time step in the direction of the last. Messages to/from pairwise factors are included in this forward pass. The backward pass sends messages in the direction from the last time step back to the first. This process is repeated until convergence. We say that BP has converged when the maximum residual error (Sutton and McCallum, 2007) over all messages is below some threshold. Upon convergence, we obtain the belief values of variables and factors as,

$$b_i(v_i) = \frac{1}{\kappa_i} \prod_{\alpha\in N(i)} \mu_{\alpha\to i}(v_i) \quad (11)$$

$$b_\alpha(v_\alpha) = \frac{1}{\kappa_\alpha} \psi_\alpha(v_\alpha) \prod_{i\in N(\alpha)} \mu_{i\to\alpha}(v_\alpha[i]) \quad (12)$$

where κ_i and κ_α are normalization constants ensuring that the beliefs for a variable i and factor α sum-to-one. In this way, we can use the beliefs as approximate marginal probabilities.

3.6 Learning and Decoding

We perform end-to-end training of the neural factor graph by following the (approximate) gradient of the log-likelihood $\sum_{i=1}^{N} \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)})$. The true gradient requires access to the marginal probabilities for each factor, e.g. $p(\mathbf{y}_\alpha|\mathbf{x})$ where $\mathbf{y}_\alpha$ denotes the subset of variables in factor α. For example, if α is a transition factor for tag m at timestep t, then $\mathbf{y}_\alpha$ would be $y_{t,m}$ and $y_{t+1,m}$. Following (Sutton et al., 2007), we replace these marginals with the beliefs $b_\alpha(\mathbf{y}_\alpha)$ from loopy belief propagation.[3] Consider the log-likelihood of a single example $\ell^{(i)} = \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)})$. The partial derivative with respect to parameter $\lambda_{g,k}$ for each type of factor $g \in \{NN, T, P\}$ is the difference of the observed features with the expected features under the model's (approximate) distribution as represented by the beliefs:

$$\frac{\partial \ell^{(i)}}{\partial \lambda_{g,k}} = \sum_{\alpha \in \mathcal{C}_g} \left(f_{g,k}(\mathbf{y}_\alpha^{(i)}) - \sum_{\mathbf{y}_\alpha} b_\alpha(\mathbf{y}_\alpha) f_{g,k}(\mathbf{y}_\alpha) \right)$$

where $\mathcal{C}_g$ denotes all the factors of type g, and we have omitted any dependence on $\mathbf{x}^{(i)}$ and t for brevity—t is accessible through the factor index α. For the neural network factors, the features are given by a biLSTM. We backpropagate through to the biLSTM parameters using the partial derivative below,

$$\frac{\partial \ell^{(i)}}{\partial f_{NN,k}(y_{t,m}^{(i)}, t)} = \lambda_{NN,k} - \sum_{y_{t,m}} b_{t,m}(y_{t,m}) \lambda_{NN,k}$$

where $b_{t,m}(\cdot)$ is the variable belief corresponding to variable $y_{t,m}$.

To predict a sequence of tag sets $\hat{\mathbf{y}}$ at test time, we use minimum Bayes risk (MBR) decoding (Bickel and Doksum, 1977; Goodman, 1996) for Hamming loss over tags. For a variable $y_{t,m}$ representing tag m at timestep t, we take

$$\hat{y}_{t,m} = \operatorname*{arg\,max}_{l \in \mathcal{Y}_m} b_{t,m}(l). \tag{13}$$

where l ranges over the possible labels for tag m.

Language Pair	HRL Train	Dev	Test
DA/SV	4,383	504	1219
RU/BG	3,850	1115	1116
FI/HU	12,217	441	449
ES/PT	14,187	560	477

Table 1: Dataset sizes. tgt_size = 100 or 1,000 LRL sentences are added to HRL Train

Language Pair	Unique Tags	Tag Sets
DA/SV	23	224
RU/BG	19	798
FI/HU	27	2195
ES/PT	19	451

Table 2: Tag Set Sizes with tgt_size=100

4 Experimental Setup

4.1 Dataset

We used the Universal Dependencies Treebank UD v2.1 (Nivre et al., 2017) for our experiments. We picked four low-resource/high-resource language pairs, each from a different family: Danish/Swedish (DA/SV), Russian/Bulgarian (RU/BG), Finnish/Hungarian (FI/HU), Spanish/Portuguese (ES/PT). Picking languages from different families would ensure that we obtain results that are on average consistent across languages.

The sizes of the training and evaluation sets are specified in Table 1. In order to simulate low-resource settings, we follow the experimental procedure from Cotterell and Heigold (2017). We restrict the number of sentences of the target language (tgt_size) in the training set to 100 or 1000 sentences. We also augment the tag sets in our training data by adding a `NULL` label for all tags that are not seen for a token. It is expected that our model will learn which tags are unlikely to occur given the variable dependencies in the factor graph. The dev set and test set are only in the target language. From Table 2, we can see there is also considerable variance in the number of unique tags and tag sets found in each of these language pairs.

[3]Using this approximate gradient is akin to the *surrogate likelihood* training of (Wainwright, 2006).

Language	Model	tgt_size = 100			tgt_size=1000		
		Accuracy	F1-Micro	F1-Macro	Accuracy	F1-Macro	F1-Micro
SV	Baseline	15.11	8.36	10.37	68.64	76.36	76.50
	Ours	**29.47**	**54.09**	**54.36**	**71.32**	**84.42**	**84.46**
BG	Baseline	**29.05**	14.32	29.62	**59.20**	**67.22**	**67.12**
	Ours	27.81	**40.97**	**42.43**	39.25	60.23	60.84
HU	Baseline	21.97	13.30	16.67	**50.75**	58.68	62.79
	Ours	**33.32**	**54.88**	**54.69**	45.90	**74.05**	**73.38**
PT	Baseline	18.91	7.10	10.33	74.22	81.62	81.87
	Ours	**58.82**	**73.67**	**74.07**	**76.26**	**87.13**	**87.22**

Table 3: Token-wise accuracy and F1 scores on mono-lingual experiments

4.2 Baseline Tagger

As the baseline tagger model, we re-implement the SPECIFIC model from Cotterell and Heigold (2017) that uses a language-specific softmax layer. Their model architecture uses a character biLSTM embedder to obtain a vector representation for each token, which is used as input in a word-level biLSTM. The output space of their model is all the tag sets seen in the training data. This work achieves strong performance on several languages from UD on the task of morphological tagging and is a strong baseline.

4.3 Training Regimen

We followed the parameter settings from Cotterell and Heigold (2017) for the baseline tagger and the neural component of the FCRF-LSTM model. For both models, we set the input embedding and linear layer dimension to 128. We used 2 hidden layers for the LSTM where the hidden layer dimension was set to 256 and a dropout (Srivastava et al., 2014) of 0.2 was enforced during training. All our models were implemented in the PyTorch toolkit (Paszke et al., 2017). The parameters of the character biLSTM and the word biLSTM were initialized randomly. We trained the baseline models and the neural factor graph model with SGD and Adam respectively for 10 epochs each, in batches of 64 sentences. These optimizers gave the best performances for the respective models.

For the FCRF, we initialized transition and pairwise parameters with zero weights, which was important to ensure stable training. We considered BP to have reached convergence when the maximum residual error was below 0.05 or if the maximum number of iterations was reached (set to 40 in our experiments). We found that in cross-lingual experiments, when $tgt\_size = 100$, the relatively large amount of data in the HRL was causing our model to overfit on the HRL and not generalize well to the LRL. As a solution to this, we upsampled the LRL data by a factor of 10 when $tgt\_size = 100$ for both the baseline and the proposed model.

Evaluation: Previous work on morphological analysis (Cotterell and Heigold, 2017; Buys and Botha, 2016) has reported scores on average token-level accuracy and F1 measure. The average token level accuracy counts a tag set prediction as correct only it is an exact match with the gold tag set. On the other hand, F1 measure is measured on a tag-by-tag basis, which allows it to give partial credit to partially correct tag sets. Based on the characteristics of each evaluation measure, Accuracy will favor tag-set prediction models (like the baseline), and F1 measure will favor tag-wise prediction models (like our proposed method). Given the nature of the task, it seems reasonable to prefer getting some of the tags correct (e.g. Noun+Masc+Sing becomes Noun+Fem+Sing), instead of missing all of them (e.g. Noun+Masc+Sing becomes Adj+Fem+Plur). F-score gives partial credit for getting some of the tags correct, while tagset-level accuracy will treat these two mistakes equally. Based on this, we believe that F-score is intuitively a better metric. However, we report both scores for completeness.

5 Results and Analysis

5.1 Main Results

First, we report the results in the case of monolingual training in Table 3. The first row for each language pair reports the results for our reimple-

Language	Model	tgt_size = 100			tgt_size=1000		
		Accuracy	F1-Micro	F1-Macro	Accuracy	F1-Macro	F1-Micro
DA/SV	Baseline	**66.06**	73.95	74.37	**82.26**	**87.88**	**87.91**
	Ours	63.22	**78.75**	**78.72**	77.43	87.56	87.52
RU/BG	Baseline	**52.76**	58.41	58.23	**71.90**	77.89	77.97
	Ours	46.89	**64.46**	**64.75**	67.56	**82.06**	**82.11**
FI/HU	Baseline	**51.74**	68.15	66.82	61.80	75.96	76.16
	Ours	45.41	**68.63**	**68.07**	**63.93**	**85.06**	**84.12**
ES/PT	Baseline	**79.40**	86.03	86.14	**85.85**	91.91	91.93
	Ours	77.75	**88.42**	**88.44**	85.02	**92.35**	**92.37**

Table 4: Token-wise accuracy and F1 scores on cross-lingual experiments

mentation of Cotterell and Heigold (2017), and the second for our full model. From these results, we can see that we obtain improvements on the F-measure over the baseline method in most experimental settings except BG with $tgt\_size = 1000$. In a few more cases, the baseline model sometimes obtains higher accuracy scores for the reason described in 4.3.

In our cross-lingual experiments shown in Table 4, we also note F-measure improvements over the baseline model with the exception of DA/SV when $tgt\_size = 1000$. We observe that the improvements are on average stronger when $tgt\_size = 100$. This suggests that our model performs well with very little data due to its flexibility to generate any tag set, including those not observed in the training data. The strongest improvements are observed for FI/HU. This is likely because the number of unique tags is the highest in this language pair and our method scales well with the number of tags due to its ability to make use of correlations between the tags in different tag sets.

Language	Transition	Pairwise	F1-Macro
HU	×	×	69.87
	✓	×	73.21
	×	✓	73.68
	✓	✓	74.05
FI/HU	×	×	79.57
	✓	×	84.41
	×	✓	84.73
	✓	✓	85.06

Table 5: Ablation Experiments (tgt_size=1000)

To examine the utility of our transition and pairwise factors, we also report results on ablation experiments by removing transition and pairwise factors completely from the model in Table 5. Ablation experiments for each factor showed decreases in scores relative to the model where both factors are present, but the decrease attributed to the pairwise factors is larger, in both the monolingual and cross-lingual cases. Removing both factors from our proposed model results in a further decrease in the scores. These differences were found to be more significant in the case when $tgt\_size = 100$.

Upon looking at the tag set predictions made by our model, we found instances where our model utilizes variable dependencies to predict correct labels. For instance, for a specific phrase in Portuguese (*um estado*), the baseline model predicted $\{$*POS: Det, Gender: Masc, Number: Sing*$\}_t$, $\{$*POS: Noun, Gender: Fem (X), Number: Sing*$\}_{t+1}$, whereas our model was able to get the gender correct because of the transition factors in our model.

5.2 What is the Model Learning?

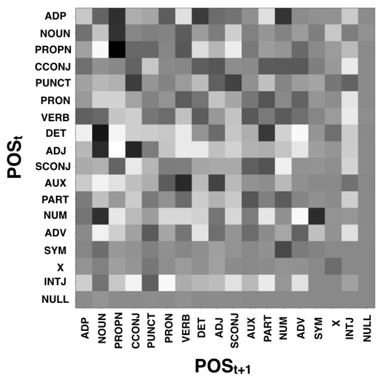


Figure 4: Generic transition weights for POS from the RU/BG model

One of the major advantages of our model is

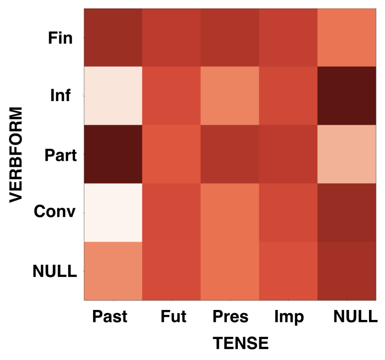


Figure 5: Generic pairwise weights between `Verbform` and `Tense` from the RU/BG model

the ability to interpret what the model has learned by looking at the trained parameter weights. We investigated both language-generic and language-specific patterns learned by our parameters:

- **Language-Generic**: We found evidence for several syntactic properties learned by the model parameters. For instance, in Figure 4, we visualize the generic ($\lambda_{\mathrm{T,\,gen}}$) transition weights of the POS tags in RU/BG. Several universal trends such as determiners and adjectives followed by nouns can be seen. In Figure 5, we also observed that infinitive has a strong correlation for `NULL` tense, which follows the universal phenomena that infinitives don't have tense.

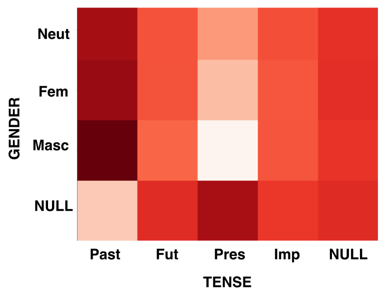


Figure 6: Language-specific pairwise weights for RU between `Gender` and `Tense` from the RU/BG model

- **Language Specific Trends**: We visualized the learnt language-specific weights and looked for evidence of patterns corresponding to linguistic phenomenas observed in a language of interest. For instance, in Russian, verbs are gender-specific in past tense but not in other tenses. To analyze this, we plotted pairwise weights for Gender/Tense in Figure 6 and verified strong correlations between the past tense and all gender labels.

6 Related Work

There exist several variations of the task of prediction of morphological information from annotated data: paradigm completion (Durrett and DeNero, 2013; Cotterell et al., 2017b), morphological reinflection (Cotterell et al., 2017a), segmentation (Creutz et al., 2005; Cotterell et al., 2016) and tagging. Work on morphological tagging has broadly focused on structured prediction models such as CRFs, and neural network models. Amongst structured prediction approaches, Müller et al. (2013); Müller and Schütze (2015) proposed the use of a higher-order CRF that is approximated using coarse-to-fine decoding. (Müller et al., 2015) proposed joint lemmatization and tagging using this framework. (Hajič, 2000) was the first work that performed experiments on multilingual morphological tagging. They proposed an exponential model and the use of a morphological dictionary. Buys and Botha (2016); Kirov et al. (2017) proposed a model that used tag projection of type and token constraints from a resource-rich language to a low-resource language for tagging.

Most recent work has focused on character-based neural models (Heigold et al., 2017), that can handle rare words and are hence more useful to model morphology than word-based models. These models first obtain a character-level representation of a token from a biLSTM or CNN, which is provided to a word-level biLSTM tagger. Heigold et al. (2017, 2016) compared several neural architectures to obtain these character-based representations and found the effect of the neural network architecture to be minimal given the networks are carefully tuned. Cross-lingual transfer learning has previously boosted performance on tasks such as translation (Johnson et al., 2016) and POS tagging (Snyder et al., 2008; Plank et al., 2016). Cotterell and Heigold (2017) proposed a cross-lingual character-level neural morphological tagger. They experimented with different strategies to facilitate cross-lingual training: a language ID for each token, a language-specific softmax and a joint language identification and tagging model. We have used this work as a baseline model for comparing with our proposed method.

In contrast to earlier work on morphological tagging, we use a hybrid of neural and graphical model approaches. This combination has several advantages: we can make use of expressive feature representations from neural models while ensuring that our model is interpretable. Our work is similar in spirit to Huang et al. (2015) and Ma and Hovy (2016), who proposed models that use a CRF with features from neural models. For our graphical model component, we used a factorial CRF (Sutton et al., 2007), which is a generalization of a linear chain CRF with additional pairwise factors between cotemporal variables.

7 Conclusion and Future Work

In this work, we proposed a novel framework for sequence tagging that combines neural networks and graphical models, and showed its effectiveness on the task of morphological tagging. We believe this framework can be extended to other sequence labeling tasks in NLP such as semantic role labeling. Due to the robustness of the model across languages, we believe it can also be scaled to perform morphological tagging for multiple languages together.

Acknowledgments

The authors would like to thank David Mortensen, Soumya Wadhwa and Maria Ryskina for useful comments about this work. We would also like to thank the reviewers who gave valuable feedback to improve the paper. This project was supported in part by an Amazon Academic Research Award and Google Faculty Award.

References

Peter J. Bickel and Kjell A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected Topics*. Holden-Day Inc., Oakland, CA, USA.

Jan Buys and Jan A. Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1954–1964.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 748–759.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 1–30.

Ryan Cotterell, Arun Kumar, and Hinrich Schütze. 2016. Morphological segmentation inside-out. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2325–2330.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017b. Paradigm completion for derivational morphology. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 714–720.

Mathias Creutz, Krista Lagus, Krister Lindén, and Sami Virpioja. 2005. Morfessor and hutmegs: Unsupervised morpheme segmentation for highly-inflecting and compounding languages .

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1185–1195.

Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings of EMNLP*.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics, pages 94–101.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 483–490.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *arXiv preprint arXiv:1606.06640* .

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 505–513.

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

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

Alexander T Ihler, W Fisher John III, and Alan S Willsky. 2005. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research* 6(May):905–936.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Christo Kirov, John Sylak-Glassman, Rebecca Knowles, Ryan Cotterell, and Matt Post. 2017. A rich morphological tagger for english: Exploring the cross-linguistic tradeoff between morphology and syntax. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. volume 2, pages 112–117.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2268–2274.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order crfs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 322–332.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 526–536.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pages 467–475.

Joakim Nivre et al. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rachel Nordlinger and Louisa Sadler. 2004. Nominal tense in crosslinguistic perspective. *Language* 80(4):776–806.

Kemal Oflazer and Ilker Kuruöz. 1994. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*. Association for Computational Linguistics, pages 144–149.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch .

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 412–418.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2008. Unsupervised multilingual learning for pos tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1041–1050.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Charles Sutton and Andrew McCallum. 2007. Improved dynamic schedules for belief propagation. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research* 8(Mar):693–723.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics, pages 1–12.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing morphologically rich languages: Introduction to the special issue. *Computational linguistics* 39(1):15–22.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2017. Word representation models for morphologically rich languages in neural machine translation. In *Proceedings of the First*

Workshop on Subword and Character Level Models in NLP. Association for Computational Linguistics, Copenhagen, Denmark, pages 103–108.

Martin J Wainwright. 2006. Estimating the“wrong”graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research* 7(Sep):1829–1859.

Global Transition-based Non-projective Dependency Parsing

Carlos Gómez-Rodríguez
Universidade da Coruña
carlos.gomez@udc.es

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

Shi, Huang, and Lee (2017a) obtained state-of-the-art results for English and Chinese dependency parsing by combining dynamic-programming implementations of transition-based dependency parsers with a minimal set of bidirectional LSTM features. However, their results were limited to projective parsing. In this paper, we extend their approach to support non-projectivity by providing the first practical implementation of the MH_4 algorithm, an $O(n^4)$ mildly non-projective dynamic-programming parser with very high coverage on non-projective treebanks. To make MH_4 compatible with minimal transition-based feature sets, we introduce a transition-based interpretation of it in which parser items are mapped to sequences of transitions. We thus obtain the first implementation of global decoding for non-projective transition-based parsing, and demonstrate empirically that it is more effective than its projective counterpart in parsing a number of highly non-projective languages.

1 Introduction

Transition-based dependency parsers are a popular approach to natural language parsing, as they achieve good results in terms of accuracy and efficiency (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Zhang and Nivre, 2011; Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016). Until very recently, practical implementations of transition-based parsing were limited to approximate inference, mainly in the form of greedy search or beam search. While cubic-time exact inference algorithms for several well-known projective transition systems had been known since the work of Huang and Sagae (2010) and Kuhlmann et al. (2011), they had been considered of theoretical interest only due to their incompatibility with rich feature models: incorporation of complex features resulted in jumps in asymptotic runtime complexity to impractical levels.

However, the recent popularization of bidirectional long-short term memory networks (bi-LSTMs; Hochreiter and Schmidhuber, 1997) to derive feature representations for parsing, given their capacity to capture long-range information, has demonstrated that one may not need to use complex feature models to obtain good accuracy (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016). In this context, Shi et al. (2017a) presented an implementation of the exact inference algorithms of Kuhlmann et al. (2011) with a minimal set of only two bi-LSTM-based feature vectors. This not only kept the complexity cubic, but also obtained state-of-the-art results in English and Chinese parsing.

While their approach provides both accurate parsing and the flexibility to use any of greedy, beam, or exact decoding with the same underlying transition systems, it does not support non-projectivity. Trees with crossing dependencies make up a significant portion of many treebanks, going as high as 63% for the Ancient Greek treebank in the Universal Dependencies[1] (UD) dataset version 2.0 and averaging around 12% over all languages in UD 2.0. In this paper, we extend Shi et al.'s (2017a) approach to mildly non-projective parsing in what, to our knowledge, is the first implementation of exact decoding for a non-projective transition-based parser.

As in the projective case, a mildly non-

[1]http://universaldependencies.org/

projective decoder has been known for several years (Cohen et al., 2011), corresponding to a variant of the transition-based parser of Attardi (2006). However, its $O(n^7)$ runtime — or the $O(n^6)$ of a recently introduced improved-coverage variant (Shi et al., 2018) — is still prohibitively costly in practice. Instead, we seek a more efficient algorithm to adapt, and thus develop a transition-based interpretation of Gómez-Rodríguez et al.'s (2011) MH_4 dynamic programming parser, which has been shown to provide very good non-projective coverage in $O(n^4)$ time (Gómez-Rodríguez, 2016). While the MH_4 parser was originally presented as a non-projective generalization of the dynamic program that later led to the arc-hybrid transition system (Gómez-Rodríguez et al., 2008; Kuhlmann et al., 2011), its own relation to transition-based parsing was not known. Here, we show that MH_4 can be interpreted as exploring a subset of the search space of a transition-based parser that generalizes the arc-hybrid system, under a mapping that differs from the "push computation" paradigm used by the previously-known dynamic-programming decoders for transition systems. This allows us to extend Shi et al. (2017a)'s work to non-projective parsing, by implementing MH_4 with a minimal set of transition-based features.

Experimental results show that our approach outperforms the projective approach of Shi et al. (2017a) and maximum-spanning-tree non-projective parsing on the most highly non-projective languages in the CoNLL 2017 shared-task data that have a single treebank. We also compare with the third-order 1-Endpoint-Crossing (1EC) parser of Pitler (2014), the only other practical implementation of an exact mildly non-projective decoder that we know of, which also runs in $O(n^4)$ but without a transition-based interpretation. We obtain comparable results for these two algorithms, in spite of the fact that the MH_4 algorithm is notably simpler than 1EC. The MH_4 parser remains effective in parsing projective treebanks, while our baseline parser, the fully non-projective maximum spanning tree algorithm, falls behind due to its unnecessarily large search space in parsing these languages. Our code, including our re-implementation of the third-order 1EC parser with neural scoring, is available at `https://github.com/tzshi/mh4-parser-acl18`.

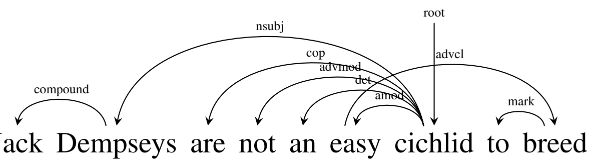


Figure 1: A non-projective dependency parse from the UD 2.0 English treebank.

2 Non-projective Dependency Parsing

In dependency grammar, syntactic structures are modeled as word-word asymmetrical subordinate relations among lexical entries (Kübler et al., 2009). These relations can be represented in a graph. For a sentence $w = w_1, ..., w_n$, we first define a corresponding set of nodes $\{0, 1, 2, ..., n\}$, where 0 is an artificial node denoting the root of the sentence. Dependency relations are encoded by edges of the form (h, m), where h is the head and m the modifier of the bilexical subordinate relation.[2]

As is conventional, we assume two more properties on dependency structures. First, each word has exactly one syntactic head, and second, the structure is acyclic. As a consequence, the edges form a directed tree rooted at node 0.

We say that a dependency structure is *projective* if it has no crossing edges. While in the CoNLL and Stanford conversions of the English Penn Treebank, over 99% of the sentences are projective (Chen and Manning, 2014) — see Fig. 1 for a non-projective English example — for other languages' treebanks, non-projectivity is a common occurrence (see Table 3 for some statistics). This paper is targeted at learning parsers that can handle non-projective dependency trees.

3 MH_4 Deduction System and Its Underlying Transition System

3.1 The MH_4 Deduction System

The MH_4 parser is the instantiation for $k = 4$ of Gómez-Rodríguez et al.'s (2011) more general MH_k parser. MH_k stands for "multi-headed with at most k heads per item": items in its deduction system take the form $[h_1, \ldots, h_p]$ for $p \leqslant k$, indicating the existence of a forest of p dependency subtrees headed by $h_1, \ldots, h_p$ such that their yields are disjoint and the union of their

[2] To simplify exposition here, we only consider the unlabeled case. We use a separately-trained labeling module to obtain labeled parsing results in §5.

$$\begin{array}{llll}
\textit{Axiom:} & \textsc{Shift}: \dfrac{[h_1,\ldots,h_m]}{[h_m,h_m+1]}\ (h_m \leqslant n) & \textsc{Combine}: \dfrac{[h_1,\ldots,h_m] \qquad [h_m,h_{m+1},\ldots,h_p]}{[h_1,\ldots,h_p]}\ (p \leqslant k) \\
{[0,1]} & & \\
\textit{Goal:} & \textsc{Link}: \dfrac{[h_1,\ldots,h_m]}{[h_1,\ldots,h_{j-1},h_{j+1},\ldots,h_m]}\ h_i \rightarrow h_j (1 \leqslant i \leqslant m \wedge 1 < j < m \wedge j \neq i) & \\
{[0,n+1]} & &
\end{array}$$

Figure 2: MH_k's deduction steps.

yields is the contiguous substring $h_1 \ldots h_p$ of the input. Deduction steps, shown in Figure 2, can be used to join two such forests that have an endpoint in common via graph union (COMBINE); or to add a dependency arc to a forest that attaches an interior head as a dependent of any of the other heads (LINK).

In the original formulation by Gómez-Rodríguez et al. (2011), all valid items of the form $[i, i+1]$ are considered to be axioms. In contrast, we follow Kuhlmann et al.'s (2011) treatment of MH_3: we consider $[0,1]$ as the only axiom and include an extra SHIFT step to generate the rest of the items of that form. Both formulations are equivalent, but including this SHIFT rule facilitates giving the parser a transition-based interpretation.

Higher values of k provide wider coverage of non-projective structures at an asymptotic runtime complexity of $O(n^k)$. When k is at its minimum value of 3, the parser covers exactly the set of projective trees, and in fact, it can be seen as a transformation[3] of the deduction system described in Gómez-Rodríguez et al. (2008) that gave rise to the projective arc-hybrid parser (Kuhlmann et al., 2011). For $k \geqslant 4$, the parser covers an increasingly larger set of non-projective structures. While a simple characterization of these sets has been lacking[4], empirical evaluation on a large number of treebanks (Gómez-Rodríguez, 2016) has shown MH_k to provide the best known tradeoff between asymptotic complexity and efficiency for $k > 4$. When $k = 4$, its coverage is second only to the 1-Endpoint-Crossing parser of Pitler et al. (2013). Both parsers fully cover well over 80% of the non-projective trees observed in the studied treebanks.

[3]Formally, it is a step refinement; see Gómez-Rodríguez et al. (2011).

[4]This is a common issue with parsers based on the general idea of arcs between non-contiguous heads, such as those deriving from Attardi (2006).

3.2 The MH_4 Transition System

Kuhlmann et al. (2011) show how the items of a variant of MH_3 can be given a transition-based interpretation under the "push computation" framework, yielding the arc-hybrid projective transition system. However, such a derivation has not been made for the non-projective case ($k > 3$), and the known techniques used to derive previous associations between tabular and transition-based parsers do not seem to be applicable in this case. The specific issue is that the deduction systems of Kuhlmann et al. (2011) and Cohen et al. (2011) have in common that the structure of their derivations is similar to that of a Dyck (or balanced-brackets) language, where steps corresponding to shift transitions are balanced with those corresponding to reduce transitions. This makes it possible to group derivation subtrees, and the transition sequences that they yield, into "push computations" that increase the length of the stack by a constant amount. However, this does not seem possible in MH_4.

Instead, we derive a transition-based interpretation of MH_4 by a generalization of that of MH_3 that departs from push computations.

To do so, we start with the MH_3 interpretation of an item $[i, j]$ given by Kuhlmann et al. (2011). This item represents a set of computations (transition sequences) that start from a configuration of the form $(\sigma, i|\beta, A)$ (where σ is the stack and $i|\beta$ is the buffer, with i being the first buffer node) and take the parser to a configuration of the form $(\sigma|i, j|\beta', A)$. That is, the computation has the net effect of placing node i on top of the previous contents of the stack, and it ends in a state where the first buffer element is j.

Under this item semantics, the COMBINE deduction step of the MH_3 parser (i.e., the instantiation of the one in Fig. 2 for $k = 3$) simply concatenates transition sequences. The SHIFT step generates a sequence with a single arc-hybrid sh transition:

$$\mathsf{sh} : (\sigma, h_m|\beta, A) \vdash (\sigma|h_m, \beta, A)$$

and the two possible instantiations of the COMBINE step when $k = 3$ take the antecedent transition sequence and add a transition to it, namely, one of the two arc-hybrid reduce transitions. Written in the context of the node indexes used in Figure 2, these are the following:

$$(\sigma|h_1|h_2, h_3|\beta, A) \vdash (\sigma|h_1, h_3|\beta, A \cup \{h_3 \rightarrow h_2\})$$
$$(\sigma|h_1|h_2, h_3|\beta, A) \vdash (\sigma|h_1, h_3|\beta, A \cup \{h_1 \rightarrow h_2\})$$

where h_1 and h_3 respectively can be simplified out to obtain the well-known arc-hybrid transitions:

$$\mathsf{la} : (\sigma|h_2, h_3|\beta, A) \vdash (\sigma, h_3|\beta, A \cup \{h_3 \rightarrow h_2\})$$
$$\mathsf{ra} : (\sigma|h_1|h_2, \beta, A) \vdash (\sigma|h_1, \beta, A \cup \{h_1 \rightarrow h_2\})$$

Now, we assume the following generalization of the item semantics: an item $[h_1, \ldots, h_m]$ represents a set of computations that start from a configuration of the form $(\sigma, h_1|\beta, A)$ and lead to a configuration of the form $(\sigma|h_1|\ldots|h_{m-1}, h_m|\beta', A)$. Note that this generalization no longer follows the "push computation" paradigm of Kuhlmann et al. (2011) and Cohen et al. (2011) because the number of nodes pushed onto the stack depends on the value of m.

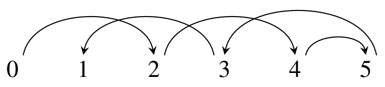
Under this item semantics, the SHIFT and COMBINE steps have the same interpretation as for MH_3. In the case of the LINK step, following the same reasoning as for the MH_3 case, we obtain the following transitions:

$$\mathsf{la} : (\sigma|h_3, h_4|\beta, A) \vdash (\sigma, h_4|\beta, A \cup \{h_4 \rightarrow h_3\})$$
$$\mathsf{ra} : (\sigma|h_2|h_3, \beta, A) \vdash (\sigma|h_2, \beta, A \cup \{h_2 \rightarrow h_3\})$$
$$\mathsf{la}' : (\sigma|h_2|h_3, h_4|\beta, A) \vdash (\sigma|h_3, h_4|\beta, A \cup \{h_3 \rightarrow h_2\})$$
$$\mathsf{ra}' : (\sigma|h_1|h_2|h_3, \beta, A) \vdash (\sigma|h_1|h_3, \beta, A \cup \{h_1 \rightarrow h_2\})$$
$$\mathsf{la}_2 : (\sigma|h_2|h_3, h_4|\beta, A) \vdash (\sigma|h_3, h_4|\beta, A \cup \{h_4 \rightarrow h_2\})$$
$$\mathsf{ra}_2 : (\sigma|h_1|h_2|h_3, \beta, A) \vdash (\sigma|h_1|h_2, \beta, A \cup \{h_1 \rightarrow h_3\})$$

These transitions give us the MH_4 transition system: a parser with four projective reduce transitions (la,ra,la′,ra′) and two Attardi-like, non-adjacent-arc reduce transitions (la$_2$ and ra$_2$).

It is worth mentioning that this MH_4 transition system we have obtained is the same as one of the variants of Attardi's algorithm introduced by Shi et al. (2018), there called ALLs_0s_1. However, in that paper they show that it can be tabularized in $O(n^6)$ using the push computation framework. Here, we have derived it as an interpretation of the $O(n^4)$ MH_4 parser.

However, in this case the dynamic programming algorithm does not cover the full search space of the transition system: while each item in the MH_4 parser can be mapped into a computation of this MH_4 transition-based parser, the opposite is not true. This tree:



can be parsed by the transition system using the computation

$\mathsf{sh}(0); \mathsf{sh}(1); \mathsf{sh}(2); \mathsf{la}_2(3 \rightarrow 1); \mathsf{sh}(3); \mathsf{sh}(4);$
$\mathsf{la}_2(5 \rightarrow 3); \mathsf{sh}(5); \mathsf{ra}(4 \rightarrow 5); \mathsf{ra}(2 \rightarrow 4); \mathsf{ra}(0 \rightarrow 2)$

but it is not covered by the dynamic programming algorithm, as no deduction sequence will yield an item representing this transition sequence. As we will see, this issue will not prevent us from implementing a dynamic-programming parser with transition-based scoring functions, or from achieving good practical accuracy.

4 Model

Given the transition-based interpretation of the MH_4 system, the learning objective becomes to find a computation that gives the gold-standard parse. For each sentence $w_1, \ldots, w_n$, we train parsers to produce the transition sequence $\mathbf{t}^*$ that corresponds to the annotated dependency structure. Thus, the model consists of two components: a parameterized scorer $S(\mathbf{t})$, and a decoder that finds a sequence $\hat{\mathbf{t}}$ as prediction based on the scoring.

As discussed by Shi et al. (2017a), there exists some tension between rich-feature scoring models and choices of decoders. Ideally, a globally-optimal decoder finds the maximum-scoring transition sequence $\hat{\mathbf{t}}$ without brute-force searching the exponentially-large output space. To keep the runtime of our exact decoder at a practical low-order polynomial, we want its feature set to be

Features	$\{\mathbf{s}_0, \mathbf{b}_0\}$	$\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$	$\{\mathbf{s}_2, \mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$
UAS	49.83	85.17	85.27

Table 1: Performance of *local* parsing models with varying number of features. We report average UAS over 10 languages on UD 2.0.

Features	$\{\mathbf{s}_0, \mathbf{b}_0\}$	Hybrid
UAS	86.79	87.27

Table 2: Performance of *global* parsing models with varying number of features.

minimal, consulting as few stack and buffer positions as possible. In what follows, we use s_0 and s_1 to denote the top two stack items and b_0 and b_1 to denote the first two buffer items.

4.1 Scoring and Minimal Features

This section empirically explores the lower limit on the number of necessary positional features. We experiment with both *local* and *global* decoding strategies. The parsers take features extracted from parser configuration c, and score each valid transition t with $S(t; c)$. The *local* parsers greedily take transitions with the highest score until termination, while the *global* parsers use the scores to find the globally-optimal solutions $\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} S(\mathbf{t})$, where $S(\mathbf{t})$ is the sum of scores for the component transitions.

Following prior work, we employ bi-LSTMs for compact feature representation. A bi-LSTM runs in both directions on the input sentence, and assigns a context-sensitive vector encoding to each token in the sentence: $\mathbf{w}_1, \ldots, \mathbf{w}_n$. When we need to extract features, say, $\mathbf{s}_0$, from a particular stack or buffer position, say s_0, we directly use the bi-LSTM vector $\mathbf{w}_{i_{s_0}}$, where i_{s_0} gives the index of the subroot of s_0 into the sentence.

Shi et al. (2017a) showed that feature vectors $\{\mathbf{s}_0, \mathbf{b}_0\}$ suffice for MH_3. Table 1 and Table 2 show the use of small feature sets for MH_4, for *local* and *global* parsing models, respectively. For a *local* parser to exhibit decent performance, we need at least $\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$, but adding $\mathbf{s}_2$ on top of that does not show any significant impact on the performance. Interestingly, in the case of *global* models, the two-vector feature set $\{\mathbf{s}_0, \mathbf{b}_0\}$ already suffices. Adding $\mathbf{s}_1$ to the global setting (column "Hybrid" in Table 2) seems attractive, but entails resolving a technical challenge that we discuss in the following section.

4.2 Global Decoder

In our transition-system interpretation of MH_k, sh transitions correspond to SHIFT and reduce transitions reflect the LINK steps. Since the SHIFT conclusions lose the contexts needed to score the transitions, we set the scores for all SHIFT rules to zero and delegate the scoring of the sh transitions to the COMBINE steps, as as in Shi et al. (2017a); for example,

$$\frac{[h_1, h_2] : v_1 \quad [h_2, h_3, h_4] : v_2}{[h_1, h_2, h_3, h_4] : v_1 + v_2 + S(\mathsf{sh}; \{\mathbf{h}_1, \mathbf{h}_2\})}$$

Here the transition sequence denoted by $[h_2, h_3, h_4]$ starts from a sh, with h_1 and h_2 taking the s_0 and b_0 positions. If we further wish to access s_1, such information is not readily available in the deduction step, apparently requiring extra bookkeeping that pushes the space and time complexity to an impractical $O(n^4)$ and $O(n^5)$, respectively. But, consider the scoring for the reduce transitions in the LINK steps:

$$\frac{[h_1, h_2, h_3, h_4] : v}{[h_1, h_2, h_4] : v + S(\mathsf{la}; \{\mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4\})}$$

$$\frac{[h_1, h_2, h_3] : v}{[h_1, h_3] : v + S(\mathsf{la}; \{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\})}$$

The deduction steps *already* keep indices for s_1 (h_2 in the first rule, h_1 in the second) and thus provide direct access without any modification. To resolve the conflict between including s_1 for richer representations and the unavailability of s_1 in scoring the sh transitions in the COMBINE steps, we propose a hybrid scoring approach — we use features $\{\mathbf{s}_0, \mathbf{b}_0\}$ when scoring a sh transition, and features $\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$ for consideration of reduce transitions. We call this method MH_4-hybrid, in contrast to MH_4-two, where we simply take $\{\mathbf{s}_0, \mathbf{b}_0\}$ for scoring all transitions.

4.3 Large-Margin Training

We train the greedy parsers with hinge loss, and the global parsers with its structured version (Taskar et al., 2005). The loss function for each sentence is formally defined as:

$$\max_{\hat{\mathbf{t}}} \left(S(\hat{\mathbf{t}}) + \mathit{cost}(\mathbf{t}^*, \hat{\mathbf{t}}) - S(\mathbf{t}^*) \right)$$

where the margin $cost(\mathbf{t}^*, \hat{\mathbf{t}})$ counts the number of mis-attached nodes for taking sequence $\hat{\mathbf{t}}$ instead of $\mathbf{t}^*$. Minimizing this loss can be thought of as optimizing for the attachment scores.

The calculation of the above loss function can be solved as efficiently as the deduction system if the *cost* function decomposes into the dynamic program. We achieve this by replacing the scoring of each reduce step by its cost-augmented version:

$$\frac{[h_1, h_2, h_3, h_4] : v}{[h_1, h_2, h_4] : v + S(\mathsf{la}_2; \{\mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4\}) + \Delta}$$

where $\Delta = \mathbf{1}(head(w_{h_3}) \neq w_{h_4})$. This loss function encourages the model to give higher contrast between gold-standard and wrong predictions, yielding better generalization results.

5 Experiments

Data and Evaluation We experiment with the Universal Dependencies (UD) 2.0 dataset used for the CoNLL 2017 shared task (Zeman et al., 2017). We restrict our choice of languages to be those with only one training treebank, for a better comparison with the shared task results.[5] Among these languages, we pick the top 10 most non-projective languages. Their basic statistics are listed in Table 3. For all development-set results, we assume gold-standard tokenization and sentence delimitation. When comparing to the shared task results on test sets, we use the provided baseline UDPipe (Straka et al., 2016) segmentation. Our models do not use part-of-speech tags or morphological tags as features, but rather leverage such information via stack propagation (Zhang and Weiss, 2016), i.e., we learn to predict them as a secondary training objective. We report unlabeled attachment F1-scores (UAS) on the development sets for better focus on comparing our (unlabeled) parsing modules. We report its labeled variant (LAS), the main metric of the shared task, on the test sets. For each experiment setting, we ran the model with 5 different random initializations, and report the mean and standard deviation. We detail the implementation details in the supplementary material.

Baseline Systems For comparison, we include three baseline systems with the same underlying feature representations and scoring paradigm. All the following baseline systems are trained with the cost-augmented large-margin loss function.

The MH_3 *parser* is the projective instantiation of the MH_k parser family. This corresponds to the global version of the arc-hybrid transition system (Kuhlmann et al., 2011). We adopt the minimal feature representation $\{\mathbf{s}_0, \mathbf{b}_0\}$, following Shi et al. (2017a). For this model, we also implement a greedy incremental version.

The *edge-factored non-projective maximal spanning tree (MST) parser* allows arbitrary non-projective structures. This decoding approach has been shown to be very competitive in parsing non-projective treebanks (McDonald et al., 2005), and was deployed in the top-performing system at the CoNLL 2017 shared task (Dozat et al., 2017). We score each edge individually, with the features being the bi-LSTM vectors $\{\mathbf{h}, \mathbf{m}\}$, where h is the head, and m the modifier of the edge.

The *crossing-sensitive third-order 1EC parser* provides a hybrid dynamic program for parsing 1-Endpoint-Crossing non-projective dependency trees with higher-order factorization (Pitler, 2014). Depending on whether an edge is crossed, we can access the modifier's grandparent g, head h, and sibling si. We take their corresponding bi-LSTM features $\{\mathbf{g}, \mathbf{h}, \mathbf{m}, \mathbf{si}\}$ for scoring each edge. This is a re-implementation of Pitler (2014) with neural scoring functions.

Main Results Table 4 shows the development-set performance of our models as compared with baseline systems. MST considers non-projective structures, and thus enjoys a theoretical advantage over projective MH_3, especially for the most non-projective languages. However, it has a vastly larger output space, making the selection of correct structures difficult. Further, the scoring is edge-factored, and does not take any structural contexts into consideration. This tradeoff leads to the similar performance of MST comparing to MH_3. In comparison, both 1EC and MH_4 are mildly non-projective parsing algorithms, limiting the size of the output space. 1EC includes higher-order features that look at tree-structural contexts; MH_4 derives its features from parsing configurations of a transition system, hence leveraging contexts within transition sequences. These considerations explain their significant improvements over MST. We also observe that MH_4 recovers more short dependencies than 1EC, while 1EC is better at longer-distance ones.

[5]When multiple treebanks are available, one can develop domain transfer strategies, which is not the focus of this work.

Language	Code	# Sent.	# Words	Sentence Coverage (%) Proj. ↓	Sentence Coverage (%) MH_4	Sentence Coverage (%) 1EC	Edge Coverage (%) Proj.	Edge Coverage (%) MH_4	Edge Coverage (%) 1EC
Basque	eu	5,396	72,974	66.48	91.48	93.29	95.98	99.27	99.42
Urdu	ur	4,043	108,690	76.97	95.89	95.77	98.89	99.83	99.81
Gothic	got	3,387	35,024	78.42	97.25	97.58	97.04	99.73	99.75
Hungarian	hu	910	20,166	79.01	98.35	97.69	98.51	99.92	99.89
Old Church Slavonic	cu	4,123	37,432	80.16	98.33	98.74	97.22	99.80	99.85
Danish	da	4,383	80,378	80.56	97.70	98.97	98.60	99.87	99.94
Greek	el	1,662	41,212	85.98	99.52	99.40	99.32	99.98	99.98
Hindi	hi	13,304	281,057	86.16	98.38	98.95	99.26	99.92	99.94
German	de	14,118	269,626	87.07	99.19	99.27	99.15	99.95	99.96
Romanian	ro	8,043	185,113	88.61	99.42	99.52	99.42	99.97	99.98

Table 3: Statistics of selected training treebanks from Universal Dependencies 2.0 for the CoNLL 2017 shared task (Zeman et al., 2017), sorted by per-sentence projective ratio.

Lan.	Global Models MH_3	Global Models MST	Global Models MH_4-two	Global Models MH_4-hybrid	Global Models 1EC	Greedy Models MH_3	Greedy Models MH_4
eu	$82.07_{\pm 0.17}$	$83.61_{\pm 0.16}$	$82.94_{\pm 0.24}$	$\mathbf{84.13}_{\pm 0.13}$	$84.09_{\pm 0.19}$	$81.27_{\pm 0.20}$	$81.71_{\pm 0.33}$
ur	$86.89_{\pm 0.18}$	$86.78_{\pm 0.13}$	$86.84_{\pm 0.26}$	$87.06_{\pm 0.24}$	$\mathbf{87.11}_{\pm 0.11}$	$86.40_{\pm 0.16}$	$86.05_{\pm 0.18}$
got	$83.72_{\pm 0.19}$	$84.74_{\pm 0.28}$	$83.85_{\pm 0.19}$	$84.59_{\pm 0.38}$	$\mathbf{84.77}_{\pm 0.27}$	$82.28_{\pm 0.18}$	$81.40_{\pm 0.45}$
hu	$83.05_{\pm 0.17}$	$82.81_{\pm 0.49}$	$83.69_{\pm 0.20}$	$\mathbf{84.59}_{\pm 0.50}$	$83.48_{\pm 0.27}$	$81.75_{\pm 0.47}$	$80.75_{\pm 0.54}$
cu	$86.70_{\pm 0.30}$	$88.02_{\pm 0.25}$	$87.57_{\pm 0.14}$	$88.09_{\pm 0.28}$	$\mathbf{88.27}_{\pm 0.32}$	$86.05_{\pm 0.23}$	$86.01_{\pm 0.11}$
da	$85.09_{\pm 0.16}$	$84.68_{\pm 0.36}$	$85.45_{\pm 0.43}$	$\mathbf{85.77}_{\pm 0.39}$	$\mathbf{85.77}_{\pm 0.16}$	$83.90_{\pm 0.24}$	$83.59_{\pm 0.06}$
el	$87.82_{\pm 0.24}$	$87.27_{\pm 0.22}$	$87.77_{\pm 0.20}$	$87.83_{\pm 0.36}$	$\mathbf{87.95}_{\pm 0.23}$	$87.14_{\pm 0.25}$	$86.95_{\pm 0.25}$
hi	$93.75_{\pm 0.14}$	$93.91_{\pm 0.26}$	$93.99_{\pm 0.15}$	$\mathbf{94.27}_{\pm 0.08}$	$94.24_{\pm 0.04}$	$93.44_{\pm 0.09}$	$93.02_{\pm 0.10}$
de	$86.46_{\pm 0.13}$	$86.34_{\pm 0.24}$	$86.53_{\pm 0.22}$	$86.89_{\pm 0.17}$	$\mathbf{86.95}_{\pm 0.32}$	$84.99_{\pm 0.26}$	$85.27_{\pm 0.32}$
ro	$89.34_{\pm 0.27}$	$88.79_{\pm 0.43}$	$89.25_{\pm 0.15}$	$\mathbf{89.53}_{\pm 0.20}$	$89.52_{\pm 0.25}$	$88.76_{\pm 0.30}$	$87.97_{\pm 0.31}$
Avg.	86.49	86.69	86.79	**87.27**	87.21	85.60	85.27

Table 4: Experiment results (UAS, %) on the UD 2.0 development set. Bold: best result per language.

In comparison to MH_4-two, the richer feature representation of MH_4-hybrid helps in all our languages.

Interestingly, MH_4 and MH_3 react differently to switching from global to greedy models. MH_4 covers more structures than MH_3, and is naturally more capable in the global case, even when the feature functions are the same (MH_4-two). However, its greedy version is outperformed by MH_3. We conjecture that this is because MH_4 explores only the same number of configurations as MH_3, despite the fact that introducing non-projectivity expands the search space dramatically.

Comparison with CoNLL Shared Task Results (Table 5) We compare our models on the test sets, along with the best single model (#1; Dozat et al., 2017) and the best ensemble model (#2; Shi et al., 2017b) from the CoNLL 2017 shared task. MH_4 outperforms 1EC in 7 out of the 10 languages. Additionally, we take our non-projective parsing models (MST, MH_4-hybrid, 1EC) and combine them into an ensemble. The average result is competitive with the best CoNLL submissions. Interestingly, Dozat et al. (2017) uses fully non-projective parsing algorithms (MST), and our ensemble system sees larger gains in the more non-projective languages, confirming the potential benefit of global mildly non-projective parsing.

Results on Projective Languages (Table 6) For completeness, we also test our models on the 10 most projective languages that have a single treebank. MH_4 remains the most effective, but by a much smaller margin. Interestingly, MH_3, which is strictly projective, matches the performance of 1EC; both outperform the fully non-projective MST by half a point.

6 Related Work

Exact inference for dependency parsing can be achieved in cubic time if the model is restricted to projective trees (Eisner, 1996). However, non-projectivity is needed for natural language parsers to satisfactorily deal with linguistic phenomena like topicalization, scrambling and extraposition, which cause crossing dependencies. In UD 2.0, 68 out of 70 treebanks were reported to contain

	Same Model Architecture					For Reference		
Lan.	MH_3	MST	MH_4-hybrid		1EC	Ensemble	CoNLL #1	CoNLL #2
eu	$78.17_{\pm 0.33}$	$79.90_{\pm 0.08}$	$\mathbf{80.22}_{\pm 0.48}$	$>$	$80.17_{\pm 0.32}$	**81.55**	81.44	79.61
ur	$\mathbf{80.91}_{\pm 0.10}$	$80.05_{\pm 0.13}$	$80.69_{\pm 0.19}$	$>$	$80.59_{\pm 0.19}$	81.37	**82.28**	81.06
got	$67.10_{\pm 0.10}$	$67.26_{\pm 0.45}$	$\mathbf{67.92}_{\pm 0.29}$	$>$	$67.66_{\pm 0.20}$	**69.83**	66.82	68.34
hu	$76.09_{\pm 0.25}$	$75.79_{\pm 0.36}$	$\mathbf{76.90}_{\pm 0.31}$	$>$	$76.07_{\pm 0.20}$	**79.35**	77.56	76.55
cu	$71.28_{\pm 0.29}$	$72.18_{\pm 0.20}$	$72.51_{\pm 0.23}$	$<$	$\mathbf{72.53}_{\pm 0.27}$	**74.38**	71.84	72.35
da	$80.00_{\pm 0.15}$	$79.69_{\pm 0.24}$	$\mathbf{80.89}_{\pm 0.17}$	$>$	$80.83_{\pm 0.27}$	82.09	**82.97**	81.55
el	$85.89_{\pm 0.29}$	$85.48_{\pm 0.25}$	$\mathbf{86.28}_{\pm 0.44}$	$>$	$86.07_{\pm 0.37}$	87.06	**87.38**	86.90
hi	$89.88_{\pm 0.18}$	$89.93_{\pm 0.12}$	$90.22_{\pm 0.12}$	$<$	$\mathbf{90.28}_{\pm 0.21}$	90.78	**91.59**	90.40
de	$76.23_{\pm 0.21}$	$75.99_{\pm 0.23}$	$\mathbf{76.46}_{\pm 0.20}$	$>$	$76.42_{\pm 0.35}$	77.38	**80.71**	77.17
ro	$83.53_{\pm 0.35}$	$82.73_{\pm 0.36}$	$83.67_{\pm 0.21}$	$<$	$\mathbf{83.83}_{\pm 0.18}$	84.51	**85.92**	84.40
Avg.	78.91	78.90	**79.57**	$>$	79.44	80.83	**80.85**	79.83

Table 5: Evaluation results (LAS, %) on the test set using the CoNLL 2017 shared task setup. The best results for each language within each block are highlighted in bold.

	Same Model Architecture				For Reference		
Lan.	MH_3	MST	MH_4-hybrid	1EC	Ensemble	CoNLL #1	CoNLL #2
ja	$\mathbf{74.29}_{\pm 0.10}$	$73.93_{\pm 0.16}$	$74.23_{\pm 0.11}$	$74.12_{\pm 0.12}$	74.51	**74.72**	74.51
zh	$\mathbf{63.54}_{\pm 0.13}$	$62.71_{\pm 0.17}$	$63.48_{\pm 0.33}$	$\mathbf{63.54}_{\pm 0.26}$	64.65	**65.88**	64.14
pl	$86.49_{\pm 0.19}$	$85.76_{\pm 0.31}$	$\mathbf{86.60}_{\pm 0.26}$	$86.36_{\pm 0.28}$	87.38	**90.32**	87.15
he	$61.47_{\pm 0.24}$	$61.28_{\pm 0.24}$	$\mathbf{61.93}_{\pm 0.22}$	$61.75_{\pm 0.22}$	62.40	**63.94**	62.33
vi	$41.26_{\pm 0.39}$	$41.04_{\pm 0.19}$	$\mathbf{41.33}_{\pm 0.32}$	$40.96_{\pm 0.36}$	**42.95**	42.13	41.68
bg	$87.50_{\pm 0.20}$	$87.03_{\pm 0.17}$	$\mathbf{87.63}_{\pm 0.17}$	$87.56_{\pm 0.14}$	88.22	**89.81**	88.39
sk	$80.48_{\pm 0.22}$	$80.25_{\pm 0.32}$	$\mathbf{81.27}_{\pm 0.14}$	$80.94_{\pm 0.25}$	82.38	**86.04**	81.75
it	$87.90_{\pm 0.07}$	$87.26_{\pm 0.23}$	$\mathbf{88.06}_{\pm 0.27}$	$87.98_{\pm 0.19}$	88.74	**90.68**	89.08
id	$\mathbf{77.66}_{\pm 0.13}$	$76.95_{\pm 0.32}$	$77.64_{\pm 0.17}$	$77.60_{\pm 0.18}$	78.27	**79.19**	78.55
lv	$69.62_{\pm 0.55}$	$69.33_{\pm 0.51}$	$\mathbf{70.54}_{\pm 0.51}$	$69.52_{\pm 0.29}$	72.34	**74.01**	71.35
Avg.	73.02	72.55	**73.27**	73.03	74.18	**75.67**	73.89

Table 6: CoNLL 2017 test set results (LAS, %) on the most projective languages (sorted by projective ratio; ja (Japanese) is fully projective).

non-projectivity (Wang et al., 2017).

However, exact inference has been shown to be intractable for models that support arbitrary non-projectivity, except under strong independence assumptions (McDonald and Satta, 2007). Thus, exact inference parsers that support unrestricted non-projectivity are limited to edge-factored models (McDonald et al., 2005; Dozat et al., 2017). Alternatives include treebank transformation and pseudo-projective parsing (Kahane et al., 1998; Nivre and Nilsson, 2005), approximate inference (e.g. McDonald and Pereira (2006); Attardi (2006); Nivre (2009); Fernández-González and Gómez-Rodríguez (2017)) or focusing on sets of dependency trees that allow only restricted forms of non-projectivity. A number of such sets, called mildly non-projective classes of trees, have been identified that both exhibit good empirical coverage of the non-projective phenomena found in natural languages and are known to have polynomial-time exact parsing algorithms; see Gómez-Rodríguez (2016) for a survey.

However, most of these algorithms have not been implemented in practice due to their prohibitive complexity. For example, Corro et al. (2016) report an implementation of the WG_1 parser, a $O(n^7)$ mildly non-projective parser introduced in Gómez-Rodríguez et al. (2009), but it could not be run for real sentences of length greater than 20. On the other hand, Pitler et al. (2012) provide an implementation of an $O(n^5)$ parser for a mildly non-projective class of structures called gap-minding trees, but they need to resort to aggressive pruning to make it practical, exploring only a part of the search space in $O(n^4)$ time.

To the best of our knowledge, the only practical system that actually implements exact inference for mildly non-projective parsing is the 1-Endpoint-Crossing (1EC) parser of Pitler (2013; 2014), which runs in $O(n^4)$ worst-case time like the MH_4 algorithm used in this paper. Thus, the system presented here is the second practical implementation of exact mildly non-projective parsing that has successfully been executed on real corpora.[6]

Comparing with Pitler (2014)'s 1EC, our parser has the following disadvantages: (-1) It has slightly lower coverage, at least on the treebanks considered by Gómez-Rodríguez (2016). (-2) The set of trees covered by MH_4 has not been characterized with a non-operational definition, while the set of 1-Endpoint-Crossing trees can be simply defined.

However, it also has the following advantages: (+1) It can be given a transition-based interpretation, allowing us to use transition-based scoring functions and to implement the analogous algorithm with greedy or beam search apart from exact inference. No transition-based interpretation is known for 1EC. While a transition-based algorithm has been defined for a strict subset of 1-Endpoint-Crossing trees, called 2-Crossing Interval trees (Pitler and McDonald, 2015), this is a separate algorithm with no known mapping or relation to 1EC or any other dynamic programming model. Thus, we provide the first exact inference algorithm for a non-projective transition-based parser with practical complexity. (+2) It is conceptually much simpler, with one kind of item and two deduction steps, while the 1-Endpoint-Crossing parser has five classes of items and several dozen distinct deduction steps. It is also a purely bottom-up parser, whereas the 1-Endpoint-Crossing parser does not have the bottom-up property. This property is necessary for models that involve compositional representations of subtrees (Dyer et al., 2015), and facilitates parallelization and partial parsing. (+3) It can be easily generalized to MH_k for $k > 4$, providing higher coverage, with time complexity $O(n^k)$. Out of the mildly non-projective parsers studied in Gómez-Rodríguez (2016), MH_4 provides the maximum coverage with respect to its complexity for $k > 4$. (+4) As shown in §5, MH_4 obtains slightly higher accuracy than 1EC on average, albeit not by a conclusive margin.

It is worth noting that 1EC has recently been extended to graph parsing by Kurtz and Kuhlmann (2017), Kummerfeld and Klein (2017), and Cao et al. (2017a,b), with the latter providing a practical implementation of a parser for 1-Endpoint-Crossing, pagenumber-2 graphs.

[6]Corro et al. (2016) describe a parser that enforces mildly non-projective constraints (bounded block degree and well-nestedness), but it is an arc-factored model, so it is subject to the same strong independence assumptions as maximum-spanning-tree parsers like McDonald et al. (2005) and does not support the greater flexibility in scoring that is the main advantage of mildly non-projective parsers over these. Instead, mild non-projectivity is exclusively used as a criterion to discard nonconforming trees.

7 Conclusion

We have extended the parsing architecture of Shi et al. (2017a) to non-projective dependency parsing by implementing the MH_4 parser, a mildly non-projective $O(n^4)$ chart parsing algorithm, using a minimal set of transition-based bi-LSTM features. For this purpose, we have established a mapping between MH_4 items and transition sequences of an underlying non-projective transition-based parser.

To our knowledge, this is the first practical implementation of exact inference for non-projective transition-based parsing. Empirical results on a collection of highly non-projective datasets from Universal Dependencies show improvements in accuracy over the projective approach of Shi et al. (2017a), as well as edge-factored maximum-spanning-tree parsing. The results are on par with the 1-Endpoint-Crossing parser of Pitler (2014) (re-implemented under the same neural framework), but our algorithm is notably simpler and has additional desirable properties: it is purely bottom-up, generalizable to higher coverage, and compatible with transition-based semantics.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments. CG has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01). TS and LL were supported in part by a Google Focused Research Grant to Cornell University. LL was also supported in part by NSF grant SES-1741441. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other sponsors.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 166–170, New York City, New York, USA.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017a. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2110–2120, Vancouver, Canada. Association for Computational Linguistics.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017b. Quasi-second-order parsing for 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 24–34, Copenhagen, Denmark. Association for Computational Linguistics.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 740–750, Doha, Qatar.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1234–1245, Edinburgh, Scotland, UK.

Caio Corro, Joseph Le Roux, Mathieu Lacroix, Antoine Rozenknop, and Roberto Wolfler Calvo. 2016. Dependency parsing with bounded block degree and well-nestedness via Lagrangian relaxation and branch-and-bound. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 355–366, Berlin, Germany. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2017. A full non-monotonic transition system for unrestricted non-projective parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 288–298, Vancouver, Canada. Association for Computational Linguistics.

Carlos Gómez-Rodríguez. 2016. Restricted non-projectivity: Coverage vs. efficiency. *Computational Linguistics*, 42(4):809–817.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology*, pages 968–976.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

Carlos Gómez-Rodríguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 291–299, Athens, Greece. Association for Computational Linguistics.

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

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1077–1086, Uppsala, Sweden.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, pages 646–652, Montreal, Quebec, Canada. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency parsing*, volume 2 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 673–682, Portland, Oregon, USA.

Jonathan K. Kummerfeld and Dan Klein. 2017. Parsing with traces: An $O(n^4)$ algorithm and a structural representation. *Transactions of the Association for Computational Linguistics*, 5:441–454.

Robin Kurtz and Marco Kuhlmann. 2017. Exploiting structure in parsing to 1-endpoint-crossing graphs. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 78–87, Pisa, Italy. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies (IWPT)*, pages 121–132, Prague, Czech Republic. Association for Computational Linguistics.

Ryan T. McDonald and Fernando C. N. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore. Association for Computational Linguistics.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70, Geneva, Switzerland. COLING.

Emily Pitler. 2014. A crossing-sensitive third-order factorization for dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:41–54.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 478–488. Association for Computational Linguistics.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association of Computational Linguistics*, 1:13–24.

Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 662–671, Denver, Colorado. Association for Computational Linguistics.

Tianze Shi, Carlos Gómez-Rodríguez, and Lillian Lee. 2018. Improving coverage and runtime complexity for exact inference in non-projective transition-based dependency parsers. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page (in press), New Orleans, Louisiana. Association for Computational Linguistics.

Tianze Shi, Liang Huang, and Lillian Lee. 2017a. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12–23, Copenhagen, Denmark.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017b. Combining global models for parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings* *of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903.

Hao Wang, Hai Zhao, and Zhisong Zhang. 2017. A transition-based system for universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 191–197, Vancouver, Canada. Association for Computational Linguistics.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 195–206.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

Constituency Parsing with a Self-Attentive Encoder

Nikita Kitaev and **Dan Klein**
Computer Science Division
University of California, Berkeley
{kitaev, klein}@cs.berkeley.edu

Abstract

We demonstrate that replacing an LSTM encoder with a self-attentive architecture can lead to improvements to a state-of-the-art discriminative constituency parser. The use of attention makes explicit the manner in which information is propagated between different locations in the sentence, which we use to both analyze our model and propose potential improvements. For example, we find that separating positional and content information in the encoder can lead to improved parsing accuracy. Additionally, we evaluate different approaches for lexical representation. Our parser achieves new state-of-the-art results for single models trained on the Penn Treebank: 93.55 F1 without the use of any external data, and 95.13 F1 when using pre-trained word representations. Our parser also outperforms the previous best-published accuracy figures on 8 of the 9 languages in the SPMRL dataset.

1 Introduction

In recent years, neural network approaches have led to improvements in constituency parsing (Dyer et al., 2016; Cross and Huang, 2016; Choe and Charniak, 2016; Stern et al., 2017a; Fried et al., 2017). Many of these parsers can broadly be characterized as following an encoder-decoder design: an encoder reads the input sentence and summarizes it into a vector or set of vectors (e.g. one for each word or span in the sentence), and then a decoder uses these vector summaries to incrementally build up a labeled parse tree. In contrast to the large variety of decoder architectures investigated in recent work, the encoders in recent parsers have predominantly been built using recurrent neural networks (RNNs), and in particular Long Short-Term Memory networks (LSTMs).

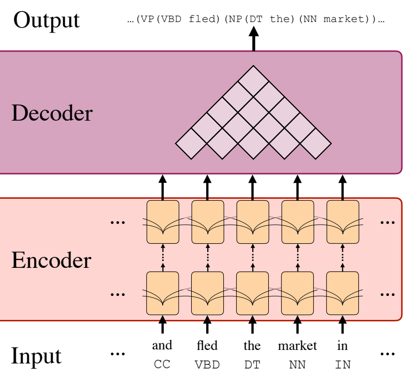


Figure 1: Our parser combines a chart decoder with a sentence encoder based on self-attention.

RNNs have largely replaced approaches such as the fixed-window-size feed-forward networks of Durrett and Klein (2015) in part due to their ability to capture global context. However, RNNs are not the only architecture capable of summarizing large global contexts: recent work by Vaswani et al. (2017) presented a new state-of-the-art approach to machine translation with an architecture that entirely eliminates recurrent connections and relies instead on a repeated neural attention mechanism. In this paper, we introduce a parser that combines an encoder built using this kind of self-attentive architecture with a decoder customized for parsing (Figure 1). In Section 2 of this paper, we describe the architecture and present our finding that self-attention can outperform an LSTM-based approach.

A neural attention mechanism makes explicit the manner in which information is transferred between different locations in the sentence, which we can use to study the relative importance of different kinds of context to the parsing task. Different locations in the sentence can attend to each other based on their positions, but also based on their contents (i.e. based on the words at or around those positions). In Section 3 we present our finding that when our parser learns to make an implicit trade-off between these two types of attention, it predominantly makes use of position-based attention, and show that explicitly factoring the two types of attention can noticeably improve parsing accuracy. In Section 4, we study our model's use of attention and reaffirm the conventional wisdom that sentence-wide global context is important for parsing decisions.

Like in most neural parsers, we find morphological (or at least sub-word) features to be important to achieving good results, particularly on unseen words or inflections. In Section 5.1, we demonstrate that a simple scheme based on concatenating character embeddings of word prefixes/suffixes can outperform using part-of-speech tags from an external system. We also present a version of our model that uses a character LSTM, which performs better than other lexical representations – even if word embeddings are removed from the model. In Section 5.2, we explore an alternative approach for lexical representations that makes use of pre-training on a large unsupervised corpus. We find that using the deep contextualized representations proposed by Peters et al. (2018) can boost parsing accuracy.

Our parser achieves 93.55 F1 on the Penn Treebank WSJ test set when not using external word representations, outperforming all previous single-system constituency parsers trained only on the WSJ training set. The addition of pre-trained word representations following Peters et al. (2018) increases parsing accuracy to 95.13 F1, a new state-of-the-art for this dataset. Our model also outperforms previous best published results on 8 of the 9 languages in the SPMRL 2013/2014 shared tasks. Code and trained English models are publicly available.[1]

2 Base Model

Our parser follows an encoder-decoder architecture, as shown in Figure 1. The decoder, described in Section 2.1, is borrowed from the chart parser of Stern et al. (2017a) with additional modifications from Gaddy et al. (2018). Their parser is architecturally streamlined yet achieves the highest performance among discriminative single-system parsers trained on WSJ data only, which is why we selected it as the starting point for our experiments with encoder variations. Sections 2.2 and 2.3 describe the base version of our encoder, where the self-attentive architecture described in Section 2.2 is adapted from Vaswani et al. (2017).

[1] https://github.com/nikitakit/self-attentive-parser

2.1 Tree Scores and Chart Decoder

Our parser assigns a real-valued score $s(T)$ to each tree T, which decomposes as

$$s(T) = \sum_{(i,j,l)\in T} s(i,j,l) \tag{1}$$

Here $s(i,j,l)$ is a real-valued score for a constituent that is located between fencepost positions i and j in a sentence and has label l. To handle unary chains, the set of labels includes a collapsed entry for each unary chain in the training set. The model handles n-ary trees by binarizing them and introducing a dummy label $\varnothing$ to nodes created during binarization, with the property that $\forall i,j : s(i,j,\varnothing) = 0$. Enforcing that scores associated with the dummy labels are always zero ensures that (1) continues to hold for all possible binarizations of an n-ary tree.

At test time, the model-optimal tree

$$\hat{T} = \arg\max_T s(T)$$

can be found efficiently using a CKY-style inference algorithm. Given the correct tree $T^\star$, the model is trained to satisfy the margin constraints

$$s(T^\star) \geq s(T) + \Delta(T, T^\star)$$

for all trees T by minimizing the hinge loss

$$\max\left(0, \max_{T\neq T^\star}\left[s(T) + \Delta(T,T^\star)\right] - s(T^\star)\right)$$

Here Δ is the Hamming loss on labeled spans, and the tree corresponding to the most-violated constraint can be found using a slight modification of the inference algorithm used at test time.

For further details, see Gaddy et al. (2018). The remainder of this paper concerns itself with the functional form of $s(i,j,l)$, which is calculated using a neural network for all $l \neq \varnothing$.

2.2 Context-Aware Word Representations

The encoder portion of our model is split into two parts: a word-based portion that assigns a context-aware vector representation y_t to each position t in the sentence (described in this section), and a chart portion that combines the vectors y_t to generate span scores $s(i,j,l)$ (Section 2.3). The architecture for generating the vectors y_t is adapted from Vaswani et al. (2017).

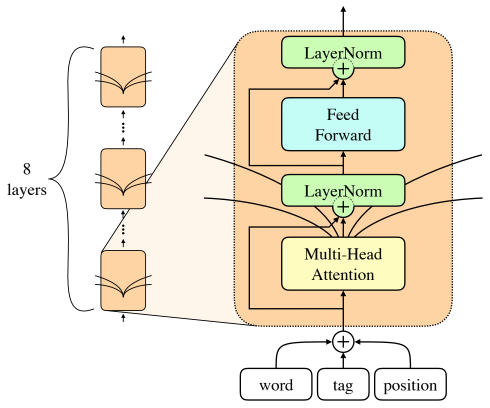


Figure 2: An overview of our encoder, which produces a context-aware summary vector for each word in the sentence. The multi-headed attention mechanism is the only means by which information may propagate between different positions in the sentence.

The encoder takes as input a sequence of word embeddings $[w_1, w_2, \ldots, w_T]$, where the first and last embeddings are of special *start* and *stop* tokens. All word embeddings are learned jointly with other parts of the model. To better generalize to words that are not seen during training, the encoder also receives a sequence of part-of-speech tag embeddings $[m_1, m_2, \ldots, m_T]$ based on the output of an external tagger (alternative lexical representations are discussed in Section 5). Additionally, the encoder stores a learned table of position embeddings, where every number $i \in 1, 2, \ldots$ (up to some maximum sentence length) is associated with a vector p_i. All embeddings have the same dimensionality, which we call d_{model}, and are added together at the input of the encoder: $z_t = w_t + m_t + p_t$.

The vectors $[z_1, z_2, \ldots, z_T]$ are transformed by a stack of 8 identical layers, as shown in Figure 2. Each layer consists of two stacked sublayers: a multi-headed attention mechanism and a position-wise feed-forward sublayer. The output of each sublayer given an input x is $\text{LayerNorm}(x + \text{SubLayer}(x))$, i.e. each sublayer is followed by a residual connection and a Layer Normalization (Ba et al., 2016) step. As a result, all sublayer outputs, including final outputs y_t, are of size d_{model}.

2.2.1 Self-Attention

The first sublayer in each of our 8 layers is a multi-headed self-attention mechanism, which is the only means by which information may propagate between positions in the sentence. The input

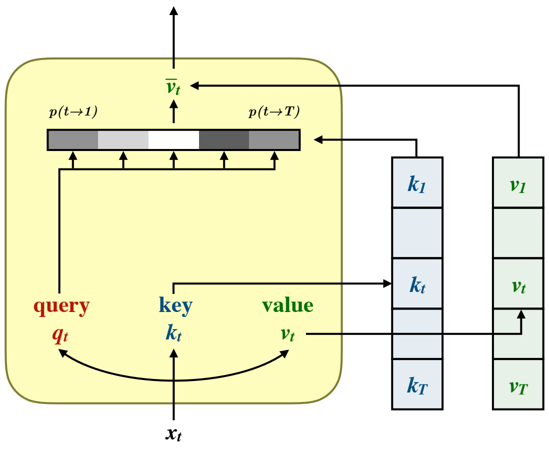


Figure 3: A single attention head. An input x_t is split into three vectors that participate in the attention mechanism: a query q_t, a key k_t, and a value v_t. The query q_t is compared with all keys to form a probability distribution $p(t \to \cdot)$, which is then used to retrieve an average value $\bar{v}_t$.

to the attention mechanism is a $T \times d_{model}$ matrix X, where each row vector x_t corresponds to word t in the sentence.

We first consider a single attention head, as illustrated in Figure 3. Learned parameter matrices W_Q, W_K, and W_V are used to map an input x_t to three vectors: a query $q_t = W_Q^\top x_t$, a key $k_t = W_K^\top x_t$, and a value $v_t = W_V^\top x_t$. Query and key vectors have the same number of dimensions, which we call d_k. The probability that word i attends to word j is then calculated as $p(i \to j) \propto \exp(\frac{q_i \cdot k_j}{\sqrt{d_k}})$. The values v_j for all words that have been attended to are aggregated to form an average value $\bar{v}_i = \sum_j p(i \to j) v_j$, which is projected back to size d_{model} using a learned matrix W_O. In matrix form, the behavior of a single attention head is:

$$\text{SingleHead}(X) = \left[\text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V\right] W_O$$
$$\text{where } Q = XW_Q;\ K = XW_K;\ V = XW_V$$

Rather than using a single head, our model sums together the outputs from multiple heads:

$$\text{MultiHead}(X) = \sum_{n=1}^{8} \text{SingleHead}^{(n)}(X)$$

Each of the 8 heads has its own trainable parameters $W_Q^{(n)}$, $W_K^{(n)}$, $W_V^{(n)}$, and $W_O^{(n)}$. This allows a word to gather information from up to 8 remote locations in the sentence at each attention sublayer.

2.2.2 Position-Wise Feed-Forward Sublayer

We use the same form as Vaswani et al. (2017):

$$\text{FeedForward}(x) = W_2\text{relu}(W_1 x + b_1) + b_2$$

Here *relu* denotes the Rectified Linear Unit nonlinearity, and distinct sets of learned parameters are used at each of the 8 instances of the feed-forward sublayer in our model.

The input and output dimensions are the same because of the use of residual connections throughout the model, but we can vary the number of parameters by adjusting the size of the intermediate vector that the nonlinearity is applied to.

2.3 Span Scores

The outputs y_t from the word-based encoder portion described in the previous section are combined to form span scores $s(i, j, \cdot)$ following the method of Stern et al. (2017a). Concretely,

$$s(i, j, \cdot) = M_2\text{relu}(\text{LayerNorm}(M_1 v + c_1)) + c_2$$

where LayerNorm denotes Layer Normalization, *relu* is the Rectified Linear Unit nonlinearity, and $v = [\overrightarrow{y}_j - \overrightarrow{y}_i; \overleftarrow{y}_{j+1} - \overleftarrow{y}_{i+1}]$ combines summary vectors for relevant positions in the sentence. A span endpoint to the right of the word potentially requires different information from the endpoint to the left, so a word at a position k is associated with *two* annotation vectors ($\overrightarrow{y}_k$ and $\overleftarrow{y}_k$).

Stern et al. (2017a) define $\overrightarrow{y}_k$ and $\overleftarrow{y}_k$ in terms of the output of the forward and backward portions, respectively, of their BiLSTM encoder; we instead construct each of $\overrightarrow{y}_k$ and $\overleftarrow{y}_k$ by splitting in half[2] the outputs y_k from Section 2.2. We also introduce a Layer Normalization step to match the use of Layer Normalization throughout our model.

2.4 Results

The model presented above achieves a score of 92.67 F1 on the Penn Treebank WSJ development set. Details regarding hyperparameter choice and optimizer settings are presented in the supplementary material. For comparison, a model that uses the same decode procedure with an LSTM-based encoder achieves a development set score of 92.24 (Gaddy et al., 2018). These results demonstrate that an RNN-based encoder is not required for building a good parser; in fact, self-attention can achieve better results.

[2]To avoid an adverse interaction with material described in Section 3, when a vector y_k is split in half the even coordinates contribute to $\overrightarrow{y}_k$ and the odd coordinates contribute to $\overleftarrow{y}_k$.

3 Content vs. Position Attention

The primary mechanism for information transfer throughout our encoder is self-attention, where words can attend to each other using both content features and position information. In Section 2, we described an encoder that takes as input a component-wise addition between a word, tag, and position embedding for each word in the sentence. Content and position information are intermingled throughout the network. While ideally the network would learn to balance the different types of information, in practice it does not. In this section we show that factoring the model to explicitly separate content and position information results in increased parsing accuracy.

To help gauge the relative importance of the two types of attention, we trained a modified version of our model that was only allowed to use position attention. This constraint was enforced by making the query and key vectors used for the attention mechanism be linear transformations of the corresponding word's position embedding: $Q^{(n)} = PW_Q^{(n)}$ and $K^{(n)} = PW_K^{(n)}$. The per-head weight matrices now multiply a matrix P containing the same position embeddings that are used at the input to the encoder, rather than the layer input X (as in Section 2.2.1). However, value vectors $V^{(n)} = XW_V^{(n)}$ remain unchanged and continue to carry content-related information.

We expected our parser to still achieve reasonable performance when restricted to only use positional attention because the resulting architecture can be viewed as a generalization of a multi-layer convolutional neural network. The 8 attention heads at each layer of our model can mimic the behavior of a size-8 convolutional filter, but can also determine their attention targets dynamically and need not respect any translation-invariance properties. Disabling content-based attention throughout all 8 layers of the network results in a development-set accuracy decrease of only 0.27 F1. While we expected reasonable parsing performance in this setting, it seems strange that content-based attention benefits our model to such a small degree.

We next investigate the possibility that intermingling content and position information in a single vector can cause one type of attention to dominate over the other and compromise the network's ability to find the optimal balance of the two. To do this we propose a factored version of our model that explicitly separates content and position information.

A first step is to replace the component-wise addition $z_t = w_t + m_t + p_t$ (where w_t, m_t, and p_t represent word, tag, and position embeddings, respectively) with a concatenation $z_t = [w_t + m_t; p_t]$. We preserve the size of the vector z_t by cutting the dimensionality of embeddings in half for the concatenative scheme. However, simply isolating the position-related components of the input vectors in this manner does not improve the performance of our network: the concatenative network achieves a development-set F1 of 92.60 (not much different from 92.67 F1 using the model in Section 2).

The issue with intermingling information is not the component-wise addition per se. In fact, concatenation and addition often perform similarly in high dimensions (especially when the resulting vector is immediately multiplied by a matrix that intermingles the two sources of information). On that note, we can examine how the mixed vectors are used later in the network, and in particular in the query-key dot products for the attention mechanism. If we have a query-key dot product $q \cdot k$ (see Section 2.2.1) where we imagine q decomposing into content and positional information as $q = q^{(c)} + q^{(p)}$ (and likewise for k), we have $q \cdot k = (q^{(c)} + q^{(p)}) \cdot (k^{(c)} + k^{(p)})$. This formulation includes cross-terms such as $q^{(c)} \cdot k^{(p)}$; for example it is possible to learn a network where the word *the* always attends to the *5th position* in the sentence. Such cross-attention seems of limited use compared to the potential for overfitting that it introduces.

To complete our factored model, we find all cases where a vector $x = [x^{(c)}; x^{(p)}]$ is multiplied by a parameter matrix, and replace the matrix multiplication $c = Wx$ with a split form $c = [c^{(c)}; c^{(p)}] = [W^{(c)}x^{(c)}; W^{(p)}x^{(p)}]$. This causes a number of intermediate quantities in our model to be factored, including all query and key vectors. Query-key dot products now decompose as $q \cdot k = q^{(c)} \cdot k^{(c)} + q^{(p)} \cdot k^{(p)}$. The result of factoring a single attention head, shown in Figure 4, can also be viewed as separately applying attention to $x^{(c)}$ and $x^{(p)}$, except that the log-probabilities in the two halves are added together prior to value lookup. The feed-forward sublayers in our model (Section 2.2.2) are likewise split into two independent portions that operate on position and content information.

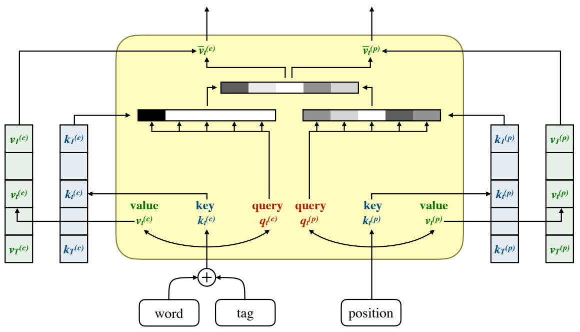


Figure 4: A single attention head, after factoring content and position information. Attention probabilities are calculated separately for the two types of information, and a combined probability distribution is then applied to both types of input information.

Alternatively, factoring can be seen as enforcing the block-sparsity constraint

$$W = \begin{bmatrix} W^{(c)} & 0 \\ 0 & W^{(p)} \end{bmatrix}$$

on parameter matrices throughout our model. We maintain the same vector sizes as in Section 2, which means that factoring strictly reduces the number of trainable parameters. For simplicity, we split each vector into equal halves that contain position and content information, cutting the number of model parameters roughly in half. This factored scheme is able to achieve 93.15 development-set F1, an improvement of almost 0.5 F1 over the unfactored model.

These results suggest that factoring different types of information leads to a better parser, but there is in principle a confound: perhaps by making all matrices block-sparse we've stumbled across a better hyperparameter configuration. For example, these gains could be due to a difference in the number of trainable parameters alone. To control for this confound we also evaluated a version of our model that enforces block-sparsity throughout, but retains the use of component-wise addition at the inputs. This model achieves 92.63 F1 (not much different from the unfactored model), which supports our hypothesis that true factoring of information is important.

Attention		
Content	Position	F1
All 8 layers	All 8 layers	93.15
All 8 layers	Disabled	72.45
Disabled	All 8 layers	90.84
First 4 layers only	All 8 layers	91.77
Last 4 layers only	All 8 layers	92.82
First 6 layers only	All 8 layers	92.42
Last 6 layers only	All 8 layers	92.90

Table 1: Development-set F1 scores when content and/or position attention is selectively disabled *at test-time only* for a subset of the layers in our model. Position attention is the most important contributor to our model, but content attention is also helpful (especially at the final layers of the encoder).

Distance	F1 (strict)	F1 (relaxed)
5	81.65	89.82
10	89.83	92.20
15	91.72	92.78
20	92.48	92.91
30	93.01	93.09
40	93.04	93.12
∞	93.15	

Table 2: Development-set F1 scores when attention is constrained to not exceed a particular distance in the sentence *at test time only*. In the *relaxed* setting, the first and last two tokens of the sentence can attend to any word and be attended to by any word, to allow for sentence-wide pooling of information.

4 Analysis of our Model

The defining feature of our encoder is the use of self-attention, which is the only mechanism for transfer of information between different locations throughout a sentence. The attention is further factored into types: content-based attention and position-based attention. In this section, we analyze the manner in which our model uses this attention mechanism to make its predictions.

4.1 Content vs. Position Attention

To examine the relative utilization of content-based vs. position-based attention in our architecture, we perturb a trained model at test-time by selectively zeroing out the contribution of either the content or the position component to any attention mechanism. This can be done independently at different layers; the results of this experiment are shown in Table 1.

We can see that our model learns to use a combination of the two attention types, with position-based attention being the most important. We also see that content-based attention is more useful at later layers in the network, which is consistent with the idea that the initial layers of our model act similarly to a dilated convolutional network while the upper layers have a greater balance between the two attention types.

4.2 Windowed Attention

We can also examine our model's use of long-distance context information by applying windowing to the attention mechanism. We begin by taking our trained model and windowing the attention mechanism at test-time only. As shown in Table 2, strict windowing yields poor results: even a window of size 40 causes a loss in parsing accuracy compared to the original model. When we began to investigate *how* the model makes use of long-distance attention, we immediately found that there are particular attention heads at some layers in our model that almost always attend to the start token. This suggests that the start token is being used as the location for some sentence-wide pooling/processing, or perhaps as a dummy target location when a head fails to find the particular phenomenon that it's learned to search for. In light of this observation, we introduce a *relaxed* variation on the windowing scheme, where the start token, first word, last word, and stop token can participate in all possible uses of attention, but pairs of other words in the sentence can only attend to each other if they are within a given window. We include three other positions in addition to the start token to do our best to cover possible locations for global pooling by our model. Results for relaxed windowing at test-time only are also shown in Table 2. Even when we allow global processing to take place at designated locations such as the start token, our model is able to make use of long-distance dependencies at up to length 40.

Next, we examine whether the parser's use of long-distance dependencies is actually essential to performing the task by retraining our model subject to windowing. To evaluate the role of global

Distance	F1 (strict)	F1 (relaxed)
5	92.74	92.94
10	92.92	93.00
20	93.06	93.17
∞	93.15	

Table 3: Development-set F1 scores when attention is constrained to not exceed a particular distance in the sentence *during training and at test time*. In the *relaxed* setting, the first and last two tokens of the sentence can attend to any word and be attended to by any word, to allow for sentence-wide pooling of information.

computation, we consider both strict and relaxed windowing. In principle we could have replaced relaxed windowing at training time with explicit provisions for global computation, but for analysis purposes we choose to minimize departures from our original architecture.

The results, shown in Table 3, demonstrate that long-distance dependencies continue to be essential for achieving maximum parsing accuracy using our model. Note that when a window of size 10 was imposed at training time, this was *per-layer* and the series of 8 layers actually had an effective context size of around *80* – which was still insufficient to recover the performance of our full parser (with either approach to windowing). The side-by-side comparison of strict and relaxed windowing shows that the ability to pool global information, using the designated locations that are always available in the relaxed scheme, consistently translates to accuracy gains but is insufficient to compensate for small window sizes. This suggests that not only must the information signal from long-distance tokens be available in principle, but that it also helps to have this information be directly accessible without an intermediate bottleneck.

5 Lexical Models

The models described in previous sections all rely on pretagged input sentences, where the tags are predicted using the Stanford tagger. We use the same pretagged dataset as Cross and Huang (2016). In this section we explore two alternative classes of lexical models: those that use no external systems or data of any kind, as well as word vectors that are pretrained in an unsupervised manner.

	Word embeddings	
	✓	✗
None	92.20	–
Tags	93.15	–
CharLSTM	93.40	93.61
CharConcat	93.32	93.35

Table 4: Development-set F1 scores for different approaches to handling morphology, with and without the addition of learned word embeddings.

5.1 Models with Subword Features

If tag embeddings are removed from our model and only word embeddings remain (where word embeddings are learned jointly with other model parameters), performance suffers by around 1 F1. To restore performance without introducing any dependencies on an external system, we explore incorporating lexical features directly into our model. The results for different approaches we describe in this section are shown in Table 4.

We first evaluate an approach (CharLSTM) that independently runs a bidirectional LSTM over the characters in each word and uses the LSTM outputs in place of part-of-speech tag embeddings. We find that this approach performs better than using predicted part-of-speech tags. We can further remove the word embeddings (leaving the character LSTMs only), which does not seem to hurt and can actually help increase parsing accuracy.

Next we examine the importance of recurrent connections by constructing and evaluating a simpler alternative. Our approach (CharConcat) is inspired by Hall et al. (2014), who found it effective to replace words with frequently-occurring suffixes, and the observation that our original tag embeddings are rather high-dimensional. To represent a word, we extract its first 8 letters and last 8 letters, embed each letter, and concatenate the results. If we use 32-dimensional embeddings, the 16 letters can be packed into a 512-dimensional vector – the same size as the inputs to our model. This size for the inputs in our model was chosen to simplify the use of residual connections (by matching vector dimensions), even though the inputs themselves could have been encoded in a smaller vector. This allows us to directly replace tag embeddings with the 16-letter prefix/suffix concatenation. For short words, embeddings of a padding token are inserted as needed. Words longer than 16 letters are represented in a lossy manner by this concatenative approach, but we hypothesize that prefix/suffix information is enough for our task. We find this simple scheme remarkably effective: it is able to outperform pretagging and can operate even in the absence of word embeddings. However, its performance is ultimately not quite as good as using a character LSTM.

Given the effectiveness of the self-attentive encoder at the sentence level, it is aesthetically appealing to consider it as a sub-word architecture as well. However, it was empirically much slower, did not parallelize better than a character-level LSTM (because words tend to be short), and initial results underperformed the LSTM. One explanation is that in a lexical model, one only wants to compute a single vector per word, whereas the self-attentive architecture is better adapted for producing context-aware summaries at multiple positions in a sequence.

5.2 External Embeddings

Next, we consider a version of our model that uses external embeddings. Recent work by Peters et al. (2018) has achieved state-of-the-art performance across a range of NLP tasks by augmenting existing models with a new technique for word representation called ELMo (Embeddings from Language Models). Their approach is able to capture both subword information and contextual clues: the embeddings are produced by a network that takes characters as input and then uses an LSTM to capture contextual information when producing a vector representation for each word in a sentence.

We evaluate a version of our model that uses ELMo as the sole lexical representation, using publicly available ELMo weights. These pre-trained word representations are 1024-dimensional, whereas all of our factored models thus far have 512-dimensional content representations; we found that the most effective way to address this mismatch is to project the ELMo vectors to the required dimensionality using a learned weight matrix. With the addition of contextualized word representations, we hypothesized that a full 8 layers of self-attention would no longer be necessary. This proved true in practice: our best development set result of 95.21 F1 was obtained with a 4-layer encoder.

Encoder Architecture	F1 (dev)	Δ
LSTM (Gaddy et al., 2018)	92.24	-0.43
Self-attentive (Section 2)	92.67	0.00
+ Factored (Section 3)	93.15	0.48
+ CharLSTM (Section 5.1)	93.61	0.94
+ ELMo (Section 5.2)	95.21	2.54

Table 5: A comparison of different encoder architectures and their development-set performance relative to our base self-attentive model.

	LR	LP	F1
Single model, WSJ only			
Vinyals et al. (2015)	–	–	88.3
Cross and Huang (2016)	90.5	92.1	91.3
Gaddy et al. (2018)	91.76	92.41	92.08
Stern et al. (2017b)	92.57	92.56	92.56
Ours (CharLSTM)	**93.20**	**93.90**	**93.55**
Multi-model/External			
Durrett and Klein (2015)	–	–	91.1
Vinyals et al. (2015)	–	–	92.8
Dyer et al. (2016)	–	–	93.3
Choe and Charniak (2016)	–	–	93.8
Liu and Zhang (2017)	–	–	94.2
Fried et al. (2017)	–	–	94.66
Ours (ELMo)	94.85	95.40	**95.13**

Table 6: Comparison of F1 scores on the WSJ test set.

6 Results

6.1 English (WSJ)

The development set scores of the parser variations presented in previous sections are summarized in Table 5. Our best-performing parser used a factored self-attentive encoder over ELMo word representations.

The results of evaluating our model on the test set are shown in Table 6. The test score of 93.55 F1 for our CharLSTM parser exceeds the previous best numbers for single-system parsers trained on the Penn Treebank (without the use of any external data, such as pre-trained word embeddings). When our parser is augmented with ELMo word representations, it achieves a new state-of-the-art score of 95.13 F1 on the WSJ test set.

Our WSJ-only parser took 18 hours to train using a single Tesla K80 GPU and can parse the

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
Dev (all lengths)										
Coavoux and Crabbé (2017)	83.07	88.35	82.35	88.75	90.34	91.22	86.78[b]	**94.0**	79.64	87.16
Ours (CharLSTM only)	**85.94**	**90.05**	84.27	91.26	90.50	92.23	**87.90**	93.94	79.34	**88.38**
Ours (CharLSTM + word embeddings)	85.59	89.31	**84.42**	**91.39**	**90.78**	**92.32**	87.62	93.76	**79.71**	88.32
Test (all lengths)										
Björkelund et al. (2014), ensemble	81.32[a]	88.24	82.53	81.66	89.80	91.72	83.81	90.50	**85.50**	86.12
Cross and Huang (2016)	–	–	83.31	–	–	–	–	–	–	–
Coavoux and Crabbé (2017)	82.92[b]	88.81	82.49	85.34	89.87	92.34	86.04	93.64	84.0	87.27
Ours (model selected on dev)	**85.61**	**89.71**	**84.06**	**87.69**	**90.35**	**92.69**	**86.59**	**93.69**	84.45	**88.32**
Δ: Ours - Best Previous	**+2.69**	**+0.90**	**+0.75**	**+2.35**	**+0.48**	**+0.35**	**+0.55**	**+0.05**	-1.05	

Table 7: Results on the SPMRL dataset. All values are F1 scores calculated using the version of `evalb` distributed with the shared task. [a]Björkelund et al. (2013) [b]Uses character LSTM, whereas other results from Coavoux and Crabbé (2017) use predicted part-of-speech tags.

1,700-sentence WSJ development set in 8 seconds. When using ELMo embeddings, training time was 13 hours (not including the time needed to pre-train the word embeddings) and parsing the development set takes 24 seconds. Training and inference times are dominated by neural network computations; our single-threaded Cython implementation of the chart decoder (Section 2.1) consumes a negligible fraction of total running time.

6.2 Multilingual (SPMRL)

We tested our model's ability to generalize across languages by training it on the nine languages represented in the SPMRL 2013/2014 shared tasks (Seddah et al., 2013). To verify that our lexical representations can function for morphologically-rich languages and smaller treebanks, we restricted ourselves to running a subset of the exact models that we evaluated on English. In particular, we evaluated the model that uses a character-level LSTM, with and without the addition of learned word embeddings. We did not evaluate ELMo in the multilingual setting because pre-trained ELMo weights were only available for English. Hyper-parameters were unchanged compared to the English model with the exception of the learning rate, which we adjusted for some of the smaller datasets in the SPMRL task (see Table 9 in the supplementary material). Results are shown in Table 7.

Development set results show that the addition of word embeddings to a model that uses a character LSTM has a mixed effect: it improves performance for some languages, but hurts for others. For each language, we selected the trained model that performed better on the development set and evaluated it on the test set. On 8 of the 9 languages, our test set result exceeds the previous best-published numbers from any system we are aware of. The exception is Swedish, where the model of Björkelund et al. (2014) continues to be state-of-the-art despite a number of approaches proposed in the intervening years that have achieved better performance on other languages. We note that their model uses ensembling (via product grammars) and a reranking step, whereas our model was only evaluated in the single-system condition.

7 Conclusion

In this paper, we show that the choice of encoder can have a substantial effect on parser performance. In particular, we demonstrate state-of-the-art parsing results with a novel encoder based on factored self-attention. The gains we see come not only from incorporating *more* information (such as subword features or externally-trained word representations), but also from structuring the architecture to *separate* different kinds of information from each other. Our results suggest that further research into different ways of encoding utterances can lead to additional improvements in both parsing and other natural language processing tasks.

Acknowledgments

NK is supported by an NSF Graduate Research Fellowship. This research used the Savio computational cluster provided by the Berkeley Research Computing program at the University of California, Berkeley.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv:1607.06450 [cs, stat]*. ArXiv: 1607.06450.

Anders Björkelund, Ozlem Cetinoglu, Agnieszka Faleńska, Richárd Farkas, Thomas Mueller, Wolfgang Seeker, and Zsolt Szántó. 2014. The IMS-Wrocław-Szeged-CIS entry at the SPMRL 2014 shared task: Reranking and morphosyntax meet unlabeled data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 97–102.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (Re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 331–336. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 228–237.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017a. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827. Association for Computational Linguistics.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017b. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio,

H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2755–2763. Curran Associates, Inc.

Pre- and In-Parsing Models for Neural Empty Category Detection

Yufei Chen, Yuanyuan Zhao, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
`yufei.chen@pku.edu.cn, zhaoyy1461@gmail.com,`
`{ws,wanxiaojun}@pku.edu.cn`

Abstract

Motivated by the positive impact of empty categories on syntactic parsing, we study neural models for pre- and in-parsing detection of empty categories, which has not previously been investigated. We find several non-obvious facts: (a) BiLSTM can capture non-local contextual information which is essential for detecting empty categories, (b) even with a BiLSTM, syntactic information is still able to enhance the detection, and (c) automatic detection of empty categories improves parsing quality for overt words. Our neural ECD models outperform the prior state-of-the-art by significant margins.

1 Introduction

Encoding unpronounced nominal elements, such as dropped pronouns and traces of dislocated elements, the empty category is an important piece of machinery in representing the (deep) syntactic structure of a sentence (Carnie, 2012). Figure 1 shows an example. In linguistic theory, e.g. Government and Binding (GB; Chomsky, 1981), empty category is a key concept bridging S-Structure and D-Structure, due to its possible contribution to trace *movements*. In practical treebanking, empty categories have been used to indicate long-distance dependencies, discontinuous constituents, and certain dropped elements (Marcus et al., 1993; Xue et al., 2005). Recently, there has been an increasing interest in automatic empty category detection (ECD; Johnson, 2002; Seeker et al., 2012; Xue and Yang, 2013; Wang et al., 2015). And it has been shown that ECD is able to improve the linear model-based dependency parsing (Zhang et al., 2017b).

There are two key dimensions of approaches for ECD: the relationship with parsing and statistical disambiguation. Considering the relationship with parsing, we can divide ECD models into three types: (1) Pre-parsing approach (e.g. Dienes and Dubey (2003)) where empty categories are identified without using syntactic analysis, (2) In-parsing approach (e.g. Cai et al. (2011); Zhang et al. (2017b)) where detection is integrated into a parsing model, and (3) Post-parsing approach (e.g. Johnson (2002); Wang et al. (2015)) where parser outputs are utilized as clues to determine the existence of empty categories. For disambiguation, while early work on dependency parsing focused on linear models, recent work started exploring deep learning techniques for the post-parsing approach (Wang et al., 2015). From the above two dimensions, we show all existing systems for ECD in Table 1. Neural models for pre- and in-parsing ECD have not been studied yet. In this paper, we fill this gap in the literature.

	Pre-Parsing	In-Parsing	Post-Parsing
Linear	✔	✔	✔
Neural	✘	✘	✔

Table 1: ECD approaches that have been investigated.

It is obvious that empty categories are highly related to surface syntactic analysis. To determine the existence of empty elements between two overt words relies on not only the *sequential contexts* but also the *hierarchical contexts*. Traditional linear structured prediction models, e.g. conditional random fields (CRF), for sequence structures are rather weak to capture hierarchical contextual information which is essentially *non-local* for their architectures. Accordingly, pre-parsing models based on linear disambiguation techniques fail to produce comparable accuracy to the other two models. In striking contrast, RNN based se-

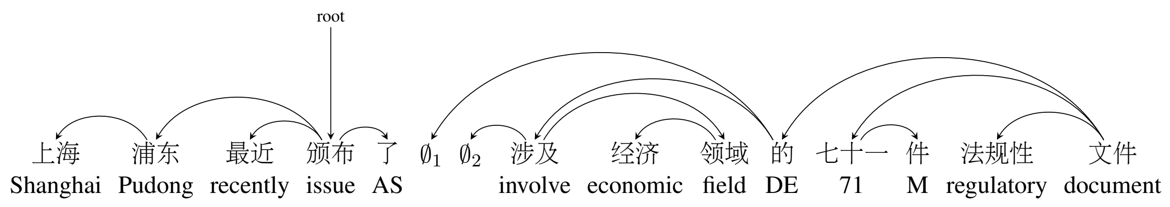


Figure 1: An example from CTB: *Shanghai Pudong recently enacted 71 regulatory documents involving the economic fields*. The dependency structure is according to Xue (2007). "$\emptyset_1$" indicates a null operator that represents empty relative pronouns. "$\emptyset_2$" indicates a trace left by relativization.

quence labeling models have been shown very powerful to capture non-local information, and therefore have great potential to advance the pre-parsing approach for ECD. In this paper, we propose a new bidirectional LSTM (BiLSTM) model for pre-parsing ECD using information about contextual words.

Previous studies highlight the usefulness of syntactic analysis for ECD. Furthermore, syntactic parsing of overt words can benefit from detection of empty elements and vice versa (Zhang et al., 2017b). In this paper, we follow Zhang et al.'s encouraging results obtained with linear models and study first- and second-order neural models for in-parsing ECD. The main challenge for neural in-parsing ECD is to encode empty element candidates and integrate the corresponding embeddings into a parsing model. We focus on the state-of-the-art parsing architecture developed by Kiperwasser and Goldberg (2016) and Dozat and Manning (2016), which use BiLSTMs to extract features from contexts followed by a nonlinear transformation to perform local scoring.

To evaluate the effectiveness of deep learning techniques for ECD, we conduct experiments on a pro-drop language, i.e. Chinese. The empirical evaluation indicates some non-obvious facts:

1. Neural ECD models outperform the prior state-of-the-art by significant margins. Even a pre-parsing model without any syntactic information outperforms the best existing linear in-parsing and post-parsing ECD models.

2. Incorporating empty elements can help neural dependency parsing. This parallels Zhang et al.'s investigation on linear models.

3. Our in-parsing neural models obtain better predictions than the pre-parsing model.

The implementation of all models is available at `https://github.com/draplater/empty-parser`.

2 Pre-Parsing Detection

2.1 Context of Empty Categories

Sequential Context Perhaps, it is the most intuitive idea to view a natural language sentence as a word-by-word sequence. Analyzing contextual information by modeling neighboring words according to this sequential structure is a very basic view for dealing with a large number of NLP tasks, e.g. POS tagging and syntactic parsing. It is also important to consider sequential contexts for ECD to derive the horizontal features that exploit the lexical context of the current pending point, presented as one or more preceding and following word tokens, as well as their part-of-speech tags (POS).

Hierarchical Context The detection of ECs requires broad contextual knowledge. Besides one-dimensional representation, vertical features are equally essential to express the empty element. The hierarchical structure is a compact reflection of the syntactic content. By integrating the hierarchical context, we can analyze the regular distributional pattern of ECs in a syntactic tree. More specifically, it means considering the head information of the EC and relevant dependencies to augment the prediction.

Both *sequential* and *hierarchical* contexts are essential to determine the existence of empty elements between two overt words. Even words close to each other in a hierarchical structure may appear far apart in sequential representations, which makes it hard for linear sequential tagging models to catch the hierarchical contextual information. RNN based sequence models have been proven very powerful to capture non-local features. In this paper, we show that LSTM is able to advance the pre-parsing ECD significantly.

Interspace:	@@ 颁布(issue)	@@ 了(AS)	@@ 涉及(involve)	@@ 经济(economic)
	O VV	O AS	*OP**T* VV	O NN
Pre2 and *Pre3*:	颁布(issue)	了(AS)	涉及(involve)	经济(economic)
	VV	AS	VV#pre1=*T*#pre2=*OP*	NN
Prepost:	颁布(issue)	了(AS)	涉及(involve)	经济(economic)
	VV	AS#post=*OP*	VV#pre1=*T*	NN

Figure 2: An example of four kinds of annotations. The phrase is cut out from the sentence in Figure 1. "@@" means interspaces between words.

2.2 A Sequence-Oriented Model

In the sequence-oriented model, we formulate ECD as a sequence labeling problem. In general, we attach ECs to surrounding overt tokens to represent their identifications, i.e. their locations and types. We explore four sets of annotation specifications, denoted as *Interspace*, *Pre2*, *Pre3* and *Prepost*, respectively. Following is the detailed descriptions.

Interspace We convert ECs' information into different tags of the interspaces between words. The assigned tag is the concatenation of ECs between the two words. If there is no EC, we just tag the interspace as *O*. Specially, according to our observation that only one EC occurs at the end of the sentence in our data set, we simply count on the heading space of sentences instead of the one standing at the end. Assume that there are n words in a given sentence, then there will be $2 * n$ items (n words and n interspaces) to tag.

Pre2 and Pre3 We stick ECs to words following them. In experiments using POS information, ECs are attached to the POS of the next word, while the normal words are just tagged with their POS. In experiments without POS information, ECs are straightly regarded as the label of the following words. Words without ECs ahead are consistently tagged using an empty marker. Similar to *Interspace*, linearly consecutive ECs are concatenated as a whole. *Pre2* means that at most two preceding consecutive ECs are considered while *Pre3* limits the considered continuous length to three. The determination of window lengths are grounded in the distribution of ECs' continuous lengths as shown in Table 2.

Prepost Considering that it may be a challenge to capture long-distance features, we introduce another labeling rule called *Prepost*. Different from *Pre2* and *Pre3*, the responsibility for presenting ECs will be shared by both the preceding and the following words. Whereas, tags heading sentences will remain unchanged. Particularly, if the amount of consecutive ECs in the current position is an odd number, we choose to attach the extra EC to the following word for consistency and clarity.

	1	2	3	4
Train	7499	3702	142	5
Dev	530	233	10	0
Test	900	433	19	0

Table 2: The distribution of ECs' continuous lengths in training, development and test data.

Take part of the sentence in Figure 1 as an example. As described above, the four kinds of representations are depicted in Figure 2. To investigate the effect of POS in the tagging process, we also conduct experiments by integrating POS to the tagging process. For *Interspace*, POS tags are individual output labels, while for other representations, the POS information is used to divide an empty category integrated tag into subtypes.

2.3 Tagging Based on LSTM-CRF

In order to capture long-range syntactic information for accurate disambiguation in pre-parsing phase, we build a LSTM-CRF model inspired by the neural network proposed in Ma and Hovy (2016). A BiLSTM layer is set up on character embeddings for extracting character-level representations of each word, which is concatenated with the pre-trained word embedding before feeding into another BiLSTM layer to capture contextual information. Thus we have obtained dense and continuous representations of the words in given sentences. The last part is to decode with linear chain CRF which can optimize the output sequence by factoring in local characteristics. Dropout layers both before and after the sentence-level network serve to prevent over-fitting.

3 In-Parsing Detection

Zhang et al. (2017b) designs novel algorithms to produce dependency trees in which empty elements are allowed. Their results show that integrating empty categories can augment the parsing of overt tokens when structured perceptron, a global linear model, is applied for disambiguation. From a different perspective, by jointing ECD and dependency parsing, we can utilize full syntactic information in the process of detecting ECs. Parallel to their work, we explore the effect of ECD on the neural dependency based parsing in this section.

3.1 Joint ECD and Dependency Parsing

To perform ECD and dependency parsing in a unified framework, we formulate the issue as an optimization problem. Assume that we are given a sentence s with n normal words. We use an index set $\mathcal{I}_o = \{(i,j)|i,j \in \{1,\cdots,n\}\}$ to denote all possible overt dependency edges, and use $\mathcal{I}_c = \{(i,\phi_j)|i,j \in \{1,\cdots,n\}\}$ to denote all possible covert dependency edges. ϕ_j denotes an empty node that precede the jth word. Then a dependency parse with empty nodes can be represented as a vector:

$$\boldsymbol{z} = \{z(i,j) : (i,j) \in \mathcal{I}_o \cup \mathcal{I}_c\}.$$

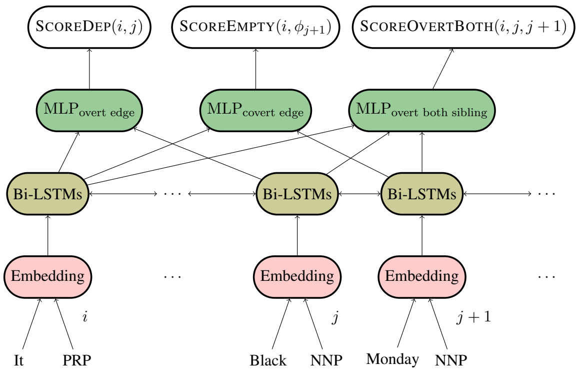
Let $\mathcal{Z}$ denote the set of all possible $\boldsymbol{z}$, and $\textsc{Part}(\boldsymbol{z})$ denote the factors in the dependency tree, including edges (and edge siblings in the second-order model). Then parsing with ECD can be defined as a search for the highest-scored $\boldsymbol{z}^*(s)$ in all compatible analyses, just like parsing without empty elements:

$$\begin{aligned}\boldsymbol{z}^*(s) &= \arg\max_{\boldsymbol{z}\in\mathcal{Z}(s)} \textsc{Score}(s,\boldsymbol{z}) \\ &= \arg\max_{\boldsymbol{z}\in\mathcal{Z}(s)} \sum_{p\in\textsc{Part}(\boldsymbol{z})} \textsc{ScorePart}(s,p)\end{aligned}$$

The graph-based parsing algorithms proposed by Zhang et al. are based on two properties: ECs can only serve as dependents and the number of successive ECs is limited. The latter trait makes it reasonable to treat consecutive ECs governed by the same head as one *word*. We also follow this set-up.

3.2 Scoring Based on BiLSTM

Kiperwasser and Goldberg (2016) proposed a simple yet effective architecture to implement neural dependency parsers. In particular, a BiLSTM is utilized as a powerful *feature extractor* to assist a dependency parser. Mainstream data-driven dependency parsers, including both transition- and graph-based ones, can apply useful word vectors provided by a BiLSTM to conduct the disambiguation. Following Kiperwasser and Goldberg (2016)'s experience on graph-based dependency parser, we implement such a parser to recover empty categories and to evaluate the impact of empty categories on surface parsing.



SCOREDEP(i,j)
SCOREEMPTY(i,ϕ_{j+1})
SCOREOVERTBOTH$(i,j,j+1)$
MLP$_{\text{overt edge}}$
MLP$_{\text{covert edge}}$
MLP$_{\text{overt both sibling}}$
Bi-LSTMs
Bi-LSTMs
Bi-LSTMs
Embedding
Embedding
Embedding
i
j
$j+1$
It
PRP
Black
NNP
Monday
NNP

Figure 3: The neural network structure when parsing sentence "It wasn't Black Monday." 5 MLPs is used for overt edges (i,j), covert edges (i,ϕ_j), overt-both siblings (i,j,k), covert-inside siblings (i,ϕ_j,k) and covert-outside siblings (i,j,ϕ_k) respectively, and 3 of them are shown in the graph.

Here we present details of the design of our parser. A vector is associated with each word or POS-tag to transform them into continuous and dense representations. We use pre-trained word embeddings and random initialized POS-tag embeddings.

The concatenation of the word embedding and the POS-tag embedding of each word in a specific sentence is used as the input of BiLSTMs to extract context related feature vectors r_i.

$$\boldsymbol{r}_{1:n} = \text{BiLSTM}(s; 1:n)$$

The context related feature vectors are fed into a non-linear transformation to perform scoring.

3.3 A First-Order Model

In the first-order model, we only consider the head and the dependent of the possible dependency arc. The two feature vectors of each word pair is scored with a non-linear transformation g as the first-order score. When words i and j are overt words, we define the score function in sentence s as follows,

$$\begin{aligned}&\textsc{ScoreDep}(s,i,j)\\&=\boldsymbol{W}_2\cdot g(\boldsymbol{W}_{1,1}\cdot\boldsymbol{r}_i+\boldsymbol{W}_{1,2}\cdot\boldsymbol{r}_j+\boldsymbol{b})\end{aligned}$$

$\boldsymbol{W}_2$, $\boldsymbol{W}_{1,1}$ and $\boldsymbol{W}_{1,2}$ denote the weight matrices in linear transformations. The score of covert edge from word i to word ϕ_j is calculated in a similar way with different parameters:

$$\begin{aligned}&\textsc{ScoreEmpty}(s,i,\phi_j)\\&=\boldsymbol{W}_2'\cdot g(\boldsymbol{W}_{1,1}'\cdot\boldsymbol{r}_i+\boldsymbol{W}_{1,2}'\cdot\boldsymbol{r}_j+\boldsymbol{b}')\end{aligned}$$

These non-linear transformations are also known as Multiple Layer Perceptrons(MLPs). The total score in our first-order model is defined as follows,

$$\begin{aligned}\textsc{Score}(s,\boldsymbol{z})=&\sum_{(i,j)\in\textsc{Dep}(\boldsymbol{z})}\textsc{ScoreDep}(s,i,j)\\&+\sum_{(i,\phi_j)\in\textsc{DepEmpty}(\boldsymbol{z})}\textsc{ScoreEmpty}(s,i,\phi_j)\end{aligned}$$

$\textsc{Dep}(\boldsymbol{z})$ and $\textsc{DepEmpty}(\boldsymbol{z})$ denote all overt and covert edges in $\boldsymbol{z}$ respectively. Because each overt and covert edge is selected independently of the others, the decoding process can be seen as calculating the maximum subtree from overt edges(we use Eisner Algorithm in our experiments) and appending each covert edge (i,ϕ_j) when $\textsc{ScoreEmpty}(i,\phi_j)>0$.

3.4 A Second-Order Model

In the second-order model, we also consider sibling arcs. We extend the neural network in section 3.3 to perform the second-order parsing. We calculate second-order scores(scores defined over sibling arcs) in a similar way. Each pair of overt sibling arcs, for example, (i,j) and (i,k) $(j<k)$, is denoted as (i,j,k) and scored with a non-linear transformation.

$$\begin{aligned}&\textsc{ScoreOvertBoth}(s,i,j,k)=\\&\boldsymbol{W}_2''\cdot g(\boldsymbol{W}_{1,1}''\cdot\boldsymbol{r}_i+\boldsymbol{W}_{1,2}''\cdot\boldsymbol{r}_j+\boldsymbol{W}_{1,3}''\cdot\boldsymbol{r}_k+\boldsymbol{b}'')\end{aligned}$$

Zhang et al. (2017b) defines two kinds of second-order scores to describe the interaction between concrete nodes and empty categories: the covert-inside sibling (i,ϕ_j,k) and covert-outside sibling (i,j,ϕ_k). Their scores can be calculated in a similar way with different parameters.

And finally, the score function over the whole syntactic analysis is defined as:

$$\begin{aligned}\textsc{Score}(s,\boldsymbol{z})=&\sum_{(i,j)\in\textsc{Dep}(\boldsymbol{z})}\textsc{ScoreDep}(s,i,j)\\&+\sum_{(i,\phi_j)\in\textsc{DepEmpty}(\boldsymbol{z})}\textsc{ScoreEmpty}(s,i,\phi_j)\\&+\sum_{(i,j,k)\in\textsc{OvertBoth}(\boldsymbol{z})}\textsc{ScoreOvertBoth}(s,i,j,k)\\&+\sum_{(i,\phi_j,k)\in\textsc{CovertIn}(\boldsymbol{z})}\textsc{ScoreCovertIn}(s,i,\phi_j,k)\\&+\sum_{(i,j,\phi_k)\in\textsc{CovertOut}(\boldsymbol{z})}\textsc{ScoreCovertOut}(s,i,j,\phi_k)\end{aligned}$$

$\textsc{OvertBoth}(\boldsymbol{z})$, $\textsc{CovertIn}(\boldsymbol{z})$ and $\textsc{CovertOut}(\boldsymbol{z})$ denotes overt-both, covert-inside and covert-outside siblings of $\boldsymbol{z}$ respectively. Totally 5 MLPs are used to calculate the 5 types of scores. The network structure is shown in Figure 3.

Labeled Parsing Similar to Kiperwasser and Goldberg (2016) and Zhang et al. (2017a), we use a two-step process to perform labeled parsing: conduct an unlabeled parsing and assign labels to each dependency edge. The labels are determined with the nonlinear classification. We use different nonlinear classifiers for edges between concrete nodes and empty categories.

Training In order to update graphs which have high model scores but are very wrong, we use a margin-based approach to compute loss from the gold tree T^* and the best prediction $\hat{T}$ under the current model.

We define the *loss* term as:

$$\max(0,\Delta(T^*,\hat{T})-\textsc{Score}(T^*)+\textsc{Score}(\hat{T}))$$

The margin objective Δ measures the similarity between the gold tree T^* and the prediction $\hat{T}$. Following Kiperwasser and Goldberg (2016)'s experience of loss augmented inference, we define Δ as the count of dependency edges in prediction results but not belonging to the gold tree.

3.5 Structure Regularization

ECD significantly increases the search space for parsing. This results in a side effect for practical parsing. Given the limit of available annotations for training, searching for more complex structures in a larger space is harmful to the generalization ability in structured prediction (Sun, 2014). To control structure-based overfitting, we train a normal dependency parser, namely parser for overt words only, and use its first- and second-order scores to augment the corresponding score functions in the joint parsing and ECD model. At the training phase, the two parsers are trained separately, while at the test phase, the scores are calculated by individual models and added for decoding.

4 Experiments

4.1 Experimental Setup

4.1.1 Data

We conduct experiments on a subset of Penn Chinese Treebank (CTB; Xue et al., 2005) 9.0. As a pro-drop language, the empty category is a very useful method for representing the (deep) syntactic analysis in Chinese language. Empty categories in CTB is divided into six classes: *pro*, *PRO*, *OP*, *T*, *RNR* and *, which were described in detail in Xue and Yang (2013); Wang et al. (2015). For comparability with the state-of-the-art, the division of training, development and testing data is coincident with the previous work (Xue and Yang, 2013).

Our experiments can be divided into two groups. The first group is conducted on the linear conditional random field (Linear-CRF) model and LSTM-CRF tagging model to evaluate gains from the introduction of neural structures. The second group is designed for the dependency-based in-parsing models.

4.1.2 Evaluation Metrics

We adopt two kinds of metrics for the evaluation of our experiments. The first one focuses on EC's position and type, in accordance with the *labeled empty elements* measure proposed by Cai et al. (2011), which can be implemented on all models in our experiments. The second one is stricter. Besides position and type, it also checks EC's head information. An EC is considered to be correct, only when all the three parts are the same as the corresponding gold standard. Thus only models involved in dependency structures can be evaluated according to the latter metric. Based on above measures of the two degrees, we evaluate our neural pre- and in-parsing models regarding each type of EC as well as overall performance.

Besides, to compare different models' abilities to capture non-local information, we design *Dependency Distance* to indicate the number of words from one EC to its head, not counting other ECs on the path. Taking the two ECs in Figure 1 as an example, $\emptyset_2$ has a *Dependency Distance* of 0 while $\emptyset_1$'s *Dependency Distance* is 3. We calculate labeled recall scores for enumerated *Dependency Distance*. A higher score means greater capability to catch and to represent long-distance details.

4.2 Results of Pre-Parsing Models

Table 3 shows overall performances of the two sequential models on development data. From the results, we can clearly see that the introduction of neural structure pushes up the scores exceptionally. The reason is that our LSTM-CRF model not only benefits from the linear weighted combination of local characteristics like ordinary CRF models, but also has the ability to integrate more contextual information, especially long-distance information. It confirms LSTM-based models' great superiority in sequence labeling problems.

Further more, we find that the difference among the four kinds of representations is not so obvious. The most performing one with LSTM-CRF model is *Interspace*, but the advantage is narrow. *Pre3* uses a larger window length to incorporate richer contextual tokens, but at the same time, the searching space for decoding grows larger. It explains that the performance drops slightly with increasing window length. In general, experiments with POS tags show higher scores as more syntactic clues are incorporated.

We compare LSTM-CRF with other state-of-the-art systems in Table 4[1]. We can see that a simple neural pre-parsing model outperforms state-of-the-art linear in-parsing systems. Analysis about results on different EC types as displayed in Table 5 shows that the sequence-oriented pre-parsing model is good at detecting *pro* compared with previous systems, which is used widely in pro-drop languages. Additionally, the model succeeds in detecting seven * EC tokens in evaluating process. * indicates trace left by passivization as well as raising, and is very rare in training data. Previous models usually cannot identify any *. This detail reflects that the LSTM-CRF model can make the most of limited training data compared with existing systems.

[1] Wang et al. (2015) reported an overall F-score of 71.7. But their result is based on the gold standard syntactic analysis.

	Linear CRF						LSTM-CRF					
	Without POS			With POS			Without POS			With POS		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Interspace	74.6	20.6	32.2	71.2	30.3	42.5	67.9	59.8	63.6	73.0	61.6	66.8
Pre2	72.4	30.1	42.5	72.8	32.4	44.8	71.1	58.3	64.1	74.8	57.4	65.0
Pre3	73.1	30.2	42.8	73.0	32.5	44.9	71.1	58.5	64.2	73.8	57.0	64.3
Prepost	70.9	32.9	45.0	74.4	30.3	43.1	71.0	57.6	63.6	72.9	58.6	65.0

Table 3: The overall performance of the two sequential models on development data.

	P	R	F_1
Pre-parsing	67.3	54.7	60.4
In-parsing	72.6	55.5	62.9
In-parsing*	70.9	54.1	61.4
(Xue and Yang, 2013)*	65.3	51.2	57.4
(Cai et al., 2011)	66.0	54.5	58.6

Table 4: The overall performance on test data. "*" indicates more stringent evaluation metrics.

EC Type	Total	Correct	P	R	F_1
pro	315	85	52.5	27.0	35.6
PRO	300	183	58.8	61.0	59.9
OP	575	338	73.0	58.8	65.1
T	580	355	73.3	61.2	66.7
RNR	34	30	62.5	88.2	73.2
*	19	7	46.7	36.8	41.2
Overall	1823	998	67.3	54.7	60.4

Table 5: Occurrences of different ECs in test data and detailed results of *Interspace* with POS information.

	First-order			Second-order		
Type	P	R	F_1	P	R	F_1
pro	52.5	16.8	25.5	54.4	19.7	28.9
PRO	59.7	47.3	52.8	60.6	58.0	59.3
OP	74.5	55.8	63.8	79.6	67.8	73.2
T	70.6	51.7	59.7	77.3	62.8	69.3
RNR	70.8	50.0	58.6	77.8	61.8	68.9
*	0.0	0.0	0.0	0.0	0.0	0.0
Overall	68.2	45.7	**54.7**	72.6	55.5	**62.9**
Evaluation with Head						
pro	50.5	16.2	24.5	52.6	19.1	28.0
PRO	58.4	46.3	51.7	57.8	55.3	56.6
OP	72.2	54.1	61.8	78.6	67.0	72.3
T	68.5	50.2	57.9	75.4	61.2	67.6
RNR	70.8	50.0	58.6	77.8	61.8	68.9
*	0.0	0.0	0.0	0.0	0.0	0.0
Overall	66.3	44.4	**53.2**	70.9	54.1	**61.4**

Table 6: The performances of the first- and second-order in-parsing models on test data.

4.3 Results of In-Parsing Models

Table 6 presents detailed results of the in-parsing models on test data. Compared with the state-of-the-art, the first-order model performs a little worse while the second-order model achieves a remarkable score. The first-order parsing model only constrains the dependencies of both the covert and overt tokens to make up a tree. Due to the loose scoring constraint of the first-order model, the prediction of empty nodes is affected little from the prediction of dependencies of overt words. The four bold numbers in the table intuitively elicits the conclusion that integrating an *empty* edge and its sibling *overt edges* is necessary to boost the performance. It makes sense because empty categories are highly related to syntactic analysis. When we conduct ECD and dependency parsing simultaneously, we can leverage more hierarchical contextual information. Comparing results regarding EC types, we can find that *OP* and *T* benefit most from the parsing information, the F_1 score increasing by about ten points, more markedly than other types.

4.4 Results on Dependency Parsing

Table 7 shows the impact of automatic detection of empty categories on parsing overt words. We compare the results of both steps in labeled parsing. We can clearly see that integrating empty elements into dependency parsing can improve the neural parsing accuracy of overt words. Besides, when jointing parsing models both without and with ECs together, we can push up the performance further. These results confirm the conclusion in Zhang et al. (2017b) that empty elements help parse the overt words. The main reason lies in that the existence of ECs provides extra structural information which can reduce approximation

	-EC	+EC	-+EC
Unlabeled	87.6	88.9	89.6
Labeled	84.6	85.9	86.6

Table 7: Accuracies of both unlabeled and labeled parsing on development data. -EC indicates parsing without empty categories. +EC indicates the second-order in-parsing models. -+EC indicates jointing parsing models both without and with ECs together.

errors in a structured prediction problem.

According to above analysis, we can draw a conclusion that ECD and syntactic parsing can promote each other mutually. That partially explains why in-parsing models can outperform pre-parsing models. Meanwhile, it provides a new approach to improving the dependency parsing quality in a unified framework.

4.5 Impact of Dependency Distance

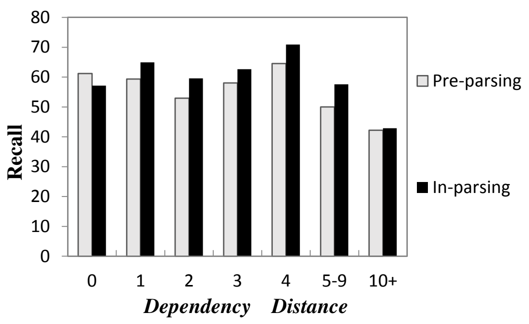


Figure 4: Recall scores of different models regarding *Dependency Distance*. ”Pre-parsing” and ”In-parsing” refer to the LSTM-CRF model and the dependency-based in-parsing model respectively.

We compare pre- and in-parsing models regarding *Dependency Distance*. The former refers to the LSTM-CRF model while the latter means the dependency-based in-parsing model. Figure 4 shows the results. The abscissa value ranges from 0 to 26, with the longest dependency arc spanning 26 non-EC word tokens. We can see that long-distance disambiguation is a challenge shared by both models. When the value of *Dependency Distance* exceeds four, the recall score drops gradually with abscissa increasing. Based on the comparison of two sets of data, we can find that in-parsing model performs better on ECs which are close to their heads. However, as for ECs which are far apart from their heads, two models have performed almost exactly alike. It demonstrates that LSTM structure is capable of capturing non-local features, making up for no exposure to parsing information.

4.6 Challenges

On the whole, the most challenging EC type is *pro*. We assume that it is because that pro-drop situations are complicated and diverse in Chinese language. According to Chinese linguistic theory, pronouns are dropped as a result of continuing from the preceding discourse or just idiomatic rules, such as the ellipsis of the first person pronoun “我/I” in the subject position. To fill this gap, we may need to extract more deep structural features.

Another difficulty is the detection of consecutive ECs. In the result of our experiments, in-parsing dependency-based model can only accurately detect up to two consecutive ECs. Too many empty elements in the same sentence conceal too much syntactic information, making it hard to disclose the original structure.

Moreover, in view of the fact that ECs play an essential role in syntactic analysis, the current detection accuracy of ECs is far from enough. We still have a long way to go.

5 Related Work

The detection of empty categories is an essential ground for many downstream tasks. For example, Chung and Gildea (2010) has proved that automatic empty category detection has a positive impact on machine translation. Zhang et al. (2017b) shows that ECD can benefit linear syntactic parsing of overt words. To accurately distinguish empty elements in sentences, there are generally three approaches. The first method is to build pre-processors before syntactic parsing. Dienes and Dubey (2003) proposed a shallow *trace tagger* which can detect discontinuities. And it can be combined with unlexicalized PCFG parsers to implement deep syntactic processing. Due to the lack of phrase structure information, it did not acquire remarkable results. The second method is to integrate ECD into parsing, as shown in Schmid (2006) and Cai et al. (2011), which involved empty elements in the process of generating parse trees. Another in-parsing system is proposed in Zhang et al. (2017b). Zhang et al. (2017b) designed algorithms to produce dependency trees in which empty elements are allowed. To add empty elements into dependency structures, they extend Eisner's first-order DP algorithm for parsing to second- and third-order algorithms. The last approach to recognizing empty elements is post-parsing methods. Johnson (2002) proposed a simple pattern-matching algorithm for recovering empty nodes in phrase structure trees while Campbell (2004) presented a rule-based algorithm. Xue and Yang (2013) conducted ECD based on dependency trees. Their methods can leverage richer syntactic information, thus have achieved more satisfying scores.

As neural networks have been demonstrated to have a great ability to capture complex features, it has been applied in multiple NLP tasks (Bengio and Schwenk, 2006; Collobert et al., 2011). Neural methods have also explored in distinguishing empty elements. For example, Wang et al. (2015) described a novel ECD solution using distributed word representations and achieved the state-of-the-art performance. Based on above work, we explore neural pre- and in-parsing models for ECD.

6 Conclusion

Neural networks have played a big role in multiple NLP tasks recently owing to its nonlinear mapping ability and the avoidance of human-engineered features. It should be a well-justified solution to identify empty categories as well as to integrate empty categories into syntactic analysis. In this paper, we study neural models to detect empty categories. We observe three facts: (1) BiLSTM significantly advances the pre-parsing ECD. (2) Automatic ECD improves the neural dependency parsing quality for overt words. (3) Even with a BiLSTM, syntactic information can enhance the detection further. Experiments on Chinese language show that our neural model for ECD exceptionally boosts the state-of-the-art detection accuracy.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61772036, 61331011) and the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Weiwei Sun is the corresponding author.

References

Yoshua Bengio and Holger Schwenk. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*. Springer, page 137186.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 212–216. http://www.aclweb.org/anthology/P11-2037.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 645–652. https://doi.org/10.3115/1218955.1219037.

A. Carnie. 2012. *Syntax: A Generative Introduction 3rd Edition and The Syntax Workbook Set*. Introducing Linguistics. Wiley. https://books.google.com/books?id=jhGKMAEACAAJ.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris Publications, Dordecht.

Tagyoung Chung and Daniel Gildea. 2010. Effects of empty categories on machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 636–645. http://www.aclweb.org/anthology/D10-1062.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(1):2493–2537.

Pétr Dienes and Amit Dubey. 2003. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sapporo, Japan, pages 431–438. https://doi.org/10.3115/1075096.1075151.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 136–143. https://doi.org/10.3115/1073083.1073107.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized pcfgs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 177–184. https://doi.org/10.3115/1220175.1220198.

Wolfgang Seeker, Richárd Farkas, Bernd Bohnet, Helmut Schmid, and Jonas Kuhn. 2012. Data-driven dependency parsing with empty heads. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1081–1090. http://www.aclweb.org/anthology/C12-2105.

Xu Sun. 2014. Structure regularization for structured prediction. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2402–2410. http://papers.nips.cc/paper/5563-structure-regularization-for-structured-prediction.pdf.

Xun Wang, Katsuhito Sudoh, and Masaaki Nagata. 2015. Empty category detection with joint context-label embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 263–271. http://www.aclweb.org/anthology/N15-1030.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11:207–238. https://doi.org/10.1017/S135132490400364X.

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistics Theories*.

Nianwen Xue and Yaqin Yang. 2013. Dependency-based empty category detection via phrase structure trees. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1051–1060. http://www.aclweb.org/anthology/N13-1125.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017a. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 665–676. http://www.aclweb.org/anthology/E17-1063.

Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2017b. The covert helps parse the overt. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 343–353. http://aclweb.org/anthology/K17-1035.

Composing Finite State Transducers on GPUs

Arturo Argueta and **David Chiang**
Department of Computer Science and Engineering
University of Notre Dame
{aargueta,dchiang}@nd.edu

Abstract

Weighted finite state transducers (FSTs) are frequently used in language processing to handle tasks such as part-of-speech tagging and speech recognition. There has been previous work using multiple CPU cores to accelerate finite state algorithms, but limited attention has been given to parallel graphics processing unit (GPU) implementations. In this paper, we introduce the first (to our knowledge) GPU implementation of the FST composition operation, and we also discuss the optimizations used to achieve the best performance on this architecture. We show that our approach obtains speedups of up to 6× over our serial implementation and 4.5× over OpenFST.

1 Introduction

Finite-state transducers (FSTs) and their algorithms (Mohri, 2009) are widely used in speech and language processing for problems such as grapheme-to-phoneme conversion, morphological analysis, part-of-speech tagging, chunking, named entity recognition, and others (Mohri et al., 2002; Mohri, 1997). Hidden Markov models (Baum et al., 1970), conditional random fields (Lafferty et al., 2001) and connectionist temporal classification (Graves et al., 2006) can also be thought of as finite-state transducers.

Composition is one of the most important operations on FSTs, because it allows complex FSTs to be built up from many simpler building blocks, but it is also one of the most expensive. Much work has been done on speeding up composition on a single CPU processor (Pereira and Riley, 1997; Hori and Nakamura, 2005; Dixon et al., 2007; Allauzen and Mohri, 2008; Allauzen et al., 2009; Ladner and Fischer, 1980; Cheng et al., 2007). Methods such as on-the-fly composition, shared data structures, and composition filters have been used to improve time and space efficiency.

There has also been some successful work on speeding up composition using multiple CPU cores (Jurish and Würzner, 2013; Mytkowicz et al., 2014; Jung et al., 2017). This is a challenge because many of the algorithms used in NLP do not parallelize in a straightforward way and previous work using multi-core implementations do not handle the reduction of identical edges generated during the composition. The problem becomes more acute on the graphics processing units (GPUs) architecture, which have thousands of cores but limited memory available. Another problem with the composition algorithm is that techniques used on previous work (such as composition filters and methods to expand or gather transitions using dictionaries or hash tables) do not translate well to the GPU architecture given the hardware limitations and communication overheads.

In this paper, we parallelize the FST composition task across multiple GPU cores. To our knowledge, this is the first successful attempt to do so. Our approach treats the composed FST as a sparse graph and uses some techniques from the work of Merrill et al. (2012); Jung et al. (2017) to explore the graph and generate the composed edges during the search. We obtain a speedup of 4.5× against OpenFST's implementation and 6× against our own serial implementation.

2 Finite State Transducers

In this section, we introduce the notation that will be used throughout the paper for the composition task. A weighted FST is a tuple $M = (Q, \Sigma, \Gamma, s, F, \delta)$, where

- Q is a finite set of states.
- Σ is a finite input alphabet.
- Γ is a finite output alphabet.
- $s \in Q$ is the start state.
- $F \subseteq Q$ are the accept states.
- $\delta : Q \times \Sigma \times \Gamma \times Q \to \mathbb{R}$ is the transition function. If $\delta(q, a, b, r) = p$, we write

$$q \xrightarrow{a:b/p} r.$$

Note that we don't currently allow epsilon transitions; this would require implementation of composition filters (Allauzen et al., 2009), which is not a trivial task on the GPU architecture given the data structures and memory needed. Hence, we leave this for future work.

For the composition task, we are given two weighted FSTs:

$$M_1 = (Q_1, \Sigma, \Gamma, s_1, F_1, \delta_1)$$
$$M_2 = (Q_2, \Gamma, \Delta, s_2, F_2, \delta_2).$$

Call Γ, the alphabet shared between the two transducers, the *inner* alphabet, and let $m = |\Gamma|$. Call Σ and Δ, the input alphabet of M_1 and the output alphabet of M_2, the *outer* alphabets.

The composition of M_1 and M_2 is the weighted FST

$$M_1 \circ M_2 = (Q_1 \times Q_2, \Sigma, \Delta, s_1 s_2, F_1 \times F_2, \delta)$$

where

$$\delta(q_1 q_2, a, b, r_1 r_2) = \sum_{b \in \Gamma} \delta_1(q_1, a, c, r_1) \cdot \delta_2(q_2, c, b, r_2).$$

That is, for each pair of transitions with the same inner symbol,

$$q_1 \xrightarrow{a:b/p_1} r_1$$
$$q_2 \xrightarrow{b:c/p_2} r_2,$$

the composed transducer has a transition

$$q_1 q_2 \xrightarrow{a:c/p_1 p_2} r_1 r_2.$$

Transitions with the same source, target, input, and output symbols are merged, adding their weights.

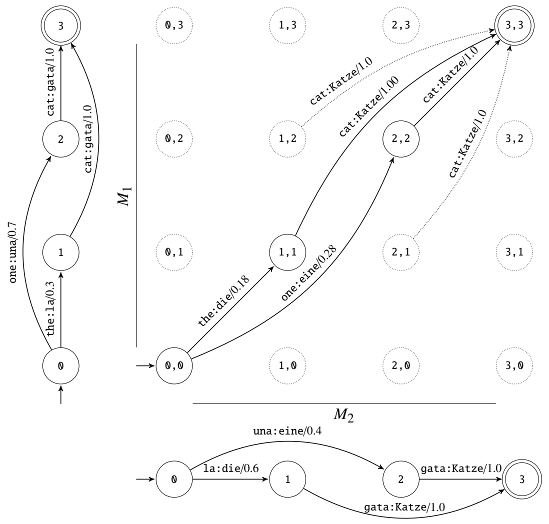


Figure 1: Example composition of two finite state transducers: M_1 translates English to Spanish, M_2 translates Spanish to German. The center of the image contains the composition of the two input transducers. This new transducer translates English to German. The dotted states and transitions are those that cannot be reached from the start state.

3 Method

In this section, we describe our composition method and its implementation.

3.1 Motivation

If implemented naïvely, the above operation is inefficient. Even if M_1 and M_2 are trim (have no states that are unreachable from the start state or cannot reach the accept state), their composition may have many unreachable states. Figure 1 shows a clear example where the transducers used for composition are trim, yet several states (drawn as dotted circles) on the output transducers cannot be reached from the start state. The example also shows composed transitions that originate from unreachable states. As a result, a large amount of time and memory may be spent creating states and composing transitions that will not be reachable nor needed in practice. One solution to avoid the problem is to compose only the edges and states that are reachable from the start state on the output transducer to avoid unnecessary computations and reduce the overall memory footprint.

We expect this problem to be more serious when the FSTs to be composed are sparse, that is, when there are many pairs of states without a transition between them. And we expect that FSTs used in

	Data	Transitions	Nonzero	% Nonzero
1k	de-en	21.7M	16.5k	0.076%
	en-de	12.3M	15.4k	0.125%
	en-es	12.5M	15.5k	0.124%
	en-it	13.1M	16.3k	0.124%
10k	de-en	394M	114k	0.029%
	en-de	138M	93.9k	0.067%
	en-es	135M	93.3k	0.068%
	en-it	143M	97.2k	0.067%
15k	de-en	634M	158k	0.025%
	en-de	201M	126k	0.062%
	en-es	195M	125k	0.064%
	en-it	209M	131k	0.062%

Table 1: FSTs used in our experiments. Key: Data = language pair used to generate the transducers; Transitions = maximum possible number of transitions; Nonzero = number of transitions with nonzero weight; % Nonzero = percent of possible transitions with nonzero weight. The left column indicates the number of parallel sentences used to generate the transducers used for testing.

natural language processing, whether they are constructed by hand or induced from data, will often be sparse.

For example, below (Section 4.1), we will describe some FSTs induced from parallel text that we will use in our experiments. We measured the sparsity of these FSTs, shown in Table 1. These FSTs contain very few non-zero connections between their states, suggesting that the output of the composition will have a large number of unreachable states and transitions. The percentage of non-zero transitions found in the transducers used for testing decreases as the transducer gets larger. Therefore, when composing FSTs, we want to construct only reachable states, using a traversal scheme similar to breadth-first search to avoid the storage and computation of irrelevant elements.

3.2 Serial composition

We first present a serial composition algorithm (Algorithm 2). This algorithm performs a breadth-first search (BFS) of the composed FST beginning from the start state, so as to avoid creating inaccessible states. As is standard, the BFS uses two data structures, a frontier queue (A) and a visited set (Q), which is always a superset of A. For each state q_1q_2 popped from A, the algorithm composes all transitions from q_1 with all transitions from q_2 that have the same inner symbol. The composed edges are added to the final transducer, and the corresponding target states q_1q_2 are pushed into A for future expansion. The search finishes once A runs out of states to expand.

Algorithm 1 Serial composition algorithm.

```
Input Transducers: $M_1 = (Q_1, \Sigma, \Gamma, s_1, F_1, \delta_1)$
                   $M_2 = (Q_2, \Gamma, \Delta, s_2, F_2, \delta_2)$
Output Transducer: $M_1 \circ M_2$
 1: $A \leftarrow \{s_1 s_2\}$                ▷ Queue of states to process
 2: $Q \leftarrow \{s_1 s_2\}$                ▷ Set of states created so far
 3: $\delta \leftarrow \emptyset$                       ▷ Transition function
 4: while $|A| > 0$ do
 5:     $q_1 q_2 \leftarrow \text{pop}(A)$
 6:     for $q_1 \xrightarrow{a:b/p_1} r_1 \in \delta_1$ do
 7:         for $q_2 \xrightarrow{b:c/p_2} r_2 \in \delta_2$ do
 8:             $\delta(q_1 q_2, a, c, r_1 r_2) \mathrel{+}= p_1 p_2$
 9:             if $r_1 r_2 \notin Q$ then
10:                 $Q \leftarrow Q \cup \{r_1 r_2\}$
11:                 push$(A, r_1 r_2)$
12: return $(Q, \Sigma, \Delta, s_1 s_2, F_1 \times F_2, \delta)$
```

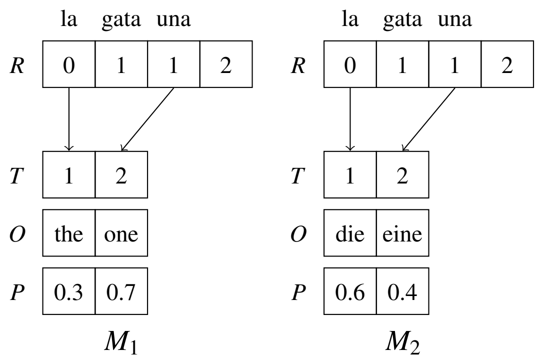


Figure 2: Example CSR-like representation of state 0 for transducers M_1 and M_2 from Figure 1.

3.3 Transducer representation

Our GPU implementation stores FST transition functions in a format similar to compressed sparse row (CSR) format, as introduced by our previous work Argueta and Chiang (2017). For the composition task we use a slightly different representation. An example of the adaptation is shown in Figure 2. The transition function δ for the result is stored in a similar fashion. The storage method is defined as follows:

- z is the number of transitions with nonzero weight.
- R is an array of length $|Q|m + 1$ containing offsets into the arrays T, O, and P. If the states are numbered $0, \dots, |Q| - 1$ and the inner symbols are numbered $0, \dots m - 1$, then state q's outgoing transitions on inner symbol b can be found starting at the offset stored in $R[qm + b]$. The last offset index, $R[|Q|m + 1]$, must equal z.
- $T[k]$ is the target state of the kth transition.
- $O[k]$ is the outer symbol of the kth transition.
- $P[k]$ is the weight of the kth transition.

Similarly to several toolkits (such as OpenFST), we require the edges in T, O, P to be sorted by their inner symbols before executing the algorithm, which allows faster indexing and simpler parallelization.

3.4 Parallel composition

Our parallel composition implementation has the same overall structure as the serial algorithm, and is shown in Algorithm 2. The two transducers to be composed are stored on the GPU in global memory, in the format described in Section 3.3. Both transducers are sorted according to their inner symbol on the CPU and copied to the device. The memory requirements for a large transducer complicates the storage of the result on the GPU global memory. If the memory of states and edges generated by both inputs does not fit on the GPU, then the composition cannot be computed using only device memory. The execution time will also be affected if the result lives on the device and there is a limited amount of memory available for temporary variables created during the execution. Therefore, the output transducer must be stored on the host using page-locked memory, with the edge transitions unsorted.

Page-locked, or pinned, memory is memory that will not get paged out by the operating system. Since this memory cannot be paged out, the amount of RAM available to other applications will be reduced. This enables the GPU to access the host memory quickly. Pinned memory provides better transfer speeds since the GPU creates different mappings to speed up `cudaMemcpy` operations on host memory. Allocating pinned memory consumes more time than a regular `malloc`,

Algorithm 2 Parallel composition algorithm.

```
Input Transducers: $M_1 = (Q_1, \Sigma, \Gamma, s_1, F_1, \delta_1)$
                   $M_2 = (Q_2, \Gamma, \Delta, s_2, F_2, \delta_2)$
Output Transducer: $M_1 \circ M_2$
 1: $A \leftarrow \{s_1 s_2\}$                          ▷ Queue of states to process
 2: $Q \leftarrow \{s_1 s_2\}$                          ▷ Set of states visited
 3: $\delta \leftarrow []$                              ▷ List of transitions
 4: while $|A| > 0$ do
 5:     $q_1 q_2 \leftarrow \text{pop}(A)$
 6:     $\delta_d \leftarrow []$
 7:     $A_d \leftarrow \emptyset$
 8:     $H \leftarrow \emptyset$
 9:     $red \leftarrow$ false
10:     parfor $b \in \Gamma$ do                        ▷ kernels
11:         parfor $q_1 \xrightarrow{a:b/p_1} r_1$ do    ▷ threads
12:             parfor $q_2 \xrightarrow{b:c/p_2} r_2$ do    ▷ threads
13:                 append $q_1 q_2 \xrightarrow{a:c/p_1 p_2} r_1 r_2$ to $\delta_d$
14:                 if $h(a, c, r_1 r_2) \in H$ then
15:                     $red \leftarrow$ true
16:                 else
17:                     add $h(a, c, r_1 r_2)$ to $H$
18:                 if $r_1 r_2 \notin Q$ then
19:                     $A_d \leftarrow A_d \cup \{r_1 r_2\}$
20:                     $Q \leftarrow Q \cup \{r_1 r_2\}$
21:     concatenate $\delta_d$ to $\delta$
22:     for $q \in A_d$ do push$(A, q)$
23:     if $red$ then
24:         sort $\delta[q_1 q_2]$
25:         reduce $\delta[q_1 q_2]$
26: return $(Q, \Sigma, \Delta, s_1 s_2, F_1 \times F_2, \delta)$
```

therefore it should be done sporadically. In this work, pinned memory is allocated only once at start time and released once the composition has been completed. Using page-locked memory on the host side as well as pre-allocating memory on the device decreases the time to both copy the results back from the GPU, and the time to reuse device structures used on different kernel methods.

Generating transitions The frontier queue A is stored on the host. For each state $q_1 q_2$ popped from A, we need to compose all outgoing transitions of q_1 and q_2 obtained from M_1 and M_2 respectively. Following previous work (Merrill et al., 2012; Jurish and Würzner, 2013), we create these in parallel, using the three **parfor** loops in lines 10–12. Although these three loops are written the same way in pseudocode for simplicity, in actuality they use two different parallelization schemes in the actual implementation of the algorithm.

The outer loop launches a CUDA kernel for each inner symbol $b \in \Gamma$. For example, to compose the start states in Figure 1, three kernels will be launched (one for *la*, *gata*, and *una*). Each of these kernels composes all outgoing transitions of q_1 with output b with all outgoing transitions of q_2 with input b. Each of these kernels is executed in a unique stream, so that a higher parallelization can be achieved. Streams are used in CUDA programming to minimize the number of idle cores during execution. A stream is a group of commands that execute in-order on the GPU. What makes streams useful in CUDA is the ability to execute several asynchronously. If more than one kernel can run asynchronously without any interdependence, the assignment of kernel calls to different streams will allow a higher speedup by minimizing the amount of idling cores during execution. All kernel calls using streams are asynchronous to the host making synchronization between several different streams necessary if there exist data dependencies between different parts of the execution pipeline. Asynchronous memory transactions can also benefit from streams, if these operations do not have any data dependencies.

We choose a kernel block size of 32 for the kernel calls since this is the amount of threads that run in parallel on all GPU streaming multiprocessors at any given time. If the number of threads required to compose a tuple of states is not divisible by 32, the number of threads is rounded up to the closest multiple. When several input tuples generate less than 32 edges, multiple cores will remain idle during execution. Our approach obtains better speedups when the input transducers are able to generate a large amount of edges for each symbol b and each state tuple on the result. In general, the kernels may take widely varying lengths of time based on the amount of composed edges; using streams enables the scheduler to minimize the number of idle cores.

The two inner loops represent the threads of the kernel; each composes a pair of transitions sharing an inner symbol b. Because these transitions are stored contiguously (Figure 2), the reads can be coalesced, meaning that the memory reads from the parallel threads can be combined into one transaction for greater efficiency. Figure 2 shows how the edges for a transducer are stored in global memory to achieve coalesced memory operations each time the edges of a symbol b associated with a state tuple q_1,q_2 need to be composed.

Figure 2 shows how the edges leaving the start state tuple for transducers M_1 and M_2 are stored. As mentioned above, three kernels will be launched to compose the transitions leaving the start states, but only two will be executed (because there are no transitions on *gata* for both start states). For $R[\texttt{la}]$ on machine M_1, only one edge can output `la` given $R[\texttt{la} + 1] - R[\texttt{la}] = 1$, and machine M_2 has one edge that reads *la* given $R[\texttt{la} + 1] - R[\texttt{la}] = 1$. For this example, $R[\texttt{la}]$ points to index 0 on T, O, P for both states. This means that only one edge will be generated from the composition $(0, 0 \xrightarrow{\texttt{the:die/0.18}} 1, 1)$. For symbol `gata`, no edges can be composed given $R[\texttt{gata} + 1] - R[\texttt{gata}] = 0$ on both machines, meaning that no edges read or output that symbol. Finally, for $R[\texttt{una}]$ on machine M_1 and M_2, one edge can be generated $(0, 0 \xrightarrow{\texttt{one:eine/0.28}} 2, 2)$ given the offsets in R for both input FSTs. If n_1 edges can be composed for a symbol b on one machine and n_2 from the other one, the kernel will generate $n_1 n_2$ edges.

The composed transitions are first appended to a pre-allocated buffer δ_d on the GPU. After processing the valid compositions leaving $q_1 q_2$, all the transitions added in δ_d are appended in bulk to δ on the host.

Updating frontier and visited set Each destination state $r_1 r_2$, if previously unvisited, needs to be added to both A and Q. Instead of adding it directly to A (which is stored on the host), we add it to a buffer A_d stored on the device to minimize the communication overhead between the host and the device. After processing $q_1 q_2$ and synchronizing all streams, A_d is appended in bulk to A using a single `cudaMemcpy` operation.

The visited set Q is stored on the GPU device as a lookup table of length $|Q_1||Q_2|$. Merrill et al. (2012) perform BFS using two stages to obtain the states and edges needed for future expansion. Similarly, our method performs the edge expansion using two steps by using the lookup table Q. The first step of the kernel updates Q and all visited states that need to be added to A_d. The second step appends all the composed edges to δ in parallel. Since several threads check the table in parallel, an atomic operation (`atomicOr`) is used to check and update each value on the table in a consistent fashion. Q also functions as a map to convert the state tuple q_1q_2 into a single integer. Each time a tuple is not in Q, the structure gets updated with the total number of states generated plus one for a specific pair of states.

Reduction Composed edges with the same source, target, input, and output labels must be merged, summing their probabilities. This is done in lines 23–25, which first sort the transitions and then merge and sum them. To do this, we pack the transitions into an array of keys and an array of values. Each key is a tuple (a, c, r_1r_2) packed into a 64-bit integer. We then use the sort-by-key and reduce-by-key operations provided by the Thrust library. The mapping of tuples to integers is required for the sort operation since the comparisons required for the sorting can be made faster than using custom data structures with a custom comparison operator. [1]

Because the above reduction step is rather expensive, lines 14–17 use a heuristic to avoid it if possible. H is a set of transitions represented as a hash table without collision resolution, so that lookups can yield false positives. If *red* is false, then there were no collisions, so the reduction step can be skipped. The hash function is simply $h(a, c, r_1r_2) = a + c|\Sigma|$. In more detail, H actually maps from hashes to integers. Clearing H (line 8) actually just increments a counter i; storing a hash k is implemented as $H[k] \leftarrow i$, so we can test whether k is a member by testing whether $H[k] = i$. An atomic operation (`atomicExch`) is used to consistently check H since several threads update this variable asynchronously.

4 Experiments

We tested the performance of our implementation by constructing several FSTs of varying sizes and comparing our implementation against other baselines.

4.1 Setup

In our previous work (Argueta and Chiang, 2017), we created transducers for a toy translation task. We trained a bigram language model (as in Figure 3a) and a one-state translation model (as in Figure 3) with probabilities estimated from GIZA++ Viterbi word alignments. Both were trained on the Europarl corpus. We then precomposed them using the Carmel toolkit (Graehl, 1997).

We used the resulting FSTs to test our parallel composition algorithm, composing a German-to-English transducer with a English-to-t transducer to translate German to language t, where t is German, Spanish, or Italian.

Our experiments were tested using two different architectures. The serial code was measured using a 16-core Intel Xeon CPU E5-2650 v2, and the parallel implementation was executed on a system with a GeForce GTX 1080 Ti GPU connected to a 24-core Intel Xeon E5-2650 v4 processor.

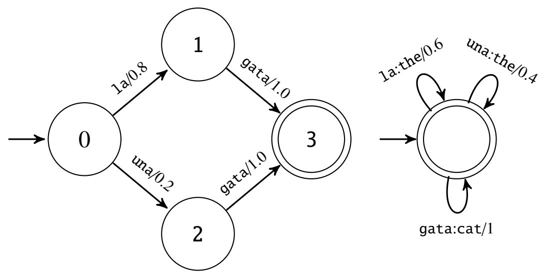
4.2 Baselines

In this work, OpenFST (Allauzen et al., 2007) and our serial implementation (Algorithm 1) were used as a baseline for comparison. OpenFST is a toolkit developed by Google as a successor of the AT&T Finite State Machine library. For consistency, all implementations use the OpenFST text file format to read and process the transducers.

4.3 Results

OpenFST's composition operation can potentially create multiple transitions (that is, two or more transitions with the same source state, destination state, input label, and output label); a separate function (`ArcSumMapper`) must be applied to merge multiple transitions and sum their weights. Previous work also requires an additional step if identical edges need to be merged. For this reason,

[1]https://thrust.github.io/



(a) (b)

Figure 3: The transducers used for testing were obtained by pre-composing: (a) a language model and (b) a translation model. These two composed together form a transducer that can translate an input sequence from one language (here, Spanish) into another language (here, English).

			Training size (lines)					
			1000		10000		15000	
Method	Hardware	Target	Time	Ratio	Time	Ratio	Time	Ratio
OpenFST	Xeon E5	DE	0.52	0.78	69.51	3.56	157.16	4.38
our serial	Xeon E5	DE	**0.21**	**0.31**	28.47	1.45	72.33	2.02
our parallel	GeForce GTX 1080	DE	0.67	1.00	**19.54**	**1.00**	**35.89**	**1.00**
OpenFST	Xeon E5	ES	0.46	0.72	55.62	2.97	137.16	4.07
our serial	Xeon E5	ES	**0.19**	**0.30**	23.30	1.24	62.42	1.85
our parallel	GeForce GTX 1080	ES	0.64	1.00	**18.72**	**1.00**	**33.71**	**1.00**
OpenFST	Xeon E5	IT	0.54	0.79	60.66	3.05	136.06	3.91
our serial	Xeon E5	IT	**0.21**	**0.31**	25.58	1.28	119.84	3.45
our parallel	GeForce GTX 1080	IT	0.68	1.00	**19.88**	**1.00**	**34.76**	**1.00**

Table 2: This table shows how the total running time of our GPU implementation compares against all other methods. Times (in seconds) are for composing two transducers using English as the shared input/output vocabulary and German as the source language of the first transducer (de-en,en-*). Ratios are relative to our parallel algorithm on the GeForce GTX 1080 Ti.

we compare our implementation against OpenFST both with and without the reduction of transitions with an identical source,target,input, and output. We analyzed the time to compose all possible edges without performing any reductions (Algorithm 1, line 8). The second setup analyzes the time it takes to compute the composition and the arc summing of identical edges generated during the process.

Table 2 shows the performance of the parallel implementation and the baselines without reducing identical edges. For the smallest transducers, our parallel implementation is slower than the baselines (0.72× compared to OpenFST and 0.30× compared to our serial version). With larger transducers, the speedups increase up to 4.38× against OpenFST and 2.02× against our serial implementation. Larger speedups are obtained for larger transducers because the GPU can utilize the streaming multiprocessors more fully. On the other hand, the overhead created by CUDA calls, device synchronization, and memory transfers between the host CPU and the device might be too expensive when the inputs are too small.

Table 3 shows the performance of all implementations with the reduction operation. Again, for the smallest transducers we can see a similar behavior, our parallel implementation is slower (0.30× against OpenFST and 0.39× against our serial version). Speedups improve with the larger transducers, eventually achieving a 4.52× speedup over OpenFST and a 6.26× speedup over our serial implementation of the composition algorithm.

4.4 Discussion

One comparison missing above is a comparison against a multicore processor. We attempted to compare against a parallel implementation using OpenMP on a single 16-core processor, but it did not yield any meaningful speedup, and even slowdowns of up to 10%. We think the reason for this is that because the BFS-like traversal of the FST makes it impractical to process states in parallel, the best strategy is to process and compose transitions in parallel. This very fine-grained parallelism does not seem suitable for OpenMP, as the overhead due to thread initialization and synchronization is higher than the time to execute the parallel sections of the code where the actual composition is calculated. According to our measurements, the average time to compose two transitions is 7.4 nanoseconds, while the average time to create an OpenMP thread is 10.2 nanoseconds. By contrast, the overhead for creating a CUDA thread seems to be around 0.4 nanoseconds. While a different parallelization strategy may exist for multicore architectures, at present, our finding is that GPUs, or other architectures with a low cost to create and destroy threads, are much more suitable for the fine grained operations used for the composition task.

			Training size (lines)					
			1000		10000		15000	
Method	Hardware	Target	Time	Ratio	Time	Ratio	Time	Ratio
OpenFST	Xeon E5	DE	**0.87**	**0.41**	148.11	3.19	374.72	4.52
our serial	Xeon E5	DE	0.96	0.45	213.27	4.59	518.97	6.26
our parallel	GeForce GTX 1080	DE	2.11	1.00	**47.70**	**1.00**	**82.88**	**1.00**
OpenFST	Xeon E5	ES	**0.60**	**0.30**	116.45	2.66	279.85	3.57
our serial	Xeon E5	ES	0.77	0.39	202.15	4.61	390.29	4.97
our parallel	GeForce GTX 1080	ES	2.00	1.00	**45.30**	**1.00**	**78.38**	**1.00**
OpenFST	Xeon E5	IT	**0.76**	**0.36**	130.61	2.87	309.28	3.79
our serial	Xeon E5	IT	1.06	0.50	158.57	3.48	427.51	5.24
our parallel	GeForce GTX 1080	IT	2.12	1.00	**47.04**	**1.00**	**81.54**	**1.00**

Table 3: This table shows how the total running time of our GPU implementation compares against all other methods. Times (in seconds) are for composing two transducers and performing edge reduction using English as the shared input/output vocabulary and German as the source language of the first transducer (de-en,en-*). Ratios are relative to our parallel algorithm on the GeForce GTX 1080 Ti.

5 Future Work

For future work, other potential bottlenecks could be addressed. The largest bottleneck is the queue used on the host to keep track of the edges to expand on the GPU. Using a similar data structure on the GPU to keep track of the states to expand would yield higher speedups. The only challenge of using such a data structure is the memory consumption on the GPU. If the two input transducers contain a large number of states and transitions, the amount of memory needed to track all the states and edges generated will grow significantly. Previous work (Harish and Narayanan, 2007) has shown that state queues on the GPU cause a large memory overhead. Therefore, if state expansion is moved to the GPU, the structures used to keep track of the states must be compressed or occupy the least amount of memory possible on the device in order to allocate all structures required on the device. The queue will also require a mechanism to avoid inserting duplicate tuples into the queue.

For the reduction step, speedups can be achieved if the sort and reduce operations can be merged with the edge expansion part of the method. The challenge of merging identical edges during expansion is the auxiliary memory that will be required to store and index intermediate probabilities. It can be doable if the transducers used for the composition are small. In that case, the reduce operation might not yield significant speedups given the fact that the overhead to compose small transducers is too high when using a GPU architecture.

6 Conclusion

This is the first work, to our knowledge, to deliver a parallel GPU implementation of the FST composition algorithm. We were able to obtain speedups of up to 4.5× over a serial OpenFST baseline and 6× over the serial implementation of our method. This parallel method considers several factors, such as host to device communication using page-locked memory, storage formats on the device, thread configuration, duplicate edge detection, and duplicate edge reduction. Our implementation is available as open-source software.[2]

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This research was supported in part by an Amazon Academic Research Award and a hardware grant from NVIDIA.

References

Cyril Allauzen and Mehryar Mohri. 2008. 3-way composition of weighted finite-state transducers. In *Implementation and Applications of Automata*, pages 262–273. Springer.

[2]`https://bitbucket.org/aargueta2/parallel_composition`

Cyril Allauzen, Michael Riley, and Johan Schalkwyk. 2009. A generalized composition algorithm for weighted finite-state transducers. In *Proceedings of the Conference of the International Speech Communication Association (ISCA)*, pages 1203–1206.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer.

Arturo Argueta and David Chiang. 2017. Decoding with finite-state transducers on GPUs. In *Proceedings of EACL*, pages 1044–1052.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Octavian Cheng, John Dines, and Mathew Magimai Doss. 2007. A generalized dynamic composition algorithm of weighted finite state transducers for large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–345. IEEE.

Paul R. Dixon, Diamantino A Caseiro, Tasuku Oonishi, and Sadaoki Furui. 2007. The Titech large vocabulary WFST speech recognition system. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 443–448.

Jonathan Graehl. 1997. Carmel finite-state toolkit. *ISI/USC*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*, pages 369–376.

Pawan Harish and P. J. Narayanan. 2007. Accelerating large graph algorithms on the GPU using CUDA. In *Proceedings of High Performance Computing (HiPC)*, volume 7, pages 197–208.

Takaaki Hori and Atsushi Nakamura. 2005. Generalized fast on-the-fly composition algorithm for WFST-based speech recognition. In *Proceedings of INTERSPEECH*, pages 557–560.

Minyoung Jung, Jinwoo Park, Johann Blieberger, and Bernd Burgstaller. 2017. Parallel construction of simultaneous deterministic finite automata on shared-memory multicores. In *Proceedings of the International Conference on Parallel Processing (ICPP)*, pages 271–281.

Bryan Jurish and Kay-Michael Würzner. 2013. Multi-threaded composition of finite-state-automata. In *Proceedings of FSMNLP*, pages 81–89.

Richard E. Ladner and Michael J. Fischer. 1980. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Duane Merrill, Michael Garland, and Andrew Grimshaw. 2012. Scalable GPU graph traversal. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 117–128.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Mehryar Mohri. 2009. Weighted automata algorithms. In *Handbook of Weighted Automata*, pages 213–254. Springer.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Todd Mytkowicz, Madanlal Musuvathi, and Wolfram Schulte. 2014. Data-parallel finite-state machines. In *Proceedings of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 529–542.

Fernando C. N. Pereira and Michael D. Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*. MIT Press.

Supervised Treebank Conversion: Data and Approaches

Xinzhou Jiang[2]*, Bo Zhang[2], Zhenghua Li[1,2], Min Zhang[1,2], Sheng Li[3], Luo Si[3]

1. Institute of Artificial Intelligence, Soochow University, Suzhou, China

2. School of Computer Science and Technology, Soochow University, Suzhou, China

`{xzjiang, bzhang17}@stu.suda.edu.cn, {zhli13,minzhang}@suda.edu.cn`

3. Alibaba Inc., Hangzhou, China

`{lisheng.ls,luo.si}@alibaba-inc.com`

Abstract

Treebank conversion is a straightforward and effective way to exploit various heterogeneous treebanks for boosting parsing accuracy. However, previous work mainly focuses on unsupervised treebank conversion and makes little progress due to the lack of manually labeled data where each sentence has two syntactic trees complying with two different guidelines at the same time, referred as *bi-tree aligned data.*

In this work, we for the first time propose the task of *supervised treebank conversion*. First, we manually construct a bi-tree aligned dataset containing over ten thousand sentences. Then, we propose two simple yet effective treebank conversion approaches (pattern embedding and treeLSTM) based on the state-of-the-art deep biaffine parser. Experimental results show that 1) the two approaches achieve comparable *conversion accuracy*, and 2) treebank conversion is superior to the widely used multi-task learning framework in multiple treebank exploitation and leads to significantly higher *parsing accuracy*.

1 Introduction

During the past few years, neural network based dependency parsing has achieved significant progress and outperformed the traditional discrete-feature based parsing (Chen and Manning, 2014; Dyer et al., 2015; Zhou et al., 2015; Andor et al., 2016). Most remarkably, Dozat and Manning (2017) propose a simple yet effective deep biaffine parser that further advances the state-of-the-art accuracy by large margin. As reported, their parser outperforms the state-of-the-art discrete-feature based parser of Bohnet and Nivre (2012) by 0.97 (93.76% − 92.79%) on the English WSJ data and 6.87 (85.38% − 78.51%) on the Chinese CoNLL-2009 data, respectively. Kindly note that all these results are obtained by training parsers on a single treebank.

Treebanks	#Tok	Grammar
Sinica (Chen et al., 2003)	0.36M	Case grammar
CTB (Xue et al., 2005)	1.62M	Phrase structure
TCT (Zhou, 2004)	1.00M	Phrase structure
PCT (Zhan, 2012)	0.90M	Phrase structure
HIT-CDT (Che et al., 2012)	0.90M	Dependency structure
PKU-CDT (Qiu et al., 2014)	1.40M	Dependency structure

Table 1: Large-scale Chinese treebanks (token number in million).

Meanwhile, motivated by different syntactic theories and practices, major languages in the world often possess multiple large-scale heterogeneous treebanks, e.g., Tiger (Brants et al., 2002) and TüBa-D/Z (Telljohann et al., 2004) treebanks for German, Talbanken (Einarsson, 1976) and Syntag (Järborg, 1986) treebanks for Swedish, ISST (Montemagni et al., 2003) and TUT[1] treebanks for Italian, etc. Table 1 lists several large-scale Chinese treebanks. In this work, we take HIT-CDT as a case study. Our next-step plan is to annotate bi-tree aligned data for PKU-CDT and then convert PKU-CDT to our guideline. For non-dependency treebanks, the straight-

* The first two (student) authors make equal contributions to this work. Zhenghua is the correspondence author.

[1] `http://www.di.unito.it/~tutreeb/`

forward choice is to convert such treebanks to dependency treebanks based on heuristic head-finding rules. The second choice is to directly extend our proposed approaches by adapting the patterns and treeLSTMs for non-dependency structures, which should be straightforward as well.

Considering the high cost of treebank construction, it has always been an interesting and attractive research direction to exploit various heterogeneous treebanks for boosting parsing performance. Though under different linguistic theories or annotation guidelines, the treebanks are painstakingly developed to capture the syntactic structures of the same language, thereby having a great deal of common grounds.

Previous researchers have proposed two approaches for multi-treebank exploitation. On the one hand, the *guiding-feature* method projects the knowledge of the source-side treebank into the target-side treebank, and utilizes extra pattern-based features as guidance for the target-side parsing, mainly for the traditional discrete-feature based parsing (Li et al., 2012). On the other hand, the *multi-task learning* method simultaneously trains two parsers on two treebanks and uses shared neural network parameters for representing common-ground syntactic knowledge (Guo et al., 2016).[2] Regardless of their effectiveness, while the guiding-feature method fails to directly use the source-side treebank as extra training data, the multi-task learning method is incapable of explicitly capturing the structural correspondences between two guidelines. In this sense, we consider both of them as *indirect exploitation* approaches.

Compared with the indirect approaches, treebank conversion aims to directly convert a source-side treebank into the target-side guideline, and uses the converted treebank as extra labeled data for training the target-side model. Taking the example in Figure 1, the goal of this work is to convert the under tree that follows the HIT-CDT guideline (Che et al., 2012) into the upper one that follows our new guideline. However, due to the lack

[2] Johansson (2013) applies the feature-sharing approach of Daumé III (2007) for multiple treebank exploitation, which can be regarded as a simple discrete-feature variant of multi-task learning.

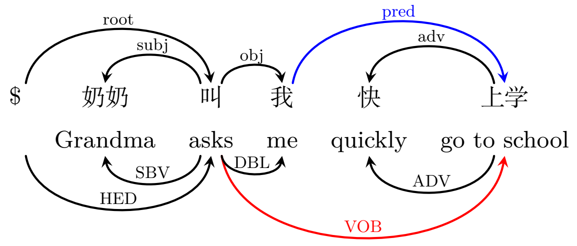


Figure 1: Example of treebank conversion from the source-side HIT-CDT tree (under) to the target-side our-CDT tree (upper).

of *bi-tree aligned data*, in which each sentence has two syntactic trees following the source-side and target-side guidelines respectively, most previous studies are based on unsupervised treebank conversion (Niu et al., 2009) or pseudo bi-tree aligned data (Zhu et al., 2011; Li et al., 2013), making very limited progress.

In this work, we for the first time propose the task of supervised treebank conversion. *The key motivation is to better utilize a large-scale source-side treebank by constructing a small-scale bi-tree aligned data.* In summary, we make the following contributions.

(1) We have manually annotated a high-quality bi-tree aligned data containing over ten thousand sentences, by re-annotating the HIT-CDT treebank according to a new guideline.

(2) We propose a pattern embedding conversion approach by retrofitting the indirect guiding-feature method of Li et al. (2012) to the direct conversion scenario, with several substantial extensions.

(3) We propose a treeLSTM conversion approach that encodes the source-side tree at a deeper level than the shallow pattern embedding approach.

Experimental results show that 1) the two conversion approaches achieve nearly the same conversion accuracy, and 2) direct treebank conversion is superior to indirect multi-task learning in exploiting multiple treebanks in methodology simplicity and performance, yet with the cost of manual annotation. We release the annotation guideline and the newly annotated data in http://hlt.suda.edu.cn/index.php/SUCDT.

2 Annotation of Bi-tree Aligned Data

The key issue for treebank conversion is that sentences in the source-side and target-side treebanks are non-overlapping. In other words, there lacks a bi-tree aligned data in which each sentence has two syntactic trees complying with two guidelines as shown in Figure 1. Consequently, we cannot train a supervised conversion model to directly learn the structural correspondences between the two guidelines. To overcome this obstacle, we construct a bi-tree aligned data of over ten thousand sentences by re-annotating the publicly available dependency-structure HIT-CDT treebank according to a new annotation guideline.

2.1 Data Annotation

Annotation guideline. Unlike phrase-structure treebank construction with very detailed and systematic guidelines (Xue et al., 2005; Zhou, 2004), previous works on Chinese dependency-structure annotation only briefly describe each relation label with a few concrete examples. For example, the HIT-CDT guideline contains 14 relation labels and illustrates them in a 14-page document.

The UD (universal dependencies) project[3] releases a more detailed language-generic guideline to facilitate cross-linguistically consistent annotation, containing 37 relation labels. However, after in-depth study, we find that the UD guideline is very useful and comprehensive, but may not be completely compact for realistic annotation of Chinese-specific syntax. After many months' investigation and trial, we have developed a systematic and detailed annotation guideline for Chinese dependency treebank construction. Our 60-page guideline employs 20 relation labels and gives detailed illustrations for annotation, in order to improve consistency and quality.

Please refer to Guo et al. (2018) for the details of our guideline, including detailed discussions on the correspondences and differences between the UD guideline and ours.

[3]http://universaldependencies.org

Partial annotation. To save annotation effort, we adopt the idea of Li et al. (2016) and only annotate the most uncertain (difficult) words in a sentence. For simplicity, we directly use their released parser and produce the uncertainty results of all HLT-CDT sentences via two-fold jack-knifing. First, we select $2,000$ most difficult sentences of lengths $[5, 10]$ for full annotation[4]. Then, we select $3,000$ most difficult sentences of lengths $[10, 20]$ from the remaining data for 50% annotation. Finally, we select $6,000$ most difficult sentences of lengths $[5, 25]$ for 20% annotation from the remaining data. The difficulty of a sentence is computed as the averaged difficulty of its selected words.

Annotation platform. To guarantee annotation consistency and data quality, we build an online annotation platform to support *strict double annotation* and subsequent inconsistency handling. Each sentence is distributed to two random annotators. If the two submissions are not the same (inconsistent dependency or relation label), a third expert annotator will compare them and decide a single answer.

Annotation process. We employ about 20 students in our university as part-time annotators. Before real annotation, we first give a detailed talk on the guideline for about two hours. Then, the annotators spend several days on systematically studying our guideline. Finally, they are required to annotate 50 testing sentences on the platform. If the submission is different from the correct answer, the annotator receives an instant feedback for self-improvement. Based on their performance, about 10 capable annotators are chosen as experts to deal with inconsistent submissions.

2.2 Statistics and Analysis

Consistency statistics. Compared with the final answers, the overall accuracy of all annotators is 87.6%. Although the overall inter-annotator dependency-wise consistency rate is 76.5%, the sentence-wise consistency rate is only 43.7%. In other words, 56.3% $(100 - 43.7)$ sentences are further checked by a third expert annotator. This shows how

[4] Punctuation marks are ruled out and unannotated.

difficult it is to annotate syntactic structures and how important it is to employ strict double annotation to guarantee data quality.

Annotation time analysis. As shown in Table 2, the averaged sentence length is 15.4 words in our annotated data, among which 4.7 words (30%) are partially annotated with their heads. According to the records of our annotation platform, each sentence requires about 3 minutes in average, including the annotation time spent by two annotators and a possible expert. The total cost of our data annotation is about 550 person-hours, which can be completed by 20 full-time annotators within 4 days. The most cost is spent on quality control via two-independent annotation and inconsistency handling by experts. This is in order to obtain very high-quality data. The cost is reduced to about 150 person-hours without such strict quality control.

Heterogeneity analysis. In order to understand the heterogeneity between our guideline and the HIT-CDT guideline, we analyze the $36,348$ words with both-side heads in the `train` data, as shown in Table 2. The consistency ratio of the two guidelines is 81.69% (UAS), without considering relation labels. By mapping each relation label in HIT-CDT (14 in total) to a single label of our guideline (20 in total), the maximum consistency ratio is 73.79% (LAS). The statistics are similar for the `dev`/`test` data.

3 Indirect Multi-task Learning

Basic parser. In this work, we build all the approaches over the state-of-the-art deep biaffine parser proposed by Dozat and Manning (2017). As a graph-based dependency parser, it employs a deep biaffine neural network to compute the scores of all dependencies, and uses viterbi decoding to find the highest-scoring tree. Figure 2 shows how to score a dependency $i \leftarrow j$.[5]

First, the biaffine parser applies multi-layer bidirectional sequential LSTMs (biSeqLSTM) to encode the input sentence. The word/tag embeddings $\mathbf{e}^{w_k}$ and $\mathbf{e}^{t_k}$ are concatenated as the input vector at w_k.

[5] The score computation of the relation labels is analogous, but due to space limitation, we refer readers to Dozat and Manning (2017) for more details.

Then, the output vector of the top-layer biSeqLSTM at w_k, denoted as $\mathbf{h}_k^{seq}$, is fed into two separate MLPs to get two lower-dimensional representation vectors.

$$\begin{aligned} \mathbf{r}_k^{\mathrm{H}} &= \mathrm{MLP}^{\mathrm{H}}\left(\mathbf{h}_k^{seq}\right) \\ \mathbf{r}_k^{\mathrm{D}} &= \mathrm{MLP}^{\mathrm{D}}\left(\mathbf{h}_k^{seq}\right) \end{aligned} \tag{1}$$

where $\mathbf{r}_k^{\mathrm{H}}$ is the representation vector of w_k as a head word, and $\mathbf{r}_k^{\mathrm{D}}$ as a dependent.

Finally, the score of the dependency $i \leftarrow j$ is computed via a biaffine operation.

$$\texttt{score}(i \leftarrow j) = \begin{bmatrix} \mathbf{r}_i^{\mathrm{D}} \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W}^b \mathbf{r}_j^{\mathrm{H}} \tag{2}$$

During training, the original biaffine parser uses the local softmax loss. For each w_i and its head w_j, its loss is defined as $-\log \frac{e^{\texttt{score}(i \leftarrow j)}}{\sum_k e^{\texttt{score}(i \leftarrow k)}}$. Since our training data is partially annotated, we follow Li et al. (2016) and employ the global CRF loss (Ma and Hovy, 2017) for better utilization of the data, leading to consistent accuracy gain.

Multi-task learning aims to incorporate labeled data of multiple related tasks for improving performance (Collobert and Weston, 2008). Guo et al. (2016) apply multi-task learning to multi-treebank exploitation based on the neural transition-based parser of Dyer et al. (2015), and achieve higher improvement than the guiding-feature approach of Li et al. (2012).

Based on the state-of-the-art biaffine parser, this work makes a straightforward extension to realize multi-task learning. We treat the source-side and target-side parsing as two individual tasks. The two tasks use shared parameters for word/tag embeddings and multi-layer biSeqLSTMs to learn common-ground syntactic knowledge, use separate parameters for the MLP and biaffine layers to learn task-specific information.

4 Direct Treebank Conversion

Task definition. As shown in Figure 1, given an input sentence $\mathbf{x}$, treebank conversion aims to convert the under source-side tree $\mathbf{d}^{src}$ to the upper target-side tree $\mathbf{d}^{tgt}$. Therefore, the main challenge is how to make full use of the given $\mathbf{d}^{src}$ to guide the construction of $\mathbf{d}^{tgt}$. Specifically, under the biaffine parser framework, the key is to utilize $\mathbf{d}^{src}$ as guidance for better scoring an arbitrary target-side dependency $i \leftarrow j$.

In this paper, we try to encode the structural information of i and j in $\mathbf{d}^{src}$ as a dense vector from two representation levels, thus leading to two approaches, i.e., the shallow pattern embedding approach and the deep treeLSTM approach. The dense vectors are then used as extra inputs of the MLP layer to obtain better word representations, as shown in Figure 2.

4.1 The Pattern Embedding Approach

In this subsection, we propose the pattern embedding conversion approach by retrofitting the indirect guiding-feature method of Li et al. (2012) to the direct conversion scenario, with several substantial extensions.

The basic idea of Li et al. (2012) is to use extra guiding features produced by the source-side parser. First, they train the source parser $Parser^{src}$ on the source-side treebank. Then, they use $Parser^{src}$ to parse the target-side treebank, leading to *pseudo* bi-tree aligned data. Finally, they use the predictions of $Parser^{src}$ as extra pattern-based guiding features and build a better target-side parser $Parser^{tgt}$.

The original method of Li et al. (2012) is proposed for traditional discrete-feature based parsing, and does not consider the relation labels in $\mathbf{d}^{src}$. In this work, we make a few useful extensions for more effective utilization of $\mathbf{d}^{src}$.

- We further subdivide their "else" pattern into four cases according to the length of the path from w_i to w_j in $\mathbf{d}^{src}$. The left part of Figure 2 shows all 9 patterns.
- We use the labels of w_i and w_j in $\mathbf{d}^{src}$, denoted as l_i and l_j.
- Inspired by the treeLSTM approach, we also consider the label of w_a, the lowest common ancestor (LCA) of w_i and w_j, denoted as l_a.

Our pattern embedding approach works as follows. Given $i \leftarrow j$, we first decide its pattern type according to the structural relationship between w_i and w_j in $\mathbf{d}^{src}$, denoted as $p_{i\leftarrow j}$. For example, if w_i and w_j are both the children of a third word w_k in $\mathbf{d}^{src}$, then $p_{i\leftarrow j} =$ "*sibling*". Figure 2 shows all 9 patterns.

Then, we embed $p_{i\leftarrow j}$ into a dense vector $\mathbf{e}^{p_{i\leftarrow j}}$ through a lookup operation in order to fit into the biaffine parser. Similarly, the three labels are also embedded into three dense vectors, i.e., $\mathbf{e}^{l_i}$, $\mathbf{e}^{l_j}$, $\mathbf{e}^{l_a}$.

The four embeddings are combined as $\mathbf{r}^{pat}_{i\leftarrow j}$ to represent the structural information of w_i and w_j in $\mathbf{d}^{src}$.

$$\mathbf{r}^{pat}_{i\leftarrow j} = \mathbf{e}^{p_{i\leftarrow j}} \oplus \mathbf{e}^{l_i} \oplus \mathbf{e}^{l_j} \oplus \mathbf{e}^{l_a} \qquad (3)$$

Finally, the representation vector $\mathbf{r}^{pat}_{i\leftarrow j}$ and the top-layer biSeqLSTM outputs are concatenated as the inputs of the MLP layer.

$$\begin{aligned} \mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j} &= \mathrm{MLP}^{\mathrm{D}}\left(\mathbf{r}^{seq}_i \oplus \mathbf{r}^{pat}_{i\leftarrow j}\right) \\ \mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j} &= \mathrm{MLP}^{\mathrm{H}}\left(\mathbf{r}^{seq}_j \oplus \mathbf{r}^{pat}_{i\leftarrow j}\right) \end{aligned} \qquad (4)$$

Through $\mathbf{r}^{pat}_{i\leftarrow j}$, the extended word representations, i.e., $\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$ and $\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$, now contain the structural information of w_i and w_j in $\mathbf{d}^{src}$.

The remaining parts of the biaffine parser is unchanged. The extended $\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$ and $\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$ are fed into the biaffine layer to compute a more reliable score of the dependency $i \leftarrow j$, with the help of the guidance of $\mathbf{d}^{src}$.

4.2 The TreeLSTM Approach

Compared with the pattern embedding approach, our second conversion approach employs treeLSTM to obtain a deeper representation of $i \leftarrow j$ in the source-side tree $\mathbf{d}^{src}$. Tai et al. (2015) first propose treeLSTM as a generalization of seqLSTM for encoding tree-structured inputs, and show that treeLSTM is more effective than seqLSTM on the semantic relatedness and sentiment classification tasks. Miwa and Bansal (2016) compare three treeLSTM variants on the relation extraction task and show that the SP-tree (shortest path) treeLSTM is superior to the full-tree and subtree treeLSTMs.

In this work, we employ the SP-tree treeLSTM of Miwa and Bansal (2016) for our treebank conversion task. Our preliminary experiments also show the SP-tree treeLSTM outperforms the full-tree treeLSTM, which is consistent with Miwa and Bansal. We did not implement the in-between subtree treeLSTM.

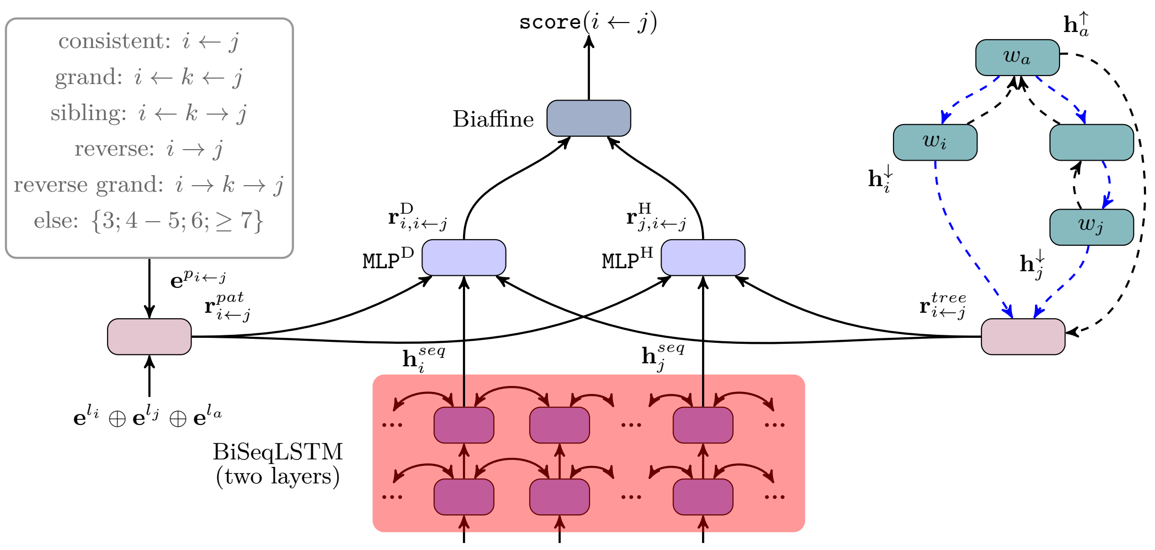

consistent: $i \leftarrow j$
grand: $i \leftarrow k \leftarrow j$
sibling: $i \leftarrow k \rightarrow j$
reverse: $i \rightarrow j$
reverse grand: $i \rightarrow k \rightarrow j$
else: $\{3; 4-5; 6; \geq 7\}$
$\texttt{score}(i \leftarrow j)$
Biaffine
$\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$
$\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$
MLP$^{\mathrm{D}}$
MLP$^{\mathrm{H}}$
$\mathbf{e}^{p_{i\leftarrow j}}$
$\mathbf{r}^{pat}_{i\leftarrow j}$
$\mathbf{h}^{seq}_i$
$\mathbf{h}^{seq}_j$
$\mathbf{e}^{l_i} \oplus \mathbf{e}^{l_j} \oplus \mathbf{e}^{l_a}$
BiSeqLSTM (two layers)
$\mathbf{h}^{\uparrow}_a$
w_a
w_i
$\mathbf{h}^{\downarrow}_i$
w_j
$\mathbf{h}^{\downarrow}_j$
$\mathbf{r}^{tree}_{i\leftarrow j}$

Figure 2: Computation of $\texttt{score}(i \leftarrow j)$ in our proposed conversion approaches. Without the source-side tree $\mathbf{d}^{src}$, the baseline uses the basic $\mathbf{r}^{\mathrm{D}}_i$ and $\mathbf{r}^{\mathrm{H}}_j$ (instead of $\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$ and $\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$).

Given w_i and w_j and their LCA w_a, the SP-tree is composed of two paths, i.e., the path from w_a to w_i and the path from w_a to w_j, as shown in the right part of Figure 2.

Different from the shallow pattern embedding approach, the treeLSTM approach runs a bidirectional treeLSTM through the SP-tree, in order to encode the structural information of w_i and w_j in $\mathbf{d}^{src}$ at a deeper level. The top-down treeLSTM starts from w_a and accumulates information until w_i and w_j, whereas the bottom-up treeLSTM propagates information in the opposite direction.

Following Miwa and Bansal (2016), we stack our treeLSTM on top of the biSeqLSTM layer of the basic biaffine parser, instead of directly using word/tag embeddings as inputs. For example, the input vector for w_k in the treeLSTM is $\mathbf{x}_k = \mathbf{h}^{seq}_k \oplus \mathbf{e}^{l_k}$, where $\mathbf{h}^{seq}_k$ is the top-level biSeqLSTM output vector at w_k, and l_k is the label between w_k and its head word in $\mathbf{d}^{src}$, and $\mathbf{e}^{l_k}$ is the label embedding.

In the bottom-up treeLSTM, an LSTM node computes a hidden vector based on the combination of the input vector and the hidden vectors of its children in the SP-tree. The right part of Figure 2 and Eq. (5) illustrate the computation at w_a.

$$\begin{aligned}
\tilde{\mathbf{h}}_a &= \sum_{k \in \mathcal{C}(a)} \mathbf{h}_k \\
\mathbf{i}_a &= \sigma\left(\mathbf{U}^{(i)}\mathbf{x}_a + \mathbf{V}^{(i)}\tilde{\mathbf{h}}_a + \mathbf{b}^{(i)}\right) \\
\mathbf{f}_{a,k} &= \sigma\left(\mathbf{U}^{(f)}\mathbf{x}_a + \mathbf{V}^{(f)}\mathbf{h}_k + \mathbf{b}^{(f)}\right) \\
\mathbf{o}_a &= \sigma\left(\mathbf{U}^{(o)}\mathbf{x}_a + \mathbf{V}^{(o)}\tilde{\mathbf{h}}_a + \mathbf{b}^{(o)}\right) \\
\mathbf{u}_a &= \tanh\left(\mathbf{U}^{(u)}\mathbf{x}_a + \mathbf{V}^{(u)}\tilde{\mathbf{h}}_a + \mathbf{b}^{(u)}\right) \\
\mathbf{c}_a &= \mathbf{i}_a \odot \mathbf{u}_a + \sum_{k \in \mathcal{C}(a)} \mathbf{f}_{a,k} \odot \mathbf{c}_k \\
\mathbf{h}_a &= \mathbf{o}_a \odot \tanh\left(\mathbf{c}_a\right)
\end{aligned} \tag{5}$$

where $\mathcal{C}(a)$ means the children of w_a in the SP-tree, and $\mathbf{f}_{a,k}$ is the forget vector for w_a's child w_k.

The top-down treeLSTM sends information from the root w_a to the leaves w_i and w_j. An LSTM node computes a hidden vector based on the combination of its input vector and the hidden vector of its single preceding (father) node in the SP-tree.

After performing the biTreeLSTM, we follow Miwa and Bansal (2016) and use the combination of three output vectors to represent the structural information of w_i and w_j in $\mathbf{d}^{src}$, i.e., the output vectors of w_i and w_j in the top-down treeLSTM, and the output vector of w_a

	#Sent	#Tok (HIT)	#Tok (our)
`train`	7,768	119,707	36,348
`dev`	998	14,863	4,839
`test`	1,995	29,975	9,679
`train-HIT`	52,450	980,791	36,348

Table 2: Data statistics. Kindly note that sentences in `train` are also in `train-HIT`.

in the bottom-up treeLSTM.

$$\mathbf{r}_{i \leftarrow j}^{tree} = \mathbf{h}_i^{\downarrow} \oplus \mathbf{h}_j^{\downarrow} \oplus \mathbf{h}_a^{\uparrow} \tag{6}$$

Similar to Eq. (4) for the pattern embedding approach, we concatenate $\mathbf{r}_{i \leftarrow j}^{tree}$ with the output vectors of the top-layer biSeqLSTM, and feed them into MLP$^{\text{H/D}}$.

5 Experiments

5.1 Experiment Settings

Data. We randomly select $1,000/2,000$ sentences from our newly annotated data as the `dev/test` datasets, and the remaining as `train`. Table 2 shows the data statistics after removing some broken sentences (ungrammatical or wrongly segmented) discovered during annotation. The "#tok (our)" column shows the number of tokens annotated according to our guideline. `Train-HIT` contains all sentences in HIT-CDT except those in `dev/test`, among which most sentences only have the HIT-CDT annotations.

Evaluation. We use the standard labeled attachment score (LAS, UAS for unlabeled) to measure the parsing and conversion accuracy.

Implementation. In order to more flexibly realize our ideas, we re-implement the baseline biaffine parser in C++ based on the lightweight neural network library of Zhang et al. (2016). On the Chinese CoNLL-2009 data, our parser achieves 85.80% in LAS, whereas the original tensorflow-based parser[6] achieves 85.54% (85.38% reported in their paper) under the same parameter settings and external word embedding.

Hyper-parameters. We follow most parameter settings of Dozat and Manning (2017). The external word embedding dictionary is trained on Chinese Gigaword (LDC2003T09) with GloVe (Pennington et al., 2014). For efficiency, we use two biSeqLSTM layers instead of three, and reduce the biSeqLSTM output dimension (300) and the MLP output dimension (200).

[6]`https://github.com/tdozat/Parser-v1`

	Training data	UAS	LAS
Multi-task	`train & train-HIT`	79.29	74.51
Pattern	`train`	86.66	82.03
TreeLSTM	`train`	**86.69**	**82.09**
Combined	`train`	86.66	81.82

Table 3: Conversion accuracy on `test` data.

For the conversion approaches, the source-side pattern/label embedding dimensions are 50 (thus $|\mathbf{r}_{i \leftarrow j}^{pat}| = 200$), and the treeLSTM output dimension is 100 (thus $|\mathbf{r}_{i \leftarrow j}^{tree}| = 300$).

During training, we use 200 sentences as a data batch, and evaluate the model on the `dev` data every 50 batches (as an epoch). Training stops after the peak LAS on `dev` does not increase in 50 consecutive epochs.

For the multi-task learning approach, we randomly sample 100 `train` sentences and 100 `train-HIT` sentences to compose a data batch, for the purpose of corpus weighting.

To fully utilize `train-HIT` for the conversion task, the conversion models are built upon multi-task learning, and directly reuse the embeddings and biSeqLSTMs of the multi-task trained model without fine-tuning.

5.2 Results: Treebank Conversion

Table 3 shows the conversion accuracy on the `test` data. As a strong baseline for the conversion task, the multi-task trained target-side parser ("multi-task") does not use $\mathbf{d}^{src}$ during both training and evaluation. In contrast, the conversion approaches use both the sentence $\mathbf{x}$ and $\mathbf{d}^{src}$ as inputs.

Compared with "multi-task", the two proposed conversion approaches achieve nearly the same accuracy, and are able to dramatically improve the accuracy with the extra guidance of $\mathbf{d}^{src}$. The gain is 7.58 (82.09 − 74.51) in LAS for the treeLSTM approach.

It is straightforward to combine the two conversion approaches. We simply concatenate $\mathbf{h}_{i/j}^{seq}$ with both $\mathbf{r}_{i \leftarrow j}^{pat}$ and $\mathbf{r}_{i \leftarrow j}^{tree}$ before feeding into MLP$^{\text{H/D}}$. However, the "combined" model leads to no further improvement. This indicates that although the two approaches try

	on `dev`		on `test`	
	UAS	LAS	UAS	LAS
Pattern (full)	86.73	81.93	86.66	82.03
w/o distance	86.73	81.75	86.57	81.94
w/o l_i	86.47	80.55	86.47	81.15
w/o l_j	86.55	81.69	86.45	81.76
w/o l_a	86.24	81.66	86.17	81.51
w/o labels	86.05	79.78	85.93	80.08
TreeLSTM (full)	86.73	81.95	86.69	82.09
w/o labels	86.55	80.32	86.20	80.56

Table 4: Feature ablation for the conversion approaches.

to encode the structural information of w_i and w_j in $\mathbf{d}^{src}$ from different perspectives, the resulted representations are actually overlapping instead of complementary, which is contrary to our intuition that the treeLSTM approach should give better and deeper representations than the shallow pattern embedding approach. We have also tried several straightforward modifications to the standard treeLSTM in Eq. (5), but found no further improvement. We leave further exploration of better treeLSTMs and model combination approaches as future work.

Feature ablation results are presented in Table 4 to gain more insights on the two proposed conversion approaches. In each experiment, we remove a single component from the full model to learn its individual contribution.

For the pattern embedding approach, all proposed extensions to the basic pattern-based approach of Li et al. (2012) are useful. Among the three labels, the embedding of l_i is the most useful and its removal leads to the highest LAS drop of 0.88 (82.03 − 81.15). This is reasonable considering that 81.69% dependencies are consistent in the two guidelines, as discussed in the heterogeneity analysis of Section 2.2. Removing all three labels decreases UAS by 0.73 (86.66−85.93) and LAS by 1.95 (82.03 − 80.08), demonstrating that the source-side labels are highly correlative with the target-side labels, and therefore very helpful for improving LAS.

For the treeLSTM approach, the source-side labels in $\mathbf{d}^{src}$ are also very useful, improving UAS by 0.49 (86.69 − 86.20) and LAS by 1.53 (82.09 − 80.56).

5.3 Results: Utilizing Converted Data

Another important question to be answered is whether treebank conversion can lead to higher parsing accuracy than multi-task learning. In terms of model simplicity, treebank conversion is better because eventually the target-side parser is trained directly on an enlarged homogeneous treebank unlike the multi-task learning approach that needs to simultaneously train two parsers on two heterogeneous treebanks.

Table 5 shows the empirical results. Please kindly note that the parsing accuracy looks very low, because the `test` data is partially annotated and only about 30% most uncertain (difficult) words are manually labeled with their heads according to our guideline, as discussed in Section 2.1.

The first-row, "single" is the baseline target-side parser trained on the `train` data.

The second-row "single (hetero)" refers to the source-side heterogeneous parser trained on `train-HIT` and evaluated on the target-side `test` data. Since the similarity between the two guidelines is high, as discussed in Section 2.2, the source-side parser achieves even higher UAS by 0.21 (76.20 − 75.99) than the baseline target-side parser trained on the small-scale `train` data. The LAS is obtained by mapping the HIT-CDT labels to ours (Section 2.2).

In the third row, "multi-task" is the target-side parser trained on `train & train-HIT` with the multi-task learning approach. It significantly outperforms the baseline parser by 4.30 (74.51 − 70.21) in LAS. This shows that the multi-task learning approach can effectively utilize the large-scale `train-HIT` to help the target-side parsing.

In the fourth row, "single (large)" is the basic parser trained on the large-scale `converted train-HIT` (homogeneous). We employ the treeLSTM approach to convert all sentences in train-HIT into our guideline.[7] We can see that

[7] For each sentence in `train`, which is already partially annotated, the conversion model actually completes the partial target-side tree into a full tree via constrained decoding. As shown by the results in Li et al. (2016), since the most difficult dependencies are known and given to the model, the parsing accuracy will be much higher than the traditional parsing without constraints.

	Training data	UAS	LAS
Single	`train`	75.99	70.95
Single (hetero)	`train-HIT`	76.20	68.43
Multi-task	`train & train-HIT`	79.29	74.51
Single (large)	`converted train-HIT`	**80.45**	**75.83**

Table 5: Parsing accuracy on `test` data. LAS difference between any two systems is statistically significant ($p < 0.005$) according to Dan Bikel' s randomized parsing evaluation comparer for significance test Noreen (1989).

Task	Training data	UAS	LAS
Conversion	`train`	93.42	90.49
Parsing (baseline)	`train`	89.66	86.41
Parsing (ours)	`converted train-HIT`	91.16	88.07

Table 6: Results on the fully annotated 372 sentences of the `test` data.

the single parser trained on the converted data significantly outperforms the parser in the multi-task learning approach by 1.32 (75.83 – 74.51) in LAS.

In summary, we can conclude that *treebank conversion is superior to multi-task learning in multi-treebank exploitation for its simplicity and better performance.*

5.4 Results on fully annotated data

We randomly divided the newly annotated data into train/dev/test, so the test set has a mix of 100%, 50% and 20% annotated sentences. To gain a rough estimation of the performance of different approaches on fully annotated data, we give the results in Table 6. We can see that all the models achieve much higher accuracy on the portion of fully annotated data than on the whole test data as shown in Table 3 and 5, since the dependencies to be evaluated are the most difficult ones in a sentence for the portion of partially annotated data. Moreover, the conversion model can achieve over 90% LAS thanks to the guidance of the source-side HIT-CDT tree. Please also note that there would still be a slight bias, because those fully annotated sentences are chosen as the most difficult ones according to the parsing model but are also very short ($[5, 10]$).

6 Conclusions and Future Work

In this work, we for the first time propose the task of supervised treebank conversion by constructing a bi-tree aligned data of over ten thousand sentences. We design two simple yet effective conversion approaches based on the state-of-the-art deep biaffine parser. Results show that 1) the two approaches achieves nearly the same conversion accuracy; 2) relation labels in the source-side tree are very helpful for both approaches; 3) treebank conversion is more effective in multi-treebank exploitation than multi-task learning, and achieves significantly higher parsing accuracy.

In future, we would like to advance this work in two directions: 1) proposing more effective conversion approaches, especially by exploring the potential of treeLSTMs; 2) constructing bi-tree aligned data for other treebanks and exploiting all available single-tree and bi-tree labeled data for better conversion.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. We are greatly grateful to all participants in data annotation for their hard work. We also thank Guodong Zhou and Wenliang Chen for the helpful discussions, and Meishan Zhang for his help on the re-implementation of the Biaffine Parser. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61502325 61432013), and was also partially supported by the joint research project of Alibaba and Soochow University.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*, pages 2442–2452.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP 2012*, pages 1455–1465.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theory*, pages 24–41.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2012. Chinese Dependency Treebank 1.0 (LDC2012T05). In *Philadelphia: Linguistic Data Consortium.*

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Keh-Jiann Chen, Chi-Ching Luo, Ming-Chung Chang, Feng-Yi Chen, Chao-Jan Chen, Chu-Ren Huang, and Zhao-Ming Gao. 2003. *Sinica treebank: Design criteria,representational issues and implementation*, chapter 13. Kluwer Academic Publishers.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML.*

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*, pages 256–263.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependecy parsing. In *Proceedings of ICLR.*

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*, pages 334–343.

Jan Einarsson. 1976. Talbankens skriftspråkskonkordans. Department of Scandinavian Languages, Lund University.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. A universal framework for inductive transfer parsing across multi-typed treebanks. In *Proceedings of COLING*, pages 12–22.

Lijuan Guo, Zhenghua Li, Xue Peng, and Min Zhang. 2018. Data annotation guideline of chinese dependency syntax for multi-domain and multi-source texts. *Journal of Chinese Information Processing.*

Jerker Järborg. 1986. Manual for syntaggning. Department of Linguistic Computation, University of Gothenburg.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proceedings of NAACL*, pages 127–137.

Xiang Li, Wenbin Jiang, Yajuan Lü, and Qun Liu. 2013. Iterative transformation of annotation guidelines for constituency parsing. In *Proceedings of ACL*, pages 591–596.

Zhenghua Li, Wanxiang Che, and Ting Liu. 2012. Exploiting multiple treebanks for parsing with quasisynchronous grammar. In *Proceedings of ACL*, pages 675–684.

Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. 2016. Active learning for dependency parsing with partial annotation. In *Proceedings of ACL.*

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. In *Proceedings of IJCNLP*, pages 59–69.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.

Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. 2003. Building the italian syntactic–semantic treebank. In Anne Abeille, editor, Building and Using Syntactically Annotated Corpora. Kluwer, Dordrecht.

Zheng-Yu Niu, Haifeng Wang, and Hua Wu. 2009. Exploiting heterogeneous treebanks for parsing. In *Proceedings of ACL*, pages 46–54.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: An introduction.* John Wiley & Sons, Inc., New York.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Likun Qiu, Yue Zhang, Peng Jin, and Houfeng Wang. 2014. Multi-view chinese treebanking. In *Proceedings of COLING*, pages 257–268.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

Heike Telljohann, Erhard Hinrichs, and Sandra Kbler. 2004. The Tüba-D/Z treebank: Annotating German with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, pages 2229–2235.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*, volume 11, pages 207–238.

Weidong Zhan. 2012. The application of treebank to assist Chinese grammar instruction: a preliminary investigation. *Journal of Technology and Chinese Language Teaching*, 3(2):16–29.

Meishan Zhang, Jie Yang, Zhiyang Teng, and Yue Zhang. 2016. Libn3l:a lightweight package for neural nlp. In *Proceedings of LREC*, pages 225–229.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of ACL*, pages 1213–1222.

Qiang Zhou. 2004. Annotation scheme for Chinese treebank. *Journal of Chinese Information Processing*, 18(4):1–8.

Muhua Zhu, Jingbo Zhu, and Minghan Hu. 2011. Better automatic treebank conversion using a feature-based approach. In *Proceedings of ACL*, pages 715–719.

Object-oriented Neural Programming (OONP) for Document Understanding

Zhengdong Lu[1] **Xianggen Liu**[2,3,4,*] **Haotian Cui**[2,3,4,*] **Yukun Yan**[2,3,4,*] **Daqi Zheng**[1]
`luz@deeplycurious.ai`,
`{liuxg16,cht15,yanyk13}@mails.tsinghua.edu.cn`, `da@deeplycurious.ai`
[1] DeeplyCurious.ai
[2] Department of Biomedical Engineering, School of Medicine, Tsinghua University
[3] Beijing Innovation Center for Future Chip, Tsinghua University
[4] Laboratory for Brain and Intelligence, Tsinghua University

Abstract

We propose Object-oriented Neural Programming (OONP), a framework for semantically parsing documents in specific domains. Basically, OONP reads a document and parses it into a predesigned object-oriented data structure that reflects the domain-specific semantics of the document. An OONP parser models semantic parsing as a decision process: a neural net-based Reader sequentially goes through the document, and builds and updates an intermediate ontology during the process to summarize its partial understanding of the text. OONP supports a big variety of forms (both symbolic and differentiable) for representing the state and the document, and a rich family of operations to compose the representation. An OONP parser can be trained with supervision of different forms and strength, including supervised learning (SL) , reinforcement learning (RL) and hybrid of the two. Our experiments on both synthetic and real-world document parsing tasks have shown that OONP can learn to handle fairly complicated ontology with training data of modest sizes.

1 Introduction

Mapping a document into a structured "machine readable" form is a canonical and probably the most effective way for document understanding. There are quite some recent efforts on designing neural net-based learning machines for this purpose, which can be roughly categorized into two groups: 1) sequence-to-sequence model with the

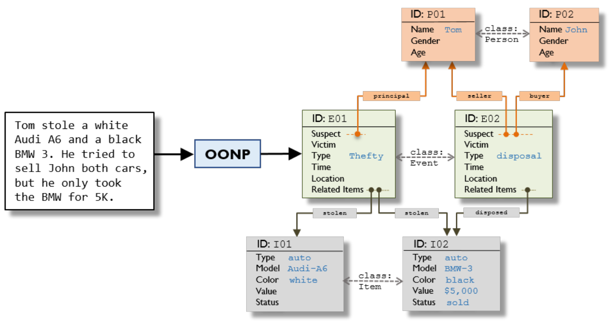


Figure 1: Illustration of OONP on a parsing task.

neural net as the black box (Liang et al., 2017), and 2) neural net as a component in a pre-designed statistical model (Zeng et al., 2014). Both categories are hindered in tackling document with complicated structures, by either the lack of effective representation of knowledge or the flexibility in fusing them in the model.

Towards solving this problem, we proposed Object-oriented Neural Programming (OONP), a framework for semantically parsing in-domain documents (illustrated in Figure 1). OONP maintains an object-oriented data structure, where objects from different classes are to represent entities (people, events, items etc) which are connected through links with varying types. Each object encapsulates internal properties (both symbolic and differentiable), allowing both neural and symbolic reasoning over complex structures and hence making it possible to represent rich semantics of documents. An OONP parser is neural net-based, but it has sophisticated architecture and mechanism designed for taking and yielding discrete structures, hence nicely combining symbolism (for interpretability and formal reasoning) and connectionism (for flexibility and learnability).

For parsing, OONP reads a document and parses it into this object-oriented data structure through a series of discrete actions along reading the document sequentially. OONP supports a rich fam-

* The work was done when these authors worked as interns at DeeplyCurious.ai.

ily of operations for composing the ontology, and flexible hybrid forms for knowledge representation. An OONP parser can be trained with supervised learning (SL), reinforcement learning (RL) and hybrid of the two.

OONP in a nutshell The key properties of OONP can be summarized as follows

1. OONP models parsing as a decision process: as the "reading and comprehension" agent goes through the text it gradually forms the ontology as the representation of the text through its action;
2. OONP uses a symbolic memory with graph structure as part of the state of the parsing process. This memory will be created and updated through the sequential actions of the decision process, and will be used as the semantic representation of the text at the end
3. OONP can blend supervised learning (SL) and reinforcement learning (RL) in tuning its parameters to suit the supervision signal in different forms and strength.

2 Related Works

2.1 Semantic Parsing

Semantic parsing is concerned with translating language utterances into executable logical forms and plays a key role in building conversational interfaces (Jonathan and Percy, 2014). Different from common tasks of semantic parsings, such as parsing the sentence to dependency structure (Buys and Blunsom, 2017) and executable commands (Herzig and Berant, 2017), OONP parses documents into a predesigned object-oriented data structure which is easily readable for both human and machine. It is related to semantic web (Berners-Lee et al., 2001) as well as frame semantics (Charles J, 1982) in the way semantics is represented, so in a sense, OONP can be viewed as a neural-symbolic implementation of semantic parsing with similar semantic representation.

2.2 State Tracking

OONP is inspired by Daumé III et al. (2009) on modeling parsing as a decision process, and the work on state-tracking models in dialogue system (Henderson et al., 2014) for the mixture of symbolic and probabilistic representations of dialogue state. For modeling a document with entities, Yang et al. (2017) use coreference links to recover entity clusters, though they only model entity mentions as containing a single word. However, entities whose names consist of multiple words are not considered. Entity Networks (Henaff et al., 2017) and EntityNLM (Ji et al., 2017) have addressed above problem and are the pioneers to model on tracking entities, but they have not considered the properties of the entities. In fact, explicitly modeling the entities both with their properties and contents is important to understand a document, especially a complex document. For example, if there are two persons named 'Avery', it is vital to know their genders or last names to avoid confusion. Therefore, we propose OONP to sketch objects and their relationships by building a structured graph for document parsing.

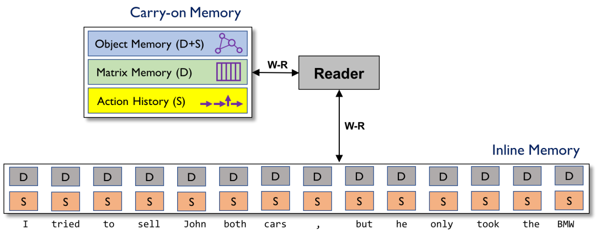


Figure 2: The overall diagram of OONP, where S stands for symbolic representation, D for distributed representation, and S+D for a hybrid of symbolic and distributed parts.

3 OONP: Overview

An OONP parser (illustrated in Figure 2) consists of a Reader equipped with read/write heads, Inline Memory that represents the document, and Carry-on Memory that summarizes the current understanding of the document at each time step. For each document to parse, OONP first preprocesses it and puts it into the Inline Memory, and then Reader controls the read-heads to sequentially go through the Inline Memory and at the same time update the Carry-on Memory. We will give a more detailed description of the major components below.

3.1 Memory

we have two types of memory, Carry-on Memory and Inline Memory. Carry-on Memory is designed to save the state in the decision process and summarize current understanding of the document based on the text that has been "read", while Inline Memory is designed to save location-specific information about the document. In a sense, the information in Inline Memory is low-level and unstructured, waiting for Reader to fuse and integrate into more structured representation.

Carry-on Memory has three compartments:

- Object Memory: denoted M_{obj}, the object-oriented data structure constructed during the parsing process;
- Matrix Memory: denoted M_{mat}, a matrix-type memory with fixed size, for differentiable read/write by the controlling neural net (Graves et al., 2014). In the simplest case, it could be just a vector as the hidden state of conventional RNN;
- Action History: symbolic memory to save the entire history of actions made during the parsing process.

Intuitively, Object Memory stores the extracted knowledge of the document with defined structure and strong evidence, while Matrix Memory keeps the knowledge that is fuzzy, uncertain or incomplete, waiting for further information to confirm, complete or clarify.

Object Memory

Object Memory stores an object-oriented representation of document, as illustrated in Figure 3. Each object is an instance of a particular class[*], which specifies the innate structure of the object, including internal properties, operations, and how this object can be connected with others. The internal properties can be of different types, for example string or category, which usually correspond to different actions in specifying them: the string-type property is usually "copied" from the original text in Inline Memory, while the category properties need to be rendered by a classifier. The links are in general directional and typed, resembling a special property viewing from the "source object". In Figure 3, there are six "linked" objects of three classes (namely, PERSON, EVENT, and ITEM) . Taking ITEM-object `I02` for example, it has five internal properties (`Type`, `Model`, `Color`, `Value`, `Status`), and is linked with two EVENT-objects through `stolen` and `disposed` link respectively.

In addition to the symbolic properties and links, each object had also its *object-embedding* as the distributed interface with Reader. For description simplicity, we will refer to the symbolic part of this hybrid representation of objects as the Ontology, with some slight abuse of this word. Object-embedding is complementary to the symbolic part

[*]We only consider flat structure of classes, but it is possible to have a hierarchy of classes with different levels of abstractness, and to allow an object to go from abstract class to its child during parsing with more information obtained.

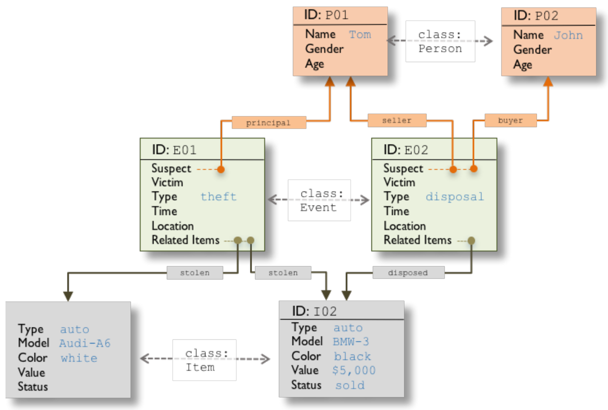


Figure 3: An example of objects of three classes.

of the object, recording all the relevant information associated with it but not represented in the Ontology, e.g., the contextual information when the object is created. Both Ontology and the object embeddings will be updated in time by the class-dependent operations driven by the actions issued by the Policy-net in Reader.

According to the way the Ontology evolves with time, the parsing task can be roughly classified into two categories: 1) **Stationary:** there is a final ground truth that does not change with time, and 2) **Dynamical:** the truth changes with time. For stationary Ontology, see Section 5.2 and 5.3 for example, and for dynamical Ontology, please see Section 5.1.

Inline Memory

Inline Memory stores the relatively raw representation of the document with the sequential structure. Basically, Inline Memory is an array of memory cells, each corresponding to a pre-defined language unit (e.g., word) in the same order as they are in the original text. Each cell can have distributed part and symbolic part, designed to save the result of preprocessing of text, e.g., plain word embedding, hidden states of RNN, or some symbolic processing.

Inline Memory provides a way to represent locally encoded "low level" knowledge of the text, which will be read, evaluated and combined with the global semantic representation in Carry-on Memory by Reader. One particular advantage of this setting is that it allows us to incorporate the local decisions of some other models, including "higher order" ones like local relations across multiple language units, as illustrated in Figure 4.

3.2 Reader

Reader is the control center of OONP, coordinating and managing all the operations of OONP. More

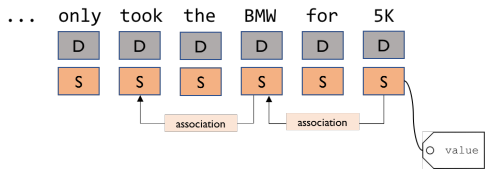


Figure 4: Inline Memory with symbolic knowledge.

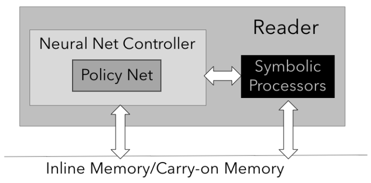


Figure 5: The overall digram of OONP

specifically, it takes the input of different forms (reading), processes it (thinking), and updates the memory (writing). As shown in Figure 5, Reader contains Neural Net Controller (NNC) and multiple symbolic processors, and NNC also has Policy-net as its sub-component. Similar to the controller in Neural Turing Machine (Graves et al., 2014), NNC is equipped with multiple read-heads and write-heads for differentiable read/write over Matrix Memory and (the distributed part of) Inline Memory, with a variety of addressing strategies (Graves et al., 2014). Policy-net however issues discrete outputs (i.e., *actions*), which gradually builds and updates the Object Memory in time. The symbolic processors are designed to handle information in symbolic form from Object Memory, Inline Memory, Action History, and Policy-net, while that from Inline Memory and Action History is eventually generated by Policy-net. In Appendix.A[†], we give a particular implementation of Reader with more details.

4 OONP: Actions

The actions issued by Policy-net can be generally categorized as the following

- `New-Assign` : determine whether to create an new object for the information at hand or assign it to a certain existed object;
- `Update.X` : determine which internal property or link of the selected object to update;
- `Update2what` : determine the content of the updating, which could be about string, category or links;

The typical order of actions is `New-Assign` $\rightarrow$ `Update.X` $\rightarrow$ `Update2what`, but it is common to have `New-Assign` action followed by nothing, when, for example, an object is mentioned but no substantial information is provided. As shown in Figure 6, we give an example of the entire episode of OONP parsing on the short text given in Figure 1, to show that a sequence of actions gradually forms the complete representation of the document.

[†]The appendix is also available at `https://arxiv.org/abs/1709.08853`

5 An examples of actions

5.1 `New-Assign`

With any information at hand (denoted as $\mathcal{S}_t$) at time t, the choices of `New-Assign` include the following three categories of actions: 1) creating (`New`) an object of a certain type, 2) assigning $\mathcal{S}_t$ to an existed object, and 3) doing nothing for $\mathcal{S}_t$ and moving on. For Policy-net, the stochastic policy is to determine the following probabilities:

$$\begin{aligned} \mathsf{prob}(c, \mathsf{new}|\mathcal{S}_t), &\quad c = 1, 2, \cdots, |\mathcal{C}| \\ \mathsf{prob}(c, k|\mathcal{S}_t), &\quad \text{for } \mathcal{O}_t^{c,k} \in \mathsf{M}_{\mathsf{obj}}^t \\ \mathsf{prob}(\mathtt{none}|\mathcal{S}_t) & \end{aligned}$$

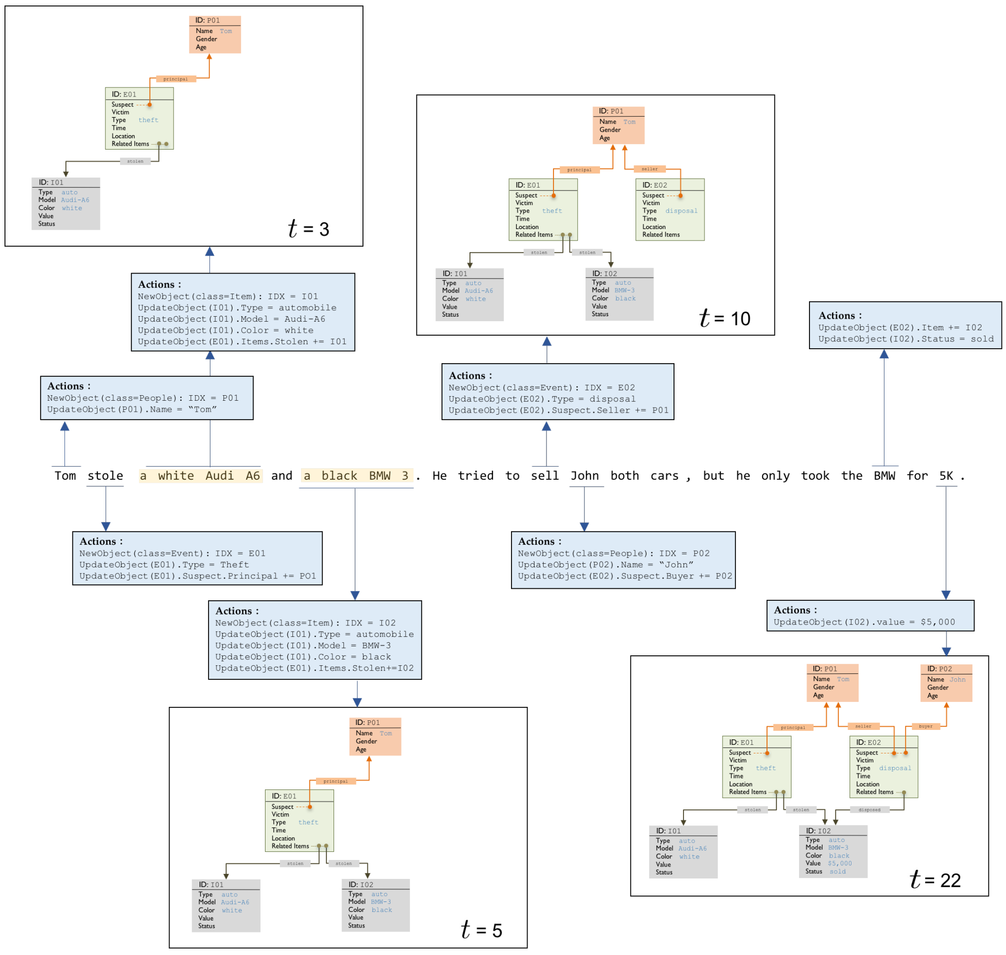
where $|\mathcal{C}|$ stands for the number of classes, $\mathcal{O}_t^{c,k}$ stands for the k^{th} object of class c at time t. Determining whether to new an object always relies on the following two signals

1. The information at hand cannot be contained by any existed objects;
2. Linguistic hints that suggest whether a new object is introduced.

Based on those intuitions, we take a score-based approach to determine the above-mentioned probability. More specifically, for a given $\mathcal{S}_t$, Reader forms a "temporary" object with its own structure (denoted $\hat{\mathcal{O}}_t$) with both symbolic and distributed sections. We also have a virtual object for the New action for each class c, denoted $\mathcal{O}_t^{c,\mathsf{new}}$, which is typically a time-dependent vector formed by Reader based on information in Matrix Memory. For a given $\hat{\mathcal{O}}_t$, we can then define the following $|\mathcal{C}| + |\mathsf{M}_{\mathsf{obj}}^t| + 1$ types of score functions:

$$\begin{aligned} \mathtt{New:}\ & \mathsf{score}_{\mathsf{new}}^{(c)}(\mathcal{O}_t^{c,\mathsf{new}}, \hat{\mathcal{O}}_t; \theta_{\mathsf{new}}^{(c)}),\ c = 1, 2, \cdots, |\mathcal{C}| \\ \mathtt{Assign:}\ & \mathsf{score}_{\mathsf{assign}}^{(c)}(\mathcal{O}_t^{c,k}, \hat{\mathcal{O}}_t; \theta_{\mathsf{assign}}^{(c)}),\ \text{for } \mathcal{O}_t^{c,k} \in \mathsf{M}_{\mathsf{obj}}^t \\ \mathtt{Do\ nothing:}\ & \mathsf{score}_{\mathsf{none}}(\hat{\mathcal{O}}_t; \theta_{\mathsf{none}}). \end{aligned}$$

to measure the level of matching between the information at hand and existed objects, as well as the likeliness for creating an object or doing nothing. This process is pictorially illustrated in Figure 7. We therefore can define the following probability for the stochastic policy



```
Tom stole a white Audi A6 and a black BMW 3 . He tried to sell John both cars , but he only took the BMW for 5K .
```

```
Actions :
NewObject(class=People): IDX = P01
UpdateObject(P01).Name = "Tom"
```

```
Actions :
NewObject(class=Item): IDX = I01
UpdateObject(I01).Type = automobile
UpdateObject(I01).Model = Audi-A6
UpdateObject(I01).Color = white
UpdateObject(E01).Items.Stolen += I01
```

```
Actions :
NewObject(class=Event): IDX = E01
UpdateObject(E01).Type = Theft
UpdateObject(E01).Suspect.Principal += P01
```

```
Actions :
NewObject(class=Item): IDX = I02
UpdateObject(I01).Type = automobile
UpdateObject(I01).Model = BMW-3
UpdateObject(I01).Color = black
UpdateObject(E01).Items.Stolen+=I02
```

```
Actions :
NewObject(class=Event): IDX = E02
UpdateObject(E02).Type = disposal
UpdateObject(E02).Suspect.Seller += P01
```

```
Actions :
NewObject(class=People): IDX = P02
UpdateObject(P02).Name = "John"
UpdateObject(E02).Suspect.Buyer += P02
```

```
Actions :
UpdateObject(E02).Item += I02
UpdateObject(I02).Status = sold
```

```
Actions :
UpdateObject(I02).value = $5,000
```

Figure 6: A pictorial illustration of a full episode of OONP parsing, where we assume the description of cars (highlighted with shadow) are segmented in preprocessing.

$$\text{prob}(c, \text{new}|\mathcal{S}_t) = \frac{e^{\text{score}^{(c)}_{\text{new}}(\mathcal{O}^{c,\text{new}}_t, \hat{\mathcal{O}}_t; \theta^{(c)}_{\text{new}})}}{Z(t)} \quad (1)$$

$$\text{prob}(c, k|\mathcal{S}_t) = \frac{e^{\text{score}^{(c)}_{\text{assign}}(\mathcal{O}^{c,k}_t, \hat{\mathcal{O}}_t; \theta^{(c)}_{\text{assign}})}}{Z(t)} \quad (2)$$

$$\text{prob}(\text{none}|\mathcal{S}_t) = \frac{e^{\text{score}_{\text{none}}(\hat{\mathcal{O}}_t; \theta_{\text{none}})}}{Z(t)} \quad (3)$$

where $Z(t) = \sum_{c' \in \mathcal{C}} e^{\text{score}^{(c')}_{\text{new}}(\mathcal{O}^{c',\text{new}}_t, \hat{\mathcal{O}}_t; \theta^{(c')}_{\text{new}})} + \sum_{(c'',k') \in \text{idx}(\mathbf{M}^t_{\text{obj}})} e^{\text{score}^{(c'')}_{\text{assign}}(\mathcal{O}^{c'',k}_t, \hat{\mathcal{O}}_t; \theta^{(c'')}_{\text{assign}})} + e^{\text{score}_{\text{none}}(\hat{\mathcal{O}}_t; \theta_{\text{none}})}$ is the normalizing factor.

5.2 Updating Objects

In `Update.X` step, Policy-net needs to choose the property or external link (or none) to update for the selected object determined by `New-Assign` step. If `Update.X` chooses to update an external link, Policy-net needs to further determine which object it links to. After that, `Update2what` updates the chosen property or links. In task with static Ontology, most internal properties and links will be "locked" after they are updated for the first time, with some exception on a few semi-structured properties (e.g., the `Description` property in the experiment in Section 7.2). For dynamical Ontology, on the contrary, some properties and links are always subject to changes.

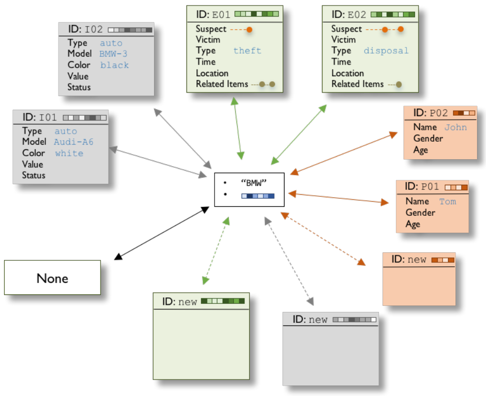


Figure 7: A pictorial illustration of what the Reader sees in determining whether to `New` an object and the relevant object when the read-head on Inline Memory reaches the last word in the text in Figure 2. The color of the arrow line stands for different matching functions for object classes, where the dashed lines are for the new object.

6 Learning

The parameters of OONP models (denoted Θ) include that for all operations and that for composing the distributed sections in Inline Memory. They can be trained with supervised learning (SL) , reinforcement learning (RL), and a hybrid of the two in different ways. With pure SL, the oracle gives the ground truth about the "right action" at each time step during the entire decision process, with which the parameter can be tuned to maximize the likelihood of the truth, with the following objective function

$$\mathcal{J}_{\text{SL}}(\Theta) = -\frac{1}{N}\sum_{i}^{N}\sum_{t=1}^{T_i}\log(\pi_t^{(i)}[a_t^\star]) \tag{4}$$

where N stands for the number of instances, T_i stands for the number of steps in decision process for the i^{th} instance, $\pi_t^{(i)}[\cdot]$ stands for the probabilities of the actions at t from the stochastic policy, and $a_t^\star$ stands for the ground truth action in step t.

With RL, the supervision is given as rewards during the decision process, for which an extreme case is to give the final reward at the end of the decision process by comparing the generated Ontology and the ground truth, e.g.,

$$r_t^{(i)} = \begin{cases} 0, & \text{if } t \neq T_i \\ \mathsf{match}(\mathsf{M}_{\text{obj}}^{T_i}, \mathcal{G}_i), & \text{if } t = T_i \end{cases} \tag{5}$$

where the $\mathsf{match}(\mathsf{M}_{\text{obj}}^{T_i}, \mathcal{G}_i)$ measures the consistency between the Ontology of in the Object Memory $\mathsf{M}_{\text{obj}}^{T_i}$ and the ground truth $\mathcal{G}^\star$. We can use policy search algorithm to maximize the expected total reward, e.g. the commonly used REINFORCE (Williams, 1992) for training, with the gradient

$$\nabla_\Theta \mathcal{J}_{\text{RL}}(\Theta) = -\mathbb{E}_{\pi_\theta}\left[\nabla_\Theta \log \pi_\Theta\left(a_t^i|s_t^i\right) r_{t:T}^{(i)}\right] \tag{6}$$

$$\approx -\frac{1}{NT_i}\sum_{i}^{N}\sum_{t=1}^{T}\nabla_\Theta \log \pi_\Theta\left(a_t^i|s_t^i\right) r_{t:T_i}^{(i)}. \tag{7}$$

When OONP is applied to real-world tasks, there is often quite natural supervision signals for both SL and RL. More specifically, for static Ontology one can infer some actions from the final ontology based on some basic assumption, e.g.,

- the system should `New` an object the first time it is mentioned;
- the system should put an extracted string (say, that for `Name`) into the right property of right object at the end of the string.

For those that can not be fully inferred, say the categorical properties of an object (e.g., `Type` for event objects), we have to resort to RL to determine the *time* of decision, while we also need SL to train Policy-net on the *content* of the decision. Fortunately it is quite straightforward to combine the two learning paradigms in optimization. More specifically, we maximize this combined objective

$$\mathcal{J}(\Theta) = \mathcal{J}_{\text{SL}}(\Theta) + \lambda\mathcal{J}_{\text{RL}}(\Theta), \tag{8}$$

where $\mathcal{J}_{\text{SL}}$ and $\mathcal{J}_{RL}$ are over the parameters within their own supervision mode and λ coordinates the weight of the two learning mode on the parameters they share. Equation (8) actually indicates a deep coupling of supervised learning and reinforcement learning, since for any episode the samples of actions related to RL might affect the inputs to the models under supervised learning.

For dynamical Ontology (see Section 7.1 for example), it is impossible to derive most of the decisions from the final Ontology since they may change over time. For those we have to rely mostly on the supervision at the time step to train the action (supervised mode) or count on OONP to learn the dynamics of the ontology evolution by fitting the final ground truth. Both scenarios are discussed in Section 7.1 on a synthetic task.

7 Experiments

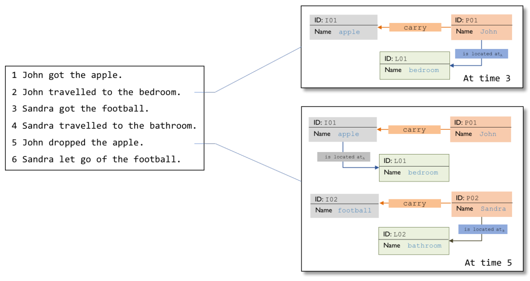
We applied OONP on three document parsing tasks, to verify its efficacy on parsing documents with different characteristics and investigate different components of OONP.

7.1 Task-I: bAbI Task

Data and Task

We implemented OONP on enriched version of bAbI tasks (Johnson, 2017) with intermediate representation for history of arbitrary length. In this experiment, we considered only the original bAbi task-2 (Weston et al., 2015), with an instance shown in the left panel Figure 8. The ontology has three types of objects: PERSON-object, ITEM-object, and LOCATION-object, and three types of links specifying relations between them (see Figure 8 for an illustration). All three types of objects have `Name` as the only internal property.

The task for OONP is to read an episode of story and recover the trajectory of the evolving ontology. We choose bAbI for its dynamical ontology that evolves with time and ground truth given for each snapshot. Comparing with the real-world tasks we will present later, bAbi has almost trivial internal properties but relatively rich opportunities for links, considering that any two objects of different types could potentially have a link.



```
1 John got the apple.
2 John travelled to the bedroom.
3 Sandra got the football.
4 Sandra travelled to the bathroom.
5 John dropped the apple.
6 Sandra let go of the football.
```

Figure 8: One instance of bAbI (6-sentence episode) and the ontology of two snapshots.

Action	Description
`NewObject`(c)	New an object of class-c.
`AssignObject`(c,k)	Assign the current information to existed object (c,k)
`Update`(c,k)`.AddLink`(c',k',ℓ)	Add an link of type-ℓ from object-(c,k) to object-(c',k')
`Update`(c,k)`.DelLink`(c',k',ℓ)	Delete the link of type-ℓ from object-(c,k) to object-(c',k')

Table 1: Actions for bAbI.

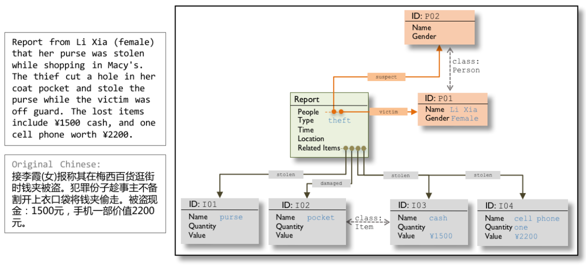
Implementation Details

For preprocessing, we have a trivial NER to find the names of people, items and locations (saved in the symbolic part of Inline Memory) and word-level bi-directional GRU for the distributed representations of Inline Memory. In the parsing process, Reader goes through the inline word-by-word in the temporal order of the original text, makes `New-Assign` action at every word, leaving `Update.X` and `Update2what` actions to the time steps when the read-head on Inline Memory reaches a punctuation (see more details of actions in Table 1). For this simple task, we use an almost fully neural Reader (with MLPs for Policy-net) and a vector for Matrix Memory, with however a Symbolic Reasoner to maintain the logical consistency after updating the relations with the actions (see Appendx.B for more details).

Results and Analysis

For training, we use 1,000 episodes with length evenly distributed from one to six. We use just REINFORCE with only the final reward defined as the overlap between the generated ontology and the ground truth, while step-by-step supervision on actions yields almost perfect result (result omitted). For evaluation, we use the F1 (Rijsbergen, 1979) between the generated links and the ground truth averaged over all snapshots of all test instances, since the links are sparse compared with all the possible pairwise relations between objects, with which we get F1= 94.80% without Symbolic Reasoner and F1= 95.30% with it.

Clearly OONP can learn fairly well on recovering the evolving ontology with such a small training set and weak supervision (RL with the final reward), showing that the credit assignment over to earlier snapshots does not cause much difficulty in the learning of OONP even with a generic policy search algorithm. It is not so surprising to observe that Symbolic Reasoner helps to improve the results on discovering the links, while it does not improve the performance on identifying the objects although it is taken within the learning.



```
Report from Li Xia (female)
that her purse was stolen
while shopping in Macy's.
The thief cut a hole in her
coat pocket and stole the
purse while the victim was
off guard. The lost items
include ¥1500 cash, and one
cell phone worth ¥2200.
```

Original Chinese:
接李霞(女)报称其在梅西百货逛街时钱夹被盗。犯罪份子趁事主不备割开上衣口袋将钱夹偷走。被盗现金：1500元，手机一部价值2200元。

Figure 9: Example of police report & its ontology.

7.2 Task-II: Parsing Police Report

Data & Task

We implement OONP for parsing Chinese police report (brief description of criminal cases written by policeman), as illustrated in the left panel of Figure 9. We consider a corpus of 5,500 cases with a variety of crime categories, including theft, robbery, drug dealing and others. Although the language is reasonably formal, the corpus covers a big variety of topics and language styles, and has a high proportion of typos. The ontology we designed for this task mainly consists of a number of PERSON-objects and ITEM-objects connected through an EVENT-object with several types of relations, as illustrated in the right panel of Figure 9. A PERSON-object has three internal properties: `Name` (string), `Gender` (categorical) and `Age` (number), and two types of external links (`suspect` and `victim`) to an EVENT-object. An ITEM-object has three internal properties: `Name` (string), `Quantity` (string) and `Value` (string), and six types of external links (`stolen`, `drug`, `robbed`, `swindled`, `damaged`, and `other`) to an EVENT-object. On average, a sample has 95.24 Chinese words and the ontology has 3.35 objects, 3.47 mentions and 5.02 relationships. Compared with bAbI in Section 7.1, the police report ontology has less pairwise links but much richer internal properties for objects of all three objects.

Implementation Details

The OONP model is to generate the ontology as illustrated in Figure 9 through a decision process with actions in Table 2. As pre-processing, we performed third party NER algorithm to find people names, locations, item etc. For the distributed part of Inline Memory, we used dilated CNN with different choices of depth and kernel size (Yu and Koltun, 2016), all of which will be jointly learned during training. In updating objects with its string-type properties (e.g., `Name` for a PERSON-object), we use Copy-Paste strategy for extracted string (whose NER tag already specifies which property in an object it goes to) as Reader sees it. For undetermined category properties in existed objects, Policy-net will determine the object to update (a `New-Assign` action without `New` option), its property to update (an `Update.X` action), and the updating operation (an `Update2what` action) at milestones of the decision process , e.g., when reaching an punctuation. For this task, since all the relations are between the single by-default EVENT-object and other objects, the relations can be reduced to category-type properties of the corresponding objects in practice. For category-type properties, we cannot recover `New-Assign` and `Update.X` actions from the label (the final ontology), so we resort RL for learning to determine that part, which is mixed with the supervised learning for `Update2what` and other actions for string-type properties.

Action	Description
`NewObject`(c)	New an object of class-c.
`AssignObject`(c,k)	Assign the current information to existed object (c,k)
`UpdateObject`(c,k)`.Name`	Set the name of object-(c,k) with the extracted string.
`UpdateObject`(PERSON,k)`.Gender`	Set the name of a PERSON-object indexed k with the extracted string.
`UpdateObject`(ITEM,k)`.Quantity`	Set the quantity of an ITEM-object indexed k with the extracted string.
`UpdateObject`(ITEM,k)`.Value`	Set the value of an ITEM-object indexed k with the extracted string.
`UpdateObject`(EVENT,1)`.Items.x`	Set the link between the EVENT-object and an ITEM-object, where `x` ∈{`stolen`, `drug`, `robbed`, `swindled`, `damaged`, `other`}
`UpdateObject`(EVENT,1)`.Persons.x`	Set the link between the EVENT-object and an PERSON-object, and `x` ∈{`victim`, `suspect`}

Table 2: Actions for parsing police report.

Results & Discussion

We use 4,250 cases for training, 750 for validation an held-out 750 for test. We consider the following four metrics in comparing the performance of different models:

Assignment Accuracy	the accuracy on `New-Assign` actions made by the model
Category Accuracy	the accuracy of predicting the category properties of all the objects
Ontology Accuracy	the proportion of instances for which the generated Objects is exactly the same as the ground truth
Ontology Accuracy-95	the proportion of instances for which the generated Objects achieves 95% consistency with the ground truth

which measures the accuracy of the model in making discrete decisions as well as generating the final ontology.

Model	Assign Acc. (%)	Type Acc. (%)	Ont. Acc. (%)	Ont. Acc-95 (%)
Bi-LSTM (baseline)	73.2 ± 0.58	-	36.4± 1.56	59.8 ± 0.83
ENTITYNLM (baseline)	87.6 ± 0.50	84.3 ± 0.80	59.6 ± 0.85	72.3 ± 1.37
OONP (neural)	88.5 ± 0.44	84.3 ± 0.58	61.4 ± 1.26	75.2 ± 1.35
OONP (structured)	91.2 ± 0.62	87.0 ± 0.40	65.4 ± 1.42	79.9 ± 1.28
OONP (RL)	**91.4** ± 0.38	**87.8** ± 0.75	**66.7** ± 0.95	**80.7** ± 0.82

Table 3: OONP on parsing police reports.

We empirically investigated two competing models, Bi-LSTM and EntityNLM , as baselines. Both models can be viewed as simplified versions of OONP. Bi-LSTM consists of a bi-directional LSTM as Inline Memory encoder and a two-layer MLP on top of that as Policy-net. Bi-LSTM does not support categorical prediction for objects due to the lack of explicit object representation, which will only be trained to perform `New-Assign` actions and evaluated on them (with the relevant metrics modified for it). EntityNLM, on the other hand, has some modest capability for modeling entities with the original purpose of predicting entity mentions (Ji et al., 2017) which has been adapted and re-implemented for this scenario. For OONP , we consider three variants:

- OONP (neural): simple version of OONP with only distributed representation in Reader;
- OONP (structured): OONP that considers the matching between two structured objects in `New-Assign` actions;
- OONP (RL): another version of OONP (structured) that uses RL[‡] to determine the time for predicting the category properties, while OONP (neural) and OONP (structured) use a rule-based approach to determine the time.

The experimental results are given in Table 3. As shown in Table 3, Bi-LSTM struggles to achieve around 73% Assignment Accuracy on test set, while OONP (neural) can boost the performance to 88.5%. Arguably, this difference in performance is due to the fact that Bi-LSTM lacks Object Memory, so all relevant information has to be stored in the Bi-LSTM hidden states along the reading process. When we start putting symbolic representation and operation into Reader, as shown in the result of OONP (structure), the performance is again significantly improved on all four metrics.

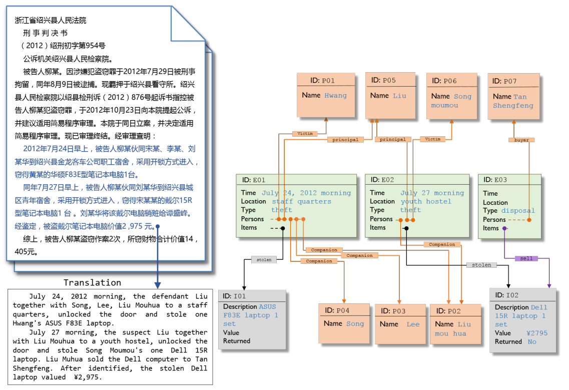
From the result of OONP (RL), RL improves not only the prediction of categorical property (and hence the overall ontology accuracy) but also tasks trained with purely SL (i.e., learning the `New-Assign` actions). This indicates there might be some deep entanglement between SL and RL through the obvious interaction between features in parsing and/or sharing of parameters.

7.3 Task-III: Parsing court judgment docs

Data and Task

Comparing with Task-II, court judgements are typically much longer, containing multiple events

[‡] A more detailed exposition of this idea can be found in (Liu et al., 2018), where RL is used for training a multi-label classifier of text


浙江省绍兴县人民法院
刑 事 判 决 书
（2012）绍刑初字第954号
公诉机关绍兴县人民检察院。
被告人柳某。因涉嫌犯盗窃罪于2012年7月29日被刑事拘留，同年8月9日被逮捕。现羁押于绍兴县看守所。绍兴县人民检察院以绍县检刑诉（2012）876号起诉书指控被告人柳某犯盗窃罪，于2012年10月23日向本院提起公诉，并建议适用简易程序审理。本院于同日立案，并决定适用简易程序审理。现已审理终结。经审理查明：
2012年7月24日早上，被告人柳某伙同宋某、李某、刘某华到绍兴县金龙客车公司职工宿舍，采用开锁方式进入，窃得黄某的华硕F83E型笔记本电脑1台。
同年7月27日早上，被告人柳某伙同刘某华到绍兴县城区青年宿舍，采用开锁方式进入，窃得宋某某的戴尔15R型笔记本电脑1 台 。刘某华将该戴尔电脑销赃给谭盛峰。经鉴定，被盗戴尔笔记本电脑价值2 ,975 元。
综上，被告人柳某盗窃作案2次，所窃财物合计价值14，405元。

Translation

July 24, 2012 morning, the defendant Liu together with Song, Lee, Liu Mouhua to a staff quarters, unlocked the door and stole one Hwang's ASUS F83E laptop.
July 27 morning, the suspect Liu together with Liu Mouhua to a youth hostel, unlocked the door and stole Song Moumou's one Dell 15R laptop. Liu Muhua sold the Dell computer to Tan Shengfeng. After identified, the stolen Dell laptop valued ¥2,975.

Figure 10: Left: the judgement document with highlighted part being the description the facts of crime; right: the corresponding ontology

of different types and large amount of irrelevant text. The dataset contains 4056 Chinese judgement documents, divided into training/dev/testing set 3256/400/400 respectively. The ontology for this task mainly consists of a number of PERSON-objects and ITEM-objects connected through a number EVENT-object with several types of links. An EVENT-object has three internal properties: `Time` (string), `Location` (string), and `Type` (category, ∈{`theft`, `restitution`, `disposal`}), four types of external links to PERSON-objects (namely, `principal`, `companion`, `buyer`, `victim`) and four types of external links to ITEM-objects (`stolen`, `damaged`, `restituted`, `disposed`). In addition to the external links to EVENT-objects , a PERSON-object has only the `Name` (string) as the internal property. An ITEM-object has three internal properties: `Description` (array of strings), `Value` (string) and `Returned`(binary) in addition to its external links to EVENT-objects , where `Description` consists of the words describing the corresponding item, which could come from multiple segments across the document. An object could be linked to more than one EVENT-object, for example a person could be the principal suspect in event A and also a companion in event B. An illustration of the judgement document and the corresponding ontology can be found in Figure 10.

Implementation Details

We use a model configuration similar to that in Section 7.2, with event-based segmentation of text given by third-party extraction algorithm (Yan et al., 2017) in Inline Memory, which enables OONP to trivially `New` EVENT-objectwith rules. OONP reads the Inline Memory, fills the EVENT-objects, creates and fills PERSON-objects and ITEM-objects, and specifies the links between them, with the actions summarized in Table 4. When an object is created during a certain event, it will be given an extra feature (not an internal property) indicating this connection, which will be used in deciding links between this object and event object, as well as in determining the future `New-Assign` actions.

Action for 2nd-round	Description
`NewObject`(c)	New an object of class-c.
`AssignObject`(c, k)	Assign the current information to existed object (c, k)
`UpdateObject`(PERSON, k)`.Name`	Set the name of the k^{th} PERSON-object with the extracted string.
`UpdateObject`(ITEM, k)`.Description`	Add to the description of an k^{th} ITEM-object with the extracted string.
`UpdateObject`(ITEM, k)`.Value`	Set the value of an k^{th} ITEM-object with the extracted string.
`UpdateObject`(EVENT, k)`.Time`	Set the time of an k^{th} EVENT-object with the extracted string.
`UpdateObject`(EVENT, k)`.Location`	Set the location of an k^{th} EVENT-object with the extracted string.
`UpdateObject`(EVENT, k)`.Type`	Set the type of the k^{th} EVENT-object among {`theft`, `disposal`, `restitution`}
`UpdateObject`(EVENT, k)`.Items.x`	Set the link between the k^{th} EVENT-object and an ITEM-object, where `x` ∈ {`stolen`, `damaged`, `restituted`, `disposed` }
`UpdateObject`(EVENT, k)`.Persons.x`	Set the link between the k^{th} EVENT-object and an PERSON-object, and `x` ∈ {`principal`, `companion`, `buyer`, `victim`}

Table 4: Actions for parsing court judgements.

Results and Analysis

We use the same metric as in Section 7.2, and compare two OONP variants, OONP (neural) and OONP (structured), with two baselines EntityNLM and Bi-LSTM. The two baselines will be tested only on the second-round reading, while both OONP variants are tested on a two-round reading. The results are shown in Table 5. OONP parsers attain accuracy significantly higher than Bi-LSTM. Among, OONP (structure) achieves over 71% accuracy on getting the entire ontology right and over 77% accuracy on getting 95% consistency with the ground truth. We omitted the RL results since the model RL model chooses to predict the type properties same as the simple rules.

Model	Assign Acc. (%)	Type Acc. (%)	Ont. Acc. (%)	Ont. Acc-95 (%)
Bi-LSTM (baseline)	84.66 ± 0.20	-	18.20 ± 0.74	36.88 ± 1.01
ENTITYNLM (baseline)	90.50 ± 0.21	96.33 ± 0.39	39.85 ± 0.20	48.29 ± 1.96
OONP (neural)	94.50 ± 0.24	97.73 ± 0.12	53.29 ± 0.26	72.22 ± 1.01
OONP (structured)	**96.90** ± 0.22	**98.80** ± 0.08	**71.11** ± 0.54	**77.27** ± 1.05

Table 5: OONP on judgement documents.

8 Conclusion

We proposed Object-oriented Neural Programming (OONP), a framework for semantically parsing in-domain documents. OONP is neural net-based, but equipped with sophisticated architecture and mechanism for document understanding, therefore nicely combining interpretability and learnability. Experiments on both synthetic and real-world datasets have shown that OONP outperforms several strong baselines by a large margin on parsing fairly complicated ontology.

Acknowledgments

We thank Fandong Meng and Hao Xiong for their insightful discussion. We also thank Classic Law Institute for providing the raw data.

References

Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific American* 284(5):34–43.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics(ACL)*. pages 1215–1226.

Fillmore Charles J. 1982. Frame semantics. *In Linguistics in the Morning Calm* pages 111–137.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)* .

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR* abs/1410.5401.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Henderson, Matthew, Blaise Thomson, , and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. pages 292–299.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics(ACL)*. pages 623–628.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP)*. Association for Computational Linguistics, pages 1830–1839.

Daniel D. Johnson. 2017. Learning graphical state transitions. In *the International Conference on Learning Representations(ICLR)*.

Berant Jonathan and Liang Percy. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Association for Computational Linguistics(ACL)*.

Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. 2018. Jumper: Learning when to make classification decisions in reading. In *IJCAI*.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR* abs/1502.05698.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Yukun Yan, Daqi Zheng, Zhengdong Lu, and Sen Song. 2017. Event identification as a decision process with non-linear representation of text. *CoRR* abs/1710.00969.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP)*. Association for Computational Linguistics, pages 1850–1859.

Fisher Yu and Vladlen Koltun. 2016. Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. *In Proceedings of COLING* .

Finding Syntax in Human Encephalography with Beam Search

John Hale[♠,△] Chris Dyer[♠] Adhiguna Kuncoro[♠,♣] Jonathan R. Brennan[♢]
[♠]DeepMind, London, UK
[♣]Department of Computer Science, University of Oxford
[♢]Department of Linguistics, University of Michigan
[△]Department of Linguistics, Cornell University
{jthale,cdyer,akuncoro}@google.com jobrenn@umich.edu

Abstract

Recurrent neural network grammars (RNNGs) are generative models of (tree, string) pairs that rely on neural networks to evaluate derivational choices. Parsing with them using beam search yields a variety of incremental complexity metrics such as word surprisal and parser action count. When used as regressors against human electrophysiological responses to naturalistic text, they derive two amplitude effects: an early peak and a P600-like later peak. By contrast, a non-syntactic neural language model yields no reliable effects. Model comparisons attribute the early peak to syntactic composition within the RNNG. This pattern of results recommends the RNNG+beam search combination as a mechanistic model of the syntactic processing that occurs during normal human language comprehension.

1 Introduction

Computational psycholinguistics has "always been...the thing that computational linguistics stood the greatest chance of providing to humanity" (Kay, 2005). Within this broad area, cognitively-plausible parsing models are of particular interest. They are mechanistic computational models that, at some level, do the same task people do in the course of ordinary language comprehension. As such, they offer a way to gain insight into the operation of the human sentence processing mechanism (for a review see Hale, 2017).

As Keller (2010) suggests, a promising place to look for such insights is at the intersection of (a) incremental processing, (b) broad coverage, and (c) neural signals from the human brain.

The contribution of the present paper is situated precisely at this intersection. It combines a probabilistic generative grammar (RNNG; Dyer et al., 2016) with a parsing procedure that uses this grammar to manage a collection of syntactic derivations as it advances from one word to the next (Stern et al., 2017, cf. Roark, 2004). Via well-known complexity metrics, the intermediate states of this procedure yield quantitative predictions about language comprehension difficulty. Juxtaposing these predictions against data from human encephalography (EEG), we find that they reliably derive several amplitude effects including the P600, which is known to be associated with syntactic processing (e.g. Osterhout and Holcomb, 1992).

Comparison with language models based on long short term memory networks (LSTM, e.g. Hochreiter and Schmidhuber, 1997; Mikolov, 2012; Graves, 2012) shows that these effects are specific to the RNNG. A further analysis pinpoints one of these effects to RNNGs' syntactic composition mechanism. These positive findings reframe earlier null results regarding the syntax-sensitivity of human processing (Frank et al., 2015). They extend work with eyetracking (e.g. Roark et al., 2009; Demberg et al., 2013) and neuroimaging (Brennan et al., 2016; Bachrach, 2008) to higher temporal resolution.[1] Perhaps most significantly, they establish a general correspondence between a computational model and electrophysiological responses to naturalistic language.

Following this Introduction, section 2 presents recurrent neural network grammars, emphasizing their suitability for incremental parsing. Sections 3 then reviews a previously-proposed

[1]Magnetoencephalography also offers high temporal resolution and as such this work fits into a tradition that includes Wehbe et al. (2014), van Schijndel et al. (2015), Wingfield et al. (2017) and Brennan and Pylkkänen (2017).

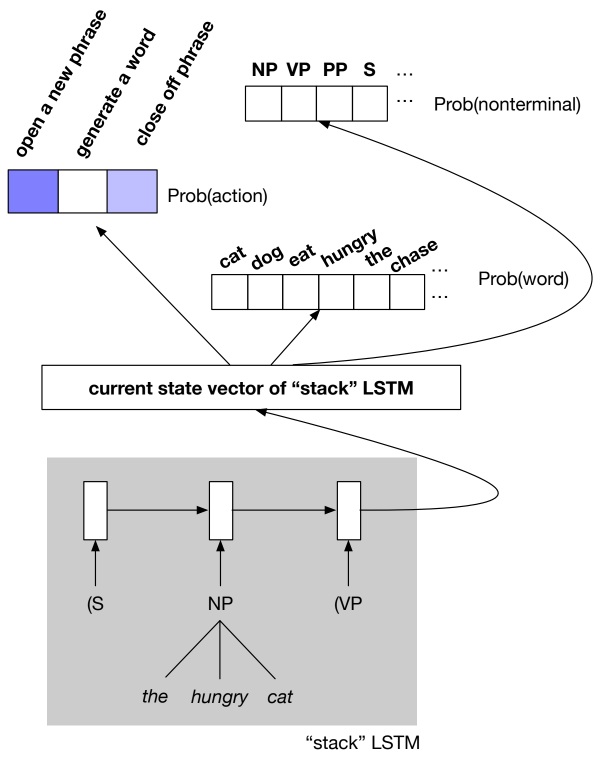


Figure 1: Recurrent neural network grammar configuration used in this paper. The absence of a lookahead buffer is significant, because it forces parsing to be incremental. Completed constituents such as [$_{\mathrm{NP}}$ the hungry cat] are represented on the stack by numerical vectors that are the output of the syntactic composition function depicted in Figure 2.

beam search procedure for them. Section 4 goes on to introduce the novel application of this procedure to human data via incremental complexity metrics. Section 5 explains how these theoretical predictions are specifically brought to bear on EEG data using regression. Sections 6 and 7 elaborate on the model comparison mentioned above and report the results in a way that isolates the operative element. Section 8 discusses these results in relation to established computational models. The conclusion, to anticipate section 9, is that syntactic processing can be found in naturalistic speech stimuli if ambiguity resolution is modeled as beam search.

2 Recurrent neural network grammars for incremental processing

Recurrent neural network grammars (henceforth: RNNGs Kuncoro et al., 2017; Dyer et al.,

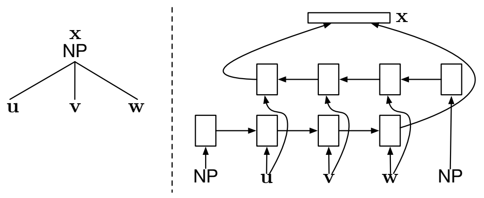


Figure 2: RNNG composition function traverses daughter embeddings $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, representing the entire tree with a single vector $\mathbf{x}$. This Figure is reproduced from (Dyer et al., 2016).

2016) are probabilistic models that generate trees. The probability of a tree is decomposed via the chain rule in terms of derivational action-probabilities that are conditioned upon previous actions i.e. they are history-based grammars (Black et al., 1993). In the vanilla version of RNNG, these steps follow a depth-first traversal of the developing phrase structure tree. This entails that daughters are announced bottom-up one by one as they are completed, rather than being predicted at the same time as the mother.

Each step of this generative story depends on the state of a stack, depicted inside the gray box in Figure 1. This stack is "neuralized" such that each stack entry corresponds to a numerical vector. At each stage of derivation, a single vector summarizing the entire stack is available in the form of the final state of a neural sequence model. This is implemented using the stack LSTMs of Dyer et al. (2015). These stack-summary vectors (central rectangle in Figure 1) allow RNNGs to be sensitive to aspects of the left context that would be masked by independence assumptions in a probabilistic context-free grammar. In the present paper, these stack-summaries serve as input to a multi-layer perceptron whose output is converted via softmax into a categorical distribution over three possible parser actions: open a new constituent, close off the latest constituent, or generate a word. A hard decision is made, and if the first or last option is selected, then the same vector-valued stack–summary is again used, via multilayer perceptrons, to decide which specific nonterminal to open, or which specific word to generate.

Phrase-closing actions trigger a syntactic composition function (depicted in Figure 2) which squeezes a sequence of subtree vectors into one single vector. This happens by applying a bidirectional LSTM to the list of daughter vectors, prepended with the vector for the mother category following §4.1 of Dyer et al. (2016).

The parameters of all these components are adaptively adjusted using backpropagation at training time, minimizing the cross entropy relative to a corpus of trees. At testing time, we parse incrementally using beam search as described below in section 3.

3 Word-synchronous beam search

Beam search is one way of addressing the search problem that arises with generative grammars — constructive accounts of language that are sometimes said to "strongly generate" sentences. Strong generation in this sense simply means that they derive both an observable word-string as well as a hidden tree structure. Probabilistic grammars are joint models of these two aspects. By contrast, parsers are programs intended to infer a good tree from a given word-string. In incremental parsing with history-based models this inference task is particularly challenging, because a decision that looks wise at one point may end up looking foolish in light of future words. Beam search addresses this challenge by retaining a collection called the "beam" of parser states at each word. These states are rated by a score that is related to the probability of a partial derivation, allowing an incremental parser to hedge its bets against temporary ambiguity. If the score of one analysis suddenly plummets after seeing some word, there may still be others within the beam that are not so drastically affected. This idea of ranked parallelism has become central in psycholinguistic modeling (see e.g. Gibson, 1991; Narayanan and Jurafsky, 1998; Boston et al., 2011).

As Stern et al. (2017) observe, the most straightforward application of beam search to generative models like RNNG does not perform well. This is because lexical actions, which advance the analysis onwards to successive words, are assigned such low probabilities compared to structural actions which do not advance to the next word. This imbalance is inevitable in a probability model that strongly generates sentences, and it causes naive beam-searchers to get bogged down, proposing more and more phrase structure rather than moving on through the sentence. To address it, Stern et al. (2017) propose a word-synchronous variant of beam search. This variant keeps searching through structural actions until "enough" high-scoring parser states finally take a lexical action, arriving in synchrony at the next word of the sentence. Their procedure is written out as Algorithm 1.

Algorithm 1 Word-synchronous beam search with fast-tracking. After Stern et al. (2017)

```
1:  thisword ← input beam
2:  nextword ← ∅
3:  while |nextword| < k do
4:      fringe ← successors of all states
                 s ∈ thisword via any
                 parsing action
5:      prune fringe to top k
6:      thisword ← ∅
7:      for each parser state s ∈ fringe do
8:          if s came via a lexical action then
9:              add s to nextword
10:         else            ▷ must have been structural
11:             add s to thisword
12:         end if
13:     end for
14: end while
15: return nextword pruned to top $k_{\text{word}} \ll k$
```

In Algorithm 1 the beam is held in a set-valued variable called *nextword*. Beam search continues until this set's cardinality exceeds the designated action beam size, k. If the beam still isn't large enough (line 3) then the search process explores one more action by going around the while-loop again. Each time through the loop, lexical actions compete against structural actions for a place among the top k (line 5). The imbalance mentioned above makes this competition fierce, and on many loop iterations *nextword* may not grow by much. Once there are enough parser states, another threshold called the word beam k_{word} kicks in (line 15). This other threshold sets the number of analyses that are handed off to the next invocation of the algorithm. In the study reported here the word beam remains at the default setting suggested by Stern and colleagues, $k/10$.

Stern et al. (2017) go on to offer a modification of the basic procedure called "fast tracking" which improves performance, particularly when the action beam k is small. Under fast tracking, an additional step is added between lines 4 and 5 of

	k=100	k=200	k=400	k=600	k=800	k=1000	k=2000
Fried et al. (2017) RNNG ppl unknown, −fast track	74.1	80.1	85.3	87.5	88.7	89.6	not reported
this paper ppl=141, −fast track	71.5	78.81	84.15	86.42	87.34	88.16	89.81
this paper ppl=141, $k_{ft} = k/100$	87.1	88.96	90.48	90.64	90.84	90.96	91.25

Table 1: Penn Treebank development section bracketing accuracies (F1) under Word-Synchronous beam search. These figures show that an incremental parser for RNNG can perform well on a standard benchmark. "ppl" indicates the perplexity of over both trees and strings for the trained model on the development set, averaged over words In all cases the word beam is set to a tenth of the action beam, i.e. $k_{\text{word}} = k/10$.

Algorithm 1 such that some small number k_{ft} of parser states are promoted directly into *nextword*. These states are required to come via a lexical action, but in the absence of fast tracking they quite possibly would have failed the thresholding step in line 5.

Table 1 shows Penn Treebank accuracies for this word-synchronous beam search procedure, as applied to RNNG. As expected, accuracy goes up as the parser considers more and more analyses. Above $k = 200$, the RNNG+beam search combination outperforms a conditional model based on greedy decoding (88.9). This demonstration re-emphasizes the point, made by Brants and Crocker (2000) among others, that cognitively-plausible incremental processing can be achieved without loss of parsing performance.

4 Complexity metrics

In order to relate computational models to measured human responses, some sort of auxiliary hypothesis or linking rule is required. In the domain of language, these are traditionally referred to as complexity metrics because of the way they quantify the "processing complexity" of particular sentences. When a metric offers a prediction on each successive word, it is an *incremental* complexity metric.

Table 2 characterizes four incremental complexity metrics that are all obtained from intermediate states of Algorithm 1. The metric denoted DISTANCE is the most classic; it is inspired by the count of "transitions made or attempted" in Kaplan (1972). It quantifies syntactic work by counting the number of parser actions explored by Algorithm 1 between each word i.e. the number of times around the while-loop on line 3. The information theoretical quantities SURPRISAL and ENTROPY came into more widespread use later. They quantify unexpectedness and uncertainty, respectively, about alternative syntactic analyses at a given point within a sentence. Hale (2016) reviews their applicability across many different languages, psycholinguistic measurement techniques and grammatical models. Recent work proposes possible relationships between these two metrics, at the empirical as well as theoretical level (van Schijndel and Schuler, 2017; Cho et al., 2018).

metric	characterization
DISTANCE	count of actions required to synchronize k analyses at the next word
SURPRISAL	log-ratio of summed forward probabilities for analyses in the beam
ENTROPY	average uncertainty of analyses in the beam
ENTROPY Δ	difference between previous and current entropy value

Table 2: Complexity Metrics

The SURPRISAL metric was computed over the word beam i.e. the k_{word} highest-scoring partial syntactic analyses at each successive word. In an attempt to obtain a more faithful estimate, ENTROPY and its first-difference are computed over *nextword* itself, whose size varies but is typically much larger than k_{word}.

5 Regression models of naturalistic EEG

Electroencephalography (EEG) is an experimental technique that measures very small voltage fluctuations on the scalp. For a review emphasizing its implications vis-á-vis computational models, see Murphy et al. (2018).

We analyzed EEG recordings from 33 participants as they passively listened to a spoken recitation of the first chapter of Alice's Adventures in Wonderland.[2] This auditory stimulus was delivered via earphones in an isolated booth. All participants scored significantly better than chance on a post-session 8-question comprehension quiz. An additional ten datasets were excluded for not meeting this behavioral criterion, six due to excessive noise, and three due to experimenter error. All participants provided written informed consent under the oversight of the University of Michigan HSBS Institutional Review Board (#HUM00081060) and were compensated \$15/h.[3]

Data were recorded at 500 Hz from 61 active electrodes (impedences < 25 kΩ) and divided into 2129 epochs, spanning -0.3–1 s around the onset of each word in the story. Ocular artifacts were removed using ICA, and remaining epochs with excessive noise were excluded. The data were filtered from 0.5–40 Hz, baseline corrected against a 100 ms pre-word interval, and separated into epochs for content words and epochs for function words because of interactions between parsing variables of interest and word-class (Roark et al., 2009).

Linear regression was used per-participant, at each time-point and electrode, to identify content-word EEG amplitudes that correlate with complexity metrics derived from the RNNG+beam search combination via the complexity metrics in Table 2. We refer to these time series as Target predictors.

Each Target predictor was included in its own model, along with several Control predictors that are known to influence sentence processing: sentence order, word-order in sentence, log word frequency (Lund and Burgess, 1996), frequency of the previous and subsequent word, and acoustic sound power averaged over the first 50 ms of the epoch.

All predictors were mean-centered. We also constructed null regression models in which the rows of the design matrix were randomly permuted.[4] β coefficients for each effect were tested against these null models at the group level across all electrodes from 0–1 seconds post-onset, using a non-parametric cluster-based permutation test to correct for multiple comparisons across electrodes and time-points (Maris and Oostenveld, 2007).

6 Language models for literary stimuli

We compare the fit against EEG data for models that are trained on the same amount of textual data but differ in the explicitness of their syntactic representations.

At the low end of this scale is the LSTM language model. Models of this type treat sentences as a sequence of words, leaving it up to backpropagation to decide whether or not to encode syntactic properties in a learned history vector (Linzen et al., 2016). We use SURPRISAL from the LSTM as a baseline.

RNNGs are higher on this scale because they explicitly build a phrase structure tree using a symbolic stack. We consider as well a degraded version, RNNG$_{-\mathrm{comp}}$ which lacks the composition mechanism shown in Figure 2. This degraded version replaces the stack with initial substrings of bracket expressions, following Choe and Charniak (2016); Vinyals et al. (2015). An example would be the length 7 string shown below

$$\text{(S} \mid \text{(NP} \mid \text{the} \mid \text{hungry} \mid \text{cat} \mid)_{NP} \mid \text{(VP}$$

Here, vertical lines separate symbols whose vector encoding would be considered separately by RNNG$_{-\mathrm{comp}}$. In this degraded representation, the noun phrase is not composed explicitly. It takes up five symbols rather than one. The balanced parentheses (NP and)$_{\mathrm{NP}}$ are rather like instructions for some subsequent agent who might later perform the kind of syntactic composition that occurs online in RNNGs, albeit in an implicit manner.

In all cases, these language models were trained on chapters 2–12 of Alice's Adventures in Wonderland. This comprises 24941 words. The stimulus that participants saw during EEG data collection, for which the metrics in Table 2 are calculated, was chapter 1 of the same book, comprising 2169 words.

RNNGs were trained to match the output trees provided by the Stanford parser (Klein and Manning, 2003). These trees conform to the Penn Treebank annotation standard but do not explicitly mark long-distance dependency or include any empty categories. They seem to adequately represent basic syntactic properties such

[2] `https://tinyurl.com/alicedata`

[3] A separate analysis of these data appears in Brennan and Hale (2018); datasets are available from JRB.

[4] Temporal auto-correlation across epochs could impact model fits. Content-words are spaced 1 s apart on average and a spot-check of the residuals from these linear models indicates that they do not show temporal auto-correlation: AR(1) < 0.1 across subjects, time-points, and electrodes.

as clausal embedding and direct objecthood; nevertheless we did not undertake any manual correction.

During RNNG training, the first chapter was used as a development set, proceeding until the per-word perplexity over all parser actions on this set reached a minimum, 180. This performance was obtained with a RNNG whose state vector was 170 units wide. The corresponding LSTM language model state vector had 256 units; it reached a per-word perplexity of 90.2. Of course the RNNG estimates the joint probability of both trees *and* words, so these two perplexity levels are not directly comparable. Hyperparameter settings were determined by grid search in a region near the one which yielded good performance on the Penn Treebank benchmark reported on Table 1.

7 Results

To explore the suitability of the RNNG + beam search combination as a cognitive model of language processing difficulty, we fitted regression models as described above in section 5 for each of the metrics in Table 2. We considered six beam sizes $k = \{100, 200, 400, 600, 800, 1000\}$. Table 3 summarizes statistical significance levels reached by these Target predictors; no other combinations reached statistical significance.

LSTM		not significant
SURPRISAL	$k = 100$	$p_{cluster} = 0.027$
DISTANCE	$k = 200$	$p_{cluster} = 0.012$
SURPRISAL	$k = 200$	$p_{cluster} = 0.003$
DISTANCE	$k = 400$	$p_{cluster} = 0.002$
SURPRISAL	$k = 400$	$p_{cluster} = 0.049$
ENTROPY Δ	$k = 400$	$p_{cluster} = 0.026$
DISTANCE	$k = 600$	$p_{cluster} = 0.012$
ENTROPY	$k = 600$	$p_{cluster} = 0.014$

Table 3: Statistical significance of fitted Target predictors in Whole-Head analysis. $p_{cluster}$ values are minima for each Target with respect to a Monte Carlo cluster-based permutation test (Maris and Oostenveld, 2007).

7.1 Whole-Head analysis

Surprisal from the LSTM sequence model did not reliably predict EEG amplitude at any timepoint or electrode. The DISTANCE predictor did derive a central positivity around 600 ms post-word onset as shown in Figure 3a. SURPRISAL predicted an early frontal positivity around 250 ms, shown in Figure 3b. ENTROPY and ENTROPY Δ seemed to drive effects that were similarly early and frontal, although negative-going (not depicted); the effect for ENTROPY Δ localized to just the left side.

7.2 Region of Interest analysis

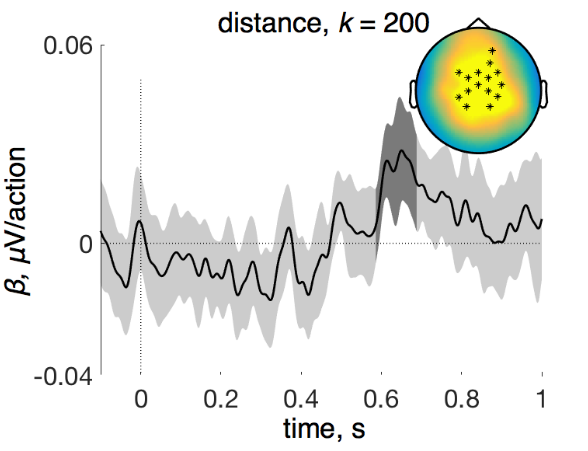
We compared RNNG to its degraded cousin, RNNG$_{-\text{comp}}$, in three regions of interest shown in Figure 4. These regions are defined by a selection of electrodes and a time window whose zero-point corresponds to the onset of the spoken word in the naturalistic speech stimulus. Regions "N400" and "P600" are well-known in EEG research, while "ANT" is motivated by findings with a PCFG baseline reported by Brennan and Hale (2018).

Single-trial data were averaged across electrodes and time-points within each region and fit with a linear mixed-effects model with fixed effects as described below and random intercepts by-subjects (Alday et al., 2017). We used a stepwise likelihood-ratio test to evaluate whether individual Target predictors from the RNNG significantly improved over RNNG$_{-\text{comp}}$, and whether a RNNG$_{-\text{comp}}$ model significantly improve a baseline regression model. The baseline regression model, denoted $\emptyset$, contains the Control predictors described in section 5 and SURPRISAL from the LSTM sequence model. Targets represent each of the eight reliable whole-head effects detailed in Table 3. These 24 tests (eight effects by three regions) motivate a Bonferroni correction of $\alpha = 0.002 = 0.05/24$.

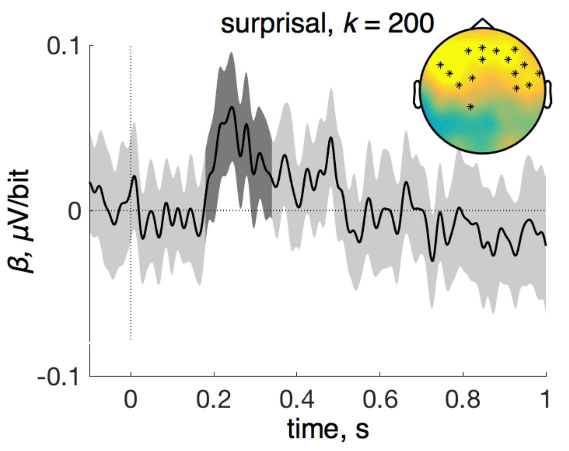
Statistically significant results obtained for DISTANCE from RNNG$_{-\text{comp}}$ in the P600 region and for SURPRISAL for RNNG in the ANT region. No significant results were observed in the N400 region. These results are detailed in Table 4.

8 Discussion

Since beam search explores analyses in descending order of probability, DISTANCE and SURPRISAL ought to be yoked, and indeed they are correlated at $r = 0.33$ or greater across all of the beam sizes k that we considered in this study. However they are reliably associated with different EEG effects. SURPRISAL manifests at anterior electrodes relatively early. This seems to be a different effect from that observed by Frank et al. (2015). Frank and colleagues relate N400 ampli-



(a) DISTANCE derives a P600 at $k = 200$.



(b) SURPRISAL derives an early response at $k = 200$.

Figure 3: Plotted values are fitted regression coefficients and 95% confidence intervals, statistically significant in the dark-shaded region with respect to a permutation test following Maris and Oostenveld (2007). The zero point represents the onset of a spoken word. Insets show electrodes with significant effects along with grand-averaged coefficient values across the significant time intervals. The diagram averages over all content words in the first chapter of Alice's Adventures in Wonderland.

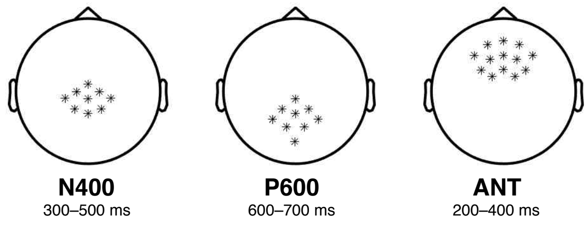


Figure 4: Regions of interest. The first region on the left, named "N400", comprises central-posterior electrodes during a time window 300–500 ms post-onset. The middle region, "P600" includes posterior electrodes 600–700 ms post-onset. The rightmost region "ANT" consists of just anterior electrodes 200-400 ms post-onset.

tude to word surprisals from an Elman-net, analogous to the LSTM sequence model evaluated in this work. Their study found no effects of syntax-based predictors over and above sequential ones. In particular, no effects emerged in the 500–700 ms window, where one might have expected a P600. The present results, by contrast, show that an explicitly syntactic model can derive the P600 quite generally via DISTANCE. The absence of an N400 effect in this analysis could be attributable to the choice of electrodes, or perhaps the modality of the stimulus narrative, i.e. spoken versus read.

The model comparisons in Table 4 indicate that the early peak, but not the later one, is attributable to the RNNG's composition function. Choe and Charniak's (2016) "parsing as language modeling" scheme potentially could explain the P600-like wave, but it would not account for the earlier peak. This earlier peak is the one derived by the RNNG under SURPRISAL, but only when the RNNG includes the composition mechanism depicted in Figure 2.

This pattern of results suggests an approach to the overall modeling task. In this approach, both grammar and processing strategy remain the same, and alternative complexity metrics, such as SURPRISAL and DISTANCE, serve to interpret the unified model at different times or places within the brain. This inverts the approach of Brouwer et al. (2017) and Wehbe et al. (2014) who interpret different *layers* of the same neural net using the same complexity metric.

9 Conclusion

Recurrent neural net grammars indeed learn something about natural language syntax, and what they learn corresponds to indices of human language processing difficulty that are manifested in electroencephalography. This correspondence, between computational model and human electrophysiological response, follows from a system that lacks an initial stage of purely string-based processing. Previous work was "two-stage" in the sense that the generative model served to

	RNNG$_{-\text{comp}} > \emptyset$			RNNG $>$ RNNG$_{-\text{comp}}$		
	χ^2	df	p	χ^2	df	p
DISTANCE, "P600" region						
$k = 200$	13.409	1	**0.00025**	4.198	1	0.04047
$k = 400$	15.842	1	**<0.0001**	3.853	1	0.04966
$k = 600$	13.955	1	**0.00019**	3.371	1	0.06635
SURPRISAL, "ANT" region						
$k = 100$	3.671	1	0.05537	13.167	1	**0.00028**
$k = 200$	3.993	1	0.04570	10.860	1	**0.00098**
$k = 400$	3.902	1	0.04824	10.189	1	**0.00141**
ENTROPY Δ, "ANT" region						
$k = 400$	10.141	1	**0.00145**	5.273	1	0.02165

Table 4: Likelihood-ratio tests indicate that regression models with predictors derived from RNNGs with syntactic composition (see Figure 2) do a better job than their degraded counterparts in accounting for the early peak in region "ANT" (right-hand columns). Similar comparisons in the "P600" region show that the model improves, but the improvement does not reach the $\alpha = 0.002$ significance threshold imposed by our Bonferroni correction (bold-faced text). RNNGs lacking syntactic composition do improve over a baseline model ($\emptyset$) containing lexical predictors and an LSTM baseline (left-hand columns).

rerank proposals from a conditional model (Dyer et al., 2016). If this one-stage model is cognitively plausible, then its simplicity undercuts arguments for string-based perceptual strategies such as the Noun-Verb-Noun heuristic (for a textbook presentation see Townsend and Bever, 2001). Perhaps, as Phillips (2013) suggests, these are unnecessary in an adequate cognitive model. Certainly, the road is now open for more fine-grained investigations of the order and timing of individual parsing operations within the human sentence processing mechanism.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. 1607441 and No. 1607251. We thank Max Cantor and Rachel Eby for helping with data collection.

References

Phillip M. Alday, Matthias Schlesewsky, and Ina Bornkessel-Schlesewsky. 2017. Electrophysiology reveals the neural dynamics of naturalistic auditory language processing: event-related potentials reflect continuous model updates. *eNeuro*, 4(6).

Asaf Bachrach. 2008. *Imaging neural correlates of syntactic complexity in a naturalistic context*. Ph.D. thesis, MIT.

Ezra Black, Fred Jelinek, John Lafrerty, David M. Magerman, Robert Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *31st Annual Meeting of the Association for Computational Linguistics*.

Marisa Ferrara Boston, John T. Hale, Shravan Vasishth, and Reinhold Kliegl. 2011. Parallel processing and sentence comprehension difficulty. *Language and Cognitive Processes*, 26(3):301–349.

Thorsten Brants and Matthew Crocker. 2000. Probabilistic parsing and psychological plausibility. In *Proceedings of 18th International Conference on Computational Linguistics COLING-2000*, Saarbrücken/Luxembourg/Nancy.

Jonathan R. Brennan and John T. Hale. 2018. Hierarchical structure guides rapid linguistic predictions during naturalistic listening. Forthcoming.

Jonathan R. Brennan and Liina Pylkkänen. 2017. MEG evidence for incremental sentence composition in the anterior temporal lobe. *Cognitive Science*, 41(S6):1515–1531.

Jonathan R. Brennan, Edward P. Stabler, Sarah E. Van Wagenen, Wen-Ming Luh, and John T. Hale. 2016. Abstract linguistic structure correlates with temporal activity during naturalistic comprehension. *Brain and Language*, 157-158:81–94.

Harm Brouwer, Matthew W. Crocker, Noortje J. Venhuizen, and John C. J. Hoeks. 2017. A neurocomputational model of the N400 and the P600 in language processing. *Cognitive Science*, 41:1318–1352.

Pyeong Whan Cho, Matthew Goldrick, Richard L. Lewis, and Paul Smolensky. 2018. Dynamic encoding of structural uncertainty in gradient symbols. In

Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018), pages 19–28.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas.

Vera Demberg, Frank Keller, and Alexander Koller. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar. *Computational Linguistics*, 39(4):1025–1066.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California.

Stefan L. Frank, Leun J. Otten, Giulia Galli, and Gabriella Vigliocco. 2015. The ERP response to the amount of information conveyed by words in sentences. *Brain and Language*, 140:1–11.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166, Vancouver, Canada.

Edward Gibson. 1991. *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown*. Ph.D. thesis, Carnegie Mellon University.

Alex Graves. 2012. *Supervised sequence labelling with recurrent neural networks*. Springer.

John Hale. 2016. Information-theoretical complexity metrics. *Language and Linguistics Compass*, 10(9):397–412.

John Hale. 2017. Models of human sentence comprehension in computational psycholinguistics. In Mark Aronoff, editor, *Oxford Research Encyclopedia of Linguistics*. Oxford University Press.

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

Ronald M. Kaplan. 1972. Augmented transition networks as psychological models of sentence comprehension. *Artificial Intelligence*, 3:77–100.

Martin Kay. 2005. ACL lifetime achievement award: A life of language. *Computational Linguistics*, 31(4).

Frank Keller. 2010. Cognitively plausible models of human language processing. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 60–67.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Eric Maris and Robert Oostenveld. 2007. Nonparametric statistical testing of EEG- and MEG-data. *Journal of Neuroscience Methods*, 164(1):177–190.

Tomáš Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Brian Murphy, Leila Wehbe, and Alona Fyshe. 2018. Decoding language from the brain. In Thierry Poibeau and Aline Editors Villavicencio, editors, *Language, Cognition, and Computational Models*, pages 53–80. Cambridge University Press.

Srini Narayanan and Daniel Jurafsky. 1998. Bayesian models of human sentence processing. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society*, University of Wisconsin-Madson.

Lee Osterhout and Phillip J. Holcomb. 1992. Event-related brain potentials elicited by syntactic anomaly. *Journal of Memory and Language*, 31:785–806.

Colin Phillips. 2013. Parser & grammar relations: We don't understand everything twice. In Montserrat Sanz, Itziar Laka, and Michael K. Tanenhaus, editors, *Language Down the Garden Path: The Cognitive and Biological Basis of Linguistic Structures*, chapter 16, pages 294–315. Oxford University Press.

Brian Roark. 2004. Robust garden path parsing. *Natural Language Engineering*, 10(1):1–24.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 324–333, Singapore.

Marten van Schijndel, Brian Murphy, and William Schuler. 2015. Evidence of syntactic working memory usage in MEG data. In *Proceedings of CMCL 2015*, Denver, Colorado, USA.

Marten van Schijndel and William Schuler. 2017. Approximations of predictive entropy correlate with reading times. In *Proceedings of CogSci 2017*, London, UK. Cognitive Science Society.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700, Copenhagen, Denmark.

David J. Townsend and Thomas G. Bever. 2001. *Sentence comprehension : the integration of habits and rules*. MIT Press.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. 2014. Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243, Doha, Qatar.

Cai Wingfield, Li Su, Xunying Liu, Chao Zhang, Phil Woodland, Andrew Thwaites, Elisabeth Fonteneau, and William D. Marslen-Wilson. 2017. Relating dynamic brain states to dynamic machine states: Human and machine solutions to the speech recognition problem. *PLOS Computational Biology*, 13(9):1–25.

Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information

Sudha Rao
University of Maryland, College Park
raosudha@cs.umd.edu

Hal Daumé III
University of Maryland, College Park
Microsoft Research, New York City
hal@cs.umd.edu

Abstract

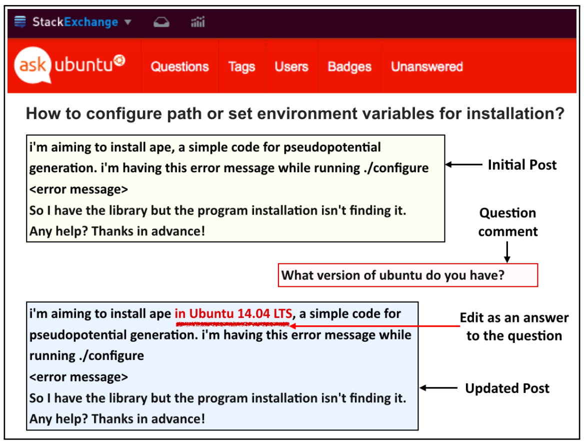
Inquiry is fundamental to communication, and machines cannot effectively collaborate with humans unless they can ask questions. In this work, we build a neural network model for the task of ranking clarification questions. Our model is inspired by the idea of expected value of perfect information: a good question is one whose expected answer will be useful. We study this problem using data from StackExchange, a plentiful online resource in which people routinely ask clarifying questions to posts so that they can better offer assistance to the original poster. We create a dataset of clarification questions consisting of ∼77K posts paired with a clarification question (and answer) from three domains of StackExchange: `askubuntu`, `unix` and `superuser`. We evaluate our model on 500 samples of this dataset against expert human judgments and demonstrate significant improvements over controlled baselines.

1 Introduction

A principle goal of asking questions is to fill information gaps, typically through clarification questions.[1] We take the perspective that a good question is the one whose *likely answer* will be useful. Consider the exchange in Figure 1, in which an initial poster (who we call "Terry") asks for help configuring environment variables. This post is underspecified and a responder ("Parker") asks a clarifying question (a) below, but could alternatively have asked (b) or (c):

(a) What version of Ubuntu do you have?

[1]We define 'clarification question' as a question that asks for some information that is currently missing from the given context.



How to configure path or set environment variables for installation?

i'm aiming to install ape, a simple code for pseudopotential generation. i'm having this error message while running ./configure
<error message>
So I have the library but the program installation isn't finding it.
Any help? Thanks in advance!

What version of ubuntu do you have?

i'm aiming to install ape in Ubuntu 14.04 LTS, a simple code for pseudopotential generation. i'm having this error message while running ./configure
<error message>
So I have the library but the program installation isn't finding it.
Any help? Thanks in advance!

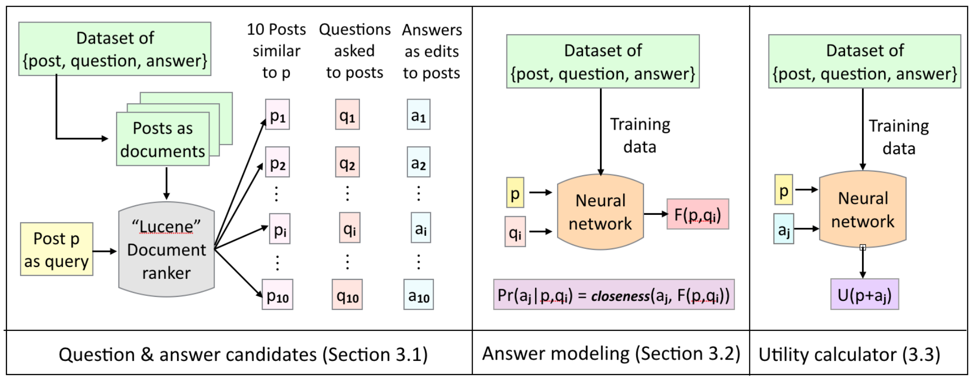
Figure 1: A post on an online Q & A forum "askubuntu.com" is updated to fill the missing information pointed out by the question comment.

(b) What is the make of your wifi card?

(c) Are you running Ubuntu 14.10 kernel 4.4.0-59-generic on an x86_64 architecture?

Parker should not ask (b) because an answer is unlikely to be useful; they should not ask (c) because it is too specific and an answer like "No" or "I do not know" gives little help. Parker's question (a) is much better: it is both likely to be useful, and is plausibly answerable by Terry.

In this work, we design a model to rank a candidate set of clarification questions by their usefulness to the given post. We imagine a use case (more discussion in § 7) in which, while Terry is writing their post, a system suggests a shortlist of questions asking for information that it thinks people like Parker might need to provide a solution, thus enabling Terry to immediately clarify their post, potentially leading to a much quicker resolution. Our model is based on the decision theoretic framework of the Expected Value of Perfect Information (EVPI) (Avriel and Williams, 1970), a measure of the value of gathering additional information. In our setting, we use EVPI to calculate which questions are most likely to elicit an answer that would make the post more informative.



Dataset of {post, question, answer}
Posts as documents
Post p as query
"Lucene" Document ranker
10 Posts similar to p
Questions asked to posts
Answers as edits to posts
p1
p2
pi
p10
q1
q2
qi
q10
a1
a2
ai
a10
Question & answer candidates (Section 3.1)
Dataset of {post, question, answer}
Training data
p
qi
Neural network
F(p,qi)
Pr(aj|p,qi) = closeness(aj, F(p,qi))
Answer modeling (Section 3.2)
Dataset of {post, question, answer}
Training data
p
aj
Neural network
U(p+aj)
Utility calculator (3.3)

Figure 2: The behavior of our model during test time: Given a post p, we retrieve 10 posts similar to post p using Lucene. The questions asked to those 10 posts are our question candidates Q and the edits made to the posts in response to the questions are our answer candidates A. For each question candidate q_i, we generate an answer representation $F(p, q_i)$ and calculate how close is the answer candidate a_j to our answer representation $F(p, q_i)$. We then calculate the utility of the post p if it were updated with the answer a_j. Finally, we rank the candidate questions Q by their expected utility given the post p (Eq 1).

Our work has two main contributions:

1. A novel neural-network model for addressing the task of ranking clarification question built on the framework of expected value of perfect information (§2).
2. A novel dataset, derived from StackExchange[2], that enables us to learn a model to ask clarifying questions by looking at the types of questions people ask (§3).

We formulate this task as a ranking problem on a set of potential clarification questions. We evaluate models both on the task of returning the original clarification question and also on the task of picking any of the candidate clarification questions marked as good by experts (§4). We find that our EVPI model outperforms the baseline models when evaluated against expert human annotations. We include a few examples of human annotations along with our model performance on them in the supplementary material. We have released our dataset of ∼77K (p, q, a) triples and the expert annotations on 500 triples to help facilitate further research in this task.[3]

2 Model description

We build a neural network model inspired by the theory of expected value of perfect information (EVPI). EVPI is a measurement of: if I were to acquire information X, how useful would that be to me? However, because we haven't acquired X yet, we have to take this quantity in expectation over all possible X, weighted by each X's likelihood. In our setting, for any given question q_i that we can ask, there is a set A of possible answers that could be given. For each possible answer $a_j \in A$, there is some probability of getting that answer, and some utility if that were the answer we got. The value of this question q_i is the expected utility, over all possible answers:

$$\text{EVPI}(q_i|p) = \sum_{a_j \in A} \mathbb{P}[a_j|p, q_i]\mathbb{U}(p + a_j) \qquad (1)$$

In Eq 1, p is the post, q_i is a potential question from a set of candidate questions Q and a_j is a potential answer from a set of candidate answers A. Here, $\mathbb{P}[a_j|p, q_i]$ measures the probability of getting an answer a_j given an initial post p and a clarifying question q_i, and $\mathbb{U}(p + a_j)$ is a utility function that measures how much more complete p would be if it were augmented with answer a_j. The modeling question then is how to model:

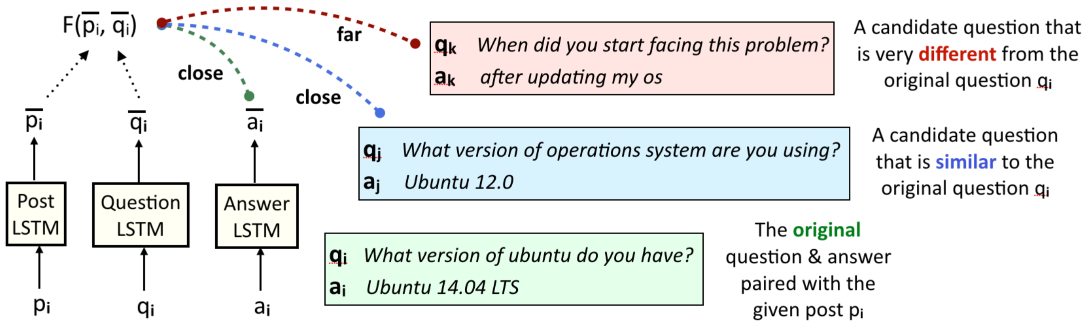
1. The probability distribution $\mathbb{P}[a_j|p, q_i]$ and
2. The utility function $\mathbb{U}(p + a_j)$.

In our work, we represent both using neural networks over the appropriate inputs. We train the parameters of the two models jointly to minimize a joint loss defined such that an answer that has a higher potential of increasing the utility of a post gets a higher probability.

Figure 2 describes the behavior of our model during test time. Given a post p, we generate a set of candidate questions and a set of candidate

[2]We use data from StackExchange; per license cc-by-sa 3.0, the data is "intended to be shared and remixed" (with attribution).

[3]https://github.com/raosudha89/ranking_clarification_questions



$F(\bar{p}_i, \bar{q}_i)$
far
close
close
$\bar{p}_i$
$\bar{q}_i$
$\bar{a}_i$
Post LSTM
Question LSTM
Answer LSTM
p_i
q_i
a_i
q_k When did you start facing this problem?
a_k after updating my os
A candidate question that is very different from the original question q_i
q_j What version of operations system are you using?
a_j Ubuntu 12.0
A candidate question that is similar to the original question q_i
q_i What version of ubuntu do you have?
a_i Ubuntu 14.04 LTS
The original question & answer paired with the given post p_i

Figure 3: Training of our answer generator. Given a post p_i and its question q_i, we generate an answer representation that is not only close to its original answer a_i, but also close to one of its candidate answers a_j if the candidate question q_j is close to the original question q_i.

answers (§2.1). Given a post p and a question candidate q_i, we calculate how likely is this question to be answered using one of our answer candidates a_j (§2.2). Given a post p and an answer candidate a_j, we calculate the utility of the updated post i.e. $\mathbb{U}(p + a_j)$ (§2.3). We compose these modules into a joint neural network that we optimize end-to-end over our data (§2.4).

2.1 Question & answer candidate generator

Given a post p, our first step is to generate a set of question and answer candidates. One way that humans learn to ask questions is by looking at how others ask questions in a similar situation. Using this intuition we generate question candidates for a given post by identifying posts similar to the given post and then looking at the questions asked to those posts. For identifying similar posts, we use Lucene[4], a software extensively used in information retrieval for extracting documents relevant to a given query from a pool of documents. Lucene implements a variant of the term frequency-inverse document frequency (TF-IDF) model to score the extracted documents according to their relevance to the query. We use Lucene to find the top 10 posts most similar to a given post from our dataset (§ 3). We consider the questions asked to these 10 posts as our set of question candidates Q and the edits made to the posts in response to the questions as our set of answer candidates A. Since the top-most similar candidate extracted by Lucene is always the original post itself, the original question and answer paired with the post is always one of the candidates in Q and A. §3 describes in detail the process of extracting the (*post, question, answer*) triples from the StackExchange datadump.

2.2 Answer modeling

Given a post p and a question candidate q_i, our second step is to calculate how likely is this question to be answered using one of our answer candidates a_j. We first generate an answer representation by combining the neural representations of the post and the question using a function $F_{ans}(\bar{p}, \bar{q}_i)$ (details in §2.4). Given such a representation, we measure the distance between this answer representation and one of the answer candidates a_j using the function below:

$$dist(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j) = 1 - cos\_sim(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j)$$

The likelihood of an answer candidate a_j being the answer to a question q_i on post p is finally calculated by combining this distance with the cosine similarity between the question q_i and the question q_j paired with the answer candidate a_j:

$$\mathbb{P}[a_j|p, q_i] = \exp^{-dist(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j)} * cos\_sim(\hat{q}_i,\ \hat{q}_j) \tag{2}$$

where $\hat{a}_j$, $\hat{q}_i$ and $\hat{q}_j$ are the average word vector of a_j, q_i and q_j respectively (details in § 2.4) and cos_sim is the cosine similarity between the two input vectors.

We model our answer generator using the following intuition: a question can be asked in several different ways. For e.g. in Figure 1, the question "What version of Ubuntu do you have?" can be asked in other ways like "What version of operating system are you using?", "Version of OS?", etc. Additionally, for a given post and a question, there can be

[4]https://lucene.apache.org/

several different answers to that question. For instance, “Ubuntu 14.04 LTS”, “Ubuntu 12.0”, “Ubuntu 9.0”, are all valid answers. To generate an answer representation capturing these generalizations, we train our answer generator on our triples dataset (§3) using the loss function below:

$$\text{loss}_{\text{ans}}(p_i, q_i, a_i, Q_i) = dist(F_{ans}(\bar{p}_i, \bar{q}_i),\ \hat{a}_i) + \sum_{j \in Q} \Big(dist(F_{ans}(\bar{p}_i, \bar{q}_i),\ \hat{a}_j) * cos\_sim(\hat{q}_i,\ \hat{q}_j) \Big) \quad (3)$$

where, $\hat{a}$ and $\hat{q}$ is the average word vectors of a and q respectively (details in §2.4), cos_sim is the cosine similarity between the two input vectors.

This loss function can be explained using the example in Figure 3. Question q_i is the question paired with the given post p_i. In Eq 3, the first term forces the function $F_{ans}(\bar{p}_i, \bar{q}_i)$ to generate an answer representation as close as possible to the correct answer a_i. Now, a question can be asked in several different ways. Let Q_i be the set of candidate questions for post p_i, retrieved from the dataset using Lucene (§ 2.1). Suppose a question candidate q_j is very similar to the correct question q_i (i.e. $cos\_sim(\hat{q}_i,\ \hat{q}_j)$ is near zero). Then the second term forces the answer representation $F_{ans}(\bar{p}_i, \bar{q}_i)$ to be close to the answer a_j corresponding to the question q_j as well. Thus in Figure 3, the answer representation will be close to a_j (since q_j is similar to q_i), but may not be necessarily close to a_k (since q_k is dissimilar to q_i).

2.3 Utility calculator

Given a post p and an answer candidate a_j, the third step is to calculate the utility of the updated post i.e. $\mathbb{U}(p + a_j)$. As expressed in Eq 1, this utility function measures how useful it would be if a given post p were augmented with an answer a_j paired with a different question q_j in the candidate set. Although theoretically, the utility of the updated post can be calculated only using the given post (p) and the candidate answer (a_j), empirically we find that our neural EVPI model performs better when the candidate question (q_j) paired with the candidate answer is a part of the utility function. We attribute this to the fact that much information about whether an answer increases the utility of a post is also contained in the question asked to the post. We train our utility calculator using our dataset of (p, q, a) triples (§3). We label all the (p_i, q_i, a_i) pairs from our triples dataset with label $y = 1$. To get negative samples, we make use of the answer candidates generated using Lucene as described in §2.1. For each $a_j \in A_i$, where A_i is the set of answer candidates for post p_i, we label the pair (p_i, q_j, a_j) with label $y = 0$, except for when $a_j = a_i$. Thus, for each post p_i in our triples dataset, we have one positive sample and nine negative samples. It should be noted that this is a noisy labelling scheme since a question not paired with the original question in our dataset can often times be a *good* question to ask to the post (§ 4). However, since we do not have annotations for such other good questions at train time, we assume such a labelling.

Given a post p_i and an answer a_j paired with the question q_j, we combine their neural representations using a function $F_{util}(\bar{p}_i, \bar{q}_j, \bar{a}_j)$ (details in § 2.4). The utility of the updated post is then defined as $\mathbb{U}(p_i + a_j) = \sigma(F_{util}(\bar{p}_i, \bar{q}_j, \bar{a}_j))$[5]. We want this utility to be close to 1 for all the positively labelled (p, q, a) triples and close to 0 for all the negatively labelled (p, q, a) triples. We therefore define our loss using the binary cross-entropy formulation below:

$$\text{loss}_{\text{util}}(y_i, \bar{p}_i, \bar{q}_j, \bar{a}_j) = y_i \log(\sigma(F_{util}(\bar{p}_i, \bar{q}_j, \bar{a}_j))) \quad (4)$$

2.4 Our joint neural network model

Our fundamental representation is based on recurrent neural networks over word embeddings. We obtain the word embeddings using the GloVe (Pennington et al., 2014) model trained on the entire datadump of StackExchange.[6]. In Eq 2 and Eq 3, the average word vector representations $\hat{q}$ and $\hat{a}$ are obtained by averaging the GloVe word embeddings for all words in the question and the answer respectively. Given an initial post p, we generate a post neural representation $\bar{p}$ using a post LSTM (long short-term memory architecture) (Hochreiter and Schmidhuber, 1997). The input layer consists of word embeddings of the words in the post which is fed into a single hidden layer. The output of each of the hidden states is averaged together to get our neural representation $\bar{p}$. Similarly, given a question q and an answer a, we generate the neural representations $\bar{q}$ and $\bar{a}$ using a question LSTM and an answer LSTM respectively. We define the function F_{ans} in our answer model as a feedforward neural network with five hidden layers on the inputs $\bar{p}$ and $\bar{q}$. Likewise, we

[5] σ is the sigmoid function.

[6] Details in the supplementary material.

define the function F_{util} in our utility calculator as a feedforward neural network with five hidden layers on the inputs $\bar{p}$, $\bar{q}$ and $\bar{a}$. We train the parameters of the three LSTMs corresponding to p, q and a, and the parameters of the two feedforward neural networks jointly to minimize the sum of the loss of our answer model (Eq 3) and our utility calculator (Eq 4) over our entire dataset:

$$\sum_i \sum_j \text{loss}_{\text{ans}}(\bar{p}_i, \bar{q}_i, \bar{a}_i, Q_i) + \text{loss}_{\text{util}}(y_i, \bar{p}_i, \bar{q}_j, \bar{a}_j) \tag{5}$$

Given such an estimate $\mathbb{P}[a_j|p, q_i]$ of an answer and a utility $\mathbb{U}(p + a_j)$ of the updated post, we rank the candidate questions by their value as calculated using Eq 1. The remaining question, then, is how to get data that enables us to train our answer model and our utility calculator. Given data, the training becomes a multitask learning problem, where we learn simultaneously to predict utility and to estimate the probability of answers.

3 Dataset creation

StackExchange is a network of online question answering websites about varied topics like academia, ubuntu operating system, latex, etc. The data dump of StackExchange contains timestamped information about the posts, comments on the post and the history of the revisions made to the post. We use this data dump to create our dataset of (*post, question, answer*) triples: where the *post* is the initial unedited post, the *question* is the comment containing a question and the *answer* is either the edit made to the post after the question or the author's response to the question in the comments section.

Extract posts: We use the post histories to identify posts that have been updated by its author. We use the timestamp information to retrieve the initial unedited version of the post.

Extract questions: For each such initial version of the post, we use the timestamp information of its comments to identify the first question comment made to the post. We truncate the comment till its question mark '?' to retrieve the question part of the comment. We find that about 7% of these are rhetoric questions that indirectly suggest a solution to the post. For e.g. *"have you considered installing X?"*. We do a manual analysis of these non-clarification questions and hand-crafted a few rules to remove them. [7]

	Train	Tune	Test
`askubuntu`	19,944	2493	2493
`unix`	10,882	1360	1360
`superuser`	30,852	3857	3856

Table 1: Table above shows the sizes of the train, tune and test split of our dataset for three domains.

Extract answers: We extract the answer to a clarification question in the following two ways:
(a) Edited post: Authors tend to respond to a clarification question by editing their original post and adding the missing information. In order to account for edits made for other reasons like stylistic updates and grammatical corrections, we consider only those edits that are longer than four words. Authors can make multiple edits to a post in response to multiple clarification questions.[8] To identify the edit made corresponding to the given question comment, we choose the edit closest in time following the question.
(b) Response to the question: Authors also respond to clarification questions as subsequent comments in the comment section. We extract the first comment by the author following the clarification question as the answer to the question.

In cases where both the methods above yield an answer, we pick the one that is the most semantically similar to the question, where the measure of similarity is the cosine distance between the average word embeddings of the question and the answer.

We extract a total of 77,097 (*post, question, answer*) triples across three domains in StackExchange (Table 1). We will release this dataset along with the the nine question and answer candidates per triple that we generate using lucene (§ 2.1). We include an analysis of our dataset in the supplementary material.

4 Evaluation design

We define our task as given a post p, and a set of candidate clarification questions Q, rank the questions according to their usefulness to the post.

[7] Details in the supplementary material.

[8] On analysis, we find that 35%-40% of the posts get asked multiple clarification questions. We include only the first clarification question to a post in our dataset since identifying if the following questions are clarifications or a part of a dialogue is non-trivial.

Since the candidate set includes the original question q that was asked to the post p, one possible approach to evaluation would be to look at how often the original question is ranked higher up in the ranking predicted by a model. However, there are two problems to this approach: 1) Our dataset creation process is noisy. The original question paired with the post may not be a useful question. For e.g. *"are you seriously asking this question?"*, *"do you mind making that an answer?"*[9]. 2) The nine other questions in the candidate set are obtained by looking at questions asked to posts that are similar to the given post.[10] This greatly increases the possibility of some other question(s) being more useful than the original question paired with the post. This motivates an evaluation design that does not rely solely on the original question but also uses human judgments. We randomly choose a total of 500 examples from the test sets of the three domains proportional to their train set sizes (`askubuntu:160`, `unix:90` and `superuser:250`) to construct our evaluation set.

4.1 Annotation scheme

Due to the technical nature of the posts in our dataset, identifying useful questions requires technical experts. We recruit 10 such experts on Upwork[11] who have prior experience in unix based operating system administration.[12] We provide the annotators with a post and a randomized list of the ten question candidates obtained using Lucene (§2.1) and ask them to select a *single* "best" (B) question to ask, and additionally mark as "valid" (V) other questions that they thought would be okay to ask in the context of the original post. We enforce that the "best" question be always marked as a "valid" question. We group the 10 annotators into 5 pairs and assign the same 100 examples to the two annotators in a pair.

4.2 Annotation analysis

We calculate the inter-annotator agreement on the "best" and the "valid" annotations using Cohen's Kappa measurement. When calculating the agreement on the "best" in the strict sense, we get a low

[9]Data analysis included in the supplementary material suggests 9% of the questions are not useful.

[10]Note that this setting is different from the distractor-based setting popularly used in dialogue (Lowe et al., 2015) where the distractor candidates are chosen randomly from the corpus.

[11]https://upwork.com

[12]Details in the supplementary material.

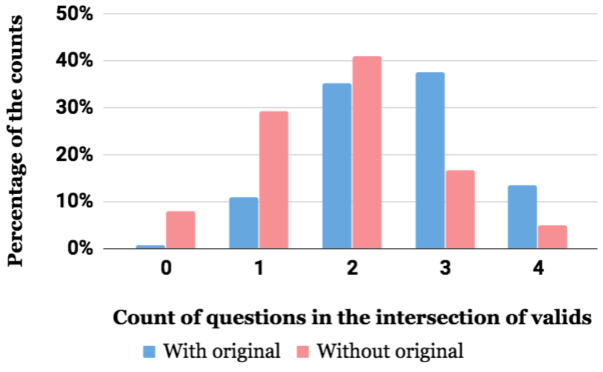


Figure 4: Distribution of the count of questions in the intersection of the "valid" annotations.

agreement of 0.15. However, when we relax this to a case where the question marked as"best" by one annotator is marked as "valid" by another, we get an agreement of 0.87. The agreement on the "valid" annotations, on the other hand, was higher: 0.58. We calculate this agreement on the binary judgment of whether a question was marked as valid by the annotator.

Given these annotations, we calculate how often is the original question marked as "best" or "valid" by the two annotators. We find that 72% of the time one of the annotators mark the original as the "best", whereas only 20% of the time both annotators mark it as the "best" suggesting against an evaluation solely based on the original question. On the other hand, 88% of the time one of the two annotators mark it as a "valid" question confirming the noise in our training data.[13]

Figure 4 shows the distribution of the counts of questions in the intersection of "valid" annotations (blue legend). We see that about 85% of the posts have more than 2 valid questions and 50% have more than 3 valid questions. The figure also shows the distribution of the counts when the original question is removed from the intersection (red legend). Even in this set, we find that about 60% of the posts have more than two valid questions. These numbers suggests that the candidate set of questions retrieved using Lucene (§2.1) very often contains useful clarification questions.

5 Experimental results

Our primary research questions that we evaluate experimentally are:

1. Does a neural network architecture improve upon non-neural baselines?

[13]76% of the time both the annotators mark it as a "valid".

Model	$B1 \cup B2$ p@1	p@3	p@5	MAP	$V1 \cap V2$ p@1	p@3	p@5	MAP	Original p@1
Random	17.5	17.5	17.5	35.2	26.4	26.4	26.4	42.1	10.0
Bag-of-ngrams	19.4	19.4	18.7	34.4	25.6	27.6	27.5	42.7	10.7
Community QA	23.1	21.2	20.0	40.2	33.6	30.8	29.1	47.0	18.5
Neural (p, q)	21.9	20.9	19.5	39.2	31.6	30.0	28.9	45.5	15.4
Neural (p, a)	24.1	**23.5**	20.6	41.4	32.3	**31.5**	29.0	46.5	18.8
Neural (p, q, a)	25.2	**22.7**	**21.3**	42.5	34.4	**31.8**	**30.1**	47.7	20.5
EVPI	**27.7**	**23.4**	**21.5**	**43.6**	**36.1**	**32.2**	**30.5**	**49.2**	**21.4**

Table 2: Model performances on 500 samples when evaluated against the union of the "best" annotations ($B1 \cup B2$), intersection of the "valid" annotations ($V1 \cap V2$) and the original question paired with the post in the dataset. The difference between the bold and the non-bold numbers is statistically significant with $p < 0.05$ as calculated using bootstrap test. p@k is the precision of the k questions ranked highest by the model and MAP is the mean average precision of the ranking predicted by the model.

2. Does the EVPI formalism provide leverage over a similarly expressive feedforward network?
3. Are answers useful in identifying the right question?
4. How do the models perform when evaluated on the candidate questions excluding the original?

5.1 Baseline methods

We compare our model with following baselines:

Random: Given a post, we randomly permute its set of 10 candidate questions uniformly.[14]

Bag-of-ngrams: Given a post and a set of 10 question and answer candidates, we construct a bag-of-ngrams representation for the post, question and answer. We train the baseline on all the positive and negative candidate triples (same as in our utility calculator (§2.3)) to minimize hinge loss on misclassification error using cross-product features between each of (*p, q*), (*q, a*) and (*p, a*). We tune the ngram length and choose n=3 which performs best on the tune set. The question candidates are finally ranked according to their predictions for the positive label.

Community QA: The recent SemEval2017 Community Question-Answering (CQA) (Nakov et al., 2017) included a subtask for ranking a set of comments according to their relevance to a given post in the Qatar Living[15] forum. Nandi et al. (2017), winners of this subtask, developed a logistic regression model using features based on string similarity, word embeddings, etc. We train this model on all the positively and negatively labelled (p, q) pairs in our dataset (same as in our utility calculator (§2.3), but without a). We use a subset of their features relevant to our task.[16]

Neural baselines: We construct the following neural baselines based on the LSTM representation of their inputs (as described in §2.4):
1. **Neural(p, q):** Input is concatenation of $\bar{p}$ and $\bar{q}$.
2. **Neural(p, a):** Input is concatenation of $\bar{p}$ and $\bar{a}$.
3. **Neural(p, q, a):** Input is concatenation of $\bar{p}$, $\bar{q}$ and $\bar{a}$.

Given these inputs, we construct a fully connected feedforward neural network with 10 hidden layers and train it to minimize the binary cross entropy across all positive and negative candidate triples (same as in our utility calculator (§ 2.3)). The major difference between the neural baselines and our EVPI model is in the loss function: the EVPI model is trained to minimize the joint loss between the answer model (defined on $F_{ans}(p, q)$ in Eq 3) and the utility calculator (defined on $F_{util}(p, q, a)$ in Eq 4) whereas the neural baselines are trained to minimize the loss directly on $F(p, q)$, $F(p, a)$ or $F(p, q, a)$. We include the implementation details of all our neural models in the supplementary material.

5.2 Results

5.2.1 Evaluating against expert annotations

We first describe the results of the different models when evaluated against the expert annotations we collect on 500 samples (§4). Since the annotators

[14]We take the average over 1000 random permutations.
[15]http://www.qatarliving.com/forum

[16]Details in the supplementary material.

had a low agreement on a single best, we evaluate against the union of the "best" annotations ($B1 \cup B2$ in Table 2) and against the intersection of the "valid" annotations ($V1 \cap V2$ in Table 2).

Among non-neural baselines, we find that the bag-of-ngrams baseline performs slightly better than random but worse than all the other models. The Community QA baseline, on the other hand, performs better than the neural baseline (Neural (p, q)), both of which are trained without using the answers. The neural baselines with answers (Neural(p, q, a) and Neural(p, a)) outperform the neural baseline without answers (Neural(p, q)), showing that answer helps in selecting the right question.

More importantly, EVPI outperforms the Neural (p, q, a) baseline across most metrics. Both models use the same information regarding the true question and answer and are trained using the same number of model parameters.[17] However, the EVPI model, unlike the neural baseline, additionally makes use of alternate question and answer candidates to compute its loss function. This shows that when the candidate set consists of questions similar to the original question, summing over their utilities gives us a boost.

5.2.2 Evaluating against the original question

The last column in Table 2 shows the results when evaluated against the original question paired with the post. The bag-of-ngrams baseline performs similar to random, unlike when evaluated against human judgments. The Community QA baseline again outperforms Neural(p, q) model and comes very close to the Neural (p, a) model.

As before, the neural baselines that make use of the answer outperform the one that does not use the answer and the EVPI model performs significantly better than Neural(p, q, a).

5.2.3 Excluding the original question

In the preceding analysis, we considered a setting in which the "ground truth" original question was in the candidate set Q. While this is a common evaluation framework in dialog response selection (Lowe et al., 2015), it is overly optimistic. We, therefore, evaluate against the "best" and the "valid" annotations on the nine other question candidates. We find that the neural models beat the non-neural baselines. However, the differences between all the neural models are statistically insignificant.[18]

6 Related work

Most prior work on question generation has focused on generating reading comprehension questions: given text, write questions that one might find on a standardized test (Vanderwende, 2008; Heilman, 2011; Rus et al., 2011; Olney et al., 2012). Comprehension questions, by definition, are answerable from the provided text. Clarification questions–our interest–are not.

Outside reading comprehension questions, Labutov et al. (2015) generate high-level question templates by crowdsourcing which leads to significantly less data than we collect using our method. Liu et al. (2010) use template question generation to help authors write better related work sections. Mostafazadeh et al. (2016) introduce a Visual Question Generation task where the goal is to generate natural questions that are not about what is present in the image rather about what can be inferred given the image, somewhat analogous to clarification questions. Penas and Hovy (2010) identify the notion of missing information similar to us, but they fill the knowledge gaps in a text with the help of external knowledge bases, whereas we instead ask clarification questions. Artzi and Zettlemoyer (2011) use human-generated clarification questions to drive a semantic parser where the clarification questions are aimed towards simplifying a user query; whereas we generate clarification questions aimed at identifying missing information in a text.

Among works that use community question answer forums, the keywords to questions (K2Q) system (Zheng et al., 2011) generates a list of candidate questions and refinement words, given a set of input keywords, to help a user ask a better question. Figueroa and Neumann (2013) rank different paraphrases of query for effective search on forums. (Romeo et al., 2016) develop a neural network based model for ranking questions on forums with the intent of retrieving similar other question. The recent SemEval-2017 Community Question-Answering (CQA) (Nakov et al., 2017) task included a subtask to rank the comments according to their relevance to the post. Our task primarily differs from this task in that we want to identify a

[17]We use 10 hidden layers in the feedforward network of the neural baseline and five hidden layers each in the two feedforward networks F_{ans} and F_{util} of the EVPI model.

[18]Results included in the supplementary material.

question comment which is not only relevant to the post but will also elicit useful information missing from the post. Hoogeveen et al. (2015) created the CQADupStack dataset using StackExchange forums for the task of duplicate question retrieval. Our dataset, on the other hand, is designed for the task of ranking clarification questions asked as comments to a post.

7 Conclusion

We have constructed a new dataset for learning to rank clarification questions, and proposed a novel model for solving this task. Our model integrates well-known deep network architectures with the classic notion of expected value of perfect information, which effectively models a pragmatic choice on the part of the questioner: how do I *imagine* the other party would answer if I were to ask this question. Such pragmatic principles have recently been shown to be useful in other tasks as well (Golland et al., 2010; Smith et al., 2013; Orita et al., 2015; Andreas and Klein, 2016). One can naturally extend our EVPI approach to a full reinforcement learning approach to handle multi-turn conversations.

Our results shows that the EVPI model is a promising formalism for the question generation task. In order to move to a full system that can help users like Terry write better posts, there are three interesting lines of future work. First, we need it to be able to generalize: for instance by constructing templates of the form “What version of ___ are you running?” into which the system would need to fill a variable. Second, in order to move from question ranking to question generation, one could consider sequence-to-sequence based neural network models that have recently proven to be effective for several language generation tasks (Sutskever et al., 2014; Serban et al., 2016; Yin et al., 2016). Third is in evaluation: given that this task requires expert human annotations and also given that there are multiple possible good questions to ask, how can we automatically measure performance at this task?, a question faced in dialog and generation more broadly (Paek, 2001; Lowe et al., 2015; Liu et al., 2016).

Acknowledgments

The authors thank the three anonymous reviewers of this paper, and the anonymous reviewers of the previous versions for their helpful comments and suggestions. They also thank the members of the Computational Linguistics and Information Processing (CLIP) lab at University of Maryland for helpful discussions.

This work was supported by NSF grant IIS-1618193. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1173–1182.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 421–432.

Mordecai Avriel and AC Williams. 1970. The value of information and stochastic programming. *Operations Research* 18(5):947–954.

Alejandro Figueroa and Günter Neumann. 2013. Learning to rank effective paraphrases from query logs for community question answering. In *AAAI*. volume 13, pages 1099–1105.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 410–419.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

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

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*. ACM, page 3.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *ACL (1)*. pages 889–898.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the*

2016 Conference on Empirical Methods in Natural Language Processing. pages 2122–2132.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2010. Automatic question generation for literature review writing support. In *International Conference on Intelligent Tutoring Systems*. Springer, pages 45–54.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 285.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1802–1813.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 27–48.

Titas Nandi, Chris Biemann, Seid Muhie Yimam, Deepak Gupta, Sarah Kohail, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iit-uhh at semeval-2017 task 3: Exploring multiple features for community question answering and implicit dialogue identification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 90–97.

Andrew McGregor Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. *D&D* 3(2):75–99.

Naho Orita, Eliana Vornov, Naomi Feldman, and Hal Daumé III. 2015. Why discourse affects speakers' choice of referring expressions. In *ACL (1)*. pages 1639–1649.

Tim Paek. 2001. Empirical methods for evaluating dialog systems. In *Proceedings of the workshop on Evaluation for Language and Dialogue Systems-Volume 9*. Association for Computational Linguistics, page 2.

Anselmo Penas and Eduard Hovy. 2010. Filling knowledge gaps in text for machine reading. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 979–987.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeno, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural attention for learning to rank questions in community question answering. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1734–1745.

Vasile Rus, Paul Piwek, Svetlana Stoyanchev, Brendan Wyse, Mihai Lintean, and Cristian Moldovan. 2011. Question generation shared task and evaluation challenge: Status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 318–320.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*. pages 3776–3784.

Nathaniel J Smith, Noah Goodman, and Michael Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in neural information processing systems*. pages 3039–3047.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge, Arlington, VA*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, pages 2972–2978.

Zhicheng Zheng, Xiance Si, Edward Chang, and Xiaoyan Zhu. 2011. K2q: Generating natural language questions from keywords with user refinements. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pages 947–955.

Let's do it "again": A First Computational Approach to Detecting Adverbial Presupposition Triggers

Andre Cianflone [*] **Yulan Feng** [*] **Jad Kabbara** [*] **Jackie Chi Kit Cheung**

School of Computer Science
McGill University
Montreal, QC, Canada

MILA
Montreal, QC, Canada

{andre.cianflone@mail, yulan.feng@mail, jad@cs, jcheung@cs}.mcgill.ca

Abstract

We introduce the task of predicting adverbial presupposition triggers such as *also* and *again*. Solving such a task requires detecting recurring or similar events in the discourse context, and has applications in natural language generation tasks such as summarization and dialogue systems. We create two new datasets for the task, derived from the Penn Treebank and the Annotated English Gigaword corpora, as well as a novel attention mechanism tailored to this task. Our attention mechanism augments a baseline recurrent neural network without the need for additional trainable parameters, minimizing the added computational cost of our mechanism. We demonstrate that our model statistically outperforms a number of baselines, including an LSTM-based language model.

1 Introduction

In pragmatics, presuppositions are assumptions or beliefs in the common ground between discourse participants when an utterance is made (Frege, 1892; Strawson, 1950; Stalnaker, 1973, 1998), and are ubiquitous in naturally occurring discourses (Beaver and Geurts, 2014). Presuppositions underly spoken statements and written sentences and understanding them facilitates smooth communication. We refer to expressions that indicate the presence of presuppositions as *presupposition triggers*. These include definite descriptions, factive verbs and certain adverbs, among others. For example, consider the following statements:

(1) John is going to the restaurant *again*.

(2) John has been to the restaurant.

(1) is only appropriate in the context where (2) is held to be true because of the presence of the presupposition trigger *again*. One distinguishing characteristic of presupposition is that it is unaffected by negation of the presupposing context, unlike other semantic phenomena such as entailment and implicature. The negation of (1), *John is not going to the restaurant again.*, also presupposes (2).

Our focus in this paper is on adverbial presupposition triggers such as *again, also* and *still*. Adverbial presupposition triggers indicate the recurrence, continuation, or termination of an event in the discourse context, or the presence of a similar event. In one study of presuppositional triggers in English journalistic texts (Khaleel, 2010), adverbial triggers were found to be the most commonly occurring presupposition triggers after existential triggers.[1] Despite their frequency, there has been little work on these triggers in the computational literature from a statistical, corpus-driven perspective.

As a first step towards language technology systems capable of understanding and using presuppositions, we propose to investigate the detection of contexts in which these triggers can be used. This task constitutes an interesting testing ground for pragmatic reasoning, because the cues that are indicative of contexts containing recurring or similar events are complex and often span more than one sentence, as illustrated in Sentences (1) and (2). Moreover, such a task has immediate practical consequences. For example, in language generation applications such as summarization and dialogue systems, adding presuppositional triggers in contextually appropriate loca-

[*] Authors (listed in alphabetical order) contributed equally.

[1] Presupposition of existence are triggered by possessive constructions, names or definite noun phrases.

tions can improve the readability and coherence of the generated output.

We create two datasets based on the Penn Treebank corpus (Marcus et al., 1993) and the English Gigaword corpus (Graff et al., 2007), extracting contexts that include presupposition triggers as well as other similar contexts that do not, in order to form a binary classification task. In creating our datasets, we consider a set of five target adverbs: *too, again, also, still*, and *yet*. We focus on these adverbs in our investigation because these triggers are well known in the existing linguistic literature and commonly triggering presuppositions. We control for a number of potential confounding factors, such as class balance, and the syntactic governor of the triggering adverb, so that models cannot exploit these correlating factors without any actual understanding of the presuppositional properties of the context.

We test a number of standard baseline classifiers on these datasets, including a logistic regression model and deep learning methods based on recurrent neural networks (RNN) and convolutional neural networks (CNN).

In addition, we investigate the potential of attention-based deep learning models for detecting adverbial triggers. Attention is a promising approach to this task because it allows a model to weigh information from multiple points in the previous context and infer long-range dependencies in the data (Bahdanau et al., 2015). For example, the model could learn to detect multiple instances involving *John* and *restaurants*, which would be a good indication that *again* is appropriate in that context. Also, an attention-based RNN has achieved success in predicting article definiteness, which involves another class of presupposition triggers (Kabbara et al., 2016).

As another contribution, we introduce a new weighted pooling attention mechanism designed for predicting adverbial presupposition triggers. Our attention mechanism allows for a weighted averaging of our RNN hidden states where the weights are informed by the inputs, as opposed to a simple unweighted averaging. Our model uses a form of self-attention (Paulus et al., 2018; Vaswani et al., 2017), where the input sequence acts as both the attention mechanism's query and key/value. Unlike other attention models, instead of simply averaging the scores to be weighted, our approach aggregates (learned) attention scores by learning a reweighting scheme of those scores through another level (dimension) of attention. Additionally, our mechanism does not introduce any new parameters when compared to our LSTM baseline, reducing its computational impact.

We compare our model using the novel attention mechanism against the baseline classifiers in terms of prediction accuracy. Our model outperforms these baselines for most of the triggers on the two datasets, achieving 82.42% accuracy on predicting the adverb "also" on the Gigaword dataset.

The contributions of this work are as follows:

1. We introduce the task of predicting adverbial presupposition triggers.

2. We present new datasets for the task of detecting adverbial presupposition triggers, with a data extraction method that can be applied to other similar pre-processing tasks.

3. We develop a new attention mechanism in an RNN architecture that is appropriate for the prediction of adverbial presupposition triggers, and show that its use results in better prediction performance over a number of baselines without introducing additional parameters.

2 Related Work

2.1 Presupposition and pragmatic reasoning

The discussion of presupposition can be traced back to Frege's work on the philosophy of language (Frege, 1892), which later leads to the most commonly accepted view of presupposition called the Frege-Strawson theory (Kaplan, 1970; Strawson, 1950). In this view, presuppositions are preconditions for sentences/statements to be true or false. To the best of our knowledge, there is no previous computational work that directly investigates adverbial presupposition. However in the fields of semantics and pragmatics, there exist linguistic studies on presupposition that involve adverbs such as "too" and "again" (e.g., (Blutner et al., 2003), (Kang, 2012)) as a pragmatic presupposition trigger. Also relevant to our work is (Kabbara et al., 2016), which proposes using an attention-based LSTM network to predict noun phrase definiteness in English. Their work demonstrates the ability of these attention-based models to pick up on contextual cues for pragmatic reasoning.

Many different classes of construction can trigger presupposition in an utterance, this includes but is not limited to stressed constituents, factive verbs, and implicative verbs (Zare et al., 2012). In this work, we focus on the class of adverbial presupposition triggers.

Our task setup resembles the Cloze test used in psychology (Taylor, 1953; E. B. Coleman, 1968; Earl F. Rankin, 1969) and machine comprehension (Riloff and Thelen, 2000), which tests text comprehension via a fill-in-the-blanks task. We similarly pre-process our samples such that they are roughly the same length, and have equal numbers of negative samples as positive ones. However, we avoid replacing the deleted words with a blank, so that our model has no clue regarding the exact position of the possibly missing trigger. Another related work on the Children's Book Test (Hill et al., 2015) notes that memories that encode sub-sentential chunks (windows) of informative text seem to be most useful to neural networks when interpreting and modelling language. Their finding inspires us to run initial experiments with different context windows and tune the size of chunks according to the Logistic Regression results on the development set.

2.2 Attention

In the context of encoder-decoder models, attention weights are usually based on an energy measure of the previous decoder hidden state and encoder hidden states. Many variations on attention computation exist. Sukhbaatar et al. (2015) propose an attention mechanism conditioned on a query and applied to a document. To generate summaries, Paulus et al. (2018) add an attention mechanism in the prediction layer, as opposed to the hidden states. Vaswani et al. (2017) suggest a model which learns an input representation by self-attending over inputs. While these methods are all tailored to their specific tasks, they all inspire our choice of a self-attending mechanism.

3 Datasets

3.1 Corpora

We extract datasets from two corpora, namely the Penn Treebank (PTB) corpus (Marcus et al., 1993) and a subset (sections 000-760) of the third edition of the English Gigaword corpus (Graff et al., 2007). For the PTB dataset, we use sections 22 and 23 for testing. For the Gigaword corpus, we use sections 700-760 for testing. For the remaining data, we randomly chose 10% of them for development, and the other 90% for training.

```
('still',
['The', 'Old', 'Granary', .../* 46
    tokens omitted */...,'has', '@@@@',
    'included', 'Bertrand', 'Russell',
    .../* 6 tokens omitted */... 'Morris
    '],
['DT', 'NNP', 'NNP', .../* 46 tokens
    omitted */..., 'VBZ', '@@@@', 'VBN',
     'NNP', 'NNP', .../* 6 tokens
    omitted */... 'NNP'])
```

Figure 1: An example of an instance containing a presuppositional trigger from our dataset.

For each dataset, we consider a set of five target adverbs: *too, again, also, still*, and *yet*. We choose these five because they are commonly used adverbs that trigger presupposition. Since we are concerned with investigating the capacity of attentional deep neural networks in predicting the presuppositional effects in general, we frame the learning problem as a binary classification for predicting the presence of an adverbial presupposition (as opposed to the identity of the adverb).

On the Gigaword corpus, we consider each adverb separately, resulting in five binary classification tasks. This was not feasible for PTB because of its small size.

Finally, because of the commonalities between the adverbs in presupposing similar events, we create a dataset that unifies all instances of the five adverbs found in the Gigaword corpus, with a label "1" indicating the presence of any of these adverbs.

3.2 Data extraction process

We define a sample in our dataset as a 3-tuple, consisting of a label (representing the target adverb, or 'none' for a negative sample), a list of tokens we extract (before/after the adverb), and a list of corresponding POS tags (Klein and Manning, 2002). In each sample, we also add a special token "@@@@" right before the head word and the corresponding POS tag of the head word, both in positive and negative cases. We add such special tokens to identify the candidate context in the passage to the model. Figure 1 shows a single positive sample in our dataset.

We first extract positive contexts that contain a triggering adverb, then extract negative contexts that do not, controlling for a number of potential confounds. Our positive data consist of cases where the target adverb triggers presupposition by modifying a certain head word which, in most cases, is a verb. We define such head word as a *governor* of the target adverb.

When extracting positive data, we scan through all the documents, searching for target adverbs. For each occurrence of a target adverb, we store the location and the governor of the adverb. Taking each occurrence of a governor as a pivot, we extract the 50 unlemmatized tokens preceding it, together with the tokens right after it up to the end of the sentence (where the adverb is)–with the adverb itself being removed. If there are less than 50 tokens before the adverb, we simply extract all of these tokens. In preliminary testing using a logistic regression classifier, we found that limiting the size to 50 tokens had higher accuracy than 25 or 100 tokens. As some head words themselves are stopwords, in the list of tokens, we do not remove any stopwords from the sample; otherwise, we would lose many important samples.

We filter out the governors of "too" that have POS tags "JJ" and "RB" (adjectives and adverbs), because such cases corresponds to a different sense of "too" which indicates excess quantity and does not trigger presupposition (e.g., "rely too heavily on", "it's too far from").

After extracting the positive cases, we then use the governor information of positive cases to extract negative data. In particular, we extract sentences containing the same governors but not any of the target adverbs as negatives. In this way, models cannot rely on the identity of the governor alone to predict the class. This procedure also roughly balances the number of samples in the positive and negative classes.

For each governor in a positive sample, we locate a corresponding context in the corpus where the governor occurs without being modified by any of the target adverbs. We then extract the surrounding tokens in the same fashion as above. Moreover, we try to control position-related confounding factors by two randomization approaches: 1) randomize the order of documents to be scanned, and 2) within each document, start scanning from a random location in the document. Note that the number of negative cases might not be exactly equal to the number of negative cases in all datasets because some governors appearing in positive cases are rare words, and we're unable to find any (or only few) occurrences that match them for the negative cases.

4 Learning Model

In this section, we introduce our attention-based model. At a high level, our model extends a bidirectional LSTM model by computing correlations between the hidden states at each timestep, then applying an attention mechanism over these correlations. Our proposed weighted-pooling (WP) neural network architecture is shown in Figure 2.

The input sequence $u = \{u_1, u_2, \ldots, u_T\}$ consists of a sequence, of time length T, of one-hot encoded word tokens, where the original tokens are those such as in Listing 1. Each token u_t is embedded with pretrained embedding matrix $W_e \in \mathbb{R}^{|V| \times d}$, where $|V|$ corresponds to the number of tokens in vocabulary V, and d defines the size of the word embeddings. The embedded token vector $x_t \in \mathbb{R}^d$ is retrieved simply with $x_t = u_t W_e$. Optionally, x_t may also include the token's POS tag. In such instances, the embedded token at time step t is concatenated with the POS tag's one-hot encoding p_t: $x_t = u_t W_e || p_t$, where $||$ denotes the vector concatenation operator.

At each input time step t, a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) encodes x_t into hidden state $h_t \in \mathbb{R}^s$:

$$h_t = \left[\overrightarrow{h_t} || \overleftarrow{h_t}\right] \tag{1}$$

where $\overrightarrow{h_t} = f(x_t, h_{t-1})$ is computed by the forward LSTM, and $\overleftarrow{h_t} = f(x_t, h_{t+1})$ is computed by the backward LSTM. Concatenated vector h_t is of size $2s$, where s is a hyperparameter determining the size of the LSTM hidden states. Let matrix $H \in \mathbb{R}^{2s \times T}$ correspond to the concatenation of all hidden state vectors:

$$H = [h_1 || h_2 || \ldots || h_T]. \tag{2}$$

Our model uses a form of self-attention (Paulus et al., 2018; Vaswani et al., 2017), where the input sequence acts as both the attention mechanism's query and key/value. Since the location of a presupposition trigger can greatly vary from one sample to another, and because dependencies can be long range or short range, we model all possible word-pair interactions within a sequence. We calculate the energy between all input tokens with a

	Training set			Test set		
Corpus	Positive	Negative	Total	Positive	Negative	Total
PTB	2,596	2,579	5,175	249	233	482
Gigaword yet	32,024	31,819	63,843	7950	7890	15840
Gigaword too	55,827	29,918	85,745	13987	7514	21501
Gigaword again	43,120	42,824	85,944	10935	10827	21762
Gigaword still	97,670	96,991	194,661	24509	24232	48741
Gigaword also	269,778	267,851	537,626	66878	66050	132928
Gigaword all	498,415	491,173	989,588	124255	123078	247333

Table 1: Number of training samples in each dataset.

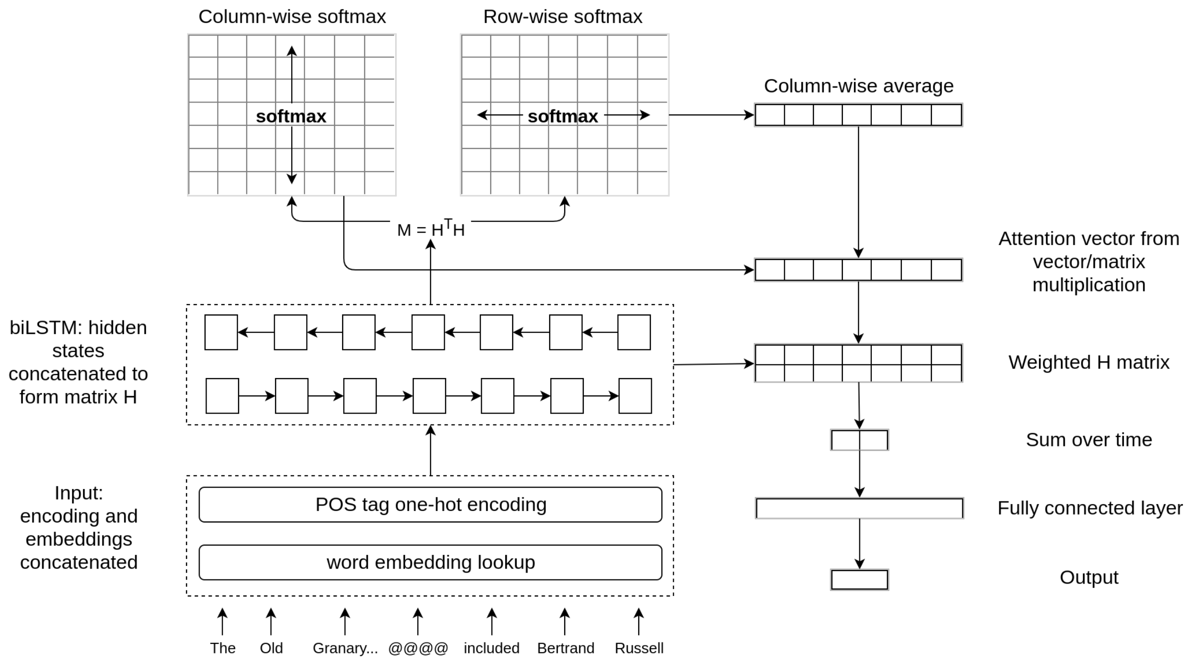


Figure 2: Our weighted-pooling neural network architecture (WP). The tokenized input is embedded with pretrained word embeddings and possibly concatenated with one-hot encoded POS tags. The input is then encoded with a bi-directional LSTM, followed by our attention mechanism. The computed attention scores are then used as weights to average the encoded states, in turn connected to a fully connected layer to predict presupposition triggering.

pair-wise matching matrix:

$$M = H^{\top}H \tag{3}$$

where M is a square matrix $\in \mathbb{R}^{T\times T}$. To get a single attention weight per time step, we adopt the attention-over-attention method (Cui et al., 2017). With matrix M, we first compute row-wise attention score M^r_{ij} over M:

$$M^r_{ij} = \frac{exp(e_{ij})}{\sum_{t=1}^{T} exp(e_{it})} \tag{4}$$

where $e_{ij} = M_{ij}$. M^r can be interpreted as a word-level attention distribution over all other words. Since we would like a single weight per word, we need an additional step to aggregate these attention scores. Instead of simply averaging the scores, we follow (Cui et al., 2017)'s approach which learns the aggregation by an additional attention mechanism. We compute column-wise softmax M^c_{ij} over M:

$$M^c_{ij} = \frac{exp(e_{ij})}{\sum_{t=1}^{T} exp(e_{tj})} \tag{5}$$

The columns of M^r are then averaged, forming vector $\beta \in \mathbb{R}^T$. Finally, β is multiplied with the column-wise softmax matrix M^c to get attention vector α:

$$\alpha = M^{r\top}\beta. \tag{6}$$

Note Equations (2) to (6) have described how we derived an attention score over our input without the introduction of any new parameters, potentially minimizing the computational effect of our attention mechanism.

As a last layer to their neural network, Cui et al. (2017) sum over α to extract the most relevant input. However, we use α as weights to combine all of our hidden states h_t:

$$c = \sum_{t=1}^{T} \alpha_t h_t \tag{7}$$

where $c \in \mathbb{R}^s$. We follow the pooling with a dense layer $z = \sigma(W_z c + b_z)$, where σ is a non-linear function, matrix $W_z \in \mathbb{R}^{64 \times s}$ and vector $b_z \in \mathbb{R}^{64}$ are learned parameters. The presupposition trigger probability is computed with an affine transform followed by a softmax:

$$\hat{y} = \text{softmax}(W_o z + b_o) \tag{8}$$

where matrix $W_o \in \mathbb{R}^{2 \times 64}$ and vector $b_o \in \mathbb{R}^2$ are learned parameters. The training objective minimizes:

$$J(\theta) = \frac{1}{m} \sum_{t=1}^{m} E(\hat{y}, y) \tag{9}$$

where $E(\cdot\,,\cdot)$ is the standard cross-entropy.

5 Experiments

We compare the performance of our WP model against several models which we describe in this section. We carry out the experiments on both datasets described in Section 3. We also investigate the impact of POS tags and attention mechanism on the models' prediction accuracy.

5.1 Baselines

We compare our learning model against the following systems. The first is the most-frequent-class baseline **(MFC)** which simply labels all samples with the most frequent class of 1. The second is a logistic regression classifier **(LogReg)**, in which the probabilities describing the possible outcomes of a single input x is modeled using a logistic function. We implement this baseline classifier with the scikit-learn package (Pedregosa et al., 2011), with a CountVectorizer including bi-gram features. All of the other hyperparameters are set to default weights.

The third is a variant LSTM recurrent neural network as introduced in (Graves, 2013). The input is encoded by a bidirectional LSTM like the WP model detailed in Section 4. Instead of a self-attention mechanism, we simply mean-pool matrix H, the concatenation of all LSTM hidden states, across all time steps. This is followed by a fully connected layer and softmax function for the binary classification. Our WP model uses the same bidirectional LSTM as this baseline LSTM, and has the same number of parameters, allowing for a fair comparison of the two models. Such a standard LSTM model represents a state-of-the-art language model, as it outperforms more recent models on language modeling tasks when the number of model parameters is controlled for (Melis et al., 2017).

For the last model, we use a slight variant of the CNN sentence classification model of (Kim, 2014) based on the Britz tensorflow implementation[2].

5.2 Hyperparameters & Additional Features

After tuning, we found the following hyperparameters to work best: 64 units in fully connected layers and 40 units for POS embeddings. We used dropout with probability 0.5 and mini-batch size of 64.

For all models, we initialize word embeddings with word2vec (Mikolov et al., 2013) pretrained embeddings of size 300. Unknown words are randomly initialized to the same size as the word2vec embeddings. In early tests on the development datasets, we found that our neural networks would consistently perform better when fixing the word embeddings. All neural network performance reported in this paper use fixed embeddings.

Fully connected layers in the LSTM, CNN and WP model are regularized with dropout (Srivastava et al., 2014). The model parameters for these neural networks are fine-tuned with the Adam algorithm (Kingma and Ba, 2015). To stabilize the RNN training gradients (Pascanu et al., 2013), we perform gradient clipping for gradients below threshold value -1, or above 1. To reduce overfitting, we stop training if the development set does not improve in accuracy for 10 epochs. All performance on the test set is reported using the best trained model as measured on the development set.

In addition, we use the CoreNLP Part-of-

[2]http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

		Accuracy						
		WSJ	Gigaword					
Models	Variants	All adverbs	All adverbs	Also	Still	Again	Too	Yet
MFC	-	51.66	50.24	50.32	50.29	50.25	65.06	50.19
LogReg	+ POS	52.81	53.65	52.00	56.36	59.49	69.77	**61.05**
	- POS	54.47	52.86	56.07	55.29	58.60	67.60	58.60
CNN	+ POS	58.84	59.12	61.53	59.54	60.26	67.53	59.69
	- POS	62.16	57.21	59.76	56.95	57.28	67.84	56.53
LSTM	+ POS	74.23	60.58	81.48	60.72	**61.81**	**69.70**	59.13
	- POS	73.18	**58.86**	81.16	**58.97**	**59.93**	**68.32**	55.71
WP	+ POS	**76.09**	**60.62**	**82.42**	**61.00**	61.59	69.38	57.68
	- POS	**74.84**	**58.87**	**81.64**	**59.03**	58.49	**68.37**	**56.68**

Table 2: Performance of various models, including our weighted-pooled LSTM (WP). MFC refers to the most-frequent-class baseline, LogReg is the logistic regression baseline. LSTM and CNN correspond to strong neural network baselines. Note that we bold the performance numbers for the best performing model for each of the "+ POS" case and the "- POS" case.

Speech (POS) tagger (Manning et al., 2014) to get corresponding POS features for extracted tokens. In all of our models, we limit the maximum length of samples and POS tags to 60 tokens. For the CNN, sequences shorter than 60 tokens are zero-padded.

6 Results

Table 2 shows the performance obtained by the different models with and without POS tags. Overall, our attention model WP outperforms all other models in 10 out of 14 scenarios (combinations of datasets and whether or not POS tags are used). Importantly, our model outperforms the regular LSTM model without introducing additional parameters to the model, which highlights the advantage of WP's attention-based pooling method. For all models listed in Table 2, we find that including POS tags benefits the detection of adverbial presupposition triggers in Gigaword and PTB datasets. Note that, in Table 2, we bolded accuracy figures that were within 0.1% of the best performing WP model as McNemar's test did not show that WP significantly outperformed the other model in these cases (p value > 0.05).

Table 3 shows the confusion matrix for the best performing model (WP,+POS). The small differences in the off-diagonal entries inform us that the model misclassifications are not particularly skewed towards the presence or absence of presupposition triggers.

		Predicted	
		Absence	Presence
Actual	Absence	54,658	11,961
	Presence	11,776	55,006

Table 3: Confusion matrix for the best performing model, predicting the presence of a presupposition trigger or the absence of such as trigger.

	WP Cor.	**WP Inc.**
LSTM Cor.	101,443	6,819
LSTM Inc.	8,016	17,123

Table 4: Contingency table for correct (cor.) and incorrect (inc.) predictions between the LSTM baseline and the attention model (WP) on the Giga_also dataset.

The contingency table, shown in Table 4, shows the distribution of agreed and disagreed classification.

7 Analysis

Consider the following pair of samples that we randomly choose from the PTB dataset (shortened for readability):

1. ...Taped just as the market closed yesterday , it offers Ms. Farrell advising , " We view

the market here as going through a relatively normal cycle ... . We **continue** to feel that the stock market is the @@@@ place to be for long-term appreciation

2. ...More people are remaining independent longer presumably because they are better off physically and financially . Careers count most for the well-to-do many affluent people @@@@ place personal success and money above family

In both cases, the head word is *place*. In Example 1, the word *continue* (emphasized in the above text) suggests that adverb *still* could be used to modify head word *place* (i.e., ... *the stock market is still the place* ...). Further, it is also easy to see that *place* refers to *stock market*, which has occurred in the previous context. Our model correctly predicts this sample as containing a presupposition, this despite the complexity of the coreference across the text.

In the second case of the usage of the same main head word *place* in Example 2, our model falsely predicts the presence of a presupposition. However, even a human could read the sentence as "many people still place personal success and money above family". This underlies the subtlety and difficulty of the task at hand. The long-range dependencies and interactions within sentences seen in these examples are what motivate the use of the various deep non-linear models presented in this work, which are useful in detecting these coreferences, particularly in the case of attention mechanisms.

8 Conclusion

In this work, we have investigated the task of predicting adverbial presupposition triggers and introduced several datasets for the task. Additionally, we have presented a novel weighted-pooling attention mechanism which is incorporated into a recurrent neural network model for predicting the presence of an adverbial presuppositional trigger. Our results show that the model outperforms the CNN and LSTM, and does not add any additional parameters over the standard LSTM model. This shows its promise in classification tasks involving capturing and combining relevant information from multiple points in the previous context.

In future work, we would like to focus more on designing models that can deal with and be optimized for scenarios with severe data imbalance. We would like to also explore various applications of presupposition trigger prediction in language generation applications, as well as additional attention-based neural network architectures.

Acknowledgements

The authors would like to thank the reviewers for their valuable comments. This work was supported by the Centre de Recherche d'Informatique de Montréal (CRIM), the Fonds de Recherche du Québec – Nature et Technologies (FRQNT) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Advances in Neural Information Processing Systems (NIPS 2015)*, pages 649–657, Montreal, Canada.

David I. Beaver and Bart Geurts. 2014. Presupposition. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, winter 2014 edition. Metaphysics Research Lab, Stanford University.

R. Blutner, H. Zeevat, K. Bach, A. Bezuidenhout, R. Breheny, S. Glucksberg, F. Happé, F. Recanati, and D. Wilson. 2003. *Optimality Theory and Pragmatics*. Palgrave Studies in Pragmatics, Language and Cognition. Palgrave Macmillan UK.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602.

G. R. Miller E. B. Coleman. 1968. A measure of information gained during prose learning. *Reading Research Quarterly*, 3(3):369–386.

Joseph W. Culhane Earl F. Rankin. 1969. Comparable cloze and multiple-choice comprehension test scores. *Journal of Reading*, 13(3):193–198.

Gottlob Frege. 1892. Über sinn und bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2007. English gigaword third edition. Technical report, Linguistic Data Consortium.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301.

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

Jad Kabbara, Yulan Feng, and Jackie Chi Kit Cheung. 2016. Capturing pragmatic knowledge in article usage prediction using lstms. In *COLING*, pages 2625–2634.

Qiang Kang. 2012. The use of too as a pragmatic presupposition trigger. *Canadian Social Science*, 8(6):165–169.

David Kaplan. 1970. What is Russell's theory of descriptions? In Wolfgang Yourgrau and Allen D. Breck, editors, *Physics, Logic, and History*, pages 277–295. Plenum Press.

Layth Muthana Khaleel. 2010. An analysis of presupposition triggers in english journalistic texts. *Of College Of Education For Women*, 21(2):523–551.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceeding of the 2015 International Conference on Learning Representation (ICLR 2015)*, San Diego, California.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 128–135, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *CoRR*, abs/1707.05589.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, Vancouver, Canada.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading Comprehension Tests As Evaluation for Computer-based Language Understanding Sytems - Volume 6*, ANLP/NAACL-ReadingComp '00, pages 13–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Robert Stalnaker. 1973. Presuppositions. *Journal of philosophical logic*, 2(4):447–457.

Robert Stalnaker. 1998. On the representation of context. *Journal of Logic, Language and Information*, 7(1):3–19.

Peter F. Strawson. 1950. On referring. *Mind*, 59(235):320–344.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28*, pages 2440–2448.

Wilson L. Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Quarterly*, 30(4):415.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Jakob Uszkoreit, Aidan N Gomez, and Ł ukasz Kaiser. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5994–6004.

Javad Zare, Ehsan Abbaspour, and Mahdi Rajaee Nia. 2012. Presupposition trigger—a comparative analysis of broadcast news discourse. *International Journal of Linguistics*, 4(3):734–743.

Author Index